|  |
| --- |
| **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION INTERNATIONALE DE NORMALISATION ISO/IEC JTC 1/SC 29/WG 5 MPEG JOINT VIDEO EXPERTS TEAM WITH ITU-T SG 16** |
| **ISO/IEC JTC 1 / SC 29 / WG 5 N 259** |
| **Online – 17–26 January 2024** |
| |  |  | | --- | --- | | **Source:** | **Convenor (Jens-Rainer Ohm)** | | **Title:** | **Technologies under consideration for future extensions of VSEI (version 3)** | | **Type:** | **General** | | **Subtype:** | **Other** | | **Status:** | **Approved** | | **Date:** | **2024-03-29** | | **Expected Action:** | **Info** | | **Action due date:** | **N/A** | | **Pages:** | **51** (not including 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**  33rd Meeting, by teleconference, 17–26 January 2024 | Document: JVET-AG2032-v1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Technologies under consideration for future extensions of VSEI (version 3)** | | |
| *Status:* | Output document approved by JVET | | |
| *Purpose:* | Draft text | | |
| *Author(s) or Contact(s):* | Sean McCarthy Jie Chen Sachin Deshpande Miska M. Hannuksela Hendry | Email: | sean.mccarthy@dolby.com  jiechen.cj@alibaba-inc.com [sdeshpande@sharplabs.com](mailto:sdeshpande@sharplabs.com) miska.hannuksela@nokia.com dr.hendry@lge.com |
| *Source:* | Editors | | |

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

# Abstract

This document contains draft text for changes under consideration for future extensions to the versatile supplemental enhancement information messages for coded video bitstreams (VSEI) standard (Rec. ITU-T H.274 | ISO/IEC 23002-7) to modify existing SEI messages, specify additional SEI messages, or specify other functionalities. This document also contains draft text for changes under consideration for future extensions to the VVC, HEVC, and AVC standards for specifying the use of SEI messages specified in VSEI in VVC, HEVC, and AVC bitstreams, respectively.

This version of the document removes draft text present in previous versions for support of encoder optimization information, source picture timing information, and object mask information SEI messages and support of modality information in VUI. Support for those items are now included in SEI messages for VSEI version 4 (Draft 1) (JVET-AG2034).

This version of the document adds draft text changes for adding support for copyright, AI masking, generative face video, and film grain region characteristics SEI message and for support of versatile SEI SBPS and versatile SEI message.

The base text for this version of the document is JVET-AF2032.

# Changes that have been integrated:

Summary: changes noted below corresponding to aspects of JVET-AG0044, JVET-AG0045, JVET-AG0051, JVET-AG0086, JVET-AG0087, JVET-AG0203, and JVET-AG0328

* Remove encoder optimization information SEI message (move to JVET-AG2034)
* Add the capability for indicating temporal sublayers where optimization has been applied into the the encoder optimization information SEI message (JVET-AG0086, item 2 option A).
* Remove source picture timing information SEI message (move to JVET-AG2034)
* Remove object mask information SEI message (move to JVET-AG2034)
* Remove modality information in VUI (move to JVET-AG2034 as modality information SEI message (JVET-AG0322))
* Remove NNPFC SEI message application information signaling (move to JVET-AG2034)
* Include copyright SEI message as proposed in JVET-AG0044 proposal 1
* Include AI marking SEI message as proposed in JVET-AG0045 proposal 1
* Include generative face video SEI message
  + as proposed in JVET-AG0203
    - Aspect 1 (gating flag for signalling the translator) (also agreed in context of JVET-AG0087)
    - Aspect 2 (predictive signalling of matrix elements)
    - Aspect 3 (more flexible interface between translator and model in terms of numbers of keypoints and matrix elements)
  + with modifications agreed from JVET-AG0087 as follows.
    - enable the capability of sending parameter translator description only at an IRAP
    - allow multiple GFV SEI messages in a picture unit, e.g., carried within suffix SEI NAL units. Additional ingredients of this item include the following:
      * Each new instance of a GFV SEI message will invoke the generative neural network (NN) inference.
      * To differentiate between repetitions and new instances of GFV SEI messages, add a counter syntax element (gfv\_cnt) in the syntax.
* Include film grain regions characteristics SEI message as proposed in JVET-AG0328
* Include versatile SEI RBSP and versatile SEI message as proposed in JVET-AG0051
  + move to a separate section together with LSEI
  + Proponents of both approaches are asked to work out possible combination

# Changes to the specification text:

*Changes to VSEI subclause 8.28 for adding the support of grouping of NNPFs*

8.28 Neural-network post-filter SEI messages

**8.28.1 General** **post-processing filtering process using NNPFs**

**8.28.1.1 General**

Input to this process is a bitstream BitstreamToFilter. Output of this process is a list of NNPF output pictures ListNnpfOutputPics.

First, BitstreamToFilter is decoded, and the list CroppedDecodedPictures is set to be the list of the cropped decoded pictures in output order resulted from decoding BitstreamToFilter.

Second, NnpfCand is set to contain any single NNPF or any single NNPF group. When NnpfCand contains an NNPF group with nnpfgc\_grouping\_type equal to 3, the subseequnt specifications of this subclause apply when NnpfCand is set to contain individually each member NNPF, if any, and each member NNPF group, if any, of the NNPF group with nnpfgc\_grouping\_type equal to 3.

Third, the filtering process for one picture, as specified in subclause 8.28.1.2, is repeatedly invoked, in output order, for each cropped decoded picture that is in CroppedDecodedPictures and for which the single NNPF contained in NnpfCand or the single NNPF group contained in NnpfCand, or one or more NNPFs or NNPF groups defined as alternatives or alternating in the NNPF group contained in NnpfCand are activated.

The order of the pictures in ListNnpfOutputPics is in output order.

Within ListNnpfOutputPics there shall be no more than one picture pertaining to any particular output time instance. [Ed. Check phrasing of this. “Pertains” is not used in a similar way anywhere in the standard.] When for any particular picture in CroppedDecodedPictures there are multiple NNPFs activated and only one of the NNPFs is allowed to be chosen to be applied although any of the NNPFs may be chosen, the above constraint shall apply regardless of which NNPF is chosen to be applied to the particular picture.

BitstreamToFilter may be processed multiple times to generate multiple different ListNnpfOutputPics through the second and third steps above.

**8.28.1.2 Filtering process for one picture**

The filtering process specified in this subclause applies to each cropped decoded picture, referred to as the current picture, that is in CroppedDecodedPictures and for which one or more NNPFs or NNPF groups in NnpfCand are activated.

An NNPF or an NNPF group to be applied to the current picture is selected as follows:

– If NnpfCand contains a single NNPF and that NNPF is activated for the current picture according to an NNPFA SEI message, that NNPF is selected to be applied to the current picture.

– Otherwise, if NnpfCand contains an NNPF group with nnpfgc\_grouping\_type equal to 2 and any NNPF of the NNPF group is activated for the current picture according to NNPFA SEI message, that NNPF is selected to be applied to the current picture.

– Otherwise, if NnpfCand contains an NNPF group with nnpfgc\_grouping\_type equal to 0 and that NNPF group is activated for the current picture according to an NNPFGA SEI message, that NNPF group is selected to be applied to the current picture.

– Otherwise (NnpfCand contains an NNPF group with nnpfgc\_grouping\_type equal to 1), the following applies:

– A set of candidate NNPFs or NNPF groups candSet is initially empty and then set to contain the following:

– The NNPFs that are activated for the current picture according to NNPFA SEI messages and are included in the NNPF group contained in NnpfCand.

– The NNPF groups that are activated for the current picture according to NNPFGA SEI messages and are included in the NNPF group contained in NnpfCand.

– For each candidate NNPF or NNPF group candFilter in candSet, the following applies:

– When one or more of the input pictures of candFilter are input pictures to the NNPF or NNPF group prevFilter that was used in any previous invocation of the filtering process specified in this subclause for the same NnpfCand, candFilter is excluded from candSet.

– Any NNPF or NNPF group remaining in candSet is selected to be applied to the current picture.

When applying an NNPF to the current picture, the following applies:

– The filtered and/or interpolated pictures are generated by the NNPF by applying the NNPF process specified in the semantics of the NNPFC SEI message, in a patch-wise manner, to the current picture.

– The order of the pictures generated by the NNPF by applying the NNPF process being stored into the output tensor of the NNPF is in output order.

– The pictures generated by the NNPF and output by the NNPF process are included into ListNnpfOutputPics, in the same order as when the pictures are stored into the output tensor of the NNPF.

When applying an NNPF group to the current picture, the following applies:

– The filtered and/or interpolated pictures are generated by applying the NNPF process specified in the semantics of the NNPFC SEI message, in a patch-wise manner, as specified in the semantics of the NNPFGA SEI message activating the NNPF group.

– The pictures in FinalNnpfgaOutputPicList are included into ListNnpfOutputPics, in the same order as the pictures are stored in FinalNnpfgaOutputPicList.

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

|  |  |
| --- | --- |
| nn\_post\_filter\_group\_characteristics( payloadSize ) { | **Descriptor** |
| **nnpfgc\_id** | ue(v) |
| **nnpfgc\_grouping\_type** | ue(v) |
| if( nnpfgc\_grouping\_type = = 0 | | nnpfgc\_grouping\_type = = 2 ) |  |
| **nnpfgc\_purpose** | u(16) |
| **nnpfgc\_num\_members\_minus2** | ue(v) |
| for( i = 0; i <= nnpfgc\_num\_members\_minus2 + 1; i++ ) { |  |
| **nnpfgc\_member\_id**[ i ] | ue(v) |
| if( nnpfgc\_grouping\_type = = 1 | | nnpfgc\_grouping\_type = = 3 ) |  |
| **nnpfgc\_member\_purpose**[ i ] | u(16) |
| } |  |
| **nnpfgc\_complexity\_info\_present\_flag** | u(1) |
| if( nnpfgc\_complexity\_info\_present\_flag ) { |  |
| **nnpfgc\_parameter\_type\_idc** | u(2) |
| if( nnpfgc\_parameter\_type\_idc != 2 ) |  |
| **nnpfgc\_log2\_parameter\_bit\_length\_minus3** | u(2) |
| **nnpfgc\_num\_parameters\_idc** | u(6) |
| **nnpfgc\_num\_kmac\_operations\_idc** | ue(v) |
| **nnpfgc\_total\_kilobyte\_size** | ue(v) |
| } |  |
| } |  |

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

The neural-network post-filter group characteristics (NNPFGC) SEI message specifies a neural network post-filter (NNPF) group. It is indicated by the SEI message if the NNPF group defines an NNPF cascade or defines NNPFs or NNPF groups of NNPF cascades that are alternatives to each other. The use of NNPF groups of NNPF cascades for specific pictures is indicated with neural-network post-filter group activation (NNPFGA) SEI messages. An NNPF group may have members that are NNPF or NNPF group. An NNPF is a direct member of an NNPF group when it is included in the list of members in the NNPFGC SEI message of the NNPF group. An NNPF is an indirect member of an NNPF group nnpfg\_1 when it is a member of another NNPF group nnpfg\_2 and nnpfg\_2 is a member of nnpfg\_1.

**nnpfgc\_id** contains an identifying number that may be used to identify an NNPF group. The value of nnpfgc\_id shall be in the range of 0 to 232 − 2, inclusive. Values of nnpfgc\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering an NNPFGC SEI message with nnpfgc\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, shall ignore the SEI message. The value of nnpfgc\_id shall not be equal to any nnpfc\_id value of any NNPFC SEI message present in the same CLVS. When the value of nnpfgc\_id of an NNPFGC SEI message nnpfgcSeiA is equal to the value of nnpfgc\_id of another NNPFGC SEI message nnpfgcSeiB present in the same CLVS, nnpfgcSeiA and nnpfgcSeiB shall be identical.

**nnpfgc\_grouping\_type** equal to 0 indicates that this SEI message specifies a group of cascaded neural-network post-filters.

nnpfgc\_grouping\_type equal to 1 indicates that the NNPFs or NNPF groups identified by the nnpfgc\_member\_id[ i ] are alternatives to each other out of which the post-processor should select only one to be applied.

nnpfgc\_grouping\_type equal to 2 indicates that this SEI message specifies a group of NNPFs that are intended to be used jointly and are activated in an alternating manner so that at most one of these NNPFs is activate for any picture.

nnpfgc\_grouping\_type equal to 3 indicates that the NNPFs or NNPF groups identified by the nnpfgc\_member\_id[ i ] are intended to be used in parallel.

nnpfgc\_grouping\_type equal to 4 indicates that the NNPFs or NNPF groups identified by the nnpfgc\_member\_id[ i ] are optional, i.e., may or may not be applied by the post-processor.

The value of nnpfgc\_grouping\_type shall be in the range of 0 to 255, inclusive. Values of nnpfgc\_grouping\_type in the range of 5 to 255, inclusive, are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPFGC SEI messages with nnpfgc\_grouping\_type in the range of 5 to 255, inclusive.

**nnpfgc\_purpose** has the semantics of nnpfc\_purpose but with the exception that the semantics are specified for the NNPF group defined by this SEI message rather than the NNPF defined by an NNPFC SEI message. When present, the value of nnpfgc\_purpose shall be the union of the values of nnpfc\_purpose of direct and indirect members of the NNPF group.

**nnpfgc\_num\_members\_minus2** plus 2indicates the number of NNPFs or NNPF groups in the NNPF group that this SEI message defines.

**nnpfgc\_member\_id**[ i ] indicates the i-th member in the NNPF group defined by this SEI message as follows:

– If there is an NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] defined in the CLVS, the i-th member in the NNPF group defined by this SEI message is an NNPF that has nnpfc\_id equal to nnpfgc\_member\_id[ i ].

– Otherwise (there is no NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] defined in the CLVS), the i-th member in the NNPF group defined by this SEI message is an NNPF group with nnpfgc\_id equal to nnpfgc\_member\_id[ i ].

When an nnpfgc\_member\_id[ i ] value references an nnpfgc\_id value of an NNPFGC SEI message nnpfgcSei, it is a requirement of bitstream conformance that the NNPFGC SEI message nnpfgcSei shall have nnpfgc\_grouping\_type equal to 0.

NOTE 1 – In other words, when a second NNPF group is a member of a first NNPF group, the type of the second NNPF group cannot be any other than a cascade of NNPFs.

When nnpfgc\_grouping\_type is equal to 0 or 2, it is a requirement of bitstream conformance that there is an NNPF with nnpfc\_id value equal to nnpfgc\_member\_id[ i ] defined in the CLVS.

NOTE 2 – In other words, the members of a cascade or alternating NNPF group are individual NNPFs and cannot be NNPF groups.

When nnpfgc\_grouping\_type is equal to 1, 3, or 4, it is a requirement of bitstream conformance that there is an NNPF with nnpfc\_id value equal to nnpfgc\_member\_id[ i ] or an NNPF group with nnpfgc\_id value equal to nnpfgc\_member\_id[ i ] defined in the CLVS.

When nnpfgc\_grouping\_type is equal to 0, the NNPFs with nnpfc\_id equal to nnpfgc\_member\_id[ i ] are performed in cascade in increasing order of i, as activated by an NNPFGA SEI message with nnpfga\_target\_id equal to nnpfgc\_id.

**nnpfgc\_member\_purpose**[ i ] has the semantics of nnpfc\_purpose but with the exception that the semantics are specified for the NNPFC or NNPFGC associated with nnpfgc\_member[ i ]. When present, the value of nnpfgc\_member\_purpose[ i ] shall be as follows:

– If nnpfgc\_member\_id[ i ] is equal to the value of nnpfc\_id of an NNPF, the value of nnpfgc\_member\_purpose[ i ] shall be equal to the value of nnpfc\_purpose of the NNPF.

– Otherwise (nnpfgc\_member\_id[ i ] is equal to the value of nnpfgc\_id of an NNPF group ), the value of nnpfgc\_member\_purpose[ i ] shall be equal to nnpfgc\_purpose of the associatied NNPF group.

nnpfgc\_complexity\_info\_present\_flag, nnpfgc\_parameter\_type\_idc, nnpfgc\_log2\_parameter\_bit\_length\_minus3, nnpfgc\_num\_parameters\_idc, nnpfgc\_num\_kmac\_operations\_idc, and nnpfgc\_total\_kilobyte\_size have the semantics of nnpfc\_complexity\_info\_present\_flag, nnpfc\_parameter\_type\_idc, nnpfc\_log2\_parameter\_bit\_length\_minus3, nnpfc\_num\_parameters\_idc, nnpfc\_num\_kmac\_operations\_idc, and nnpfc\_total\_kilobyte\_size, respectively, but with the exception that the semantics are specified for the NNPF group defined by this SEI message rather than the NNPF defined by an NNPFC SEI message. When nnpfgc\_grouping\_type is equal to 1, nnpfgc\_complexity\_info\_present\_flag shall be equal to 0.

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

|  |  |
| --- | --- |
| nn\_post\_filter\_group\_activation( payloadSize ) { | **Descriptor** |
| **nnpfga\_target\_id** | ue(v) |
| **nnpfga\_cancel\_flag** | u(1) |
| if( !nnpfga\_cancel\_flag ) { |  |
| **nnpfga\_persistence\_flag** | u(1) |
| **nnpfga\_no\_prev\_clvs\_flag** | u(1) |
| if( nnpfga\_persistence\_flag ) |  |
| **nnpfga\_no\_foll\_clvs\_flag** | u(1) |
| **nnpfga\_num\_filters\_minus2** | ue(v) |
| for( i = 0; i <= nnpfga\_num\_filters\_minus2 + 1; i++ ) { |  |
| **nnpfga\_target\_base\_flag**[ i ] | u(1) |
| **nnpfga\_input\_all\_pics\_flag**[ i ] | u(1) |
| if( !nnpfga\_input\_all\_pics\_flag[ i ] ) { |  |
| **nnpfga\_num\_input\_pics\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= nnpfga\_num\_input\_pics\_minus1[ i ]; j++ ) |  |
| **nnpfga\_input\_pic\_skip\_count**[ i ][ j ] | ue(v) |
| } |  |
| **nnpfga\_num\_output\_entries**[ i ] | ue(v) |
| for( j = 0; j < nnpfga\_num\_output\_entries[ i ]; j++ ) |  |
| **nnpfga\_output\_flag**[ i ][ j ] | u(1) |
| } |  |
| **nnpfga\_num\_output\_pic\_update** | ue(v) |
| for( i = 0; i < nnpfga\_num\_output\_pic\_update; i++ ) |  |
| **nnpfga\_output\_pic\_update\_flag**[ i ] | u(1) |
| } |  |
| } |  |

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

The neural-network post-filter group activation (NNPFGA) SEI message activates or de-activates the possible use of the target neural-network post-processing filter group (NNPFG) of NNPF groups, identified by nnpfga\_target\_id, for post-processing filtering of a set of pictures.

nnpfgc\_grouping\_type for the identfied NNPF group, which has nnpfgc\_id equal to nnpfga\_target\_id, shall be equal to 0 (cascade) or 1 (alternatives). When nnpfgc\_grouping\_type of the identified NNPF group is equal to 1, each member of the group shall have the same number of input pictures and NNPF output pictures.

For a particular picture for which the NNPFG is activated, the associated NNPFGC SEI message is the last NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id, that precedes the first VCL NAL unit of the current picture in decoding order. If the nnpfgc\_grouping\_type in the associated NNPGC SEI message is equal to 0, the target NNPFG is the NNPFG specified by the associated NNPFGC SEI message. Otherwise (nnpfgc\_grouping\_type in the associated NNPFGC SEI message is equal to 1), the target NNPFG is any NNPFG, if any, that is a member of the NNPFG specified by the associated NNPFGC SEI message. The NNPFs of the target NNPFG are defined by the NNPFC SEI messages that have nnpfc\_id equal to any nnpfgc\_member\_id[ i ] value of the target NNPFG and are present in the current picture unit or precede the current picture in decoding order.

NOTE 1 – The members of the target NNPFG are individual NNPFs and cannot be NNPF groups.

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

– Input picture width and height in units of luma samples, denoted herein by InitCroppedWidth[ idx ] and InitCroppedHeight[ idx ], respectively, of the candidate input pictures with index idx in the range of 0 to numCandInputPics − 1, inclusive, that may be used as input for the NNPFG.

– Luma sample array InitCroppedYPic[ idx ] and chroma sample arrays InitCroppedCbPic[ idx ] and InitCroppedCrPic[ idx ], when present, of the candidate input pictures with index idx in the range of 0 to numCandInputPics − 1, inclusive, that may be used as input for the NNPFG.

– Bit depth BitDepthY for the luma sample array of the candidate input pictures.

– Bit depth BitDepthC for the chroma sample arrays, if any, of the candidate input pictures.

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

– When nnpfc\_auxiliary\_inp\_idc is equal to 1, a filtering strength control value array StrengthControlVal[ idx ] that shall contain real numbers in the range of 0 to 1, inclusive, of the candidate input pictures with index idx in the range of 0 to numCandInputPics − 1, inclusive.

Candidate input picture with index 0 corresponds to the picture for which the NNPFG is activated by this NNPFGA SEI message. Candidate input picture with index i in the range of 1 to numCandInputPics − 1, inclusive, precedes the candidate input picture with index i − 1 in output order. Let candInputPicList[ 0 ] be the list of candidate input pictures in inverse output order.

**nnpfga\_target\_id** indicates the target NNPFG, which is specified by the NNPFGC SEI message that pertains to the current picture and have nnpfgc\_id equal to nnpfga\_target\_id. [Ed. Check phrasing of this. “Pertains” is not used in a similar way anywhere in the standard.]

The value of nnpfga\_target\_id shall be in the range of 0 to 232 − 2, inclusive.

An NNPFGA SEI message with a particular value of nnpfga\_target\_id shall not be present in a current PU unless there is an NNPFGC SEI message with nnpfgc\_id equal to the particular value of nnpfga\_target\_id and nnpfgc\_grouping\_type equal to 0 present in the current PU or in a PU that precedes the current PU in decoding order within the current CLVS.

When a PU contains both an NNPFGC SEI message with a particular value of nnpfgc\_id and an NNPFGA SEI message with nnpfga\_target\_id equal to the particular value of nnpfgc\_id, the NNPFGC SEI message shall precede the NNPFGA SEI message in decoding order.

**nnpfga\_cancel\_flag** equal to 1 indicates that the persistence of the target NNPFG established by any previous NNPFGA SEI message with the same nnpfga\_target\_id as the current SEI message is cancelled, i.e., the target NNPFG is no longer used unless it is activated by another NNPFGA SEI message with the same nnpfga\_target\_id as the current SEI message and nnpfga\_cancel\_flag equal to 0. nnpfga\_cancel\_flag equal to 0 indicates that the target NNPFG is activated for use.

**nnpfga\_persistence\_flag** specifies the persistence of the target NNPFG for the current layer.

nnpfga\_persistence\_flag equal to 0 specifies that the target NNPFG may be used for post-processing filtering for the current picture only.

nnpfga\_persistence\_flag equal to 1 specifies that the target NNPFG may be used for post-processing filtering for the current picture and all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer associated with a NNPFGA SEI message with the same nnpfga\_target\_id as the current SEI message that follows the current picture in output order.

NOTE 2 – The target NNPFG is not applied for this subsequent picture in the current layer associated with a NNPFGA SEI message with the same nnpfga\_target\_id as the current SEI message.

Let the nnpfgcTargetPictures be the set of pictures to which the last NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id that precedes the current NNPFGA SEI message in decoding order pertains. [Ed. Check phrasing of this. “Pertains” is not used in a similar way anywhere in the standard.] Let nnpfgaTargetPictures be the set of pictures for which the target NNPFG is activated by the current NNPFGA SEI message. It is a requirement of bitstream conformance that any picture included in nnpfgaTargetPictures shall also be included in nnpfgcTargetPictures.

**nnpfga\_no\_prev\_clvs\_flag** equal to 1 specifies that the input pictures for the NNPF group do not originate from a previous CLVS. nnpfga\_no\_prev\_clvs\_flag equal to 0 specifies that the input pictures for the NNPF group may or may not originate from a previous CLVS.

NOTE 3 – The value of nnpfga\_no\_prev\_clvs\_flag can be changed from 0 to 1, when the current CLVS is spliced from another bitstream next to the previous CLVS and this NNPFGA SEI message would cause one or more input pictures to be selected from one or more previous CLVSs and therefore is likely to impact the output of the target NNPF group negatively.

**nnpfga\_no\_foll\_clvs\_flag** equal to 1 specifies that when this NNPFGA SEI message persists for the last PU of a CLVS in output order, the NNPFGA SEI message is treated like it persisted for the last PU, in output order, of the current layer within the bitstream. When this NNPFGA SEI message does not persist for the last PU, in output order, of a CLVS in output order or nnpfga\_no\_foll\_clvs\_flag is equal to 0, the value of nnpfga\_no\_foll\_clvs\_flag causes no specific impact.

NOTE 4 – The value of nnpfga\_no\_foll\_clvs\_flag can be changed from 0 to 1 for an NNPF group including a picture-rate-upsampling NNPF, when the following CLVS is spliced from a different bitstream next to the current CLVS. Consequently, the NNPF process of the picture-rate-upsampling NNPF interpolates pictures up to the end of the current CLVS using input pictures originating from the current CLVS only.

**nnpfga\_num\_filters\_minus2** plus 2indicates the number of NNPFs in the NNPFG that this SEI message activates. The value of nnpfga\_num\_filters\_minus2 shall be equal to the value of nnpfgc\_num\_members\_minus2 in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id.

**nnpfga\_target\_base\_flag**[ i ] equal to 1 specifies that the i-th NNPF in the target NNPFG is the base NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id. nnpfga\_target\_base\_flag[ i ] equal to 0 specifies that the i-th NNPF in the target NNPFG is the NNPF specified by the last NNPFC SEI message that has nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id, precedes the first VCL NAL unit of the current picture in decoding order, and is not a repetition of the NNPFC SEI message that contains the base NNPF.

**nnpfga\_input\_all\_pics\_flag**[ i ] equal to 1 specifies that the input pictures to the i-th NNPF are selected from the list of candidate input pictures candInputPicList[ i ] without skipping. nnpfga\_input\_all\_pics\_flag[ i ] equal to 0 specifies that the input pictures to the i-th NNPF are selected from the list of candidate input pictures candInputPicList[ i ] in a manner that some candidate input pictures are skipped.

**nnpfga\_num\_input\_pics\_minus1**[ i ] specifies the number of input pictures for the i-th NNPF in the target NNPFG. When present, nnpfga\_num\_input\_pics\_minus1[ i ] shall be equal to nnpfc\_num\_input\_pics\_minus1 for an NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] of an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id. When not present, nnpfga\_num\_input\_pics\_minus1[ i ] is inferred to be equal to nnpfc\_num\_input\_pics\_minus1 for an NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id.

**nnpfga\_input\_pic\_skip\_count**[ i ][ j ] specifies a j-th picture count that is skipped in the list of candidate input pictures candInputPicList[ i ] when selecting input pictures for the NNPF activated by the i-th loop entry. When nnpfga\_input\_pic\_skip\_count[ i ][ j ] is not present, it is inferred to be equal to 0 for all values of j in the range of 0 to nnpfga\_num\_input\_pics\_minus1[ i ], inclusive. The variable numCandInputPics, which indicates the number of candidate input pictures to the NNPFG, is derived as follows:

numCandInputPics = 0  
for( j = 0; j <= nnpfga\_num\_input\_pics\_minus1[ 0 ]; j++ )  
 numCandInputPics += 1 + nnpfga\_input\_pic\_skip\_count[ 0 ][ j ] (xx)

Let candInputPicList[ m ] for m in the range of 1 to nnpfga\_num\_filters\_minus2 + 1, inclusive, be a list of pictures in inverse output order that is initially empty and formed in decreasing order of n in the range of 0 to m − 1, inclusive, by including each picture that is output by the NNPF process of the n-th loop entry that has no corresponding picture already present in candInputPicList[ m ], and lastly including each picture present in candInputPicList[ 0 ] that has no corresponding picture already present in candInputPicList[ m ].

When a candidate input picture candInputPicList[ m ][ idx ] for any value of m in the range of 1 to nnpfga\_num\_filters\_minus2 + 1, inclusive, is an NNPF output picture of the n-th NNPF process with the value of n being less than the value of m, the width and height of the candidate input picture are respectively equal to nnpfcOutputPicWidth and nnpfcOutputPicHeight of the NNPF output picture.

The list of input pictures inputPicList[ m ] to the NNPF of the m-th loop entry is derived as follows:

candIdx = 0  
for( k = 0; k <= nnpfga\_num\_input\_pics\_minus1[ m ]; k++ ) {  
 candIdx += nnpfga\_input\_pic\_skip\_count[ m ][ k ]  
 inputPicList[ m ][ k ] = candInputPicList[ m ][ candIdx ] (xx)  
 candIdx++  
}

It is a requirement of bitstream conformance that candIdx shall not exceed the number of pictures in candInputPicList[ m ].

It is a requirement of bitstream conformance that the pictures present in inputPicList[ m ], for any value of m in the range of 1 to nnpfga\_num\_filters\_minus2 + 1, inclusive, shall have the same width, height, bit depth, and chroma format.

For purposes of interpretation of the NNPFC SEI message with nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id, the following variables are specified for the i-th loop entry:

– The variables BitDepthY, BitDepthC, and ChromaFormatIdc are used as provided for the interpretation of this SEI message.

– CroppedWidth and CroppedHeight are set equal to the width and height of the pictures in inputPicList[ i ], respectively, in units of luma samples.

– For each input picture k in the range of 0 to nnpfga\_num\_input\_pics\_minus1[ i ], inclusive, the following applies:

– CroppedYPic[ k ], CroppedCbPic[ k ], and CroppedCrPic[ k ], when present, are set equal to respective sample array of inputPicList[ i ][ k ]

– When nnpfc\_auxiliary\_inp\_idc is equal to 1 for the NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id, the following applies:

– It is a requirement of bitstream conformance that inputPicList[ i ][ k ] is the same as candInputPicList[ 0 ][ idx ] for any value of idx in the range of 0 to numCandInputPics − 1, inclusive.

– StrengthControlVal[ k ] is set equal to InitStrengthControlVal[ idx ].

**nnpfga\_num\_output\_entries**[ i ] specifies the number of nnpfga\_output\_flag[ i ][ j ] syntax elements present in the NNPFGA SEI message. The value of nnpfga\_num\_output\_entries[ i ] shall be in the range of 0 to NumInpPicsInOutputTensor, inclusive, for an NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] of an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id..

**nnpfga\_output\_flag**[ i ][ j ] equal to 1 specifies that the NNPF-generated picture that corresponds to the input picture having index InpIdx[ j ] derived for the i-th NNPF of the target NNPFG is output by the NNPF process activated by this loop entry, where the NNPF process is specified in the semantics of the NNPFC SEI message. nnpfga\_output\_flag[ i ][ j ] equal to 0 specifies that the NNPF-generated picture that corresponds to the input picture having index InpIdx[ j ] derived for the i-th NNPF of the target NNPFG is not output by the NNPF process activated by this loop entry. When nnpfga\_num\_output\_entries[ i ] is less than NumInpPicsInOutputTensor derived for the i-th NNPF of the target NNPFG, nnpfga\_output\_flag[ i ][ j ] is inferred to be equal to 1 for each value of i in the range of nnpfga\_num\_output\_entries[ i ] to NumInpPicsInOutputTensor − 1, inclusive.

Let NnpfgaOutputPicList, which is the list of pictures output by NNPF process of the NNPFG in output order, be initially empty and formed in decreasing order of n in the range of 0 to nnpfga\_num\_filters\_minus2 + 1, inclusive, by including each picture that is output by the NNPF process of the n-th loop entry that has no corresponding picture already present in NnpfgaOutputPicList.

**nnpfga\_num\_output\_pic\_update** specifies the number of output picture update flag that is present. When nnpfga\_num\_output\_pic\_update is greater than 0, nnpfga\_output\_pic\_update\_flag[ i ] for i in the range from 0 to nnpfga\_num\_output\_pic\_update − 1, inclusive, are present. The value of nnpfga\_num\_output\_pic\_update shall be in the range from 0 to number of pictures in NnpfgaOutputPicList, inclusive.

**nnpfga\_output\_pic\_update\_flag**[ i ] equal to 1 specifies that the i-th picture NnpfgaOutputPicList is included in the final output picture list. nnpfga\_output\_pic\_update\_flag[ i ] equal to 0 specifies that the i-th picture NnpfgaOutputPicList is not included in the final output picture list. When the value of nnpfga\_num\_output\_pic\_update is less than the number of pictures in NnpfgaOutputPicList, the value of nnpfga\_output\_pic\_update\_flag [ i ] for i in the range from nnpfga\_num\_output\_pic\_update to the number of pictures in NnpfgaOutputPicList − 1, inclusive, is inferred to be equal to 1.

Let numCandidateOutputPics be the number of pictures in NnpfgaOutputPicList. The list of output picture flag candOutputPicFlag[ i ] for i in the range from 0 to numCandidateOutputPics − 1, inclusive is derived as follows:

for( i = 0; i < numCandidateOutputPics; i++ )  
 candOutputPicFlag[ i ] = 1  
for( i = 0; i < nnpfga\_num\_output\_pic\_update; i++ ) (xx)  
 candOutputPicFlag[ i ] = nnpfga\_output\_pic\_update\_flag[ i ]

Let FinalNnpfgaOutputPicList, which is the list of pictures output by NNPF process of the NNPFG in output order, be derived as follows:

j = 0  
for( i = 0; i < numCandidateOutputPics; i++ )  
 if( candOutputPicFlag[ i ] ) (xx)  
 FinalNnpfgaOutputPicList[ j++ ] = NnpfgaOutputPicList[ i ]

*Alternative changes to VSEI subclause 8.28 and VVC subclause D.12.11 for adding the support of grouping of PFs (both NNPFs and non-NN PFs)*

**8.28 General post-processing filtering process and neural-network post-filter SEI messages**

**8.28.1 General post-processing filtering process using PPFs**

**8.28.1.1 General**

Input to this process is a bitstream BitstreamToFilter. Output of this process is a list of output pictures ListOutputPics.

A post-processing filter (PPF) may be indicated by an SEI message for which the payloadType value is in SeiProcessingOrderSeiList, specified in the semantics of the SEI processing order (SPO) SEI message.

For each picture, there can be multiple PPFs activated and belonging to one or more PPF groups. PPF groups are alternative to each other, i.e., at most one group can be chosen to be applied.

A special PPF cascading case is defined as the case when such two PPFs are both activated for a picture: the two PPFs are both NNPFs (i.e., the payloadType value for the PPFs indicates the NNPFC SEI message), one of the two NNPFs has nnpfc\_purpose equal to 4 and the other has multiple input pictures, and neither of the two NNPFs is associated with an SEI processing order SEI message. In this case, the two NNPFs are implicitly considered as belonging to one PPF group, and the NNPF with nnpfc\_purpose equal to 4 is applied first.

Except for the special PPF cascading case, each PPF group containing multiple PPFs is associated with an SPO SEI message with a particular value of po\_id.

Except for the special PPF cascading case, any PPF not associated with an SPO SEI message is in its own PPF group.

One or more PPFs in the chosen PPF group can be applied. When multiple PPFs (in the chosen PPF group) are applied, they are applied in the cascading manner, meaning that they are applied in the order indicated by the SEI processing order SEI message associated with the chosen PPF group, and for each applied PPF that is not the last applied PPF, the output is used as the input of the next applied PPF.

First, BitstreamToFilter is decoded, the list CroppedDecodedPictures is set to be the list of the cropped decoded pictures in output order resulted from decoding BitstreamToFilter, and the list ListOutputPics is initialized to be the same as CroppedDecodedPictures.

Second, the filtering process for one picture, as specified in subclause 8.28.1.2, is repeatedly invoked, in output order, for each cropped decoded picture that is in CroppedDecodedPictures and for which one or more PPFs of one or more PPF groups are activated and only one of the groups is chosen to be applied.

The order of the pictures in ListOutputPics is in output order.

Within ListOutputPics there shall be no more than one picture pertaining to any particular output time instance. [Ed. Check phrasing of this. “Pertains” is not used in a similar way anywhere in the standard.] When for any particular picture in CroppedDecodedPictures there are PPFs of multiple PPF groups activated, the above constraint shall apply regardless of which group of PPFs is chosen to be applied when the particular picture is the current picture.

For any particular pair of pictures inputPicA and inputPicB consecutive in output order in CroppedDecodedPictures, when there are one or more pictures interpolatedPicSetA in ListOutputPics between inputPicA and inputPicB in output order, the pictures in interpolatedPicSetA shall be among the pictures that were output by applying a particular PPF ppfA when a particular picture currPicA in CroppedDecodedPictures was the current picture. The application of any other PPF that was used in the filtering process for one picture when currPicA was the current picture or the application of any PPF (including ppfA) that was used in the filtering process for one picture when any other picture currPicB in CroppedDecodedPictures was the current picture shall not output any picture between the inputPicA and inputPicB in output order.

NOTE – The intent of the constraints expressed in the above paragraph is to disallow generating PPF output pictures between any particular pair of consecutive input pictures more than once.

**8.28.1.2 Filtering process for one picture**

The filtering process specified in this subclause applies to each cropped decoded picture, referred to as the current picture, that is in CroppedDecodedPictures and for which one or more groups of PPFs are activated, only one of the PPF groups is chosen to be applied, and the number of PPFs (in the chosen PPF group) to be applied is greater than 0.

The filtering process for one picture using one PPF, as specified in subclause 8.28.1.3, is repeatedly invoked for each of the PPFs to be applied. When the number of PPFs to be applied is greater than 1, the following applies:

– If the special PPF cascading case applies for the chosen PPF group, the NNPF with nnpfc\_purpose equal to 4 is applied first, followed by the NNPF with multiple input pictures.

– Otherwise (the special PPF cascading case does not apply for the chosen PPF group), the PPFs are applied in the preferred order indicated by the SEI processing order SEI message associated with the chosen group of PPFs.

**8.28.1.3 Filtering process for one picture using one PPF**

The filtering process specified in this subclause applies when a particular PPF is applied when a particular picture is the current picture.

Before the PPF is applied, when the PPF is the first PPF to be applied, the list CandInputPicList is set to be identical to CroppedDecodedPictures.

When applying a PPF to the current picture, the input pictures for the PPF are selected from the list CandInputPicList, and the order of the pictures generated and output by the PPF are in output order.

When applying a PPF that is an NNPF to the current picture, the following applies:

– The filtered and/or interpolated pictures are generated by the NNPF by applying the NNPF process specified in the semantics of the NNPFC SEI message, in a patch-wise manner, to the current picture.

– The order of the pictures generated by the NNPF by applying the NNPF process being stored into the output tensor of the NNPF is in output order.

After the PPF is applied, the lists CandInputPicList and ListOutputPics are both updated, in the same manner, by 1) replacing each of those pictures in the list having a corresponding PPF output picture of the PPF with the corresponding PPF output picture, and 2) inserting those interpolated pictures, if any, into the list and placing them such that all pictures in the updated list are in output order.

**8.28.2 Neural-network post-filter characteristics SEI message**

**8.28.2.1 Neural-network post-filter characteristics SEI message syntax**

(no change)

**8.28.2.2 Neural-network post-filter characteristics SEI message semantics**

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

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

– The list CandInputPicList that contains a list of pictures in output order from which the input pictures for the NNPF are selected.

NOTE 1 – This list is updated by the PPF filtering process each time when a PPF is applied for the current picture unless the PPF is the last PPF that is applied for the current picture.

– Input picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.

– Luma sample array CroppedYPic[ idx ] and chroma sample arrays CroppedCbPic[ idx ] and CroppedCrPic[ idx ], when present, of the input pictures with index idx in the range of 0 to numInputPics − 1, inclusive, that are used as input for the NNPF.

– Bit depth BitDepthY for the luma sample array of the input pictures.

– Bit depth BitDepthC for the chroma sample arrays, if any, of the input pictures.

– Chroma format indicator ChromaFormatIdc, as described in subclause 7.3, of the input pictures.

– When nnpfc\_auxiliary\_inp\_idc is equal to 1, a filtering strength control value array StrengthControlVal[ idx ] that shall contain real numbers in the range of 0 to 1, inclusive, of the input pictures with index idx in the range of 0 to numInputPics − 1, inclusive.

Input picture with index 0 is the picture in CandInputPicList corresponding to the picture for which the NNPF defined by this NNPFC SEI message is activated by an NNPFA SEI message. Input picture with index i in the range of 1 to numInputPics − 1, inclusive, precedes the input picture with index i − 1 in output order.

NOTE 2 – The picture in CandInputPicList corresponding to a picture is either the cropped decoded output picture of that picture or a filtered version of the cropped decoded output picture that was an output picture of a previously applied PPF.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

…

**nnpfc\_absent\_input\_pic\_zero\_flag** equal to 1 indicates that the NNPF expects an input picture corresponding to a picture that is not present in the bitstream to be represented by sample arrays with sample values equal to 0. nnpfc\_absent\_input\_pic\_zero\_flag equal to 0 indicates that the NNPF expects an input picture inputPicA corresponding to a picture that is not present in the bitstream to be represented by the input picture inputPicB that is the closest to inputPicA in output order and the picture corresponding to inputPicB is present in the bitstream.

…

**nnpfa\_no\_prev\_clvs\_flag** equal to 1 specifies that the pictures corresponding to the input pictures for the NNPF do not originate from a previous CLVS. nnpfa\_no\_prev\_clvs\_flag equal to 0 specifies that the pictures corresponding to the input pictures for the NNPF may or may not originate from a previous CLVS.

NOTE 4 – The value of nnpfa\_no\_prev\_clvs\_flag can be changed from 0 to 1, when the current CLVS is spliced from another bitstream next to the previous CLVS and this NNPFA SEI message would cause one or more input pictures to be selected that have corresponding pictures from one or more previous CLVSs and therefore is likely to impact the output of the target NNPF negatively.

**nnpfa\_no\_foll\_clvs\_flag** equal to 1 specifies that when this NNPFA SEI message persists for the last PU of a CLVS in output order, the NNPFA SEI message is treated like it persisted for the last PU, in output order, of the current layer within the bitstream. When this NNPFA SEI message does not persist for the last PU, in output order, of a CLVS in output order or nnpfa\_no\_foll\_clvs\_flag is equal to 0, the value of nnpfa\_no\_foll\_clvs\_flag causes no specific impact.

NOTE 5 – The value of nnpfa\_no\_foll\_clvs\_flag can be changed from 0 to 1 for a picture-rate-upsampling NNPF, when the following CLVS is spliced from a different bitstream next to the current CLVS. Consequently, the NNPF process interpolates pictures up to the end of the current CLVS using input pictures corresponding to pictures originating from the current CLVS only.

…

**8.28.3 Neural-network post-filter activation SEI message**

(no change)

**D.12.11 Use of the post-processing filter SEI messages, including the neural network post-filter characteristics SEI message and the neural network post-filter activation SEI message**

A post-processing filter (PPF) may be indicated by an SEI message for which the payloadType value is in SeiProcessingOrderSeiList, which is specified in the semantics of the SEI processing order SEI message.

Let currPic be the cropped decoded output picture for which the post-processing filter (PPF), e.g., neural-network post-processing filter (NNPF) defined by the neural-network post-filter characteristics (NNPFC) SEI message, is activated, e.g., by a neural-network post-filter activation (NNPFA) SEI message, and currLayerId be the nuh\_layer\_id value of currPic.

The list candInputPicList contains a list of pictures in output order from which the input pictures for the PPF are selected.

NOTE 1 – This list is updated by the PPF filtering process each time when a PPF is applied for the current picture unless the PPF is the last PPF that is applied for the current picture.

When the PPF is not an NNPF, the PPF is considered to have only one input picture that is the picture in candInputPicList corresponding to the current picture.

NOTE 2 – The picture in candInputPicList corresponding to a cropped decoded output picture is either the cropped decoded output picture itself or a filtered version of the cropped decoded output picture that was an output picture of a previously applied PPF.

If the PPF is the first PPF that is applied for the current picture, the following applies:

– CroppedWidth is set equal to the value of pps\_pic\_width\_in\_luma\_samples − ‌SubWidthC \* ‌ ( pps\_conf\_win\_left\_offset + pps\_conf\_win\_right\_offset ) for currPic.

– CroppedHeight is set equal to the value of pps\_pic\_height\_in\_luma\_samples − ‌SubHeightC \* ‌ ( pps\_conf\_win\_top\_offset + pps\_conf\_win\_bottom\_offset ) for currPic.

– BitDepthY and BitDepthC are both set equal to BitDepth.

– ChromaFormatIdc is set equal to sps\_chroma\_format\_idc.

Otherwise (the PPF is not the first PPF that is applied for the current picture), the following applies:

– CroppedWidth is set equal to the picture width in units of luma samples of the picture in candInputPicList corresponding to the current picture.

– CroppedHeight is set equal to the picture height in units of luma samples of the picture in candInputPicList corresponding to the current picture.

– BitDepthY is set equal to the bit depth BitDepthY for the luma sample array of the picture in candInputPicList corresponding to the current picture.

– BitDepthC is set equal to the bit depth BitDepthC for the chroma sample arrays, if any, of the picture in candInputPicList corresponding to the current picture.

– ChromaFormatIdc is set equal to the chroma format indicator ChromaFormatIdc of the the picture in candInputPicList corresponding to the current picture.

The remainder of this subclause applies when the PPF is an NNPF.

It is a requirement of bitstream conformance that when a picture unit contains an NNPFA SEI message, the value of ph\_pic\_output\_flag in the picture header contained in that picture unit shall be equal to 1.

NOTE 3 – Since when the NNPF is the first PPF that is applied for currPic only cropped decoded output pictures are used as input pictures of the NNPF, the value of ph\_pic\_output\_flag in the picture header of the coded picture corresponding to each input picture of the NNPF is equal to 1.

The variable pictureRateUpsamplingFlag is set equal to ( ( nnpfc\_purpose & 0x08 ) > 0 ) ? 1 : 0.

The variable numInputPics is set equal to nnpfc\_num\_input\_pics\_minus1 + 1.

The variable numInferences is derived as follows:

– If all of the following conditions are true, the variable numPostRoll is set equal to the value of i such that nnpfc\_interpolated\_pics[ i ] is greater than 0 and the variable numInferences is set equal to 1 + numPostRoll:

– nnpfc\_purpose is equal to 8 (i.e., the only purpose for the NNPF is picture rate upsampling).

– nnpfa\_persistence\_flag is equal to 1.

– nnpfc\_interpolated\_pics[ i ] is greater than 0 only for a single value of i that is greater than 0.

– Either of the following conditions is true:

– currPic is the last picture of the bitstream in output order that has nuh\_layer\_id equal to currLayerId.

– currPic is the last picture in the CLVS in output order and nnpfa\_no\_foll\_clvs\_flag is equal to 1.

– Otherwise, if all of the following conditions are true, the variable numPostRoll is set equal to InpIdx[ i ] for the value of i such that nnpfa\_output\_flag[ i ] is equal to 1, and the variable numInferences is set equal to 1 + numPostRoll:

– pictureRateUpsamplingFlag is equal to 0.

– numInputPics is greater than 1.

– nnpfa\_persistence\_flag is equal to 1.

– nnpfa\_output\_flag[ idx ] is equal to 1 for a single value of idx in the range of 0 to NumInpPicsInOutputTensor − 1, inclusive, and for that single value of idx, InpIdx[ idx ] is greater than 0.

– Either of the following conditions is true:

– currPic is the last picture of the bitstream in output order that has nuh\_layer\_id equal to currLayerId.

– currPic is the last picture in the CLVS in output order and nnpfa\_no\_foll\_clvs\_flag is equal to 1.

– Otherwise, the variable numInferences is set equal to 1.

For each value of j in the range of 0 to numInferences − 1, inclusive, the following applies for the derivation of the input pictures for the NNPF such that each input picture is a picture in candInputPicList:

– The arrays inputPic[ i ] and inputPresentFlag[ i ] for i in the range of 0 to numInputPics − 1, inclusive, representing all the input pictures and the presence of input pictures, respectively, are specified as follows:

– When j is greater than 0, for each value of k in the range of 0 to j − 1, inclusive, inputPic[ k ] is set to be the picture in candInputPicList corresponding to currPic and inputPresentFlag[ k ] is set equal to 0.

– The j-th input picture, inputPic[ j ], is set to be the picture in candInputPicList corresponding to currPic and inputPresentFlag[ j ] is set equal to 1.

– When numInputPics is greater than 1, the following applies for each value of i in the range of j + 1 to numInputPics − 1, inclusive, in increasing order of i:

– If both of the following conditions are true, inputPic[ i ] is set to be the picture in candInputPicList corresponding to prevPic and inputPresentFlag[ i ] is set equal to 1:

– Either of the following conditions is true:

– pictureRateUpsamplingFlag is equal to 1 and currPic is associated with a frame packing arrangement SEI message with frame\_packing\_arrangement\_type equal to 5 and a particular value of fp\_current\_frame\_is\_frame0\_flag, and there is a cropped decoded output picture prevPic that is the last picture in output order among all cropped decoded output pictures that have nuh\_layer\_id equal to currLayerId, precede inputPic[ i − 1 ] in output order, and are associated with a frame packing arrangement SEI message with frame\_packing\_arrangement\_type equal to 5 and the same value of fp\_current\_frame\_is\_frame0\_flag.

– pictureRateUpsamplingFlag is equal to 0 or currPic is not associated with a frame packing arrangement SEI message with frame\_packing\_arrangement\_type equal to 5, and there is a cropped decoded output picture prevPic that is the last picture in output order among all cropped decoded output pictures that have nuh\_layer\_id equal to currLayerId and precede inputPic[ i − 1 ] in output order.

– nnpfa\_no\_prev\_clvs\_flag is equal to 0 or the coded picture corresponding to prevPic and the current picture are present in the same CLVS.

– Otherwise, the following applies:

– inputPic[ i ] is set to be the same picture as inputPic[ i − 1 ] and inputPresentFlag[ i ] is set equal to 0.

– It is a requirement of bitstream conformance that, when pictureRateUpsamplingFlag is equal to 1, nnpfc\_interpolated\_pics[ i − 1 ] shall be equal to 0.

– It is a requirement of bitstream conformance that when inputPresentFlag[ i ] is equal to 0 and nnpfc\_input\_pic\_output\_flag[ i ] is equal to 1, the value of nnpfa\_output\_flag[ idx ] shall be equal to 0 for the value of idx such that InpIdx[ idx ] is equal to i.

– For purposes of interpretation of the NNPFC SEI message, the following variables are specified:

– The luma sample arrays CroppedYPic[ i ] and the chroma sample arrays CroppedCbPic[ i ] and CroppedCrPic[ i ], when present, are derived as follows for each value of i in the range of 0 to numInputPics − 1, inclusive:

– The variable sourcePic is derived as follows:

– If inputPresentFlag[ i ] is equal to 1 or nnpfc\_absent\_input\_pic\_zero\_flag is equal to 0, sourcePic is set to be inputPic[ i ].

– Otherwise (inputPresentFlag[ i ] is equal to 0 and nnpfc\_absent\_input\_pic\_zero\_flag is equal to 1), sourcePic is set to be a picture with a luma sample array of CroppedWidth × CroppedHeight samples equal to 0 and Cb and Cr sample arrays of ( CroppedWidth / SubWidthC ) × ( CroppedHeight / SubHeightC ) samples equal to 0.

– The luma sample array CroppedYPic[ i ] and the chroma sample arrays CroppedCbPic[ i ] and CroppedCrPic[ i ], when present, are set to be the 2-dimensional arrays of decoded sample values of the Y, Cb and Cr components, respectively, of sourcePic.

– The array StrengthControlVal[ i ] for all values of i in the range of 0 to numInputPics − 1, inclusive, specifying the filtering strength control value for the input pictures for the NNPF, is derived as follows:

– StrengthControlVal[ i ] is set equal to the value of ( firstSliceQpY + QpBdOffset ) ÷ ( 63 + QpBdOffset ), where firstSliceQpY is equal to SliceQpY of the first slice of the cropped decoded output picture corresponding to inputPic[ i ].

There shall not be more than two NNPFC SEI messages present in a picture unit with the same value of nnpfc\_id. When there are two NNPFC SEI messages present in a picture unit with the same value of nnpfc\_id, these SEI messages shall have different content. When two NNPFC SEI messages with the same nnpfc\_id and different content are present in the same picture unit, both of these NNPFC SEI messages shall be in the same SEI NAL unit.

*Addition of post-processing filter gain SEI message*

* + 1. **Post-processing filter gain SEI message**

[Ed. (YK): Note that the JVET agreement was "to include the approach of separate SEI (applicable for any PF) to TuC". Therefore, this SEI message should be renamed and changed accordingly such that it applies to both NN post-processing filters and non-NN post-processing filters. And the subclause should not be a subclause of 8.28. For example, it could be subclause 8.35. (MH): My mistake, sorry for the oversight, and thanks for catching the deviation from the documented adoption. I have edited the text to make the SEI message generic to any type of post-filters. I didn't move the text yet to out from 8.28 to make it easier to review my edits to generalize this SEI message to apply to any type of post-filters. Also, we may consider generalizing 8.28 to be an umbrella for all post-filter related SEI messages and subclauses, now that 8.28.1 is also generic to any type of post-filters.]

* + - 1. **Post-propcessing filter gain SEI message syntax**

|  |  |
| --- | --- |
| post\_processing\_filter\_gain( payloadSize ) { | **Descriptor** |
| **pf\_gain\_id** | ue(v) |
| **pf\_gain\_num\_entries\_minus1** | ue(v) |
| for( i = 0; i <= pf\_gain\_num\_entries\_minus1; i++ ) |  |
| **pf\_gain\_metric\_idc**[ i ] | ue(v) |
| for( i = 0; i <= pf\_gain\_num\_entries\_minus1; i++ ) { |  |
| **pf\_gain\_per\_component\_flag**[ i ] | u(1) |
| if( pf\_gain\_per\_component\_flag[ i ] ) { |  |
| **pf\_gain\_y\_flag**[ i ] | u(1) |
| **pf\_gain\_cb\_flag**[ i ] | u(1) |
| if( pf\_gain\_y\_flag[ i ] + pf\_gain\_cb\_flag[ i ] > 0 ) |  |
| **pf\_gain\_cr\_flag**[ i ] | u(1) |
| } |  |
| **pf\_gain\_word\_cnt\_minus1**[ i ] | ue(v) |
| for( j = 0; j < pfGainNumValuesPerEntry[ i ]; j++ ) |  |
| **pf\_gain\_value**[ i ][ j ] | u(v) |
| } |  |
| } |  |

* + - 1. **Post-processing filter gain SEI message semantics**

The post-processing filter gain SEI message indicates the quality improvement that is achieved by applying an indicated post-processing filter (PPF) or an indicated PPF group.

**pf\_gain\_id** indicates the PPF or PPF group which this SEI message is associated with. When pf\_gain\_id is equal to any nnpfc\_id or nnpfgc\_id value present in the current CLVS, this SEI message indicates one or more values characterizing the gain obtained by the NNPF identified by the nnpfc\_id value or the NNPFG identified by the nnpfgc\_id value, respectively. The variables specified for the NNPF with nnpfc\_id equal to nnpf\_gain\_id or the NNPFG with nnpfgc\_id equal to nnpf\_gain\_id apply below.

**pf\_gain\_num\_entries\_minus1** plus 1 specifies the number of entries in this syntax structure.

**pf\_gain\_metric\_idc**[ i ] identifies the metric used in the gain values provided for the i-th entry in this syntax structure. pf\_gain\_metric\_idc[ i ] equal to 0 indicates a peak signal-to-noise ratio based metric specified below. The value of pf\_gain\_metric\_idc[ i ] shall be in the range of 0 to 65 535, inclusive. The value of pf\_gain\_metric\_idc[ i ] shall be equal to 0 in bitstreams conforming to this version of this Specification. Values of 1 to 65 535, inclusive, for pf\_gain\_metric\_idc[ i ] are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. When the value of pf\_gain\_metric\_idc[ i ] is in the range of 1 to 65 535, inclusive, decoders conforming to this version of this Specification shall ignore all the syntax elements for the i-th entry in this syntax structure.

**pf\_gain\_per\_component\_flag**[ i ] equal to 1 specifies that gain values are provided on colour component basis. pf\_gain\_per\_component\_flag[ i ] equal to 0 specifies that the i-th entry in this syntax structure has one and only one gain value, which may be derived collectively from several colour components.

**pf\_gain\_y\_flag**[ i ], when present and equal to 1, specifies that the 0-th gain value for the i-th entry in this syntax structure is derived from the luma component. pf\_gain\_y\_flag[ i ], when present and equal to 0, specifies that the i-th entry in this syntax structure does not have a gain value derived from the luma component.

**pf\_gain\_cb\_flag**[ i ], when present and equal to 1, specifies that the ( pf\_gain\_y\_flag[ i ] )-th gain value for the i-th entry in this syntax structure is derived from the Cb component. pf\_gain\_cb\_flag[ i ], when present and equal to 0, specifies that the i-th entry in this syntax structure does not have a gain value derived from the Cb component.

**pf\_gain\_cr\_flag**[ i ], when present and equal to 1 or when inferred to be equal to 1, specifies that the ( pf\_gain\_y\_flag[ i ] + pf\_gain\_cb\_flag[ i ] )-th gain value for the i-th entry in this syntax structure is derived from the Cr component. pf\_gain\_cr\_flag[ i ], when present and equal to 0, specifies that the i-th entry in this syntax structure does not have a gain value derived from the Cr component. When pf\_gain\_y\_flag[ i ] and pf\_gain\_cb\_flag[ i ] are present and both equal to 0, pf\_gain\_cr\_flag[ i ] is inferred to be equal to 1.

The value of pfGainNumValuesPerEntry[ i ], which specifies the count of gain values for the i-th entry in this syntax structure, is derived as follows:

pfGainNumValuesPerEntry[ i ] = pf\_gain\_per\_component\_flag[ i ] ?  
 pf\_gain\_y\_flag[ i ] + pf\_gain\_cb\_flag[ i ] + pf\_gain\_cr\_flag[ i ] : 1 (xx)

**pf\_gain\_word\_cnt\_minus1**[ i ] plus 1 specifies the length of each gain value for the i-th entry in units of 16 bits.

**pf\_gain\_value**[ i ][ j ] specifies the j-th gain value for the i-th entry in this syntax structure. The semantics and the representation format of the gain value are determined by the value of pf\_gain\_metric\_idc[ i ]. The length of pf\_gain\_value[ i ][ j ] is 16 \* ( pf\_gain\_word\_cnt\_minus1[ i ] + 1 ) bits.

When pf\_gain\_metric\_idc[ i ] is equal to 0, it is a requirement of bitstream conformance that all the following constraints apply:

– pf\_gain\_per\_component\_flag[ i ] is equal to 1.

– pf\_gain\_word\_cnt\_minus1[ i ] is equal to 0.

– nnpfc\_out\_format\_idc is equal to 1.

– When pf\_gain\_y\_flag[ i ] is equal to 1, all of the following constraints apply:

– nnpfcOutputPicWidth is equal to CroppedWidth.

– nnpfcOutputPicHeight is equal to CroppedHeight.

– When pf\_gain\_cb\_flag[ i ] is equal to 1 or pf\_gain\_cr\_flag[ i ] is equal to 1, all of the following constraints apply:

– nnpfcOutputPicWidth / outSubWidthC is equal to CroppedWidth / SubWidthC.

– nnpfcOutputPicHeight / outSubHeightC is equal to CroppedHeight / SubHeightC.

The function Log10( x ) is defined as the base-10 logarithm of x.

Given sample arrays origSampleArray and testSampleArray, the function Psnr( origSampleArray, testSampleArray, width, height, origBitDepth, testBitDepth ) returns the value of psnrVal, which is the average peak signal-to-noise ratio of testSampleArray in comparison to origSampleArray, derived as follows:

maxBitDepth = (origBitDepth > testBitDepth) ? origBitDepth : testBitDepth  
origShift = maxBitDepth − origBitDepth  
testShift = maxBitDepth − testBitDepth  
maxVal = 255 << ( maxBitDepth − 8 )  
maxSquaredVal = maxVal \* maxVal  
sumSquaredError = 0  
for( y = 0; y < height; y++ )  
 for( x = 0; x < width; x++ ) {  
 sampleError =   
 ( testSampleArray[ x ][ y ] << testShift ) − ( origSampleArray[ x ][ y ] << origShift ) (xx)  
 sumSquaredError += sampleError \* sampleError  
 }  
if( sumSquaredError = = 0 )  
 psnrVal = 999.99  
else  
 psnrVal = 10 \* Log10( maxSquaredVal ÷ ( sumSquaredError ÷ width ÷ height ) )

When pf\_gain\_metric\_idc[ i ] is equal to 0, the following applies

– Let OriginalPictures be the list of pictures given as input to encoding in output order and OrigBitDepth be the bit depth of the sample arrays of the pictures in OriginalPictures.

– CroppedDecodedPictures and ListOutputPics are derived as specified in subclause 8.28.1.1.

– It is a requirement of bitstream conformance that the number of pictures, assigned to variable numPics, in OriginalPictures, CroppedDecodedPictures and ListOutputPics is the same.

– avgPsnrDiff is derived as follows:

sumPsnrDiff = 0  
for( a = 0; a < numPics; a++ ) {  
 if( j = = 0 && pf\_gain\_y\_flag [ i ] = = 1 )  
 cIdx = 0  
 else if( ( j = = 0 && pf\_gain\_y\_flag[ i ] = = 0 && pf\_gain\_cb\_flag[ i ] = = 1 ) | |   
 ( j = = 1 && pf\_gain\_y\_flag[ i ] = = 1 && pf\_gain\_cb\_flag[ i ] = = 1 )  
 cIdx = 1  
 else  
 cIdx = 2  
 if( cIdx = = 0 ) { (xx)  
 arrayWidth = CroppedWidth  
 arrayHeight = CroppedHeight  
 } else {  
 arrayWidth = CroppedWidth / SubWidthC  
 arrayHeight = CroppedHeight / SubHeightC  
 }  
 if( cIdx = = 0 )  
 sumPsnrDiff += Psnr( OriginalPictures[ a ][ cIdx ], ListOutputPics[ a ][ cIdx ], width, height,   
 OrigBitDepth, outTensorBitDepthY ) − Psnr( OriginalPictures[ a ][ cIdx ],   
 CroppedDecodedPictures[ a ][ cIdx ], width, height, OrigBitDepth, BitDepthY )  
 else  
 sumPsnrDiff += Psnr( OriginalPictures[ a ][ cIdx ], ListOutputPics[ a ][ cIdx ], width, height,   
 OrigBitDepth, outTensorBitDepthC ) − Psnr( OriginalPictures[ a ][ cIdx ],   
 CroppedDecodedPictures[ a ][ cIdx ], width, height, OrigBitDepth, BitDepthC )  
}  
avgPsnrDiff = sumPsnrDiff ÷ numPics

– If avgPsnrDiff is greater than or equal to 256, pf\_gain\_value[ i ][ j ] is set equal to 65 535. Otherwise (avgPsnrDiff is less than 256), pf\_gain\_value[ i ][ j ] is set to a value such that pf\_gain\_value[ i ][ j ] >> 8 specifies the integer part of avgPsnrDiff and pf\_gain\_value[ i ][ j ] & 255 specifies the fractional part of avgPsnrDiff.

*Text to be added to a future edition or ammendment of Recommendation ITU-T H.266 | International Standard ISO/IEC 23090-3:*

**Use of the post-processing filter gain SEI message in VVC**

When there is a post-processing filter gain SEI message that has pf\_gain\_id equal to any nnpfc\_id or nnpfgc\_id value in the same CLVS, it shall be present in an SEI NAL unit that contains an NNPFC SEI message with nnpfc\_id equal to pf\_gain\_id or an NNPFGC SEI message with nnpfgc\_id equal to to pf\_gain\_id.

*Text to be added to a future edition or ammendment of Recommendation ITU-T H.265 | International Standard ISO/IEC 23008-2:*

**Use of the post-processing filter gain SEI message in HEVC**

When there is a post-processing filter gain SEI message that has pf\_gain\_id equal to any nnpfc\_id or nnpfgc\_id value in the same CLVS, it shall be present in an SEI NAL unit that contains an NNPFC SEI message with nnpfc\_id equal to pf\_gain\_id or an NNPFGC SEI message with nnpfgc\_id equal to pf\_gain\_id.

*Text to be added to a future edition or ammendment of Recommendation ITU-T H.264 | International Standard ISO/IEC 14496-10:*

**Use of the post-processing filter gain SEI message in AVC**

When there is a post-processing filter gain SEI message that has pf\_gain\_id equal to any nnpfc\_id or nnpfgc\_id value in the same CVS, it shall be present in an SEI NAL unit that contains an NNPFC SEI message with nnpfc\_id equal to pf\_gain\_id or an NNPFGC SEI message with nnpfgc\_id equal to pf\_gain\_id.

*Modify clause 8.30 as indicated with the highlighting:*

* 1. **Encoder optimization information SEI message**
     1. **Encoder optimization information SEI message syntax**

|  |  |
| --- | --- |
| encoder\_optimization\_info(payloadSize ) { | **Descriptor** |
| **eoi\_cancel\_flag** | u(1) |
| if( !eoi\_cancel\_flag ) { |  |
| **eoi\_persistence\_flag** | u(1) |
| **eoi\_for\_human\_viewing\_idc** | u(2) |
| **eoi\_for\_machine\_analysis\_idc** | u(2) |
| **eoi\_type** | u(16) |
| if( eoi\_persistence\_flag ) { |  |
| **eoi\_num\_sublayers** | u(3) |
| if( eoi\_num\_sublayers < 7 ) |  |
| **eoi\_min\_tid** | u(3) |
| } |  |
| if( EoiObjectBasedFlag ) |  |
| … |  |

* + 1. **Encoder optimization information SEI message semantics**

…

**eoi\_persistence\_flag** specifies the persistence of the optimization information provided in this SEI message. eoi\_persistence\_flag equal to 0 specifies that the optimization information applies for the current picture only. eoi\_persistence\_flag equal to 1 specifies that the optimization information applies for the pictures in the temporal sublayers indicated by eoi\_num\_sublayers and eoi\_min\_tid, when present, among the current picture and all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

…

**eoi\_num\_sublayers** equal to 7 specifies that this SEI message applies to all temporal sublayers. eoi\_num\_sublayers less than 7 specifies the number of temporal sublayers to which this SEI message applies. eoi\_num\_sublayers equal to 0 is reserved.

**eoi\_min\_tid** specifies the smallest temporal sublayer identifier value to which this SEI message applies. The value of eoi\_min\_tid shall be in the range of 0 to 6 − eoi\_num\_sublayers, inclusive. When eoi\_persistence\_flag is equal to 1 and eoi\_min\_tid is not present, eoi\_min\_tid is inferred to be equal to 0.

When eoi\_persistence\_flag is equal to 1, the variable eoiMaxSublayer, which is the greatest sublayer identifier value to which this SEI message applies, is derived as follows:

eoiMaxSublayer = eoi\_min\_tid + eoi\_num\_sublayers − 1

When eoi\_persistence\_flag is equal to 1, the semantics of eoi\_for\_human\_viewing\_flag and eoi\_for\_machine\_analysis\_flag apply to the sublayer representation of eoiMaxSubLayer and the semantics of eoi\_type, eoi\_object\_based\_idc (when present), eoi\_temporal\_resampling\_type\_flag (when present), and eoi\_num\_int\_pics (when present) apply to the sublayers with a sublayer identifier in the range of eoi\_min\_tid to eoiMaxSublayer, inclusive.

…

When multiple encoder optimization information SEI messages are present in a particular PU, the following constraints apply for these SEI messages:

– All of the SEI messages shall have the same value of eoi\_cancel\_flag.

– When eoi\_persistence\_flag is present in the SEI messages, all the SEI messages shall have the same value of eoi\_persistence\_flag.

– When any of the SEI messages has eoi\_num\_sublayers equal to 7, all the SEI messages shall have the same SEI payload content.

– When an SEI message among the SEI messages has eoi\_num\_sublayers less than 7, the following constraints apply:

– All the SEI messages with the same values of eoi\_num\_sublayers and eoi\_min\_tid, which are assigned to numSubLayersA and minTidA, respectively, shall have the same SEI payload content.

– There shall be no such SEI message among the SEI messages for which both of the following are true:

– eoi\_num\_sublayers is not equal to numSubLayersA or eoi\_min\_tid is not equal to minTidA.

– eoi\_min\_tid is greater than minTidA and less than minTidA + numSubLayersA.

*Add VSEI subclauses 8.34 to 8.38*

8.34 Image format metadata SEI messages

8.34.1 Exif metadata SEI message

8.34.1.1 Exif metadata SEI message syntax

|  |  |
| --- | --- |
| exif\_metadata( payloadSize ) { | Descriptor |
| **exif\_cancel\_flag** | u(1) |
| if( !exif\_cancel\_flag ) { |  |
| **exif\_persistence\_flag** | u(1) |
| **exif\_mode\_id** | u(8) |
| if( exif\_mode\_id = = 0 ) |  |
| for( i = 1; i < payloadSize; i++ ) |  |
| **exif\_data\_payload\_byte** | b(8) |
| else if( exif\_mode\_id = = 1 ) |  |
| **exif\_data\_uri** | st(v) |
| } |  |
| } |  |

8.34.1.2 Exif metadata SEI message semantics

The Exchangeable Image File (Exif) Format for digital still cameras includes a set of metadata that captures information regarding the digital photography process that was used to record an image. Such metadata includes: the acceleration vector of the camera at the time the image was captured, camera lens information, GPS data, the color space used, spectral sensitivity, maximum lens aperture, and more.

Exif metadata is specified by any of the Exchangeable image file (Exif) format for digital still cameras standards developed jointly by the Camera & Imaging Products Association (CIPA) and the Japan Electronics and Information Technology Industries Association (JEITA), for the associated video source pictures prior to encoding, e.g., for camera-captured content. To date, there are multple versions of Exif deployed, each with its own indicator to signal the version number in use.

[Ed. Note: Neither CIPA nor JEITA are listed in the organizations for which normative references may be made to documents published by CIPA or JEITA. The list of such organizations is available at: <https://www.itu.int/net4/ITU-T/lists/sdo.aspx>. To add either CIPA or JEITA to the list, the A.4 process needs to be executed for one or both organizations, and their status as A.5 organizations needs to be approved by SG16. There would probably also need to be a specific list of documents referenced rather than the concept of “any of the” standards produced by those organizations.]

The Exif metadata SEI message specifies an SEI message for which the payload of Exif metadata can be carried in the video bitstream.

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

**exif\_persistence\_flag** specifies the persistence of the Exif metadata SEI message for the current layer.

exif\_persistence\_flag equal to 0 specifies that the Exif metadata SEI message applies to the current decoded picture only.

exif\_persistence\_flag equal to 1 specifies that the Exif metadata SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an Exif metadata SEI message is output that follows the current picture in output order.

**exif\_mode\_id** specifies the mode by which the Exif metadata is obtained.

– If exif\_mode\_id is equal to 0, the Exif metadata is obtained directly from the payload of the SEI message.

– Otherwise if exif\_mode\_id is equal to 1, the Exif metadata is obtained from a URI with syntax and semantics as specified in IETF Internet Standard 66.

All other values of exif\_mode\_id are reserved.

**exif\_data\_payload\_byte** shall be a byte containing data having syntax and semantics as specified by any of the Exchangeable image file format for digital still cameras standards developed jointly by the Camera & Imaging Products Association (CIPA) and the Japan Electronics and Information Technology Industries Association (JEITA).

**exif\_data\_uri** shall contain a URI with syntax and semantics as specified in IETF Internet Standard 66 identifying the Exif metadata.

8.34.2 JFIF metadata SEI message

8.34.2.1 JFIF metadata SEI message syntax

|  |  |
| --- | --- |
| jfif\_metadata( payloadSize ) { | Descriptor |
| **jfif\_cancel\_flag** | u(1) |
| **jfif\_type\_id** | u(8) |
| if( !jfif\_cancel\_flag ) { |  |
| **jfif\_persistence\_flag** | u(1) |
| if( jfif\_type\_id = = 0 ) |  |
| for( i = 1; i < payloadSize; i++ ) |  |
| **jfif\_data\_payload\_byte** | b(8) |
| else if( jfif\_type\_id = = 1 ) |  |
| for( i = 1; i < payloadSize; i++ ) |  |
| **jfif\_extension\_payload\_byte** | b(8) |
| else if( jfif\_type\_id = = 2 ) |  |
| for( i = 1; i < payloadSize; i++ ) |  |
| **jfif\_header\_payload\_byte** | b(8) |
| } |  |
| } |  |

*[Ed. (GJS): Are some of those grid lines thicker than others?]*

8.34.2.2 JFIF metadata SEI message semantics

The JFIF metadata SEI message contains JFIF metadata, as specified by ITU-T Recommendation T.871 | ISO/IEC International Standard 10918-5.

JFIF metadata and its semantics are specified by ITU-T Recommendation T.871 | ISO/IEC International Standard 10918-5 (hereafter, the JFIF standard). Of particular importance is the distinction by JFIF between an APP0 marker that contains information describing the organization of the image data, and an APP0 marker that contains “extension” information. [Ed. Note: Needs formal reference (in clause 2). Does that document define what an “APP0 marker” is?]

A single APP0 marker that carries the information to describe the organization of the image is signalled by the zero-terminated string “JFIF” (‘0x4A46494600’) within that APP0 marker. One or more subsequent APP0 markers may follow the first APP0 marker in which the subsequent APP0 marker(s) carries “extension” information. In particular, the JFIF extension mechanism can be used to carry image thumbnails, although there is no specification by the JFIF standard for the precise type of data carried in JFIF extensions.

Approaches for carriage of JFIF within an SEI message include:

– The JFIF payload is carried entirely as a single “container” within an SEI message. In this approach, the SEI payload includes the payloads from the first and all, if any, subsequent JFIF APP0 markers for an image.

– The JFIF payload from only extension APP0 markers, i.e., all JFIF APP0 markers that appear subsequent to the first APP0 marker, is carried as a single container. This approach facilitates the carriage of thumbnail images without the overhead of the carriage of the first JFIF APP0 marker.

– The JFIF payload from only the first JFIF APP0 marker is carried as a single container.

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

**jfif\_persistence\_flag** specifies the persistence of the JFIF metadata SEI message for the current layer.

jfif\_persistence\_flag equal to 0 specifies that the JFIF metadata SEI message applies to the current decoded picture only.

jfif\_persistence\_flag equal to 1 specifies that the JFIF metadata SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an JFIF metadata SEI message is output that follows the current picture in output order.

**jfif\_type\_id** specifies the type of JFIF metadata.

– If jfif\_type\_id is equal to 0, the remaining SEI payload bytes shall contain JFIF metadata from one or more concatenated APP0 marker segments without the APP0 marker segment identifiers themselves, i.e., an APP0 marker segment with bytes containing 0x4A46494600, i.e, the zero-terminated string "JFIF", according to Rec. ITU-T T.50 or ISO 646 coding, and one or more APP0 marker segments containing 0x4A46585800, i.e, the zero-terminated string "JFXX", according to Rec. ITU-T T.50 or ISO 646 coding.

– Otherwise if jfif\_type\_id is equal to 1, the remaining SEI payload bytes shall contain bytes from one or more JFIF extension markers, i.e., one or more APP0 markers that include 0x4A46585800, i.e, the zero-terminated string "JFXX", according to Rec. ITU-T T.50 or ISO 646 coding.

– Otherwise if jfif\_type\_id is equal to 2, the remaining SEI payload bytes shall contain bytes obtained from the first APP0 marker following the 0xFFD8 Start of Image marker within the image. The first APP0 marker includes 0x4A46494600, i.e, the zero-terminated string "JFIF", according to Rec. ITU-T T.50 or ISO 646 coding.

All other values of jfif\_type\_id are reserved.

**jfif\_data\_payload\_byte** shall be a byte containing data from one or more concatenated APP0 JPEG marker segments without the bytes 0xFFE0 from each APP0 marker segment identifier, each such marker segment having syntax and semantics specified by ITU-T Recommendation T.871 | ISO/IEC International Standard 10918-5.

**jfif\_extension\_payload\_byte** shall be a byte containing data from one or more concatenated APP0 JPEG marker segments, without the bytes 0xFFE0 from each APP0 marker segment identifier, each such marker segment having syntax and semantics of a JFIF extension APP0 marker segment specified by ITU-T Recommendation T.871 | ISO/IEC International Standard 10918-5.

**jfif\_header\_payload\_byte** shall be a byte containing data from the first APP0 JPEG marker segment following the 0xFFD8 Start of Image marker segment, without the bytes 0xFFE0 from the APP0 marker segment identifier, such marker segment having syntax and semantics of the first APP0 marker segment immediately following the 0xFFD8 Start of Image marker segment specified by ITU-T Recommendation T.871 | ISO/IEC International Standard 10918-5.

8.34.3 XMP metadata SEI message

8.34.3.1 XMP metadata SEI message syntax

|  |  |
| --- | --- |
| xmp\_metadata( payloadSize ) { | Descriptor |
| **xmp\_cancel\_flag** | u(1) |
| if( !xmp\_cancel\_flag ) { |  |
| **xmp\_persistence\_flag** | u(1) |
| for( i = 1; i < payloadSize; i++ ) |  |
| **xmp\_data\_payload\_byte** | b(8) |
| } |  |
| } |  |

8.34.3.1 XMP metadata SEI message semantics

The XMP metadata SEI message specifies an SEI message for XMP metadata.

Extensible Metadata Platform (XMP) metadata is specified in ISO 16684-1: Graphic Technology – Extensible metadata platform (XMP) specification. [Ed. Note (GJS): Formal reference needed in clause 2.] XMP is widely deployed by digital cameras and digital image editing packages to record provenance and editing history with an image. The most common set of metadata included in XMP refers to a vocabulary defined by the Dublin Core Metadata Initiative (DCMI)[[1]](#footnote-1), e.g., to store information such as digital rights ownership and names of software packages used to modify the image.

The XMP metadata SEI message indicates the XMP metadata, as specified by [1] in XML or [3] in JSON. [Ed. Note: Fix or remove unresolved references and undefined abbreviations.]

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

**xmp\_persistence\_flag** specifies the persistence of the XMP metadata SEI message for the current layer.

xmp\_persistence\_flag equal to 0 specifies that the XMP metadata SEI message applies to the current decoded picture only.

xmp\_persistence\_flag equal to 1 specifies that the XMP metadata SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an XMP metadata SEI message is output that follows the current picture in output order.

**xmp\_data\_payload\_byte** shall be a byte containing data having syntax and semantics as specified by ISO 16684-1: Graphic Technology – Extensible metadata platform (XMP) specification – Part 1 Data model, serialization and core properties or ISO 16684-3: Graphic Technology – Extensible metadata platform (XMP) specification – Part 3 JSON-LD serialization of XMP.

**8.35 Multiplane image information SEI message**

8.35.1 Multiplane image information SEI message syntax

|  |  |
| --- | --- |
| multiplane\_image\_information( payloadSize ) { | **Descriptor** |
| **mpii\_num\_layers\_minus1** | ue(v) |
| **mpii\_layer\_depth\_equal\_distance\_flag** | u(1) |
| if( mpii\_layer\_depth\_equal\_distance\_flag ) { |  |
| depth\_rep\_info\_element( ZNearSign, ZNearExp, ZNearMantissa, ZNearManLen ) |  |
| depth\_rep\_info\_element( ZFarSign, ZFarExp, ZFarMantissa, ZFarManLen ) |  |
| } else |  |
| for( i = 0; i <= mpii\_num\_layer\_minus1; i++ ) |  |
| depth\_rep\_info\_element( ZSign[ i ], ZExp[ i ], ZMantissa[ i ], ZManLen[ i ] ) |  |
| **mpii\_texture\_opacity\_interleave\_flag** | u(1) |
| if( mpii\_texture\_opacity\_interleave\_flag = = 0 ) |  |
| **mpii\_texture\_opacity\_arrangement\_flag** /\* 0: Top-and-Bottom, 1: Side-by-Side \*/ | u(1) |
| **mpii\_picture\_num\_layers\_in\_height\_minus1** | ue(v) |
| } |  |

8.35.2 Muliplane image information SEI message semantics

The multiplane image information (MPII) SEI message specifies multiplane image (MPI) scene representation information that may be used for view synthesis.

When an MPII SEI message is present in any AU of a CLVS, an MPII SEI message shall be present in the first AU of the CLVS and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an MPII SEI message is output that follows the current picture in output order.

NOTE 1 – This SEI message can work together with the multiview acquisition information SEI message for view synthesis. The multiview acquisition information SEI message specifies the intrinsic and extrinsic parameters for current camera view. When multiple views are available, the reconstructed novel views can be rendered from multiplane images of nearby views.

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 CroppedWidth and CroppedHeight, respectively.

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

– A cropped decoded picture array decPicCurr0[ cIdx ][ x ][ y ], with cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, x = 0..( cIdx = = 0 ) ? CroppedWidth : CroppedWidth / SubWidthC − 1, y = 0..( cIdx = = 0 ) ? CroppedHeight : CroppedHeight / SubHeightC − 1.

– In output order a temporally following cropped decoded picture array decPicCurr1[ cIdx ][ x ][ y ], with cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, x = 0..( cIdx = = 0 ) ? CroppedWidth : CroppedWidth / SubWidthC − 1, y = 0..( cIdx = = 0 ) ? CroppedHeight : CroppedHeight / SubHeightC − 1.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

**mpii\_num\_layers\_minus1** plus 1 specifies the number of texture and opacity layers for the MPI representation.

**mpii\_layer\_depth\_equal\_distance\_flag** equal to 1 indicates that equal distances are used to generate MPI layers and depth parameters for each layer. In this case, Z[ i ] can be derived using the nearest depth value ZNear and the farthest depth value ZFar.

The depth value for i-th MPI layer, Z[ i ], is derived as follows:

Z[ i ] = i \* ( ZFar − Znear ) ÷ (mpi\_num\_layers\_minus1 ) + ZNear (xx)

mpii\_layer\_depth\_equal\_distance\_flag equal to 0 indicates that the depth information for each layer follows next in the SEI message.

The variables in the x column of Table xx are derived from the respective variables in the s, e, n and v columns of Table xx as follows:

– If the value of e is in the range of 0 to 127, exclusive, x is set equal to ( − 1)s \* 2( e − 31 ) \* ( 1 + n ÷ 2v ).

– Otherwise (e is equal to 0), x is set equal to ( − 1)s \* 2−( 30 + v ) \* n.

NOTE 2 – The above specification is similar to that found in IEC 60559:1989.

**Table xx – Association between depth parameter variables and syntax elements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **s** | **e** | **n** | **v** |
| ZNear | ZNearSign | ZNearExp | ZNearMantissa | ZNearManLen |
| ZFar | ZFarSign | ZFarExp | ZFarMantissa | ZFarManLen |
| Z[ i ] | ZSign[ i ] | ZExp | ZMantissa | ZManLen |

NOTE 3 – In some applications, disparity is used instead of depth (the disparity value D and depth value Z relationship is D = 1 ÷ Z). Correspoding to Equation (xx), the disparity value for the i-th MPI layer is D[ i ] = i \* ( DFar − Dnear ) ÷ (mpi\_num\_layers\_minus1 ) + DNear.

**mpii\_texture\_opacity\_interleave\_flag** equal to 1 indicates decoded output pictures correspond to temporally interleaved texture and opacity constituent pictures in output order as illustrated in Figure XX. mpii\_texture\_opacity\_interleave\_flag equal to 0 indicates decoded output pictures correspond to spatially packed texture and opacity constituent pictures as illustrated in Figures XX and XX.

**mpii\_texture\_opacity\_arrangement\_flag** equal to 0 indicates decoded output pictures represent texture and opacity constituent pictures in a top-bottom packing arrangement as illustrated in Figure XX. mpii\_texture\_opacity\_arrangement\_flagequal to 1 indicates decoded output pictures represent texture and opacity constituent pictures in a side-by-side packing arrangement as illustrated in Figure XX.

For each specified picture packing arrangement scheme, there are two constituent pictures that are referred to as picture 0 and picture 1. When mpii\_texture\_opacity\_interleave\_flag is equal to 0, the constituent picture associated with the upper-left sample of the decoded picture is considered to be constituent picture 0 and the other constituent picture is considered to be constituent picture 1. When mpii\_texture\_opacity\_interleave\_flag is equal to 1, the first decoded picture in the current CLVS is constituent picture 0 and the next decoded picture in output order is constituent picture 1 and the display time of the constituent picture 0 should be delayed to coincide with the display time of constituent picture 1. The two constituent pictures form the spatially packed texture and opacity picture of a MPI, with picture 0 being associated with the spatially packed texture picture and picture 1 being associated with the spatially packed opacity picture.

**mpii\_picture\_num\_layers\_in\_height\_minus1** plus 1 specifies the number of spatially packed layers in height for picture 0 and picture 1. The variable hLayers is set equal to mpii\_picture\_num\_layers\_in\_height\_minus1 + 1 and the variable wLayers is set equal to (mpii\_num\_layers\_minus1 + 1) / hLayers.

Let variable fWidth and fHeight specify the width and height of picture 0 and picture 1, respectively, and are derived as follows:

– If mpii\_texture\_opacity\_interleave\_flag is equal to 1, the following applies:

fWidth = CroppedWidth

fHeight = CroppedHeight

– Otherwise (mpii\_texture\_opacity\_interleave\_flag is equal to 0)

– If mpii\_texture\_opacity\_arrangement\_flag is equal to 0, the following applies:

fWidth = CroppedWidth , fHeight = CroppedHeight / 2

– Otherwise (mpii\_texture\_opacity\_arrangement\_flag is equal to 1), the following applies:

fWidth = CroppedWidth / 2 , fHeight = CroppedHeight

Let variable cWidth = fWidth / subWidthC and variable cHeight = fHeight / subHeightC.

Let array picture0[ cIdx ][ x ][ y ] specify samples in picture 0 and array picture1[ cIdx ][ x ][ y ] specify samples in picture 1, with cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, x = 0..( cIdx = = 0 ) ? fWidth: cWidth − 1, y = 0..( cIdx = = 0 ) ? fHeight : cHeight − 1 and are derived as follows:

– If mpii\_texture\_opacity\_interleave\_flag is equal to 1, the following applies:

picture0[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x ][ y ]

picture1[ cIdx ][ x ][ y ] = decPicCurr1[ cIdx ][ x ][ y ]

– Otherwise (mpii\_texture\_opacity\_interleave\_flag is equal to 0)

– Let variable cW = ( cIdx = = 0 )? fWidth : cWidth

– Let variable cH = ( cIdx = = 0 )? fHeight : cHeight

– If mpii\_texture\_opacity\_arrangement\_flag is equal to 0, the following applies:

picture0[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x ][ y ]

picture1[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x ][ y + cH ]

– Otherwise (mpii\_texture\_opacity\_arrangement\_flag is equal to 1), the following applies:

picture0[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x ][ y ]

picture1[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x + cW ][ y ]

Let variable layerWidth and layerHeight specify the width and height for decoded MPI layer, respectively. The variables are derived as follows:

layerWidth = fWidth / wLayers

layerHeight = fHeight / hLayers

The reconstruction of MPI process is described as follows:

The outputs of this process are:

– a 4D MPI texture layer array recTextureLayer[ i ][ cIdx ][ w ][ h ] with i = 0..mpii\_num\_layers\_minus1, cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, w = 0..( cIdx = = 0 ) ? layerWidth : layerWidth / SubWidthC − 1, and h = 0..( cIdx = = 0 ) ? layerHeight : layerHeight / SubHeightC − 1.

– a 3D MPI opacity layer array recOpacityLayer[ i ][ w ][ h ] with i = 0..mpii\_num\_layers\_minus1, x = 0..layerWidth − 1, and y = 0..layerHeight − 1.

The array recTextureLayer and array recOpacityLayer are derived as follows:

for( i = 0; i  <=  mpii\_num\_layers\_minus1; i++ ) {  
 k = i % wLayers  
 m = ( i − k ) / hLayers  
 for( cIdx = 0; cIdx < ChromaFormatIdc = = 0 ) ? 1 : 3; cIdx++ )  
 for( h = 0; h < ( cIdx = = 0 ) ? layerHeight : layerHeight / SubHeightC ; h++ )  
 for( w = 0; w < ( cIdx = = 0 ) ? layerWidth : layerWidth / SubWidthC ; w++ ) {  
 u = k \* ( cIdx  = =  0 ) ? layerWidth : layerWidth / SubWidthC + w v = m \* ( cIdx  = =  0 ) ? layerHeight : layerHeight / SubHeightC + h  
 recTextureLayer[ i ][ cIdx ][ w ][ h ] = picture0[ cIdx ][ u ][ v ]  
 }  
 for( h = 0; h < layerHeight; h++ )  
 for( w = 0; w < layerWidth; w++ )  
 recOpacityLayer[ i ][ w ][ h ] = picture1[ 0 ][ k \* layerWidth + w ][ m \* layerHeight + h ]  
}

**8.36 Copyright SEI message**

8.35.1 Copyright SEI message syntax

|  |  |
| --- | --- |
| copyright\_information( payloadSize ) { | **Descriptor** |
| **ci\_cancel\_flag** | u(1) |
| if( !ci\_cancel\_flag ) { |  |
| **ci\_persistence\_flag** | u(1) |
| **ci\_information** | st(v) |
| } |  |
| } |  |

8.35.1 Copyright SEI message semantics

**ci\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous copyright information SEI message in output order that applies to the current layer. ci\_cancel\_flagequal to 0 indicates that copyright information follows.

**ci\_persistence\_flag** specifies the persistence of the copyright information SEI message for the current layer.

ci\_persistence\_flag equal to 0 specifies that the copyright information applies to the current decoded picture only.

ci\_persistence\_flag equal to 1 specifies that the copyright information SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with a copyright information SEI message is output that follows the current picture in output order.

**ci\_information** specifies copyright information that pertains to the picture(s) in the persistence scope defined by ci\_cancel\_flag and ci\_persistence\_flag.

Informative note: A typical content of ci\_information includes the string “Copyright” or the “©” symbol, information identifying the copyright owner such as a personal or company name, the year of creation of the content according to the Gregorian calendar, and a use/licensing statement such as “all rights reserved”.

**8.36** **AI marking SEI message**

8.36.1 AI marking SEI message syntax

|  |  |
| --- | --- |
| ai\_mark( payloadSize ) { | **Descriptor** |
| **ai\_mark\_cancel\_flag** | u(1) |
| if (!ai\_mark\_cancel\_flag) { |  |
| **ai\_mark \_persistence\_flag** | u(1) |
| **ai\_mark\_information** | st(v) |
| } |  |
| } |  |

8.36.1 AI marking SEI message semantics

**ai\_mark\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous AI marking SEI message in output order that applies to the current layer. Ai\_mark\_cancel\_flagequal to 0 indicates that copyright information follows.

**ai\_mark\_persistence\_flag** specifies the persistence of the AI mark SEI message for the current layer.

ai\_mark\_persistence\_flag equal to 0 specifies that the AI Marking information applies to the current decoded picture only.

ai\_mark\_persistence\_flag equal to 1 specifies that the AI Marking information SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an AI Marking SEI message is output that follows the current picture in output order.

**ai\_mark\_information** specifies AI marking information that pertains to the picture(s) in the persistence scope defined by ai\_mark\_cancel\_flag and ai\_mark\_persistence\_flag.

Informative note: the content of ai\_mark\_information will typically be chosen based on the legal requirements for AI marking as they may exist at the location of the AI engine that generates the AI-based content as well as at the intended place of consumption of the coded video bitstream containing the AI marking SEI message.

**8.37 Generative face video SEI message**

8.37.1 Generative face video SEI message syntax

|  |  |
| --- | --- |
| generative\_face\_video ( payloadSize ) { | **Descriptor** |
| **gfv\_id** | ue(v) |
| **gfv\_cnt** | ue(v) |
| **gfv\_base\_pic\_flag** /\*indicate if current decoded output picture is a base picture\*/ | u(1) |
| if( gfv\_base\_pic\_flag ) { /\*specify TranslatorNN( )\*/ |  |
| **gfv\_nn\_present\_flag** | u(1) |
| if( gfv\_nn\_present\_flag ) { |  |
| **gfv\_nn\_base\_flag** | u(1) |
| **gfv\_nn\_mode\_idc** | ue(v) |
| if( gfv\_nn\_mode\_idc  = =  1 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **gfv\_nn\_reserved\_zero\_bit\_a** | u(1) |
| **gfv\_nn\_tag\_uri** | st(v) |
| **gfv\_nn\_uri** | st(v) |
| } |  |
| } |  |
| } else /\* current decoded output picture is a driving picture\*/ |  |
| **gfv\_drive\_pic\_fusion\_flag** /\*indicate if DrivePicture is input to GenerativeNN( )\*/ | ue(v) |
| **gfv\_coordinate\_present\_flag** | u(1) |
| if( gfv\_coordinate\_present\_flag ) { |  |
| **gfv\_coordinate\_precision\_factor\_minus1** | ue(v) |
| **gfv\_num\_kp\_minus1** | ue(v) |
| **gfv\_kp\_pred\_flag** | u(1) |
| **gfv\_coordinate\_z\_present\_flag** | u(1) |
| if(gfv\_coordinate\_z\_present\_flag ) |  |
| **gfv\_coordinate\_z\_max\_value\_minus1** | ue(v) |
| for( i = 0; i  <=  num\_kp\_minus1; i++ ) { |  |
| if(!gfv\_kp\_pred\_flag) { |  |
| **gfv\_coordinate\_x\_abs**[ i ] | u(v) |
| if( gfv\_coordinate\_x\_abs[ i ] ) |  |
| **gfv\_coordinate\_x\_sign\_flag**[ i ] | u(1) |
| **gfv\_coordinate\_y\_abs**[ i ] | u(v) |
| if( gfv\_coordinate\_y\_abs[ i ] ) |  |
| **gfv\_coordinate\_y\_sign\_flag**[ i ] | u(1) |
| if( gfv\_coordinate\_z\_present\_flag ) { |  |
| **gfv\_coordinate\_z\_abs**[ i ] | u(v) |
| if( gfv\_coordinate\_z\_abs[ i ] ) |  |
| **gfv\_coordinate\_z\_sign\_flag**[ i ] | u(1) |
| } |  |
| } else { |  |
| **gfv\_coordinate\_dx\_abs**[ i ] | u(v) |
| if(gfv\_coordinate\_dx\_abs[ i ] ) |  |
| **gfv\_coordinate\_dx\_sign\_flag**[ i ] | u(1) |
| **gfv\_coordinate\_dy\_abs**[ i ] | u(v) |
| if( gfv\_coordinate\_dy\_abs[ i ] ) |  |
| **gfv\_coordinate\_dy\_sign\_flag**[ i ] | u(1) |
| if( gfv\_coordinate\_z\_present\_flag ) { |  |
| **gfv\_coordinate\_dz\_abs**[ i ] | u(v) |
| if( gfv\_coordinate\_dz\_abs[ i ] ) |  |
| **gfv\_coordinate\_dz\_sign\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| } |  |
| } |  |
| **gfv\_matrix\_present\_flag** | u(1) |
| if(gfv\_matrix\_present\_flag ) { |  |
| **gfv\_matrix\_element\_precision\_factor\_minus1** | ue(v) |
| **gfv\_num\_matrix\_types\_minus1** | ue(v) |
| if( !gfv\_base\_pic\_flag ) |  |
| **gfv\_matrix\_pred\_flag** | u(1) |
| for( i = 0; i  <=  num\_matrix\_types\_minus1; i++ ) { |  |
| **gfv\_matrix\_type\_idx**[ i ] | u(6) |
| if( gfv\_matrix\_type\_idx[ i ]  = =  0  | |  gfv\_matrix\_type\_idx[ i ]  = =  1 ) { |  |
| if( gfv\_coordinate\_present\_flag ) |  |
| **gfv\_num\_matrices\_equal\_to\_num\_kps\_flag**[ i ] | u(1) |
| if( !gfv\_coordinate\_present\_flag  | |  !gfv\_num\_matrix\_equal\_to\_num\_kps\_flag[ i ] ) |  |
| **gfv\_num\_matrices\_info**[ i ] | ue(v) |
| }else if( gfv\_matrix\_type\_idx[ i ]  = = 2  | |  gfv\_matrix\_type\_idx[ i ]  = =  3  | |  gfv\_matrix\_type\_idx[ i ]  >=  7 ){ |  |
| if( gfv\_matrix\_type\_idx[ i ]  >=  7 ) |  |
| **gfv\_num\_matrices\_minus1**[ i ] | ue(v) |
| **gfv\_matrix\_width\_minus1**[ i ] | ue(v) |
| **gfv\_matrix\_height\_minus1**[ i ] | ue(v) |
| }else if( gfv\_matrix\_type\_idx[ i ]  >=  4  &&  gfv\_matrix\_type\_idx[ i ]  <=  6  &&  !gfv\_coordinate\_present\_flag ){ |  |
| **gfv\_matrix\_for\_3D\_space\_flag**[ i ] | u(1) |
| for( j = 0; j < numMatrices[ i ]; j++ ) |  |
| for( k = 0; k < matrixHeight[ i ]; k++ ) |  |
| for( m = 0; m <matrixWidth[ i ]; m++ ) { |  |
| if( !gfv\_matrix\_pred\_flag ) { |  |
| **gfv\_matrix\_element\_int**[ i ][ j ][ k ][ m ] | ue(v) |
| **gfv\_matrix\_element\_dec**[ i ][ j ][ k ][ m ] | u (v) |
| if( gfv\_matrix\_element\_int[ i][ j ][ k ][ m ]  | |  gfv\_matrix\_element\_dec[ i ][ j ][ k ][ m ] ) |  |
| **gfv\_matrix\_element\_sign\_flag**[ i ][ j ][ k ][ m ] | u(1) |
| } |  |
| else { |  |
| **gfv\_matrix\_delta\_element\_int**[ i ][ j ][ k ][ m ] | ue(v) |
| **gfv\_matrix\_delta\_element\_dec**[ i ][ j ][ k ][ m ] | u (v) |
| if( gfv\_matrix\_delta\_element\_int[ i][ j ][ k ][ m ]  | |  gfv\_matrix\_delta\_element\_dec[ i ][ j ][ k ][ m ] ) |  |
| **gfv\_matrix\_delta\_element\_sign\_flag**[ i ][ j ][ k ][ m ] | u(1) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |
| if( gfv\_nn\_present\_flag ) |  |
| if( gfv\_nn\_mode\_idc  = =  0 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **gfv\_nn\_reserved\_zero\_bit\_b** | u(1) |
| for( i = 0; more\_data\_in\_payload( ); i++ ) |  |
| **gfv\_nn\_payload\_byte**[ i ] | b(8) |
| } |  |
| } |  |

8.37.1 Generative face video SEI message semantics

The generative face video (GFV) SEI message indicates facial parameters and specifies a facial parameter translator network, denoted as TranslatorNN( ), that may be used to convert various formats of facial parameters signaled in the SEI message into a fixed format of parameters, and a face picture generator neural network, denoted as GenerativeNN( ), that may be used to generate output pictures using the fixed format of facial parameters and previously decoded output pictures.

NOTE 1 – Facial parameters could be determined from source pictures prior to encoding. Such source pictures may be referred to as driving pictures.

NOTE 2 – Previously decoded output pictures input to GenerativeNN( ) may be a base picture (a decoded output picture that provides the reference texture from which the face pictures may be generated) and, optionally, a picture that can be fused by GenerativeNN( ) to improve background texture and facial details. When the current picture is not a base picture, the GFV SEI message may be used to generate a face picture based on the previously decoded base picture, the facial parameters conveyed by the GFV SEI message, and, optionally, the current decoded picture for fusion pupose.

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

* Input picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
* Luma sample array baseCroppedYPic and chroma sample arrays baseCroppedCbPic and baseCroppedCrPic for a decoded output picture, denoted as BasePicture, corresponding to a source base picture.
* Luma sample array driveCroppedYPic and chroma sample arrays driveCroppedCbPic and driveCroppedCrPic for a decoded output picture, denoted as DrivePicture, corresponding to a source driving picture.
* Bit depth BitDepthY for the luma sample array of the input pictures.
* Bit depth BitDepthC for the chroma sample arrays, if any, of the input pictures.
* A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

**gfv\_id** contains an identifying number that may be used to identify face feature information and specify a neural network that may be used as TranslatorNN( ). The value of gfv\_id shall be in the range of 0 to 232–− 2, inclusive. Values of gfv\_id from 256 to 511, inclusive, and from 231 to 232–− 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering a GFV SEI message with gfv\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232–− 2, inclusive, shall ignore the SEI message.

NOTE 3 – Different values of gfv\_id in different GFV SEI messages could be used to identify different faces when more than one face is present in an output picture, for example.

**gfv\_cnt** specifies a GFV SEI message instance count value for this gfv\_id value within a picture unit.

The gfv\_cnt of the first GFV SEI message, in decoding order, with a particular value of gfv\_id within picture unit shall be equal to 0. When gfv\_cnt assigned to currGfvCnt is greater than 0, a GFV SEI message with the same gfv\_id value and gfv\_cnt equal to currGfvCnt − 1 shall precede the current GFV SEI message in decoding order in the same picture unit.

The value of gfv\_cnt shall be in the range of 0 to 65 535, inclusive.

**gfv\_base\_pic\_flag** equal to 1 indicates the current decoded output picture corresponds to a base picture. gfv\_base\_pic\_flag equal to 0 indicates the current decoded output picture does not correspond to a base picture or this SEI message does not specify syntax elements for a base picture. When gfv\_cnt is greater than 0, gfv\_base\_pic\_flag shall be equal to 0.

The following constraints apply to the value of gfv\_base\_pic\_flag:

– When a GFV SEI message is the first GFV SEI message, in decoding order, that has a particular gfv\_id value within the current CLVS, the value of gfv\_base\_pic\_flag shall be equal to 1.

– When a GFV SEI message that has a particular gfv\_id value has gfv\_base\_pic\_flag being equal to 0, 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 up to but excluding the decoded picture that follows the current decoded picture in output order within the current CLVS and is associated with a subsequent GFV SEI message, in decoding order, having gfv\_base\_pic\_flag equal to 0 and that particular gfv\_id value within the current CLVS, whichever is earlier.

**gfv\_nn\_present\_flag** equal to 1 indicatesa neural network that may be used as a TranslatorNN( ) is contained or indicated by the SEI message. gfv\_nn\_present\_flag equal to 0 indicatesa neural network that may be used as a TranslatorNN( ) is not contained or indicated by the SEI message. When gfv\_nn\_present\_flag is not present, it is inferred to be 0.

When a GFV SEI message with a particular value of gfv\_id is present in an IRAP picture unit or in a picture unit that follows an IRAP picture unit in output order and is not preceded in output order by any picture unit that follows the IRAP picture in output order and has a GFV SEI message with that particular value of gfv\_id, gfv\_nn\_present\_flag shall be present and equal to 1.

When gfv\_nn\_present\_flag is equal to 0 and TranslatorNN is referenced in the semantics of this SEI message, the applicable TranslatorNN is defined by the GFV SEI message that is present in the last preceding picture unit in output order that has that has the same value of gfv\_id as that in the current GFV SEI message and gfv\_nn\_present\_flag equal to 1.

**gfv\_nn\_base\_flag**, **gfv\_nn\_mode\_idc**, **gfv\_nn\_reserved\_zero\_bit\_a**, **gfv\_nn\_tag\_uri**, **gfv\_nn\_uri**, **gfv\_nn\_payload\_byte**[ i ] specify a neural network that may be used as a TranslatorNN( ). gfv\_nn\_base\_flag, gfv\_nn\_mode\_idc, gfv\_nn\_reserved\_zero\_bit\_a, gfv\_nn\_tag\_uri, gfv\_nn\_uri, gfv\_nn\_payload\_byte[ i ] have the same syntax and semantics as **nnpfc\_base\_flag**, **nnpfc\_mode\_idc**, **nnpfc\_reserved\_zero\_bit\_a**, **nnpfc\_tag\_uri**, **nnpfc\_uri**, **nnpfc\_payload\_byte**[ i ], respectively.

When either of the following conditions is true, GFV SEI messages shall have the same SEI payload content:

– The GFV SEI messages are present in the same picture unit, have gfv\_cnt equal to 0, have gfv\_nn\_base\_flag present, and have the same value of gfv\_id and gfv\_nn\_base\_flag,

– The GFV SEI messages are present in the same picture unit, have the same value of gfv\_cnt that is greater than 0, and have the same value of gfv\_id.

**gfv\_drive\_pic\_fusion\_flag**, when present, equal to 1 indicates the current decoded picture, which corresponds to a driving picture that may be used for fusion, may be input to GenerativeNN( ). gfv\_drive\_pic\_fusion\_flag equal to 0 indicates the current decoded picture should not be input to GenerativeNN( ).

NOTE 3 – A gfv\_drive\_pic\_fusion\_flag value of 1 can be used, for example, to indicate that the current decoded picture can be used to improve face details or handle background changes.

NOTE 4 – Fusion takes the three inputs: the base picture, features from keypoints and/or matrices carried in the GFV SEI message, and the current decoded picture, and outputs a picture.

NOTE 5 – When current decoded picture corresponds to a driving picture, it should be marked as not for output purpose.

**gfv\_coordinate\_present\_flag** equal to 1 indicates that coordinate information of keypoints is present. gfv\_coordinate\_present\_flag equal to 0 indicates that coordinate information of keypoints is not present.

It is a requirement of bitstream conformance that when gfv\_matrix\_type\_idx[ i ] for any i from 0 to gfv\_num\_matrix\_types\_minus1 is equal to 0 or 1, the value of gfv\_coordinate\_present\_flag shall be equal to 1.

**gfv\_coordinate\_precision\_factor\_minus1** plus 1 indicates the length, in bits, of syntax elements gfv\_coordinate\_x\_abs[i], gfv\_coordinate\_y\_abs[i], gfv\_coordinate\_z\_abs[i], gfv\_coordinate\_dx\_abs[ i ], gfv\_coordinate\_dy\_abs[ i ] and gfv\_coordinate\_dz\_abs[ i ].

**gfv\_num\_kps\_minus1** plus 1indicates the number of keypoints. The value of gfv\_num\_kp\_minus1 shall be in the range of 0 to 210 – 1, inclusive.

**gfv\_kp\_pred\_flag** equal to 1 indicates syntax elements gfv\_coordinate\_dx\_abs[ i ] ,gfv\_coordinate\_dy\_abs[ i ], and gfv\_coordinate\_dz\_abs[ i ] are present and syntax elements gfv\_coordinate\_dx\_sign\_flag[ i ], gfv\_coordinate\_dy\_sign\_flag[ i ] and gfv\_coordinate\_dz\_sign\_flag[ i ] may be present. gfv\_kp\_pred\_flag equal to 0 indicates gfv\_coordinate\_x\_abs[ i ] ,gfv\_coordinate\_y\_abs[ i ], and gfv\_coordinate\_z\_abs[ i ] are present and syntax elements gfv\_coordinate\_x\_sign\_flag[ i ], gfv\_coordinate\_y\_sign\_flag[ i ] and gfv\_coordinate\_z\_sign\_flag[ i ] may be present.

**gfv\_coordinate\_z\_present\_flag** equal to 1 indicates that z-axis coordinate information of the keypoints is present. coordinate\_z\_present\_flagequal to 0 indicates that the z-axis coordinate information of the keypoints is not present.

**gfv\_coordinate\_z\_max\_value\_minus1** plus 1indicates the maximum absolute value of z-axis coordinates of keypoints.

**gfv\_coordinate\_x\_abs**[ i ] indicates the normalized absolute value of the x-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_x\_sign\_flag**[ i ] specifies the sign of the x-axis coordinate of the i-th keypoint. When gfv\_coordinate\_x\_sign\_flag[ i ] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_y\_abs**[ i ] specifies the normalized absolute value of y-axis coordinate of i-th keypoint.

**gfv\_coordinate\_y\_sign\_flag**[ i ] specifies the sign of the y-axis coordinate of the i-th keypoint. When gfv\_coordinate\_y\_sign\_flag[i] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_z\_abs**[ i ] specifies the normalized absolute value of z-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_z\_sign\_flag**[ i ] specifies the sign of the z-axis coordinate of the i-th key point. When gfv\_coordinate\_z\_sign\_flag[ i ] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_dx\_abs**[ i ] indicates the absolute difference value of the normalized value of the x-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_dx\_sign\_flag**[ i ] specifies the sign of the difference value of the x-axis coordinate of the i-th keypoint. When gfv\_coordinate\_dx\_sign\_flag[ i ] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_dy\_abs**[ i ] specifies the absolute difference value of the normalized y-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_dy\_sign\_flag**[ i ] specifies the sign of the difference value of the y-axis coordinate of the i-th keypoint. When gfv\_coordinate\_yd\_sign\_flag[i] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_dz\_abs**[ i ] specifies the absolute difference value of the normalized z-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_dz\_sign\_flag**[ i ] specifies the sign of the difference value of the z-axis coordinate of the i-th key point. When gfv\_coordinate\_dz\_sign\_flag[ i ] is not present, it is inferred to be equal to 0.

The variables coordinateDeltaX[ i ], coordinateDeltaY[ i ] and coordinateDeltaZ[ i ] indicating the delta x-axis coordinate, delta y-axis coordinate and delta z-axis coordinate of the i-th keypoint, respectively, are derived as follows:

coordinateDeltaX[ i ] =

coordinateDeltaY[i] =

coordinateDeltaZ[i] =

The variables coordinateX[ i ], coordinateY[ i ] and coordinateZ[ i ] indicating the x-axis coordinate, y-axis coordinate and z-axis coordinate of the i-th keypoint, respectively, are derived as follows:

When gfv\_kp\_pred\_flag is equal to 0,

coordinateX[ i ] =

coordinateY[i] =

coordinateZ[i] =

when gfv\_kp\_pred\_flag is equal to 1,

if( gfv\_base\_pic\_flag ) {  
 coordinateX[ i ] = (( i > 0 ) ? coordinateX[ i - 1 ] : 0 ) + coordinateDeltaX[ i ]  
 coordinateY[ i ] = (( i > 0 ) ? coordinateY[ i - 1 ] : 0 ) + coordinateDeltaY[ i ]  
 coordinateZ[ i ] = (( i > 0 ) ? coordinateZ[ i - 1 ] : 0 ) + coordinateDeltaZ[ i ]  
}  
else {  
 coordinateX[ i ] = BaseKpCoordinateX[ i ] + coordinateDeltaX[ i ]  
 coordinateY[ i ] = BaseKpCoordinateY[ i ] + coordinateDeltaY[ i ]  
 coordinateZ[ i ] = BaseKpCoordinateZ[ i ] + coordinateDeltaZ[ i ]  
}

where BaseKpCoordinateX[ i ], BaseKpCoordinateY[ i ], BaseKpCoordinateZ[ i ] indicating the x-axis, y-axis and z-axis coordinates, respectively, of the i-th keypoint for the base picture are derived as follows:

if( gfv\_base\_pic\_flag ) {  
 BaseKpCoordinateX[ i ] = coordinateX[ i ]  
 BaseKpCoordinateY[ i ] = coordinateY[ i ]  
 BaseKpCoordinateZ[ i ] = coordinateZ[ i ]  
}

**gfv\_matrix\_present\_flag** equal to 1 indicates that matrix parameters are present. gfv\_matrix\_present\_flag equal to 0 indicates that matrix parameters are not present.

**gfv\_matrix\_element\_precision\_factor\_minus1** plus 1 indicates the length, in bits, of syntax elements gfv\_matrix\_element\_dec[ i ][ j ][ k ][ m ] and gfv\_matrix\_delta\_element\_dec[ i ][ j ][ k ][ m ].

**gfv\_num\_matrix\_types\_minus1** plus 1indicates the number of matrix types signalled in the SEI message. The value of gfv\_matrix\_type\_num\_minus1 shall be in the range of 0 to 26 – 1, inclusive.

**gfv\_matrix\_pred\_flag** equal to 1 indicates syntax elements gfv\_matrix\_element\_int [ i ][ j ][ k ][ m ] , gfv\_matrix\_element\_dec [ i ][ j ][ k ][ m ] are present and gfv\_matrix\_element\_sign\_flag [ i ][ j ][ k ][ m ] may be present. gfv\_matrix\_pred\_flag equal to 0 indicates gfv\_matrix\_delta\_element\_int [ i ][ j ][ k ][ m ] , gfv\_matrix\_delta\_element\_dec [ i ][ j ][ k ][ m ] are present and syntax element gfv\_matrix\_delta\_element\_sign\_flag [ i ][ j ][ k ][ m ] may be present. When gfv\_matrix\_pred\_flag is not present, it is inferred to be 0.

**gfv\_matrix\_type\_idx**[ i ]indicates the index of the i-th matrix type as specified in Table 3.

**Table 3 - Specification of gfv\_matrix\_type\_idx**

|  |  |
| --- | --- |
| **Value** | **Specification** |
| 0 | Affine translation matrix with the size of 2\*2 or 3\*3. |
| 1 | Covariance matrix with size of 2\*2 or 3\*3. |
| 2 | Mouth matrix representing mouth motion. |
| 3 | Eye matrix representing the open-close status and level of eyes. |
| 4 | Head rotation paramters with the size of 2\*2 or 3\*3 representing the head rotation in 2D space or 3D space. |
| 5 | Head translation matrix with the size of 1\*2 or 1\*3 representing head translationin 2D space or 3D space. |
| 6 | Head location matrix with size of 1\*2 or 1\*3 representing the head location in 2D space or 3D space. |
| 7 | Compact feature matrix with the size being specified by gfv\_matrix\_width\_minus1[i] and gfv\_matrix\_height\_minus1[i]. |
| 8…31 | Other matrix that may be used as determined by the application with the size being specified by gfv\_matrix\_width\_minus1[i] and gfv\_matrix\_height\_minus1[i]. |
| 32…63 | Reserved |

NOTE 6 . The undefined matrxi type is used to represent the matrxi type rather than affine translation matrix, covariance matrix, rotation matrix, translation matrix and compact feature matrix. It is may be used by the user to extend the matrix type.

**gfv\_num\_matrices\_equal\_to\_num\_kps\_flag**[ i ] equal to 1 indicates that the number of matrices of the i-th matrix type is equal to gfv\_num\_kps\_minus1 + 1. gfv\_num\_matrices\_equal\_to\_num\_kps\_flag[ i ] equal to 0 indicates the number of matrices of the i-th matrix type is not equal to gfv\_num\_coordinates\_minus1 + 1.

**gfv\_num\_matrices\_info**[ i ] provides information to derive the number of the matrices of the i-th matrix type.

**gfv\_matrix\_width\_minus1**[ i ] plus 1 indicates the width of the matrix of the i-th matrix type.

**gfv\_matrix\_height\_minus1**[ i ] plus 1 indicates the height of the matrix of the i-th matrix type.

**gfv\_matrix\_for\_3D\_space\_flag**[ i ] equal to 1 indicates the matrix of the i-th matrix type is a matrix defined in three-dimensional space. gfv\_matrix\_for\_3D\_space\_flag[ i ] equal to 0 indicates the matrix of the i-th matrix type is a matrix defined in two-dimensional space.

When gfv\_matrix\_width\_minus1[ i ] is not present, it is inferred as follows:

–– If gfv\_matrix\_type\_idx[ i ] is equal to 0, 1 or 4, and one of coordinate\_z\_present\_flag and gfv\_matrix\_for\_3D\_space\_flag[ i ] is present and equal to 1, gfv\_matrix\_width\_minus1[i] is inferred to be equal to 2

–– otherwise, if matrix\_type\_idx[ i ] is equal to 0, 1 or 4, and one of coordinate\_z\_present\_flag and gfv\_matrix\_for\_3D\_space\_flag[ i ] is present and equal to 0, gfv\_matrix\_width\_minus1[ i ] is inferred to be equal to 1

–– otherwise (matrix\_type\_idx[ i ] is equal to 5 or 6), gfv\_matrix\_width\_minus1[ i ] is inferred to be equal to 0

When gfv\_matrix\_height\_minus1[ i ] is not present, it is inferred as follows:

–– If matrix\_type\_idx is equal to 0, 1, 4, 5 or 6, and one of gfv\_coordinate\_z\_present\_flag and gfv\_matrix\_for\_3D\_space\_flag[ i ] is present and equal to 1, gfv\_matrix\_height\_minus1[ i ] is inferred to be equal to 2.

–– otherwise (gfv\_matrix\_type\_idx is equal to 0, 1, 4, 5 or 6, and one of gfv\_coordinate\_z\_present\_flag and gfv\_matrix\_for\_3D\_space\_flag[ i ] is 0), gfv\_matrix\_height \_minus1[ i ] is inferred to be equal to 1.

The variables matrixWidth[ i ] and matrixHeight[ i ] indicating the width and height of the matrix of the i-th matrix type are derived as follows

matrixWidth[ i ] = gfv\_matrix\_width\_minus1[ i ] + 1

matrixHeight[ i ] = gfv\_matrix\_height\_minus1[ i ] + 1

**gfv\_num\_matrices\_minus1**[ i ] plus 1 indicates the number of matrices of the i-th matrix type.

The variable numMatrices[ i ] indicating the number of the matrices of the i-th matrix type is derived as follows:

if( gfv\_matrix\_type\_idx[ i ] == 0 || gfv\_matrix\_type\_idx[ i ] == 1 ) {

if( gfv\_coordinate\_present\_flag )

numMatrices[ i ] = gfv\_num\_matrices\_equal\_to\_num\_kps\_flag[ i ] ? gfv\_num\_kps\_minus1 + 1 : ( gfv\_num\_matrices\_info[ i ] < gfv\_num\_kp\_minus1 ? gfv\_num\_matrices\_info [ i ] + 1 : gfv\_num\_matrices\_info [ i ] + 2 )

else

numMatrices[ i ] = gfv\_num\_matrices\_info[ i ] + 1

}

else if( gfv\_matrix\_type\_idx[ i ] >= 2 && gfv\_matrix\_type\_idx[ i ] < 7)

numMatrices[ i ] = 1

else

numMatrices[ i ] = gfv\_num\_matrices\_minus1[ i ] + 1

**gfv\_matrix\_element\_int**[ i ][ j ][ k ][ m ] indicates the integer part of the value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type.

**gfv\_matrix\_element\_dec**[ i ][ j ][ k ][ m ] indicates the decimal part of the value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type.

**gfv\_matrix\_element\_sign\_flag**[ i ][ j ][ k ][ m ] indicates the sign of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type. When gfv\_matrix\_element\_sign\_flag[ i ][ j ][ k ][ m ]is not present, it is inferred to be equal to 0.

**gfv\_matrix\_delta\_element\_int**[ i ][ j ][ k ][ m ] indicates the integer part of the difference value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type.

**gfv\_matrix\_delta\_element\_dec**[ i ][ j ][ k ][ m ] indicates the decimal part of the difference value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type.

**gfv\_matrix\_delta\_element\_sign\_flag**[ i ][ j ][ k ][ m ] indicates the sign of the difference value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type. When gfv\_matrix\_element\_sign\_flag[ i ][ j ][ k ][ m ]is not present, it is inferred to be equal to 0.

The variable matrixElementDeltaVal[ i ][ j][ k ][ m ] representing the difference value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type is derived as follows:

matrixElementDeltaVal[ i][ j ][ k ][ m ] =

The variable matrixElementVal[ i ][ j][ k ][ m ] representing the value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type is derived as follows:

when gfv\_matrix\_pred\_flag is equal to 0

matrixElementVal[ i][ j ][ k ][ m ] =   
if( gfv\_base\_pic\_flag )   
 BaseMatrixElementVal[ i][ j ][ k ][ m ] = matrixElementVal[ i][ j ][ k ][ m ]

when gfv\_matrix\_pred\_flag is equal to 1

matrixElementVal[ i][ j ][ k ][ m ] = BaseMatrixElementVal[ i][ j ][ k ][ m ] + matrixElementDeltaVal[ i][ j ][ k ][ m ] +

For a particular gfv\_id value, the following process is used in increasing order of gfv\_cnt to generate a video picture per each GFV SEI message that has gfv\_base\_pic\_flag equal to 0 and a unique value of gfv\_cnt within a picture unit:

DeriveSigParam( )   
TranslatorNN (sigKeyPoint , sigMatrix)   
DeriveInputTensors( )   
if( gfv\_base\_pic\_flag == 0 && gfv\_drive\_pic\_fusion\_flag == 0) {  
 if(ChromaFormatIdc == 0 )   
 GenerativeNN( inputBaseY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix)   
 else   
 GenerativeNN( inputBaseY, inputBaseCb, inputBaseCr, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix)   
}  
else if(gfv\_base\_pic\_flag == 0 && gfv\_drive\_pic\_fusion\_flag == 1) {  
 if(ChromaFormatIdc == 0 )   
 GenerativeNN( inputBaseY, inputDriveY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix)   
 else   
 GenerativeNN( inputBaseY, inputBaseCb, inputBaseCr, inputDriveY, inputDriveCb, inputDriveCr , inputBaseKeyPoint, inputBaseMatrix,, inputDriveKeyPoint, inputDriveMatrix)   
}

StoreOutputTensors( )

The process DeriveSigParam () for deriving the inputs of TranslatorNN () is specified as follows:

The keypoint coordinate array sigKeyPoint and the matrix sigMatrix are derived as follows:

if( gfv\_coordinate\_present\_flag ) {  
 for ( i = 0; i<= gfv\_num\_kps\_minus1; i++ ) {  
 sigKeyPoint[ i ][ 0 ] = coordinateX[ i ]  
 sigKeyPoint[ i ][ 1 ] = coordinateY[ i ]  
 if ( gfv\_coordinate\_z\_present\_flag )   
 sigKeyPoint[ i ][ 2 ] = coordinateZ[ i ]  
 }  
}  
else {  
 for ( i = 0; i <=num\_kps\_minus1; i++ ) {  
 sigKeyPoint [ i ][ 0 ] = 0  
 sigKeyPoint [ i ][ 1 ] = 0  
 if ( gfv\_coordinate\_z\_present\_flag )   
 sigKeyPoint [ i ][ 2 ] = 0  
 }  
}  
if( gfv\_matrix\_present\_flag ) {  
 for ( i = 0; i <= gfv\_num\_matrix\_types\_minus1; i++ ) {  
 for ( j = 0; j < numMatrices[ i ]; j++ ) {  
 for( k = 0; k < matrixHeight [ i ]; k++ ) {  
 for ( l = 0;l < matrixWidth [ i ]; l++) {  
 sigMatrix[ i ][ j ][ k ][ l ] = matrixElementVal[ i ][ j][ k][ l ]   
 }  
 }  
 }  
 }  
}  
else {  
 for ( i = 0; i <= gfv\_num\_matrix\_types\_minus1; i++ ) {  
 for ( j = 0; j < numMatrices[ i ]; j++ ) {  
 for( k = 0; k < matrixHeight [ i ]; k++ ) {  
 for ( l = 0;l < matrixWidth [ i ]; l++) {  
 sigMatrix [ i ][ j ][ k ][ l ] = 0  
 }  
 }  
 }  
 }  
}

TranslatorNN( ) is a process to translate the various formats of the facial parameters carried in the SEI message to the fixed format of the facial parameters to be input to the generative network to generate the output picture

Inputs to TranslatorNN() are:

* sigKeyPoint and sigMatrix

Outputs of TranslatorNN() are:

* convKeyPoint and convNumKeyPoint
* convMatrix and convNumMatrix, convMatrixWidth, convMatrixHeight

The process DeriveInputTensors( ) for deriving the inputs of GenerativeNN ( ) is specified as follows:

When gfv\_base\_pic\_flag is equal to 1, the BasePicture input tensor inputBaseY, inputBaseCb and inputBaseCr are derived as follows:

for( x = 0; x < CroppedWidth; x++ ) {  
 for ( y = 0; y < CroppedHeight; y++ ) {  
 inputBaseY[ x ][ y ] = InpY( baseCroppedYPic[ x ][ y ] )   
 }  
}  
if (ChromaFormatIdc !=0) {  
 for( x = 0; x < CroppedWidth/ SubWidthC; x++ ) {  
 for ( y = 0; y < CroppedHeight/ SubHeightC; y++ ) {  
 inputBaseCb[ x][ y ] = InpC( baseCroppedCbPic[ x][ y ] )  
 inputBaseCr[ x][ y ] = InpC( baseCroppedCrPic[ x ][ y ] )  
 }  
 }  
}

When gfv\_drive\_pic\_fusion\_flag is equal to 1, the DrivePicture luma sample array inputDriveY, inputDriveCb and input DriveCr are derived as follows:

for( x = 0; x< CroppedWidth; x++ ) {  
 for ( y = 0; y< CroppedHeight; y++ ) {  
 inputDriveY[ x ][ y ] = InpY( driveCroppedYPic[ x ][ y ] )  
 }  
}  
if (ChromaFormatIdc !=0) {  
 for( x = 0; x< CroppedWidth/ SubWidthC; x++ ) {  
 for ( y = 0; y < CroppedHeight/ SubHeightC; y++ ) {  
 InputDriveCb[ x][ y ] = InpC( driveCroppedCbPic[ x][ y ] )  
 InputDriveCr[ x][ y ] = InpC( driveCroppedCrPic[ x ][ y ] )   
 }  
 }  
}

When gfv\_base\_pic\_flag is equal to 0, the keypoint coordinate array inputDriveKeyPoint and the matrix inputDriveMatrix for the current picture are derived as follows:

for ( i = 0; i <= convNumKeyPoint; i++ ) {  
 inputDriveKeyPoint[ i ][ 0 ] = convKeyPoint[ i ][ 0 ]  
 inputDriveKeyPoint [ i ][ 1 ] = convKeyPoint[ i ][ 1 ]  
 inputDriveKeyPoint [ i ][ 2 ] = convKeyPoint[ i ][ 2 ]  
 }

for( j = 0; j < convNumMatrix; j++ ) {  
 for( k=0; k< convMatrixHeight; k++ ) {  
 for ( m=0;m< convMatrixWidth; m++) {  
 inputDriveMatrix[ j ][ k ][ m ] = convMatrix [ j ][ k ][ m ]   
 }  
 }  
}

When gfv\_base\_pic\_flag is equal to 1, the keypoint coordinate array inputBaseKeyPoint and the matrix inputBaseMatrix for the base picture are derived as follows:

for ( i = 0; i <= convNumKeyPoint; i++ ) {  
 inputBaseKeyPoint[ i ][ 0 ] = convKeyPoint[ i ][ 0 ]  
 inputBaseKeyPoint [ i ][ 1 ] = convKeyPoint[ i ][ 1 ]  
 inputBaseKeyPoint [ i ][ 2 ] = convKeyPoint[ i ][ 2 ]  
}  
for( j = 0; j < convNumMatrix; j++) {  
 for( k=0; k< convMatrixHeight; k++ ) {  
 for ( l=0;l< convMatrixWidth; l++) {  
 inputBaseMatrix[ j ][ k ][ l ] = convMatrix [ j ][ k ][ l ]   
 }  
 }  
}

where the functions InpY( ) and InpC( ) are specified as follows:

InpY( x ) = x ÷ ( ( 1  <<  BitDepthY ) – 1 )

InpC( x ) = x ÷ ( ( 1  <<  BitDepthC ) – 1 )

GenerativeNN ( ) is a process to generate the sample values of an output picture corresponding to a driving picture. It is only invoked when gfc\_base\_pic\_flag is equal to 0. Input values to GenerativeNN( ) and output values from GenerativeNN( ) are real numbers.

Inputs to GenerativeNN() are:

* When gfv\_base\_pic\_flag is equal to 0 and gfv\_drive\_pic\_fusion\_flag is equal to 0 and ChromaFormatIdc is equal to 0: inputBaseY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix
* When gfv\_base\_pic\_flag is equal to 0 and gfv\_drive\_pic\_fusion\_flag is equal to 0 and ChromaFormatIdc is not equal to 0: inputBaseY, inputBaseCb, inputBaseCr, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix
* When gfv\_base\_pic\_flag is equal to 0 and gfv\_drive\_pic\_fusion\_flag is equal to 1 and ChromaFormatIdc is equal to 0: inputBaseY, inputDriveY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix
* When gfv\_base\_pic\_flag is equal to 0 and gfv\_drive\_pic\_fusion\_flag is equal to 1 and ChromaFormatIdc is not equal to 0: inputBaseY, inputBaseCb, inputBaseCr, inputDriveY, inputDriveCb, inputDriveCr , inputBaseKeyPoint, inputBaseMatrix,, inputDriveKeyPoint, inputDriveMatrix

Outputs of GenerativeNN( ) are:

* A luma sample array genY
* When ChromaFormatIdc is not equal to 0, two chroma sample arrays genCb and genCr.

The process StoreOutputTensors( ) for deriving the output is specified as follows:

when gfv\_base\_pic\_flag is equal to 0, the output sample array outYPic[ x ][ y ], outCbPic[ x ][ y ], and outCrPic[ x ][ y ] are derived as follows:

for(x=0; x< CroppedWidth; x++){  
 for(y=0; y< CroppedHeight; y++){  
 outputYPic[ x ][ y ] = OutY( genY[ x ][ y ] )  
 }  
}

if(ChromaFormatIdc != 0) {  
 for(x=0; x< CroppedWidth/ SubWidthC; x++){  
 for(y=0; y< CroppedHeight/ SubHeightC; y++){  
 outputCbPic[ x ][ y ] = OutC( genCb[ x ][ y ] )  
 outputCrPic[ x][ y ] = OutC( genCr[ x ][ y ] )  
 }  
 }  
}

when gfv\_base\_pic\_flag is equal to 1, the output sample array outYPic[ x ][ y ], outCbPic[ x ][ y ], and outCrPic[ x ][ y ] are derived as follows:

for(x=0; x< CroppedWidth; x++){  
 for(y=0; y< CroppedHeight; y++){  
 outputYPic[ x ][ y ] = baseCroppedYPic [ x ][ y ]   
 }  
}

if(ChromaFormatIdc != 0) {  
 for(x=0; x< CroppedWidth/ SubWidthC; x++){  
 for(y=0; y< CroppedHeight/ SubHeightC; y++){  
 outputCbPic[ x ][ y ] = baseCroppedCbPic [ x ][ y ]   
 outputCrPic[ x][ y ] = baseCroppedCbPic [ x ][ y ]   
 }  
 }  
}

where the functions OutY( ) and OutC( ) are specified as follows:

OutY( x ) = Clip3( 0, ( 1  <<  BitDepthY ) – 1 , x \* ( ( 1  <<  BitDepthY ) – 1 )   
OutC( x ) = Clip3( 0, ( 1  <<  BitDepthC ) – 1 , x \* ( ( 1  <<  BitDepthC ) – 1 )

**8.38 Film grain regions characteristics SEI message**

8.38.1 Film grain regions characteristics SEI message syntax

|  |  |
| --- | --- |
| film\_grain\_regions\_characteristics( payloadSize ) { | **Descriptor** |
| **fgr\_cancel\_flag** | u(1) |
| if( !fgr\_cancel\_flag ) { |  |
| **fgr\_model\_id** | u(2) |
| **fgr\_separate\_colour\_description\_present\_flag** | u(1) |
| if( fgr\_separate\_colour\_description\_present\_flag ) { |  |
| **fgr\_bit\_depth\_luma\_minus8** | u(3) |
| **fgr\_bit\_depth\_chroma\_minus8** | u(3) |
| **fgr\_full\_range\_flag** | u(1) |
| **fgr\_colour\_primaries** | u(8) |
| **fgr\_transfer\_characteristics** | u(8) |
| **fgr\_matrix\_coeffs** | u(8) |
| } |  |
| **fgr\_blending\_mode\_id** | u(2) |
| **fgr\_log2\_scale\_factor** | u(4) |
| for( c = 0; c < 3; c++ ) |  |
| **fgr\_comp\_model\_present\_flag**[ c ] | u(1) |
| for( c = 0; c < 3; c++ ) |  |
| if( fgr\_comp\_model\_present\_flag[ c ] ) { |  |
| **fgr\_num\_intensity\_intervals\_minus1**[ c ] | u(8) |
| **fgr\_num\_model\_values\_minus1**[ c ] | u(3) |
| for( i = 0; i <= fgr\_num\_intensity\_intervals\_minus1[ c ]; i++ ) { |  |
| **fgr\_intensity\_interval\_lower\_bound**[ c ][ i ] | u(8) |
| **fgr\_intensity\_interval\_upper\_bound**[ c ][ i ] | u(8) |
| for( j = 0; j <= fgr\_num\_model\_values\_minus1[ c ]; j++ ) |  |
| **fgr\_comp\_model\_value**[ c ][ i ][ j ] | se(v) |
| **}** |  |
| **}** |  |
| **fgr\_region\_information\_present\_flag** | u(1) |
| if( fgr\_region\_information\_present\_flag ) { |  |
| **fgr\_region\_based\_adaptation\_flag** | u(1) |
| if( fgr\_region\_based\_adaptation\_flag ) { |  |
| **fgr\_active\_regions\_number\_minus1** | u(8) |
| for( i = 0; i <= **fgr\_active\_regions\_number\_minus1**; i++ ) { |  |
| **fgr\_region\_top[** i ] | u(16) |
| **fgr\_region\_left**[ i ] | u(16) |
| **fgr\_region\_width**[ i ] | u(16) |
| **fgr\_region\_height**[ i ] | u(16) |
| **fgr\_film\_grain\_enabled\_flag**[ i ] | u(1) |
| if ( fgr\_film\_grain\_enabled\_flag[ i ] ) { |  |
| for( c = 0; c < 3; c++ ) |  |
| if ( fgr\_comp\_model\_present\_flag[ c ] ) { |  |
| **fgr\_region\_interval\_min**[ i ][ c ] | **u(8)** |
| **fgr\_region\_interval\_max**[ i ][ c ] | **u(8)** |
| } |  |
| } |  |
| } |  |
| } |  |
| else |  |
| **fgr\_alpha\_channel\_adaptation\_flag** | u(1) |
| } |  |
| **fgr \_persistence\_flag** | u(1) |
| } |  |
| } |  |

8.38.1 Film grain regions characteristics SEI message semantics

This SEI message provides the decoder with a parameterized model for a film grain synthesis process. The film grain synthesis process should be applied to the decoded pictures prior to their display.

Compared to the FGC SEI message defined in 8.5, this SEI message also provides the decoder with parameters defining how to apply locally the film grain synthesis.

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

– A picture width and picture height in units of luma samples, denoted herein by PicWidthInLumaSamples and PicHeightInLumaSamples, respectively.

– When the syntax element rdf\_separate\_colour\_description\_present\_flag of the film grain regions characteristics SEI message is equal to 0, the following additional variables:

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in clause 7.3.

– A bit depth for the samples of the luma component, denoted herein by BitDepthY, and when ChromaFormatIdc is not equal to 0, a bit depth for the samples of the two associated chroma components, denoted herein by BitDepthC.

The film grain models specified in the film grain regions characteristics SEI message are expressed for application to decoded pictures that have 4:4:4 colour format with luma and chroma bit depths corresponding to the luma and chroma bit depths of the film grain model and use the same colour representation domain as the identified film grain model. When the colour format of the decoded video is not 4:4:4 or the decoded video uses a different luma or chroma bit depth from that of the film grain model or uses a different colour representation domain from that of the identified film grain model, an unspecified conversion process is expected to be applied to convert the decoded pictures to the form that is expressed for application of the film grain model.

NOTE 4 – Because the use of a specific method is not required for performing the film grain generation function used by the display process, a decoder could, if desired, down-convert the model information for chroma in order to simulate film grain for other chroma formats (4:2:0 or 4:2:2) rather than up-converting the decoded video (using a method not specified in this Specification) before performing film grain generation.

**fgr\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous film grain regions characteristics SEI message in output order that applies to the current layer. fgr\_cancel\_flag equal to 0 indicates that film grain regions characteristics follows.

**fgr\_model\_id** identifies the film grain simulation model as specified in Table 5. The value of fgr\_model\_id shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for fgr\_model\_id are reserved for future use by ITU‑T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore film grain regions characteristics SEI messages with fgr\_model\_id equal to 2 or 3.

**Table 5 – fgr\_model\_id values**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | Frequency filtering |
| 1 | Auto-regression |

**fgr\_separate\_colour\_description\_present\_flag** equal to 1 indicates that a distinct combination of luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients for the film grain characteristics specified in the SEI message is present in the film grain regions characteristics SEI message syntax. fgr\_separate\_colour\_description\_present\_flag equal to 0 indicates that the combination of luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients for the film grain characteristics specified in the SEI message are the same as indicated in VUI parameters for the CLVS.

NOTE 5 – When fgr\_separate\_colour\_description\_present\_flag is equal to 1, any of the luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients specified for the film grain characteristics specified in the SEI message could differ from that for the pictures in the CLVS.

When VUI parameters are not present for the CLVS or the value of vui\_colour\_description\_present\_flag is equal to 0, and equivalent information to that conveyed when vui\_colour\_description\_present\_flag is equal to 1 is not conveyed by external means, fgr\_separate\_colour\_description\_present\_flag shall be equal to 1.

The input image Î, which may be the decoded picture or converted from the decoded picture, used in the equations in this clause is in the same colour representation domain as the simulated film grain signal. Therefore, when any of these parameters does differ from that for the pictures in CLVS, the input image Î used in the equations in this clause would be in a different colour representation domain than that for the pictures in the CLVS. For example, when the value of fgr\_bit\_depth\_luma\_minus8 + 8 is greater than BitDepthY (i.e., the bit depth of the luma component of the pictures in the CLVS), the bit depth of the input image Î used in the equations in this clause is also greater than BitDepthY. In such a case, the input image Î would be generated by converting the actual decoded picture to be in the same colour representation domain as the simulated film grain signal. The process for converting an actual decoded picture to the 4:4:4 colour format with same colour representation domain as the simulated film grain signal is not specified in this Specification.

**fgr\_bit\_depth\_luma\_minus8** plus 8 specifies the bit depth used for the luma component of the film grain characteristics specified in the SEI message. When fgr\_bit\_depth\_luma\_minus8 is not present in the film grain regions characteristics SEI message, the value of fgr\_bit\_depth\_luma\_minus8 is inferred to be equal to BitDepthY − 8.

The value of fgBitDepth[ 0 ] is derived as follows:

fgBitDepth[ 0 ] = fgr\_bit\_depth\_luma\_minus8 + 8 (19)

**fgr\_bit\_depth\_chroma\_minus8** plus 8 specifies the bit depth used for the Cb and Cr components of the film grain characteristics specified in the SEI message. When fgr\_bit\_depth\_chroma\_minus8 is not present in the film grain regions characteristics SEI message, the value of fgr\_bit\_depth\_chroma\_minus8 is inferred to be equal to BitDepthC − 8.

The value of fgBitDepth[ c ] for c = 1 and 2 is derived as follows:

fgBitDepth[ c ] = fgr\_bit\_depth\_chroma\_minus8 + 8, with c = 1, 2 (20)

**fgr\_full\_range\_flag** has the same semantics as specified in clause 7.3 for the vui\_full\_range\_flag syntax element, except as follows:

– fgr\_full\_range\_flag specifies the video full range flag of the film grain characteristics specified in the SEI message, rather than the video full range flag used for the CLVS.

– When fgr\_full\_range\_flag is not present in the film grain regions characteristics SEI message, the value of fgr\_full\_range\_flag is inferred to be equal to vui\_full\_range\_flag.

**fgr\_colour\_primaries** has the same semantics as specified in clause 7.3 for the vui\_colour\_primaries syntax element, except as follows:

– fgr\_colour\_primaries specifies the colour primaries of the film grain characteristics specified in the SEI message, rather than the colour primaries used for the CLVS.

– When fgr\_colour\_primaries is not present in the film grain regions characteristics SEI message, the value of fgr\_colour\_primaries is inferred to be equal to vui\_colour\_primaries.

**fgr\_transfer\_characteristics** has the same semantics as specified in clause 7.3 for the vui\_transfer\_characteristics syntax element, except as follows:

– fgr\_transfer\_characteristics specifies the transfer characteristics of the film grain characteristics specified in the SEI message, rather than the transfer characteristics used for the CLVS.

– When fgr\_transfer\_characteristics is not present in the film grain regions characteristics SEI message, the value of fgr\_transfer\_characteristics is inferred to be equal to vui\_transfer\_characteristics.

**fgr\_matrix\_coeffs** has the same semantics as specified in clause 7.3 for the vui\_matrix\_coeffs syntax element, except as follows:

– fgr\_matrix\_coeffs specifies the matrix coefficients of the film grain characteristics specified in the SEI message, rather than the matrix coefficients used for the CLVS.

– When fgr\_matrix\_coeffs is not present in the film grain regions characteristics SEI message, the value of fgr\_matrix\_coeffs is inferred to be equal to vui\_matrix\_coeffs.

– The values allowed for fgr\_matrix\_coeffs are not constrained by the chroma format of the decoded video pictures that is indicated by the value of ChromaFormatIdc for the semantics of the VUI parameters.

**fgr\_blending\_mode\_id** identifies the blending mode used to blend the simulated film grain with the input images as specified in Table 6. fgr\_blending\_mode\_id shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for fgr\_blending\_mode\_id are reserved for future use by ITU‑T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore film grain characteristic SEI messages with fgr\_blending\_mode\_id equal to 2 or 3.

**Table 6 – fgr\_blending\_mode\_id values**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | Additive |
| 1 | Multiplicative |

Depending on the value of fgr\_blending\_mode\_id, the blending mode is specified as follows:

– If fgr\_blending\_mode\_id is equal to 0, the blending mode is additive as specified by:

Igrain[ c ][ x ][ y ] = Clip3( 0, ( 1 << fgBitDepth[ c ] ) − 1, Î[ c ][ x ][ y ] + G[ c ][ x ][ y ] ) (21)

– Otherwise (fgr\_blending\_mode\_id is equal to 1), the blending mode is multiplicative as specified by:

Igrain[ c ][ x ][ y ] = Clip3( 0, ( 1 << fgBitDepth[ c ] ) − 1, Î[ c ][ x ][ y ] + (22)  
 Round( ( Î[ c ][ x ][ y ] \* G[ c ][ x ][ y ] ) ÷ ( ( 1  <<  fgBitDepth[ c ] ) − 1 ) ) )

where Î[ c ][ x ][ y ] represents the sample value at coordinates x, y of the colour component c of the input image Î, G[ c ][ x ][ y ] is the simulated film grain value at the same position and colour component, and fgBitDepth[ c ] is the number of bits used for each sample in a fixed-length unsigned binary representation of the arrays Igrain[ c ][ x ][ y ], Î[ c ][ x ][ y ], and G[ c ][ x ][ y ], where c = 0..2, x = 0..PicWidthInLumaSamples − 1, and y = 0..PicHeightInLumaSamples − 1.

**fgr\_log2\_scale\_factor** specifies a scale factor used in the film grain characterization equations.

**fgr\_comp\_model\_present\_flag**[ c ] equal to 0 indicates that film grain is not modelled on the c-th colour component, where c equal to 0 refers to the luma component, c equal to 1 refers to the Cb component, and c equal to 2 refers to the Cr component. fgr\_comp\_model\_present\_flag[ c ] equal to 1 indicates that syntax elements specifying modelling of film grain on colour component c are present in the SEI message.

When fgr\_separate\_colour\_description\_present\_flag is equal to 0 and ChromaFormatIdc is equal to 0, the value of fgr\_comp\_model\_present\_flag[ 1 ] and fgr\_comp\_model\_present\_flag[ 2 ] shall be equal to 0.

**fgr\_num\_intensity\_intervals\_minus1**[ c ] plus 1 specifies the number of intensity intervals for which a specific set of model values has been estimated.

NOTE 6 – The intensity intervals could overlap in order to simulate multi-generational film grain.

**fgr\_num\_model\_values\_minus1**[ c ] plus 1 specifies the number of model values present for each intensity interval in which the film grain has been modelled. The value of fgr\_num\_model\_values\_minus1[ c ] shall be in the range of 0 to 5, inclusive.

**fgr\_intensity\_interval\_lower\_bound**[ c ][ i ] specifies the lower bound of the i-th intensity interval for which the set of model values applies.

**fgr\_intensity\_interval\_upper\_bound**[ c ][ i ] specifies the upper bound of the i-th intensity interval for which the set of model values applies.

The variable intensityIntervalIdx[ c ][ x ][ y ][ j ] represents the j-th index to the list of intensity intervals selected for the sample value Î[ c ][ x ][ y ] for c = 0..2, x = 0..PicWidthInLumaSamples − 1, y = 0..PicHeightInLumaSamples − 1, and j = 0..numApplicableIntensityIntervals[ c ][ x ][ y ] − 1, where numApplicableIntensityIntervals[ c ][ x ][ y ] is derived below.

The minimum and maximum interval index for the sample of colour component c at coordinate x, y is derived as follows:

* If fgr\_region\_based\_adaptation\_flag is equal to 0, intensityIntervalMin[ c ][ x ][ y ] is set equal to 0 and intensityIntervalMax[ c ][ x ][ y ] is set equal to fgr\_num\_intensity\_intervals\_minus1[ c ]
* Otherwise, intensityIntervalMax[ c ][ x ][ y ] and intensityIntervalMin[ c ][ x ][ y ] are derived as follows:

intensityIntervalMin[ c ][ x ][ y ] = 1  
intensityIntervalMax[ c ][ x ][ y ] = 0  
for( i = 0; i <= fgr\_active\_regions\_number\_minus1; i++ )  
 if ( x >= fgr\_region\_left[ i ] && x < fgr\_region\_left[ i ] + fgr\_region\_width[ i ] &&  
 y >= fgr\_region\_top[ i ] && y < fgr\_region\_top[ i ] + fgr\_region\_height[ i ] )  
 if ( fgr\_film\_grain\_enabled\_flag[ i ] ) { (23)  
 intensityIntervalMin[ c ][ x ][ y ] = fgr\_region\_interval\_min[ i ][ c ]  
 intensityIntervalMax[ c ][ x ][ y ] = fgr\_region\_interval\_max[ i ][ c ]  
 } else {  
 intensityIntervalMin[ c ][ x ][ y ] = 1  
 intensityIntervalMax[ c ][ x ][ y ] = 0  
 }

Depending on the value of fgr\_model\_id, the selection of the one or more intensity intervals for the sample value Î[ c ][ x ][ y ] is specified as follows:

– The variable numApplicableIntensityIntervals[ c ][ x ][ y ] is initially set equal to 0.

– If fgr\_model\_id is equal to 0, the following applies:

* The top-left sample location ( xB, yB ) of the current 8x8 block b that contains the sample value Î[ c ][ x ][ y ] is derived as ( xB, yB ) = ( x / 8, y / 8 ).
* The average value bavg of the current 8x8 block b is derived as follows:

sum8x8 = 0  
for( i = 0; i < 8; i++ )  
 for( j = 0; j < 8; j++ )  
 sum8x8 += Î[ c ][ xB \* 8 + i ][yB \* 8 + j ] (23)  
bavg = Clip3( 0, 255, ( sum8x8 + ( 1  <<  ( fgBitDepth[ c ] − 3 ) ) )  >>  ( fgBitDepth[ c ] − 2 ) )

* The value of intensityIntervalIdx[ c ][ x ][ y ][ j ] is derived as follows:

for( i = intensityIntervalMin[ c ][ x ][ y ], j = 0; i <= intensityIntervalMax[ c ][ x ][ y ]; i++ )  
 if( bavg  >=  fgr\_intensity\_interval\_lower\_bound[ c ][ i ]  &&  
 bavg  <=  fgr\_intensity\_interval\_upper\_bound[ c ][ i ] ) {  
 intensityIntervalIdx[ c ][ x ][ y ][ j ] = i (24)  
 j++  
 }  
numApplicableIntensityIntervals[ c ][ x ][ y ] = j

– Otherwise (fgr\_model\_id is equal to 1), the value of intensityIntervalIdx[ c ][ x ][ y ][ j ] is derived as follows:

I8[ c ][ x ][ y ] = ( fgBitDepth[ c ]  = =  8 ) ? ( Î[ c ][ x ][ y ] :  
 Clip3( 0, 255, ( Î[ c ][ x ][ y ] + ( 1  <<  ( fgBitDepth[ c ] − 9 ) ) )  >>  ( fgBitDepth[ c ] − 8 ) )  
for( i = intensityIntervalMin[ c ][ x ][ y ], j = 0; i <= intensityIntervalMax[ c ][ x ][ y ]; i++ )  
 if( I8[ c ][ x ][ y ]  >=  fgr\_intensity\_interval\_lower\_bound[ c ][ i ]  && (25)  
 I8[ c ][ x ][ y ]  <=  fgr\_intensity\_interval\_upper\_bound[ c ][ i ] ) {  
 intensityIntervalIdx[ c ][ x ][ y ][ j ] = i  
 j++  
 }  
numApplicableIntensityIntervals[ c ][ x ][ y ] = j

Samples that do not fall into any of the defined intervals (i.e., those samples for which the value of numApplicableIntensityIntervals[ c ][ x ][ y ] is equal to 0) are not modified by the grain generation function. Samples that fall into more than one interval (i.e., those samples for which the value of numApplicableIntensityIntervals[ c ][ x ][ y ] is greater than 1) will originate multi-generation grain. Multi-generation grain results from adding the grain computed independently for each of the applicable intensity intervals.

In the equations in the remainder of this clause, the variable sj in each instance of the list fgr\_comp\_model\_value[ c ][ sj ] is the value of intensityIntervalIdx[ c ][ x ][ y ][ j ] derived for the sample value Î[ c ][ x ][ y ].

**fgr\_comp\_model\_value**[ c ][ i ][ j ] specifies the j-th model value for the colour component c and the i-th intensity interval. The set of model values has different meaning depending on the value of fgr\_model\_id.

The value of fgr\_comp\_model\_value[ c ][ i ][ j ] is constrained as follows, and could be additionally constrained as specified elsewhere in this clause:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ j ] shall be in the range of 0 to 2fgBitDepth[ c ] − 1, inclusive.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ j ] shall be in the range of −2( fgBitDepth[ c ] − 1 ) to 2( fgBitDepth[ c ] − 1 ) − 1, inclusive.

Depending on the value of fgr\_model\_id, the synthesis of the film grain is modelled as follows:

– If fgr\_model\_id is equal to 0, a frequency filtering model enables simulating the original film grain for c = 0..2, x = 0..PicWidthInLumaSamples − 1, and y = 0..PicHeightInLumaSamples − 1 as specified by:

G[ c ][ x ][ y ] = ( fgr\_comp\_model\_value[ c ][ sj ][ 0 ] \* Q[ c ][ x ][ y ] + fgr\_comp\_model\_value[ c ][ sj ][ 5 ] \*  
 G[ c − 1 ][ x ][ y ] ) >> fgr\_log2\_scale\_factor (26)

where Q[ c ] is a two-dimensional random process generated by filtering 16x16 blocks gaussRv with random-valued elements gaussRvij generated with a normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance) and where the value of an element G[ c − 1 ][ x ][ y ] used in the right-hand side of the equation is inferred to be equal to 0 when c − 1 is less than 0.

NOTE 7 – A normalized Gaussian random variable can be generated from two independent, uniformly distributed random values over the interval from 0 to 1 (and not equal to 0), denoted as uRv0 and uRv1, using the Box-Muller transformation specified by:

gaussRvi,j = Sqrt( −2 \* Ln( uRv0 ) ) \* Cos( 2 \* π \* uRv1 ) (27)

where π is Archimedes' constant 3.141 592 653 589 793....

The band-pass filtering of blocks gaussRv can be performed in the discrete cosine transform (DCT) domain as follows:

for( y = 0; y < 16; y++ )  
 for( x = 0; x < 16; x++ )  
 if( ( x < fgr\_comp\_model\_value[ c ][ sj ][ 3 ]  &&  y < fgr\_comp\_model\_value[ c ][ sj ][ 4 ] )  | | (28)  
 x > fgr\_comp\_model\_value[ c ][ sj ][ 1 ]  | |  y > fgr\_comp\_model\_value[ c ][ sj ][ 2 ] )  
 gaussRv[ x ][ y ] = 0  
filteredRv = IDCT16x16( gaussRv )

where IDCT16x16( z ) refers to a unitary inverse discrete cosine transformation (IDCT) operating on a 16x16 matrix argument z as specified by:

IDCT16x16( z ) = r \* z \* rT (29)

where the superscript T indicates a matrix transposition and r is the 16x16 matrix with elements rij specified by:

(30)

where π is Archimedes' constant 3.141 592 653 589 793....

Q[ c ] is formed by the frequency-filtered blocks filteredRv.

NOTE 8 – Coded model values are based on blocks of size 16x16, but a decoder implementation could use other block sizes. For example, decoders implementing the IDCT on 8x8 blocks could down-convert by a factor of two the set of coded model values fgr\_comp\_model\_value[ c ][ sj ][ i ] for i equal to 1..4.

NOTE 9 – To reduce the degree of visible blocks that result from mosaicking the frequency-filtered blocks filteredRv, decoders could apply a low-pass filter to the boundaries between frequency-filtered blocks.

– Otherwise (fgr\_model\_id is equal to 1), an auto-regression model enables simulating the original film grain for c = 0..2, x = 0..PicWidthInLumaSamples − 1, and y = 0..PicHeightInLumaSamples − 1 as specified by:

G[ c ][ x ][ y ] = ( fgr\_comp\_model\_value[ c ][ sj ][ 0 ] \* n[ c ][ x][ y ] +  
 fgr\_comp\_model\_value[ c ][ sj ][ 1 ] \* ( G[ c ][ x − 1 ][ y ] +  
 ( ( fgr\_comp\_model\_value[ c ][ sj ][ 4 ] \* G[ c ][ x ][ y − 1 ] ) >>  
 fgr\_log2\_scale\_factor ) ) + fgr\_comp\_model\_value[ c ][ sj ][ 3 ] \*  
 ( ( fgr\_comp\_model\_value[ c ][ sj ][ 4 ] \* ( G[ c ][ x − 1 ][ y − 1 ] + G[ c ][ x + 1 ][ y − 1 ] ) ) >> (31)  
 fgr\_log2\_scale\_factor ) +  
 fgr\_comp\_model\_value[ c ][ sj ][ 5 ] \* ( G[ c ][ x − 2 ][ y ] +  
 ( ( fgr\_comp\_model\_value[ c ][ sj ][ 4 ] \* fgr\_comp\_model\_value[ c ][ sj ][ 4 ] \* G[ c ][ x ][ y − 2 ] ) >>  
 ( 2 \* fgr\_log2\_scale\_factor ) ) ) +  
 fgr\_ comp\_model\_value[ c ][ sj ][ 2 ] \* G[ c − 1 ][ x ][ y ] ) >> fgr\_log2\_scale\_factor

where n[ c ][ x ][ y ] is a random value with normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance for each value of c, x, and y) and where the value of an element G[ c ][ x ][ y ] used in the right-hand side of the equation is inferred to be equal to 0 when any of the following conditions are true:

– c is less than 0,

– x is less than 0,

– y is less than 0.

fgr\_comp\_model\_value[ c ][ i ][ 0 ] provides the first model value for the model as specified by fgr\_model\_id. fgr\_comp\_model\_value[ c ][ i ][ 0 ] corresponds to the standard deviation of the Gaussian noise term in the generation functions specified in Equations 26 through 31.

fgr\_comp\_model\_value[ c ][ i ][ 1 ] provides the second model value for the model as specified by fgr\_model\_id. When fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 1 ] shall be greater than or equal to 0 and less than 16.

When not present in the film grain regions characteristics SEI message, fgr\_comp\_model\_value[ c ][ i ][ 1 ] is inferred as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 1 ] is inferred to be equal to 8.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 1 ] is inferred to be equal to 0.

fgr\_comp\_model\_value[ c ][ i ][ 1 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 1 ] indicates the horizontal high cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 1 ] indicates the first order spatial correlation for neighbouring samples ( x − 1, y ) and ( x, y − 1 ).

fgr\_comp\_model\_value[ c ][ i ][ 2 ] provides the third model value for the model as specified by fgr\_model\_id. When fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 2 ] shall be greater than or equal to 0 and less than 16.

When not present in the film grain regions characteristics SEI message, fgr\_comp\_model\_value[ c ][ i ][ 2 ] is inferred as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 2 ] is inferred to be equal to fgr\_comp\_model\_value[ c ][ i ][ 1 ]

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 2 ] is inferred to be equal to 0.

fgr\_comp\_model\_value[ c ][ i ][ 2 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 2 ] indicates the vertical high cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 2 ] indicates the colour correlation between consecutive colour components.

fgr\_comp\_model\_value[ c ][ i ][ 3 ] provides the fourth model value for the model as specified by fgr\_model\_id. When fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 3 ] shall be greater than or equal to 0 and less than or equal to fgr\_comp\_model\_value[ c ][ i ][ 1 ].

When not present in the film grain regions characteristics SEI message, fgr\_comp\_model\_value[ c ][ i ][ 3 ] is inferred to be equal to 0.

fgr\_comp\_model\_value[ c ][ i ][ 3 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 3 ] indicates the horizontal low cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 3 ] indicates the first order spatial correlation for neighbouring samples ( x − 1, y − 1 ) and ( x + 1, y − 1 ).

fgr\_comp\_model\_value[ c ][ i ][ 4 ] provides the fifth model value for the model as specified by fgr\_model\_id. When fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 4 ] shall be greater than or equal to 0 and less than or equal to fgr\_comp\_model\_value[ c ][ i ][ 2 ].

When not present in the film grain regions characteristics SEI message, fgr\_comp\_model\_value[ c ][ i ][ 4 ] is inferred to be equal to fgr\_model\_id.

fgr\_comp\_model\_value[ c ][ i ][ 4 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 4 ] indicates the vertical low cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 4 ] indicates the aspect ratio of the modelled grain.

fgr\_comp\_model\_value[ c ][ i ][ 5 ] provides the sixth model value for the model as specified by fge\_model\_id.

When not present in the film grain regions characteristics SEI message, fge\_comp\_model\_value[ c ][ i ][ 5 ] is inferred to be equal to 0.

fgr\_comp\_model\_value[ c ][ i ][ 5 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 5 ] indicates the colour correlation between consecutive colour components.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 5 ] indicates the second order spatial correlation for neighbouring samples ( x, y − 2 ) and ( x − 2, y ).

**fgr\_region\_based\_adaptation\_flag** equal to 1 indicates that the spatial adaption of the film grain is defined per each region of the present SEI message and described by a rectangle. When not present in the film grain regions characteristics SEI message, fgr\_region\_based\_adaptation\_flag is inferred to be equal to 0.

**fgr\_active\_regions\_number\_minus1** plus 1 indicates the number of regions for which the film grain adaptation is specified. fgr\_active\_regions\_number\_minus1 shall be in the range of 0 to 255, inclusive.

**fgr\_region\_top**[ i ], **fgr\_region\_left**[ i ], **fgr\_region\_width**[ i ], and **fgr\_region\_height**[ i ] specify the coordinates of the top-left corner and the width and height, respectively, of the bounding box of the i-th region in the picture.

The value of fgr\_region\_left[ i ] shall be in the range of 0 to PicWidthInLumaSamples − 1, inclusive.

The value of fgr\_region\_top[ i ] shall be in the range of 0 to PicHeightInLumaSamples  − 1, inclusive.

The value of fgr\_region\_width[ i ] shall be in the range of 0 to PicWidthInLumaSamples  − fgr\_region\_left [ i ], inclusive.

The value of fgr\_region\_height[ i ] shall be in the range of 0 to PicHeightInLumaSamples  − fgr\_region\_top[ i ], inclusive.

**fgr\_film\_grain\_enabled\_flag**[ i ] equal to 1 indicates that the film grain characteristics using region intervals are applied on the i-th region. fge\_film\_grain\_enabled\_flag[ i ] equals to 0 indicates that no film grain synthesis is applied to the i-th region of the current picture.

**fgr\_region\_interval\_min**[ i ][ c ] and **fgr\_region\_interval\_max**[ i ][ c ] specify the range of interval indexes for color component c that are applicable to the i-th region. fgr\_region\_interval\_min[ i ][ c ] shall be in the range of 0 to fg\_num\_intensity\_intervals\_minus1[ c ], inclusive, and fgr\_region\_interval\_max[ i ][ c ] shall be in the range of fgr\_region\_interval\_min[ i ][ c ] to fg\_num\_intensity\_intervals\_minus1[ c ], inclusive.

NOTE – Overlapping regions are processed in the increasing order of region index i, which means that a sample belonging to the region with index i=N with min and max film grain intervals indexes equal to x and y, respectively, and also belonging to region with index i=M (with M>N) with min and max film grain intervals indexes equal to x’ and y’, respectively, is processed with film grain characteristics defined with min and max film grain intervals indexes equal to x’ and y’.

**fgr\_alpha\_channel\_adaptation\_flag** equal to 1 indicates that the decoded alpha planes coded in auxiliary pictures of type AUX\_ALPHA are applied to the simulated film grain values defined in the present SEI message prior to be applied are associated primary picture.

When an AU contains an SDI SEI message with sdi\_aux\_id[ i ] equal to 1 for at least one value of i, the SDI SEI message shall precede the FRGC SEI message in decoding order.

When an access unit contains an auxiliary picture picA in a layer, with nuh\_layer\_id equal to nuhLayerIdA, that is indicated as an alpha auxiliary layer by an SDI SEI message, the alpha channel sample values of picA persist in output order until one or more of the following conditions are true:

– The next picture, in output order, with nuh\_layer\_id equal to nuhLayerIdA is output.

– A CLVS containing the auxiliary picture picA ends.

– The bitstream ends.

– A CLVS of any associated primary layer of the auxiliary picture layer with nuh\_layer\_id equal to nuhLayerIdA ends.

If the current PU contains an SDI SEI message with sdi\_aux\_id[ i ] equal to 1 for at least one value of i, the simulated film grain values are weighted by the auxiliary data as follow:

– If fgr\_blending\_mode\_id is equal to 0, the blending mode is additive as specified by:

Igrain[ c ][ x ][ y ] = Clip3( 0, ( 1 << fgBitDepth[ c ] ) − 1, Î[ c ][ x ][ y ] + G[ c ][ x ][ y ]\* Iaux[ c ][ x ][ y ])

– Otherwise (fgr\_blending\_mode\_id is equal to 1), the blending mode is multiplicative as specified by:

Igrain[ c ][ x ][ y ] = Clip3( 0, ( 1 << fgBitDepth[ c ] ) − 1, Î[ c ][ x ][ y ] +  
 Round( ( Î[ c ][ x ][ y ] \* G[ c ][ x ][ y ]\* Iaux[ c ][ x ][ y ]) ) ÷ ( ( 1  <<  fgBitDepth[ c ] ) − 1 ) ) )

where Î[ c ][ x ][ y ] represents the sample value at coordinates x, y of the colour component c of the input image Î, G[ c ][ x ][ y ] is the simulated film grain value at the same position and colour component, and fgBitDepth[ c ], is the number of bits used for each sample in a fixed-length unsigned binary representation of the arrays Igrain[ c ][ x ][ y ], Î[ c ][ x ][ y ], and G[ c ][ x ][ y ], where c = 0..2, x = 0..PicWidthInLumaSamples − 1, and y = 0..PicHeightInLumaSamples – 1, and where Iaux[ c ][ x ][ y ] represents the sample value at coordinates x, y of the colour component c of the decoded auxiliary picture.

**fgr\_persistence\_flag** specifies the persistence of the film grain regions characteristics SEI message for the current layer.

fgr\_persistence\_flag equal to 0 specifies that the film grain regions characteristics SEI message applies to the current decoded picture only.

fgr\_persistence\_flag equal to 1 specifies that the film grain regions characteristics SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with a film grain regions characteristics SEI message is output that follows the current picture in output order.

*Addition(s) related to large SEI message and versatile SEI RBSP / versatile SEI message*

*[Ed: Proponents of both approaches are asked to work out possible combination]*

X.1 Image format metadata SEI messages

X.1.1 Large supplemental enhancement information message syntax

|  |  |
| --- | --- |
| lsei\_message( ) { | **Descriptor** |
| **lsei\_position** | u(2) |
| **lsei\_relevance** | u(2) |
| **lsei\_reserved** | u(4) |
| **lsei\_payload\_type\_byte** | u(8) |
| **lsei\_payload\_size\_16bits** | u(16) |
| lsei\_payload( lseiPayloadType, lseiPayloadSize ) |  |
| } |  |

X.1.2 Large supplemental enhancement information message semantics

Each large SEI message consists of the variables specifying the type payloadType and size payloadSize of the large SEI message payload. Large SEI message payloads are specified in Annex D. The derived large SEI message payload size payloadSize is specified in bytes and shall be equal to the number of RBSP bytes in the large SEI message payload.

NOTE – The NAL unit byte sequence containing the large SEI message might include one or more emulation prevention bytes (represented by emulation\_prevention\_three\_byte syntax elements). Since the payload size of a large SEI message is specified in RBSP bytes, the quantity of emulation prevention bytes is not included in the size payloadSize of a large SEI payload.

**lsei\_position** indicates if the SEI message corresponds to the PREFIX\_SEI\_NUT and SUFFIX\_SEI\_NUT. **lsei\_position** equal 0 indicates that the SEI message is treated as PREFIX\_SEI\_NUT. **lsei\_position** equal 1 indicates that the SEI message is treated as SUFFIX\_SEI\_NUT. Values 3 and 4 of **lsei\_position** are reserved for future use and shall be ignored.

**lsei\_relevance** indicates the relevance of the SEI message for the target application. **lsei\_relevance** ranges from 0 to 3, with 0 being the least relevant and 3 being the most relevant.

NOTE – The relevance of an SEI message is an arbitrary decision and its use is to be specified by the target application.

**lsei\_reserved** is reserved for future use and shall be ignored.

**lsei\_payload\_type\_byte** is a byte of the payload type of a large SEI message. payloadType = **lsei\_payload\_type\_byte.**

**payload\_size\_16bits** is the payload size in bits of a large SEI message. payloadSize = **payload\_size\_16bits.**

X.2 Versatile SEI RBSP

X.2.1 Versatile SEI RBSP syntax

|  |  |
| --- | --- |
| vsei\_rbsp( ) { | Descriptor |
| **vsei\_importance** | u(1) |
| **vsei\_reserved** | u(7) |
| versatile\_sei\_message() |  |
| rbsp\_trailing\_bits( ) |  |
| } |  |

X.2.2 Versatile SEI RBSP semantics

Versatile Supplemental enhancement information RBSP contains information that is not necessary to decode the samples of coded pictures from VCL NAL units. A VSEI RBSP contains one VSEI message.

**vsei\_importance** equal 1 indicates that the versatile SEI message may be an important or required. Vsei\_importance equal 0 indicates that the versatile SEI message does not have particular importance.

NOTE – Entity that is aware of this flag may use this information when it needs to make decision whether or not to deliver / drop the verstatile SEI message.

**vsei\_reserved** is reserved for future use and shall be ignored.

X.2.3 Versatile SEI message syntax

|  |  |
| --- | --- |
| versatile\_sei\_message( ) { | **Descriptor** |
| **vsei\_payload\_type\_byte** | u(8) |
| vsei\_payload( vseiPayloadType ) |  |
| } |  |

X.2.4 Versatile SEI message semantics

Versatile Supplemental enhancement information RBSP contains information that is not necessary to decode the samples of coded pictures from VCL NAL units. A VSEI RBSP contains one VSEI message.

Each versatile SEI message consists of the variables specifying the importance and type payloadType of the SEI message payload. Versatile SEI message payloads are specified in XXX.

NOTE – The NAL unit byte sequence containing the SEI message might include one or more emulation prevention bytes (represented by emulation\_prevention\_three\_byte syntax elements).

**vsei\_payload\_type\_byte** is a byte of the payload type of a versatile SEI message.

1. https://www.dublincore.org [↑](#footnote-ref-1)