|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extensions**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  11th Meeting: Geneva, CH, 12 – 18 February 2015 | Document: JCT3V-K0050\_v1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Proposed HLS Cleanup for 3D-HEVC** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Gerhard Tech  Karsten Sühring  Detlev Marpe | Email: | gerhard.tech@hhi.fraunhofer.de  karsten.suehring@hhi.fraunhofer.de  detlev.marpe@hhi.fraunhofer.de |
| *Source:* | Fraunhofer HHI | | |

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

# Abstract

Fixes for several issues of the 3D-HEVC HLS and for further alignment with MV-HEVC are proposed, which are in particular related to:

* Flexible camera parameter signalling
* Slice level indication of presence of pictures required as reference for inter-component prediction.
* Constraints on diagonal reference pictures required for residual prediction.

# Camera parameter signalling

## Current design

Several 3D-HEVC coding tools utilize depth information from depth layers for decoding of texture layers. For this, depth values of a depth layer of a current view are converted to disparities related to another view. Conversion is performed by scaling and offsetting. Scale and offsets are signalling in the VPS when constant for the video sequences. When scale and offsets vary over time, the signalling is performed in the slice headers.

Scale and offset depend on spatial position of the other view relative to the current view. For this, multiple scale and offset pairs are signalled for a current view, such that each pair is related to one of the views preceding the current view in coding order.

E.g., assuming that the bitstream includes views with view order indices 0, 2, 4, intention of the current design would be to signal the following shifts and offsets:

* for view 0: none
* for view 2: related to view 0
* for view 4: related to view 0 and related to view 2

In the current design parsing of scale and offsets is performed, e.g. in the VPS in two loops:

|  |  |
| --- | --- |
| vps\_3d\_extension( ) { | **Descriptor** |
| ... |  |
| for( n = 1; n < NumViews; n++ ) { |  |
| i = ViewOIdxList[ n ] |  |
| ... |  |
| for( m = 0; m < n; m++ ) { |  |
| j = ViewOIdxList[ m ] |  |
| **vps\_cp\_scale**[ i ][ j ] | se(v) |
| **vps\_cp\_off**[ i ][ j ] | se(v) |
| ... |  |
| } |  |
| } |  |
| **...** |  |
| } |  |

NumViews specifies the number of views (e.g., 3 in the above example) and ViewOIdxList maps the index m, which indicates the positions a pair in the bitstream, to a view order index (e.g. ViewOIdxList is [ 0 2 4 ], in the above example).

ViewOIdxList is derived in the VPS extension by iterating over all layers present in the bitstream and appending the view order index of a layer to the end of ViewOIdxList, when it is not yet present in ViewOIdxList.

Assuming, e.g. following VPS information:

i = 0; lId = 0; ViewOrderIdx[ lId ] = 0; VpsDepthFlag[ lId ] = 0  
 i = 1; lId = 1; ViewOrderIdx[ lId ] = 0; VpsDepthFlag[ lId ] = 1  
 i = 2; lId = 2; ViewOrderIdx[ lId ] = 2; VpsDepthFlag[ lId ] = 0  
 i = 3; lId = 3; ViewOrderIdx[ lId ] = 2; VpsDepthFlag[ lId ] = 1  
 i = 4; lId = 4; ViewOrderIdx[ lId ] = 4; VpsDepthFlag[ lId ] = 0  
 i = 5; lId = 5; ViewOrderIdx[ lId ] = 4; VpsDepthFlag[ lId ] = 1

would result in ViewOIdxList = { 0, 2, 4 }

Note that, since the current 3D-HEVC specification requires the depth layer of a view to directly follow the texture layer, the order of values in ViewOIdxList corresponds to the coding order of views, when texture and depth is present for all layers.

## Problem statement

**Issue 1: Auxiliary pictures**

At the Strasbourg meeting the 3D Main profile has been introduced. The 3D Main profile allows the presence of auxiliary picture layers interleaved with primary picture layers for backwards compatibility with MV-HEVC.

The following VPS information for a bitstream including a auxiliary layer

i = 0; lId = 0; ViewOrderIdx[ lId ] = 0; VpsDepthFlag[ lId ] = 0; AuxId[ lId ] = 0  
 i = 1; lId = 1; ViewOrderIdx[ lId ] = 4; VpsDepthFlag[ lId ] = 0; AuxId[ lId ] = 1  
 i = 2; lId = 2; ViewOrderIdx[ lId ] = 0; VpsDepthFlag[ lId ] = 1; AuxId[ lId ] = 0  
 i = 3; lId = 3; ViewOrderIdx[ lId ] = 2; VpsDepthFlag[ lId ] = 0; AuxId[ lId ] = 0  
 i = 4; lId = 4; ViewOrderIdx[ lId ] = 2; VpsDepthFlag[ lId ] = 1; AuxId[ lId ] = 0  
 i = 5; lId = 5; ViewOrderIdx[ lId ] = 4; VpsDepthFlag[ lId ] = 0; AuxId[ lId ] = 0  
 i = 6; lId = 6; ViewOrderIdx[ lId ] = 4; VpsDepthFlag[ lId ] = 1; AuxId[ lId ] = 0

would result in ViewOIdxList = { 0, 4, 2 } and hence to the presence of following scale and offsets:

* for view 0: none
* for view 4: related to view 0
* for view 2: related to view 0 and related to view 4

It can be seen that for view 4, shift and offset values to view 2 are missing, since order in ViewOIdxList does not correspond to the order of non-auxiliary layers.

**Issue 2:** Bitstream extraction / VPS rewriting

Scale and offset values can be signalled in the VPS or in the slice headers. However, the number of signalled scale and offset pairs depends currently on ViewOIdxList, which depends on the layers present in bitstream as signalled in the VPS. Thus, when layers are discarded from the bitstream (and the VPS is rewritten), the current design requires also rewriting of all slice segment headers, which is undesirable.

E.g., assuming that ViewOIdxList is equal to { 0, 2, 4 }, and that views 2 and 4 only depend on view 0, and that view 2 is discarded and the VPS is rewritten, also rewriting of all slice segments header of layers of view 4 would be required to remove camera parameters related to view 2.

**Issue 3:** Number of signalled camera

For increasing number of views, the number of signalled camera parameters for a particular view would increase with order N, although not all parameters would be required for a view. Considering cases that camera parameters can be signalled in the slice header this can result in an significant signalling overhead.

**Issue 4:** Availability of camera parameters

For some views camera parameters might not be present, such that no reasonable values can be signalled.

**Issue 5:** Camera parameters for dependent slices

Currently camera parameters are signaled for both dependent and independent slices. However, signalling for independent slices would be sufficient.

## Proposed fixes

To resolve issues are more flexible camera parameter is proposed as follows:

1. Signal the following in the VPS for each view (with ViewOrderIdx i) present in the bitstream:
   1. The number of related views num\_cp[ i ] for which scale and offset are present.
   2. For each m in the range of 0 to num\_cp[ i ] − 1, inclusive, a syntax element cp\_ref\_voi[ i ][ m ] specifying the ViewOrderIdx of the m-th view related to the view with ViewOrderIdx i.
   3. When it is indicated that scales and offsets for the view with ViewOrderIdx i are present in the VPS, for each m in the range of 0 to num\_cp[ i ] − 1, inclusive, a scale and offset values related to the view with ViewOrderIdx cp\_ref\_voi[ i ][ m ].
2. When it is indicated that scales and offsets for the view with ViewOrderIdx i are present in the slice header and the current slice header has ViewOrderIdx i, signal for each m in the range of 0 to num\_cp[ i ] − 1, inclusive, a scale and offset values related to the view with ViewOrderIdx cp\_ref\_voi[ i ][ m ].
3. Enable tools requiring camera parameters only, when camera parameters are present.

This more flexible signalling approach has following benefits:

* Issue 1, 3, and 4 are resolved, since related views can be explicitly specified.
* Issue 2 is resolved, since slice level signaling does not depend on ViewOIdxList anymore, but only on num\_cp and cp\_ref\_voi, which can kept unchanged for views included in the bitstream, when rewriting the VPS.

To resolve 5 it is proposed to not signal camera parameters in dependent slices, hence to move syntax elements behind the slice\_loop\_filter\_across\_slices\_enabled\_flag syntax element.

### Specification changes

|  |  |
| --- | --- |
| vps\_3d\_extension( ) { | **Descriptor** |
| **cp\_precision** | ue(v) |
| for( n = 1; n < NumViews; n++ ) { |  |
| i = ViewOIdxList[ n ] |  |
| **~~cp\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| ~~if( cp\_present\_flag[ i ] ) {~~ |  |
| **num\_cp**[ i ] | u(6) |
| if( num\_cp[ i ] > 0 ) { |  |
| **cp\_in\_slice\_segment\_header\_flag**[ i ] | u(1) |
| for( m = 0; m < num\_cp[ i ]; m++ ) { |  |
| **cp\_ref\_voi**[ i ][ m ] | ue(v) |
| if( !cp\_in\_slice\_segment\_header\_flag[ i ] ) { |  |
| ~~for( m = 0; m < n; m++ ) {~~ |  |
| ~~j = ViewOIdxList[ m ]~~ |  |
| j = cp\_ref\_voi[ i ][ m ] |  |
| **vps\_cp\_scale**[ i ][ j ] | se(v) |
| **vps\_cp\_off**[ i ][ j ] | se(v) |
| **vps\_cp\_inv\_scale\_plus\_scale**[ i ][ j ] | se(v) |
| **vps\_cp\_inv\_off\_plus\_off**[ i ][ j ] | se(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

**~~cp\_present\_flag~~**~~[ i ]~~~~equal to 1 specifies that the syntax elements vps\_cp\_scale[ i ][ j ], vps\_cp\_off[ i ][ j ], vps\_cp\_inv\_scale\_plus\_scale[ i ][ j ], and vps\_cp\_inv\_off\_plus\_off[ i ][ j ] are present in the VPS or that cp\_scale[ j ], cp\_off[ j ], cp\_inv\_scale\_plus\_scale[ j ], and cp\_inv\_off\_plus\_off[ j ] are present in slice segment headers with nuh\_layer\_id equal to layerId and ViewOrderIdx[ layerId ] equal to i. cp\_present\_flag[ i ]~~~~equal to 0 indicates that camera parameters are not present.~~

~~For layerId in the range of 0 to MaxLayersMinus1, inclusive, the variable cpRequiredFlag[ layerId ] is derived as specified in the following:~~

* ~~If one or more of the following conditions are true, cpRequiredFlag[ layerId ] is set equal to 1:~~
  + ~~A coded picture with nuh\_layer\_id equal to layerId and DepthRefinementFlag equal to 1 refers to the VPS.~~
  + ~~A coded picture with nuh\_layer\_id equal to layerId and ViewSynthesisPredFlag equal to 1 refers to the VPS~~
  + ~~A coded picture with nuh\_layer\_id equal to layerId, DepthFlag equal to 1 and IvMvPredFlag equal to 1 refers to the VPS.~~
* ~~Otherwise, cpRequiredFlag[ layerId ] is set equal to 0.~~

~~When, for any value of layerId, cpRequiredFlag[ layerId ] is equal to 1, the value of cp\_present\_flag[ ViewOrderIdx[ layerId ] ]~~~~shall be equal to 1. When not present, the value of cp\_present\_flag[ i ]~~~~is inferred to be equal to 0.~~

**num\_cp**[ i ] specifies the number of camera parameters signalled for the view with ViewIdx equal to i.

**cp\_ref\_voi**[ i ][ m ] specifies the ViewIdx of the view to which the m-th camera parameters signalled for the view with ViewIdx equal to i are related to. The value of cp\_ref\_voi[ i ][ m ] shall be in the range of 0 to 65535. It is a requirement of bitstream conformance that cp\_ref\_voi[ i ][ x ] is not equal to cp\_ref\_voi[ i ][ y ] for any value of x and y in the range of 0 to num\_cp[ i ] − 1.

For i and j in the range of 0 to NumViews − 1, inclusive, the variable CpPresentFlag[ i ][ j ] is set equal to 0 and modified as specified in the following:

for( n = 1; n < NumViews; n++ ) {  
 i = ViewOIdxList[ n ]  
 for( m = 0; m < num\_cp[ i ]; m++ )  
 CpPresentFlag[ i ][ cp\_ref\_voi[ i ][ m ] ] = 1  
 }

|  |  |
| --- | --- |
| slice\_segment\_header( ) { | **Descriptor** |
| .. |  |
| if( nuh\_layer\_id > 0 && cp\_in\_slice\_segment\_header\_flag[ ViewIdx ] ) |  |
| ~~for ( m = 0; ViewOIdxList[ m ] < ViewIdx; j++ ) {~~ |  |
| ~~j = ViewOIdxList[ m ]~~ |  |
| for( m = 0; m < num\_cp[ ViewIdx ]; m++ ) { |  |
| j = cp\_ref\_voi[ ViewIdx ][ m ] |  |
| **cp\_scale**[ j ] | se(v) |
| **cp\_off**[ j ] | se(v) |
| **cp\_inv\_scale\_plus\_scale**[ j ] | se(v) |
| **cp\_inv\_off\_plus\_off**[ j ] | se(v) |
| } |  |
| ... |  |
| } |  |

I.7.4.7.1 General slice segment header semantics

The specifications in clause F.7.4.7.1 apply with the following modifications and additions.

The variable DepthFlag is set equal to VpsDepthFlag[ nuh\_layer\_id ] and the variable ViewIdx is set equal to ViewOrderIdx[ nuh\_layer\_id ].

The variables depthOfRefViewsAvailFlag, ~~and~~  textOfCurViewAvailFlag, and cpAvailableFlag are set equal to 0 and modified as specified in the following:

* If DepthFlag is equal to 0, depthOfRefViewsAvailFlag and cpAvailableFlag ~~is~~ are set equal to 1 and the following applies:
  + For i in the range of 0 to NumRefListLayers[ nuh\_layer\_id ] − 1, inclusive, the following applies:
    - The variable refViewIdx is set equal to ViewOrderIdx[ IdRefListLayer[ nuh\_layer\_id][ i ] ].
    - When CpPresentFlag[ ViewIdx ][ refViewIdx ] is equal to 0, cpAvailableFlag is set equal to 0.
    - The variable curDepthAvailableFlag is set equal to 0.
    - For j in the range of 0 to MaxLayersMinus1, inclusive, the following applies:
      * When all of the following conditions are true, curDepthAvailableFlag is set equal to 1:
        + direct\_dependency\_flag[ LayerIdxInVps[ nuh\_layer\_id ] ][ j ] is equal to 1
        + VpsDepthFlag[ layer\_id\_in\_nuh[ j ] ] is equal to 1
        + ViewOrderIdx[ layer\_id\_in\_nuh[ j ] ] is equal to ViewOrderIdx[ IdRefListLayer[ nuh\_layer\_id][ i ] ]
        + DependencyId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
        + AuxId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
    - When curDepthAvailableFlag is equal to 0, depthOfRefViewsAvailFlag is set equal to 0.
* Otherwise (DepthFlag is equal to 1), the following applies:
  + For j in the range of 0 to MaxLayersMinus1, inclusive, the following applies:
    - When all of the following conditions are true, textOfCurViewAvailFlag is set equal to 1:
      * direct\_dependency\_flag[ LayerIdxInVps[ nuh\_layer\_id ] ][ j ] is equal to 1
      * VpsDepthFlag[  layer\_id\_in\_nuh[ j ] ] equal to 0
      * ViewOrderIdx[ layer\_id\_in\_nuh[ j ] ] equal to ViewIdx
      * DependencyId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
      * AuxId[ layer\_id\_in\_nuh[ j ] ] is equal to 0

The variables IvMvPredFlag, IvMvScalingFlag, SubPbSize, IvResPredFlag, ViewSynthesisPredFlag, DepthBasedBlkPartFlag, DepthRefinementFlag, MpiSubPbSize, IntraContourFlag, IntraSdcWedgeFlag, QtPredFlag, InterSdcFlag and DisparityDerivationFlag are derived as specified in the following:

* 1. IvMvPredFlag = ( nuh\_layer\_id > 0 ) &&  
      NumRefListLayers[ nuh\_layer\_id ] > 0 && iv\_mv\_pred\_flag[ DepthFlag ] (I−12)
  2. IvMvScalingFlag = ( nuh\_layer\_id > 0 ) && iv\_mv\_scaling\_flag[ DepthFlag ] (I−13)
  3. SubPbSize = 1 << ( nuh\_layer\_id > 0 ? log2\_sub\_pb\_size\_minus3[ DepthFlag ] + 3  : CtbLog2SizeY ) (I−14)
  4. IvResPredFlag = ( nuh\_layer\_id > 0 ) &&  
      NumRefListLayers[ nuh\_layer\_id ] > 0 && iv\_res\_pred\_flag[ DepthFlag ] (I−15)
  5. ViewSynthesisPredFlag = ( nuh\_layer\_id > 0 ) && NumRefListLayers[ nuh\_layer\_id ] > 0 &&  
      view\_synthesis\_pred\_flag[ DepthFlag ] && depthOfRefViewsAvailFlag && cpAvailableFlag (I−16)
  6. DepthBasedBlkPartFlag = ( nuh\_layer\_id > 0 ) &&  
      depth\_based\_blk\_part\_flag[ DepthFlag ] && depthOfRefViewsAvailFlag (I−17)
  7. DepthRefinementFlag = ( nuh\_layer\_id > 0 ) &&  
      depth\_refinement\_flag[ DepthFlag ] && depthOfRefViewsAvailFlag && cpAvailableFlag (I−18)
  8. MpiFlag = ( nuh\_layer\_id > 0 ) && mpi\_flag[ DepthFlag ] && textOfCurViewAvailFlag (I−19)
  9. MpiSubPbSize = 1 << ( log2\_mpi\_sub\_pb\_size\_minus3[ DepthFlag ] + 3 ) (I−20)
  10. IntraContourFlag = ( nuh\_layer\_id > 0 ) &&   
       intra\_contour\_flag[ DepthFlag ] && textOfCurViewAvailFlag (I−21)
  11. IntraSdcWedgeFlag = ( nuh\_layer\_id > 0 ) && intra\_sdc\_wedge\_flag[ DepthFlag ] (I−22)
  12. QtPredFlag = ( nuh\_layer\_id > 0 ) && qt\_pred\_flag[ DepthFlag ] && textOfCurViewAvailFlag (I−23)
  13. InterSdcFlag = ( nuh\_layer\_id > 0 ) && inter\_sdc\_flag[ DepthFlag ] (I−24)
  14. IntraSingleFlag = ( nuh\_layer\_id > 0 ) && intra\_single\_flag[ DepthFlag ] (I−25)
  15. DisparityDerivationFlag = IvMvPredFlag | | IvResPredFlag | |  
       ViewSynthesisPredFlag | | DepthBasedBlkPartFlag (I−25)

# Availability of reference pictures for inter component prediction

## Current design

When decoding a current picture several new 3D-HEVC tools refer to pictures, which a) belong to a component different from the component of the current picture; b) are included in the same AU; and are c) not in the reference picture list. More specifically:

1. When decoding a depth picture, MPI, QTL, and Intra\_Contour mode refer to the texture picture belonging to the same view as the depth picture.
2. When decoding a texture picture, VSP, Depth refinement and DBBP refer to the depth pictures belonging to the views of the texture reference pictures of the current texture picture.

In the current specification these tools are disabled, in case that required reference layers for inter component prediction are not present or available for inter-layer prediction in the bitstream. For this two flags are derived at slice level, and denoted as textOfCurViewAvailFlag (for depth coding) and depthOfRefViewsAvailFlag (for texture coding).

Note that, textOfCurViewAvailFlag and depthOfRefViewsAvailFlag only disable the corresponding tools, based on the presence of the layers required for inter component prediction in the bitstream and not based on the presence of a particular required inter-component reference picture from such a layer in the current AU.

## Problem statement

### Presence of pictures of layers for inter component prediction

The current 3D-HEVC design, as inherited from MV-HEVC, does not require pictures of all layers to be present in an AU. Hence, a picture of layer which is required for inter component prediction might not be present in a particular AU.

**Example**: The following table shows a scenario with half frame rate for layer 1. X marks the presence of a picture in a particular layer.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| POC | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |
| LId 0, View 0 Texture | X | X | X | X | X | X | X | X | X |  |
| LId 1, View 0 Depth | X |  | X |  | X |  | X |  | X |  |
| LId 2, View 1 Texture | X | X | X | X | X | X | X | X | X |  |
| LId 3, View 1 Depth | X | X | X | X | X | X | X | X | X |  |

In the example the VSP decoding process for the texture of view 1 would fail at odd POC values, since the depth pictures of view 0 are required, but not present.

### Temporal ids of pictures of layers for inter component prediction

In the MV-HEVC design inter-layer prediction is disabled for particular sub-layers based on VPS information (max\_tid\_il\_ref\_pics\_plus1, sub\_layers\_vps\_max\_minus1). The current inter-component prediction as carried out by 3D-HEVC does not take these constraints into account. Hence, semantics of inherited syntax elements are not correct in the scope of 3D-HEVC.

**Example 1**: The following table shows a scenario with different temporal sub layers, with numbers in the table indicating the temporal id of a particular picture.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| POC | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |
| LId 0, View 0, Texture, | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |
| LId 1, View 0 Depth | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |
| LId 2, View 1 Texture | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |
| LId 3,View 1 Depth | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |

In MV-HEVC, inter-layer prediction for from layer 0 to layer 2 can be disabled for pictures with temporal id equal to 1 by setting max\_tid\_il\_ref\_pics\_plus1[ 0 ][ 2 ] to 1. However, setting max\_tid\_il\_ref\_pics\_plus1[ 1 ][ 2 ] to 1 to disable inter-layer prediction from layer 1 to layer 2 (which is inter-component prediction as e.g. carried out by VSP) for pictures with temporal id equal to 1 would currently not have an effect, although semantics of max\_tid\_il\_ref\_pics\_plus1 specify this currently.

**Example 2**: The following table shows a scenario with different temporal sub layers and half frame rate for layer 1. Numbers in the table indicate the temporal id of a particular picture.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| POC | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |
| LId 0, View 0, Texture, | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |
| LId 1, View 0 Depth | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  |
| LId 2, View 1 Texture | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |
| LId 3, View 1 Depth | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |  |

In MV-HEVC, inter-layer prediction from layer 1 to layer 3, is disabled for pictures with temporal id equal to 1, since sub\_layers\_vps\_max\_minus1[ 1 ] is equal 0. However, for 3D-HEVC inter-layer prediction from layer 1 to layer 2 (which is inter-component prediction) is not disabled for pictures with temporal id equal to 1, although sub\_layers\_vps\_max\_minus1[ 1 ] is equal 0.

### Required reference layers for inter-component prediction of texture pictures

In the current specification a condition for enabling inter-component prediction from depth (as VSP, DBBP, Depth Refinement) is that depth layers of all views, whose texture layers might be used for inter-view prediction of the current texture layer, are available as reference layer of the current texture layer. Reason for this condition is that the disparity vector derived in NBDV might be related of one of these depth layers.

More specifically, in the current design the depth layers of views including the texture layers with nuh\_layer\_id values included in the list IdRefListLayer are required for enabling the inter-component prediction. IdRefListLayer is derived from the VPS and specifies the texture layers that might be used for inter-view prediction of any texture picture of the current layer.

However, at slice level there is a second list, denoted as RefPicLayerId, which is a subset of IdRefListLayer. RefPicLayerId includes only nuh\_layer\_id values of the texture layers that might actually be used for inter-view prediction of the current texture picture. For this, also the availability check of depth layers to determine whether inter-component prediction should be enabled can be narrowed at slice level by regarding layers from RefPicLayerId instead of layers from IdRefListLayer.

## Proposed solution 1

To resolve issue 2.2.1, the following is proposed:

* If all reference layers for inter component prediction are available in the bitstream and corresponding tools are enabled in the SPS, signal a flag in the slice header indicating the presence of all pictures required for inter component prediction in the current AU. Otherwise, infer this flag equal to 0.
* Require pictures of the reference layers for inter component prediction to be present in the DPB when the flag is equal to 1.
* Disable tools referring to one of the pictures required for inter component prediction, when the flag is equal to 0.

To resolve issue 2.2.2, it is proposed to constraint the availability of inter component prediction based on the temporal id of the current picture and the syntax elements sub\_layers\_vps\_max\_minus1 and max\_tid\_il\_ref\_pics\_plus1 in the same manner as done for inter layer prediction in MV-HEVC.

To resolve issue 2.2.3 it is proposed to replace IdRefListLayer by RefPicLayerId.

### Syntax and semantics

Changes related to 2.2.1 are marked red.

Changes related to 2.2.2 are marked blue.

Changes related to 2.2.3 are marked green.

**I.7.3.6.1 General slice segment header syntax**

|  |  |
| --- | --- |
| slice\_segment\_header( ) { | **Descriptor** |
| ... |  |
| **inter\_layer\_pred\_layer\_idc[**i ] | u(v) |
| } |  |
| } |  |
| if( inCompPredFlag ) |  |
| **in\_comp\_pred\_flag** | u(1) |
| if( sample\_adaptive\_offset\_enabled\_flag ) { |  |
| **slice\_sao\_luma\_flag** | u(1) |
| if( ChromaArrayType != 0 ) |  |
| ... |  |
| } |  |

General slice segment header semantics

The variable DepthFlag is set equal to VpsDepthFlag[ nuh\_layer\_id ] and the variable ViewIdx is set equal to ViewOrderIdx[ nuh\_layer\_id ].

The lists inCompRefViewIdcs[ i ], refCompLayerPresent[ i ] and the variable inCompPredFlag are derived as specified in the following:

* ~~The variables depthOfRefViewsAvailFlag and textOfCurViewAvailFlag are set equal to 0 and modified as specified in the following:~~
* If DepthFlag is equal to 0, ~~depthOfRefViewsAvailFlag is set equal to 1 and~~ the following applies:
  + For i in the range of 0 to NumActiveRefLayerPics~~NumRefListLayers[ nuh\_layer\_id ] NumRefListLayers[ nuh\_layer\_id ]~~ − 1, inclusive, the following applies:
    - The variable inCompRefViewIdcs[ i ] is set equal to ViewOrderIdx[ ~~RefPicLayerId[ nuh\_layer\_id]~~RefPicLayerId[ i ] ].
    - The variable refCompLayerPresentFlag[ i ] is set equal to 0.
    - ~~The variable curDepthAvailableFlag is set equal to 0.~~
    - For j in the range of 0 to MaxLayersMinus1, inclusive, the following applies:
      * ~~When all of the following conditions are true, curDepthAvailableFlag is set equal to 1:~~
      * When all of the following conditions are true, refCompLayerPresentFlag[ i ] is set equal to 1.
        + direct\_dependency\_flag[ LayerIdxInVps[ nuh\_layer\_id ] ][ j ] is equal to 1
        + VpsDepthFlag[ layer\_id\_in\_nuh[ j ] ] is equal to 1
        + ViewOrderIdx[ layer\_id\_in\_nuh[ j ] ] is equal to ViewOrderIdx[ IdRefListLayer[ nuh\_layer\_id][ i ] ]
        + DependencyId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
        + AuxId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
        + sub\_layers\_vps\_max\_minus1[ j ] is greater than or equal to TemporalId
        + TemporalId is equal to 0 or max\_tid\_il\_ref\_pics\_plus1[ j ][ LayerIdxInVps[ nuh\_layer\_id ] ] is greater than TemporalId
    - ~~When curDepthAvailableFlag is equal to 0, depthOfRefViewsAvailFlag is set equal to 0.~~
  + The variable depthOfRefViewsAvailFlag is derived as specified in the following:
    - If refCompLayerPresentFlag[ i ] is equal to 1, for all i the range of 0 to ~~NumRefListLayers[ nuh\_layer\_id ]~~NumActiveRefLayerPics − 1, inclusive, depthOfRefViewsAvailFlag is set equal to 1.
    - Otherwise, depthOfRefViewsAvailFlag is set equal to 0.
  + The variable inCompPredFlag is derived as specified in the following:
    - 1. inCompPredFlag = depthOfRefViewsAvailFlag && ( view\_synthesis\_pred\_flag[ DepthFlag ] | |   
          depth\_based\_blk\_part\_flag[ DepthFlag ] | | depth\_refinement\_flag[ DepthFlag ] ) (I−12)
* Otherwise (DepthFlag is equal to 1), the following applies:
  + The variable inCompRefViewIdcs[ 0 ] is set equal to ViewIdx.
  + The variable refCompLayerPresentFlag[ 0 ] is set equal to 0.
  + For j in the range of 0 to MaxLayersMinus1, inclusive, the following applies:
    - When all of the following conditions are true, refCompLayerPresentFlag[ 0 ] is set equal to 1.
    - ~~When all of the following conditions are true, textOfCurViewAvailFlag is set equal to 1:~~
      * direct\_dependency\_flag[ LayerIdxInVps[ nuh\_layer\_id ] ][ j ] is equal to 1
      * VpsDepthFlag[  layer\_id\_in\_nuh[ j ] ] equal to 0
      * ViewOrderIdx[ layer\_id\_in\_nuh[ j ] ] equal to ViewIdx
      * DependencyId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
      * AuxId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
      * sub\_layers\_vps\_max\_minus1[ j ] is greater than or equal to TemporalId
      * TemporalId is equal to 0 or max\_tid\_il\_ref\_pics\_plus1[ j ][ LayerIdxInVps[ nuh\_layer\_id ] ] is greater than TemporalId
  + The variable inCompPredFlag is derived as specified in the following:
    - 1. inCompPredFlag = refCompLayerPresentFlag[ 0 ] && ( intra\_contour\_flag[ DepthFlag ] | |   
          qt\_pred\_flag[ DepthFlag ] | | mpi\_flag[ DepthFlag ] ) ) (I−12)

**in\_comp\_pred\_flag** equal to 0 specifies that reference pictures required for inter-component prediction of the current picture might not be present and that inter-component prediction of the current picture is disabled. in\_comp\_pred\_flag[ i ] equal to 1 specifies all reference pictures required for inter-component prediction of the current picture are present and inter-component prediction of the current picture is enabled. When not present, in\_comp\_pred\_flag is inferred to be equal to 0.

When in\_comp\_pred\_flag is equal to 1, the following applies for i in the range of 0 to ( DepthFlag ? 0 : ~~NumRefListLayers[ nuh\_layer\_id ]~~NumActiveRefLayerPics − 1 ), inclusive:

* + It is a requirement of bitstream conformance that there is a picture in the DPB with PicOrderCntVal equal to the PicOrderCntVal of the current picture, and a nuh\_layer\_id value refNuhLayerId such that VpsDepthFlag[ refNuhLayerId ] not equal to DepthFlag, ViewOrderIdx[ refNuhLayerId ] is equal to inCompRefViewIdcs[ i ], DependencyId[ refNuhLayerId ] is equal to DependencyId[ nuh\_layer\_id ], and AuxId[ refNuhLayerId ] is equal to AuxId[ nuh\_layer\_id ].

The variables IvMvPredFlag, IvMvScalingFlag, SubPbSize, IvResPredFlag, ViewSynthesisPredFlag, DepthBasedBlkPartFlag, DepthRefinementFlag, MpiSubPbSize, IntraContourFlag, IntraSdcWedgeFlag, QtPredFlag, InterSdcFlag and DisparityDerivationFlag are derived as specified in the following:

* 1. ...
  2. ViewSynthesisPredFlag = ( nuh\_layer\_id > 0 ) && NumRefListLayers[ nuh\_layer\_id ] > 0 &&  
      view\_synthesis\_pred\_flag[ DepthFlag ] && in\_comp\_pred\_flag~~depthOfRefViewsAvailFlag~~ (I−16)
  3. DepthBasedBlkPartFlag = ( nuh\_layer\_id > 0 ) &&  
      depth\_based\_blk\_part\_flag[ DepthFlag ] && in\_comp\_pred\_flag~~depthOfRefViewsAvailFlag~~ (I−17)
  4. DepthRefinementFlag = ( nuh\_layer\_id > 0 ) &&  
      depth\_refinement\_flag[ DepthFlag ] && in\_comp\_pred\_flag~~depthOfRefViewsAvailFlag~~ (I−18)
  5. MpiFlag = ( nuh\_layer\_id > 0 ) && mpi\_flag[ DepthFlag ] && in\_comp\_pred\_flag~~textOfCurViewAvailFlag~~ (I−19)
  6. IntraContourFlag = ( nuh\_layer\_id > 0 ) &&   
      intra\_contour\_flag[ DepthFlag ] && in\_comp\_pred\_flag~~textOfCurViewAvailFlag~~ (I−21)
  7. QtPredFlag = ( nuh\_layer\_id > 0 ) && qt\_pred\_flag[ DepthFlag ] && in\_comp\_pred\_flag~~textOfCurViewAvailFlag~~ (I−23)

## Proposed solution 2

In solution 1, inter component is disabled for the whole picture when a reference layers or reference picture for inter-component prediction used by any CU in the slice is not available. However, in the case of prediction from depth to texture, some CUs would still be able to perform inter-component prediction when their respective inter-component reference picture is present. Thus, the following solution would be possible in such cases.:

* Signal a slice header flag for all reference layers for inter-component prediction that are available for the current slice for inter-component prediction.
* Infer the flag to zero for all reference layers for inter-component prediction that might be required, but are not available.
* Require pictures of the reference layers for inter-component prediction having the flag equal to 1 to be present in the DPB.
* Disable a tool using inter-component prediction at CU level, when the flag of its respective reference layer is equal to 0.

### Syntax and semantics

Changes related to 2.2.1 are marked red.

Changes related to 2.2.2 and 2.2.3 would be the same as for the first solution.

|  |  |
| --- | --- |
| slice\_segment\_header( ) { | **Descriptor** |
| ... |  |
| **inter\_layer\_pred\_layer\_idc[**i ] | u(v) |
| } |  |
| } |  |
| if( inCompPredFlag ) |  |
| for( i = 0; i < NumRefListLayers[ nuh\_layer\_id ]; i++ ) |  |
| if( refCompLayerPresentFlag[ i ] ) |  |
| **in\_comp\_pred\_flag**[ i ] | u(1) |
| if( sample\_adaptive\_offset\_enabled\_flag ) { |  |
| **slice\_sao\_luma\_flag** | u(1) |
| if( ChromaArrayType != 0 ) |  |
| ... |  |
| } |  |

**I.7.3.6.1 General slice segment header syntax**

The specifications in clause F.7.4.7.1 apply with the following modifications and additions.

The variable DepthFlag is set equal to VpsDepthFlag[ nuh\_layer\_id ] and the variable ViewIdx is set equal to ViewOrderIdx[ nuh\_layer\_id ].

~~The variables depthOfRefViewsAvailFlag and textOfCurViewAvailFlag are set equal to 0 and modified as specified in the following:~~

The lists inCompRefViewIdcs[ i ], refCompLayerPresent[ i ] and the variable inCompPredFlag are derived as specified in the following:

* If DepthFlag is equal to 0, ~~depthOfRefViewsAvailFlag is set equal to 1 and~~ the following applies:
  + For i in the range of 0 to NumRefListLayers[ nuh\_layer\_id ] − 1, inclusive, the following applies:
    - ~~The variable curDepthAvailableFlag is set equal to 0.~~
    - The variable inCompRefViewIdcs[ i ] is set equal to ViewOrderIdx[ RefPicLayerId[ nuh\_layer\_id][ i ] ].
    - The variable refCompLayerPresentFlag[ i ] is set equal to 0.
    - For j in the range of 0 to MaxLayersMinus1, inclusive, the following applies:
      * ~~When all of the following conditions are true, curDepthAvailableFlag is set equal to 1:~~
      * When all of the following conditions are true, refCompLayerPresentFlag[ i ] is set equal to 1.
        + direct\_dependency\_flag[ LayerIdxInVps[ nuh\_layer\_id ] ][ j ] is equal to 1
        + VpsDepthFlag[ layer\_id\_in\_nuh[ j ] ] is equal to 1
        + ViewOrderIdx[ layer\_id\_in\_nuh[ j ] ] is equal to ViewOrderIdx[ IdRefListLayer[ nuh\_layer\_id][ i ] ]
        + DependencyId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
        + AuxId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
  + ~~When curDepthAvailableFlag is equal to 0, depthOfRefViewsAvailFlag is set equal to 0.~~
  + The variable inCompPredFlag is derived as specified in the following:
    - 1. inCompPredFlag = ( view\_synthesis\_pred\_flag[ DepthFlag ] | |   
          depth\_based\_blk\_part\_flag[ DepthFlag ] | | depth\_refinement\_flag[ DepthFlag ] ) (I−12)
* Otherwise (DepthFlag is equal to 1), the following applies:
  + The variable inCompRefViewIdcs[ 0 ] is set equal to ViewIdx.
  + The variable refCompLayerPresentFlag[ 0 ] is set equal to 0.
  + For j in the range of 0 to MaxLayersMinus1, inclusive, the following applies:
    - When all of the following conditions are true, refCompLayerPresentFlag[ 0 ] is set equal to 1.
    - ~~When all of the following conditions are true, textOfCurViewAvailFlag is set equal to 1:~~
      * refCompLayerPresentFlag[ 0 ] is equal to 0.
      * direct\_dependency\_flag[ LayerIdxInVps[ nuh\_layer\_id ] ][ j ] is equal to 1
      * VpsDepthFlag[  layer\_id\_in\_nuh[ j ] ] equal to 0
      * ViewOrderIdx[ layer\_id\_in\_nuh[ j ] ] equal to ViewIdx
      * DependencyId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
      * AuxId[ layer\_id\_in\_nuh[ j ] ] is equal to 0
  + The variable inCompPredFlag is derived as specified in the following:
    1. inCompPredFlag = intra\_contour\_flag[ DepthFlag ] | |   
        qt\_pred\_flag[ DepthFlag ] | | mpi\_flag[ DepthFlag ] ) (I−12)

**in\_comp\_pred\_flag**[ i ] equal to 0 specifies that a reference picture of the view with ViewIdx equal to inCompRefViewIdcs[ i ] required for inter-component prediction of the current picture might not be present and that inter-component prediction of the current picture from a reference picture of the view with ViewIdx equal to inCompRefViewIdcs[ i ] is disabled. in\_comp\_pred\_flag[ i ] equal to 1 specifies a reference pictures of the view with ViewIdx equal to inCompRefViewIdcs[ i ] required for inter-component prediction of the current picture is present and inter-component prediction of the current picture from a reference picture of the view with ViewIdx equal to inCompRefViewIdcs[ i ] is enabled. When not present, in\_comp\_pred\_flag[ i ] is inferred to be equal to 0.

The following applies for i in the range of 0 to ( DepthFlag ? 0 : NumRefListLayers[ nuh\_layer\_id ] − 1 ), inclusive:

* When in\_comp\_pred\_flag[ i ] is equal to 1, it is a requirement of bitstream conformance that there is a picture in the DPB with PicOrderCntVal equal to the PicOrderCntVal of the current picture, and a nuh\_layer\_id value refNuhLayerId such that VpsDepthFlag[ refNuhLayerId ] not equal to DepthFlag, ViewOrderIdx[ refNuhLayerId ] is equal to inCompRefViewIdcs[ i ], DependencyId[ refNuhLayerId ] is equal to DependencyId[ nuh\_layer\_id ], and AuxId[ refNuhLayerId ] is equal to AuxId[ nuh\_layer\_id ].

The variables IvMvPredFlag, IvMvScalingFlag, SubPbSize, IvResPredFlag, ViewSynthesisPredFlag, DepthBasedBlkPartFlag, DepthRefinementFlag, MpiSubPbSize, IntraContourFlag, IntraSdcWedgeFlag, QtPredFlag, InterSdcFlag and DisparityDerivationFlag are derived as specified in the following:

* 1. ...
  2. ViewSynthesisPredFlag = ( nuh\_layer\_id > 0 ) && NumRefListLayers[ nuh\_layer\_id ] > 0 &&  
      view\_synthesis\_pred\_flag[ DepthFlag ] ~~&& depthOfRefViewsAvailFlag~~ (I−16)
  3. DepthBasedBlkPartFlag = ( nuh\_layer\_id > 0 ) && depth\_based\_blk\_part\_flag[ DepthFlag ] ~~&& depthOfRefViewsAvailFlag~~ (I−17)
  4. DepthRefinementFlag = ( nuh\_layer\_id > 0 ) && depth\_refinement\_flag[ DepthFlag ] ~~&& depthOfRefViewsAvailFlag~~ (I−18)
  5. MpiFlag = ( nuh\_layer\_id > 0 ) && mpi\_flag[ DepthFlag ] ~~&& textOfCurViewAvailFlag~~ (I−19)
  6. IntraContourFlag = ( nuh\_layer\_id > 0 ) && intra\_contour\_flag[ DepthFlag ] ~~&& textOfCurViewAvailFlag~~ (I−21)
  7. QtPredFlag = ( nuh\_layer\_id > 0 ) && qt\_pred\_flag[ DepthFlag ] ~~&& textOfCurViewAvailFlag~~  (I−23)
  8. ...

Let maxViewIdx be the maximum value of ViewIdx of all layers in the bitstream.

For v in the range of 0 to maxViewIdx − 1, inclusive, inCompPredEnabledFlag[ v ] is set equal to 0.

For i in the range of 0 to ( DepthFlag ? 0 : NumRefListLayers[ nuh\_layer\_id ] − 1 ), inclusive, when in\_comp\_pred\_flag[ i ] is equal to 1, inCompPredEnabledFlag[ inCompRefViewIdcs[ i ] ] is set equal to 1.

At CU level a particular tool using inter-component prediction from a particular reference picture of a view with ViewIdxA is disabled when one or more of the following conditions is true:

* + inCompPredEnabledFlag[ ViewIdxA ] is equal to 0.
  + the corresponding slice level tool enabling flag is equal to 0. (e.g., MpiFlag for MPI)

Disabling is either directly done by testing both conditions or indirectly by requiring a potential tool enabling flag parsed from the CU to be equal to 0.

# Availability of reference pictures for residual prediction

## Current design

The coding unit (CU) decoding process for residual prediction refers to three reference pictures. The pictures can be identified by a reference picture order count (POC), a reference view order index (VOI), the current POC, and the current VOI.

The *reference POC* is the POC of a picture which is a) included in the reference picture list *RefPicList* of the current picture; and b) included in the current layer. The *reference POC* is derived from syntax elements of the VPS and the slice header of the current picture and is constant for all CUs of the current picture.

The *reference VOI* is the VOI of a picture which is a) included in the reference picture list *RefPicList* of the current picture; and b) included in the current access unit (AU). It is derived from syntax elements of the VPS, the slice header, and the current CU and can thus vary for CUs of the current picture.

The three reference pictures referred by a CU are included in:

1. the current view and an AU with the *reference POC* (denoted as picture *A*)
2. the view with the *reference VOI* and the current AU (denoted as picture *V*)
3. the view with the *reference VOI* and the AU with the *reference POC* (denoted as picture *VA*)

When a *reference POC* and a *reference VOI* can be derived and all three reference pictures are available, residual prediction is enabled for the current CU.

The availability of picture *V* and picture *A* is implicitly guaranteed since:

* The *reference POC* and the *reference VOI* are selected among the POCs and VOIs, respectively, of pictures included in the reference picture list (*RefPicList*) of the current picture.
* *RefPicList* includes a subset of pictures included in the union of picture sets *RefPicSetLtCurr, RefPicSetStCurrBefore, RefPicSetStCurrAfter, RefPicSetInterLayer1, and RefPicSetInterLayer0*, which are derived from the slice header of the current picture and the VPS.
* Pictures included in *RefPicSetLtCurr, RefPicSetStCurrBefore, RefPicSetStCurrAfter, RefPicSetInterLayer1*, and *RefPicSetInterLayer0* are required (by a bitstream constraint) to be in the decoded picture buffer (DPB), when decoding the current picture.

Picture *VA* is available, when the decoded picture buffer (DPB) includes a picture which is marked as "used for reference" and has the *reference POC* and the *reference VOI*. Thus, whether the decoding process for residual prediction is invoked for a current CU depends on the state of the DPB.

Note that, although not explicitly tested in the decoding process, the picture *VA* is available when both of the following conditions are fulfilled:

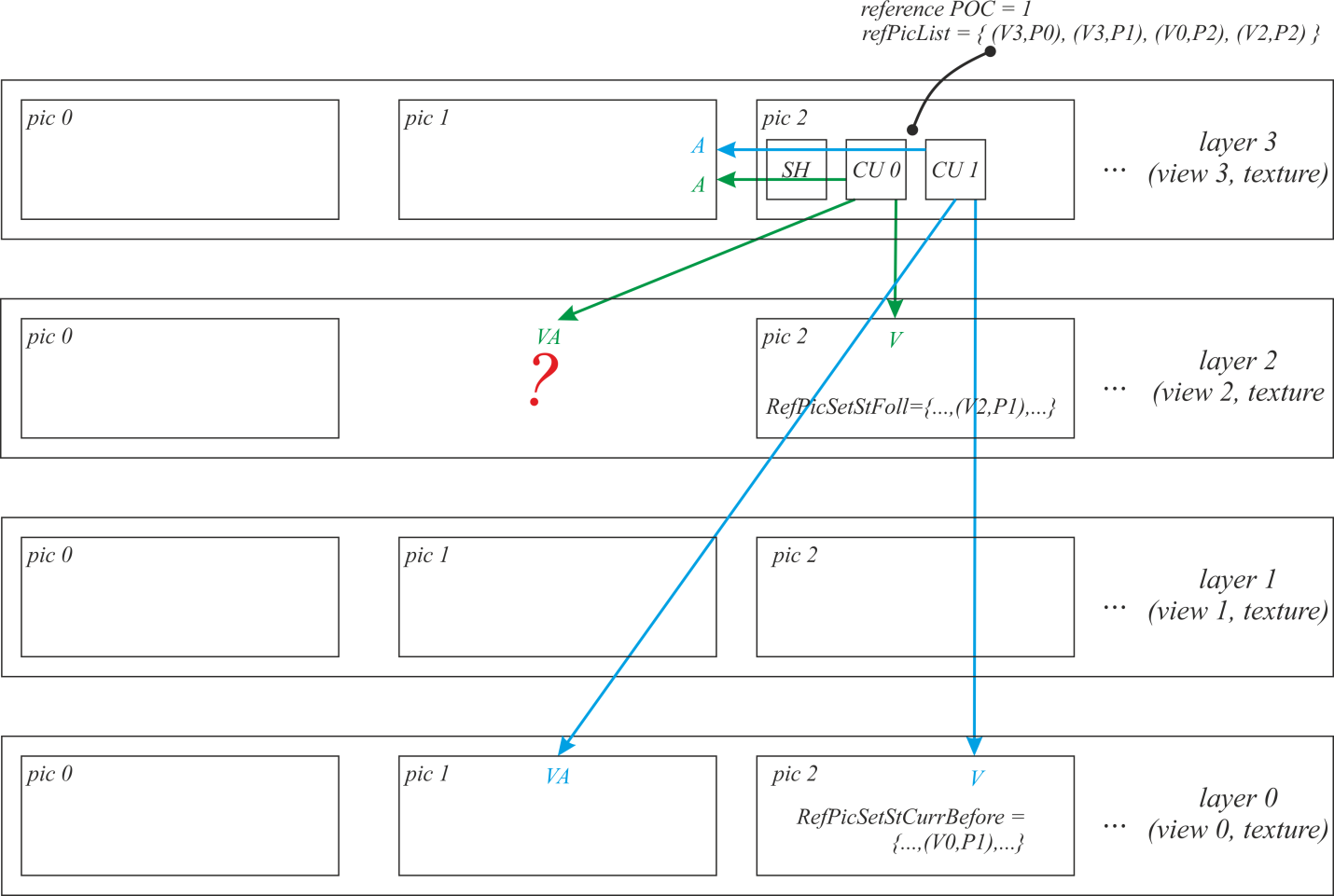
1. The bitstream contains a picture with *reference POC* and the *reference VOI.*
2. The syntax elements of the picture *V* indicate that the picture *VA* might be used by as reference by one of the following:
   1. Picture *V*. This is indicated, when picture *VA* is included in one of the sets *RefPicSetLtCurr, RefPicSetStCurrBefore,* or *RefPicSetStCurrAfter* derived from the VPS and the slice header of picture *V*.
   2. Pictures following the *V* picture in the view with the *reference* *VOI* in coding order. This is indicated when picture *VA*, is included in *RefPicSetLtFoll* or *RefPicSetStFoll* derived from the VPS and the slice header of picture *V*.

## Problem statement

Problem of the current residual prediction decoding process is the dependency on the presence of a picture with the reference POC and VOI (picture *VA*) in the DPB. Such a dependency on the state of the DPB is in general avoided in HEVC specification by design choice. A drawback of the design is that a loss of picture *VA* is not necessarily detected when decoding the current picture.

Thus, when decoding a current CU and the picture VA is not present in the DPB, it is not clear whether it is lost or intentionally not present.

**Example:** Picture 1 in view 2 has been lost. Thus, condition 1 (as provided above) is not fulfilled and picture *VA* for CU 0 is not in the DPB, such that residual prediction is disabled for CU 0. However, the erroneous decoding cannot be detected when decoding the current picture, since picture 1 in view 2 is not required to be present. Note that, the presence of picture 1 of view 2 in *RefPicSetStFoll* or *RefPicSetLtFoll* of picture 2 in view 2 does not require its presence in the bitstream or in the DPB.



**Figure 1: Example for residual prediction in case of picture loss.**

## Solution

Basic idea to allow a direct detection of a loss of picture *VA* is to enable residual prediction only, when picture *VA* it is included in *RefPicSetLtCurr, RefPicSetStCurrBefore,* or *RefPicSetStCurrAfter* of picture *V*. It is a bitstream requirement, that pictures included in these sets are present in the DPB, when decoding picture *V*. Their presence is conventionally checked when decoding picture *V.* Thus, the loss of the picture *VA* would be detected already when decoding picture *V*, hence before decoding the current picture. Note that, between decoding of picture *V* and decoding the current picture, pictures included in *RefPicSetLtCurr, RefPicSetStCurrBefore,* or *RefPicSetStCurrAfter* of picture *V* are not removed from the DPB. Thus, the presence of picture *VA* when decoding picture *V* is sufficient to guarantee presence of picture *VA* in DPB when decoding the current picture.

Note, moreover that

* cache misses or additional picture buffering might be reduced, since pictures of the layer including picture *V*, which cannot be used to decode picture *V* cannot be used for residual prediction of the current picture.
* in common use cases (and CTC) the temporal prediction structures of texture layers are aligned, such that the additional restriction to the "*Curr*" picture sets does not impose a coding penalty.
* in other unaligned scenarios it would be possible to include the additional picture *VA* to the "*Curr*" set of picture *V* if desired.

## Specification changes

I.8.3.9 Derivation process for the target reference index for residual prediction

This process is invoked when the current slice is a P or B slice.

Let currPic be the current picture.

The variables RpRefIdxL0 and RpRefIdxL1 are set equal to −1, and the variables RpRefPicAvailFlagL0 and RpRefPicAvailFlagL1 are set equal to 0.

The following applies for X in the range of 0 to 1, inclusive:

* When X is equal to 0 or the current slice is a B slice the following applies:
  + The variable pocDiff is set equal to 215 − 1.
  + For i in the range of 0 to num\_ref\_idx\_lX\_active\_minus1, inclusive, the following applies:
    - The variable currPocDiff is set equal to Abs( PicOrderCnt( RefPicListX[ i ] ) − PicOrderCntVal ).
    - When currPocDiff is not equal to 0 and currPocDiff is less than pocDiff, the following applies:
      * 1. pocDiff = currPocDiff (I−52)
        2. RpRefIdxLX = i (I−53)
        3. RpRefPicAvailFlagLX = 1 (I−54)

1. The variable RpRefPicAvailFlag is derived as specified in the following:
   1. RpRefPicAvailFlag = ( RpRefPicAvailFlagL0  | |  RpRefPicAvailFlagL1 )  &&  DispAvailFlag (I−55)
2. When RpRefPicAvailFlag is equal to 1, the following applies for X in the range of 0 to 1, inclusive:

* For refViewOrderIdx in the range of 0 to MaxLayersMinus1, inclusive, the following applies:
  + The variable RefRpRefAvailFlagLX[ refViewOrderIdx ] is set equal to 0.
* When X is equal to 0 or the current slice is a B slice the following applies:
  + For i in the range of 0 to NumActiveRefLayerPics − 1, inclusive, the following applies:
    - ~~The variable refViewOrderIdx is set equal to ViewOrderIdx( RefPicLayerId[ i ] ).~~
    - Let picV the picture in the current AU with nuh\_layer\_id equal to RefPicLayerId[ i ].
    - When RpRefPicAvailFlagLX is equal to 1 and there is a picture picA in ~~the DPB~~ one of the reference picture sets RefPicSetLtCurr, RefPicSetStCurrBefore, RefPicSetStCurrAfter of picV with PicOrderCnt( picA ) equal to PicOrderCnt( RefPicListX[ RpRefIdxLX ] ), ~~ViewIdx( picA ) equal to refViewOrderIdx, OtherDimsEqualFlag( currPic, picA, DIM\_VIEW ) equal to 1, and marked as "used for reference"~~, RefRpRefAvailFlagLX[ refViewOrderIdx ] is set equal to 1.

When RpRefPicAvailFlag is equal to 1 and RefRpRefAvailFlagL0[ refViewOrderIdx ] is equal to 1 for any refViewOrderIdx in the range of 0 to MaxLayersMinus1, inclusive, it is a requirement of bitstream conformance that PicOrderCnt( RefPicList0[ RpRefIdxL0 ] ) shall be the same for all slices of a coded picture.

When RpRefPicAvailFlag is equal to 1 and RefRpRefAvailFlagL1[ refViewOrderIdx ] is equal to 1 for any refViewOrderIdx in the range of 0 to MaxLayersMinus1, inclusive, it is a requirement of bitstream conformance that PicOrderCnt( RefPicList1[ RpRefIdxL1 ] ) shall be the same for all slices of a coded picture.

# Patent rights declaration(s)

Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V. may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).