|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  14th Meeting: Vienna, AT, 25 July – 2 Aug. 2013 | Document: JCTVC-N0216 |

|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extensions**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  5th Meeting: Vienna, AT, 27 July – 2 Aug. 2013 | Document: JCT3V-E0096 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **On reference picture list modification** | | |
| *Status:* | Input Document to JCT-VC and JCT-3V | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Yong He, Xiaoyu Xiu, Yan Ye 9710 Scranton R-D, #250  San Diego, CA 92121  USA | Tel: Email: | 1-858-210-4807 Yong.He@InterDigital.com |
| *Source:* | InterDigital Communications, Inc. | | |

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

# Abstract

It is proposed to signal the position of inter-layer reference picture set in reference picture list modification (RPLM) to save the bits. In current common test condition, reference picture list modification is invoked to allocate inter-layer reference picture (ILRP) to the end of the active reference lists when the number of active reference pictures is less than the total number of available reference pictures. To avoid signaling both temporal and ILRP reference picture one by one within current list modification framework, extra syntax elements in RPLM are proposed to signal the position of inter-layer reference picture set directly. The average bit saving of RPLM is around 57% for SHVC RA cases.

# Introduction

In the 13th JCTVC meeting, a number of contributions ([1][2][3][4][5][6]) were proposed to modify reference picture order of the reference picture list (RPL) or reference picture set (RPS) initialization so as to save bits used for reference picture list modification. For current common test conditions (CTC) test cases, reference list modifications (RPLM) are invoked to relocate the position of the inter-layer reference picture (ILRP) to the end of the lists when the number of active reference pictures is less than the number of available reference pictures. The relative positions of temporal reference pictures are not changed in those test cases.

In this contribution, we propose to add extra syntax elements in RPLM to achieve efficient signalling of inter-layer or inter-view reference picture positions in the RPL.

# Proposed syntax and decoding process

Table 1 shows the proposed RPLM syntax table.

#### Reference picture list modification syntax

|  |  |
| --- | --- |
| ref\_pic\_lists\_modification( ) { | Descriptor |
| **ref\_pic\_list\_modification\_flag\_l0** | u(1) |
| if( ref\_pic\_list\_modification\_flag\_l0 ) |  |
| if (nuh\_layer\_id > 0 && InterRefEnabledInRPLFlag) { |  |
| **ilrps\_l0\_modification\_only\_flag** | u(1) |
| if (ilrps\_l0\_modification\_only\_flag) |  |
| **list\_l0\_ilrps\_position** | u(v) |
| } |  |
| if (ilrp\_l0\_modification\_only\_flag == 0) |  |
| for( i = 0; i <= num\_ref\_idx\_l0\_active\_minus1; i++ ) |  |
| **list\_entry\_l0**[ i ] | u(v) |
| if( slice\_type = = B ) { |  |
| **ref\_pic\_list\_modification\_flag\_l1** | u(1) |
| if( ref\_pic\_list\_modification\_flag\_l1 ) |  |
| if (nuh\_layer\_id > 0 && InterRefEnabledInRPLFlag) { |  |
| **ilrps\_l1\_modification\_only\_flag** | u(1) |
| if (ilrps\_l1\_modification\_only\_flag) |  |
| **list\_l1\_ilrps\_position** | u(v) |
| } |  |
| if (ilrps\_l1\_modification\_only\_flag == 0) |  |
| for( i = 0; i <= num\_ref\_idx\_l1\_active\_minus1; i++ ) |  |
| **list\_entry\_l1**[ i ] | u(v) |
| } |  |
| } |  |

**ilrps\_l0\_modification\_only\_flag** equal to 1 indicates that refernce picture list 0 is specified explicitly by list\_l0\_ilrps\_position values. ilrps\_l0\_modification\_only\_flag equal to 0 indicates that reference picture list 0 is specified explicitly by a list of list\_entry\_l0[ i ] values. When ilrps\_l0\_modification\_only\_flag is not present in the slice header, it is inferred to be equal to 0.

**list\_l0\_ilrps\_position** specifies the beginning position of the inter-layer reference picture set in refPicListTemp0. The length of the list\_l0\_ilrps\_position syntax element is Ceil(Log2(num\_ref\_idx\_l0\_active\_minus1+1)) bits. The value of list\_l0\_ilrps\_position shall be in the range of 0 to num\_ref\_idx\_l0\_active\_minus1, inclusive. When the syntax element list\_l0\_ilrps\_position is not present in the slice header, it is inferred to be equal to 0.

**ilrps\_l1\_modification\_only\_flag** equal to 1 indicates that refernce picture list 1 is specified explicitly by list\_l1\_ilrps\_position values. ilrps\_l1\_modification\_only\_flag equal to 0 indicates that reference picture list 0 is specified explicitly by a list of list\_entry\_l1[ i ] values. When ilrps\_l1\_modification\_only\_flag is not present in the slice header, it is inferred to be equal to 0.

**list\_l1\_ilrps\_position** specifies the beginning position of the inter-layer reference picture set in refPicListTemp0. The length of the list\_l1\_ilrps\_position syntax element is Ceil(Log2(num\_ref\_idx\_l1\_active\_minus1+1)) bits. The value of list\_l1\_ilrps\_position shall be in the range of 0 to num\_ref\_idx\_l1\_active\_minus1, inclusive. When the syntax element list\_l1\_ilrps\_position is not present in the slice header, it is inferred to be equal to 0.

The decoding process for reference picture lists construction can be modified as follows:

The variable NumRpsCurrTempList0 is set equal to Max( num\_ref\_idx\_l0\_active\_minus1 + 1, NumPocTotalCurr ) and the list RefPicListTemp0 is constructed as follows:

rIdx = 0  
while( rIdx < NumRpsCurrTempList0 –NumActiveRefLayerPics ) {  
 if( InterRefEnabledInRPLFlag ) {  
 for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetStCurrBefore[ i ]  
 for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList0; rIdx++, i++ ) (‑53)   
 RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i ]  
 for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i ]  
 }

}

if (ilrps\_l0\_modification\_only\_flag)

{

for ( i = NumRpsCurrTempList0 – 1; i >= list\_l0\_ilrps\_position + NumActiveRefLayerPics; i --)

RefPicListTemp0[i] = RefPicListTemp0[i –NumActiveRefLayerPics ]

for ( i = 0; i < NumActiveRefLayerPics; i++ )

RefPicListTemp0[list\_l0\_ilrps\_position + i] = RefPicSetInterLayer[i]

}

The list RefPicList0 is constructed as follows:

for( rIdx = 0; rIdx <= num\_ref\_idx\_l0\_active\_minus1; rIdx++) (‑54)  
 RefPicList0[ rIdx ] = ref\_pic\_list\_modification\_flag\_l0 ? RefPicListTemp0[ list\_entry\_l0[ rIdx ] ] :  
 RefPicListTemp0[ rIdx ]

When the slice is a B slice, the variable NumRpsCurrTempList1 is set equal to Max( num\_ref\_idx\_l1\_active\_minus1 + 1, NumPocTotalCurr ) and the list RefPicListTemp1 is constructed as follows:

rIdx = 0  
while( rIdx < NumRpsCurrTempList1 –NumActiveRefLayerPics ) {  
 if( InterRefEnabledInRPLFlag ) {  
 for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetStCurrAfter[ i ]  
 for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ ) (‑55)  
 RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i ]  
 for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i ]  
 }  
   
   
}

if (ilrps\_l1\_modification\_only\_flag)

{

for ( i = NumRpsCurrTempList1 – 1; i >= list\_l1\_ilrps\_position + NumActiveRefLayerPics; i --)

RefPicListTemp1[i] = RefPicListTemp1[i –NumActiveRefLayerPics ]

for ( i = 0; i < NumActiveRefLayerPics; i++ )

RefPicListTemp1[list\_l1\_ilrps\_position + i] = RefPicSetInterLayer[i]

}

When the slice is a B slice, the list RefPicList1 is constructed as follows:

for( rIdx = 0; rIdx <= num\_ref\_idx\_l1\_active\_minus1; rIdx++) (‑56)  
 RefPicList1[ rIdx ] = ref\_pic\_list\_modification\_flag\_l1 ? RefPicListTemp1[ list\_entry\_l1[ rIdx ] ] :  
 RefPicListTemp1[ rIdx ]

# Results

The results for the bit saving on enhancement layer reference picture list modification (RPLM) with proposed method are presented in Table 2. The anchor is SHM2.0 software with existing RPLM method under common test condition. The bit count includes all bits of RPLM syntax structure. The overall bit saving is around 57% in CTC RA cases.

#### RPLM Bits comparison (RA)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sequences | Intra Period | Total frames | Anchor (bit) | N0216 (bit) | Bit saving percentage |
| Kimono1  ParkScene | 24 | 240 | 3832 | 1654 | 56.84% |
| PeopleOnStreet  Traffic | 32 | 150 | 2394 | 1032 | 56.89% |
| BasketballDrive  Cactus | 48 | 500 | 8214 | 3510 | 57.27% |
| BQTerrace | 64 | 600 | 9952 | 4264 | 57.15% |

# Reference

1. JCTVC-M0081, AHG9: On initialization process of reference picture lists for HEVC extensions, April 2013
2. JCTVC-M0104, AHG 9: On order of inter-layer reference pictures in reference picture lists, April 2013
3. JCTVC-M0105, AHG 9: On position of inter-layer reference pictures in reference picture lists, April 2013
4. JCTVC-M0148, Signaling inter-layer reference picture positions, April 2013
5. JCTVC-M0150, Inter-layer reference picture set initialization, April 2013
6. JCTVC3V-D0198, AHG7: Inter-layer reference pictures in reference picture list initialization, April 2013

# Patent rights declaration(s)

**InterDigital Communications, Inc. 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).**