|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  8th Meeting: San José, CA, USA, 1–10 February, 2012 | Document: JCTVC-H0321-r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG18: Resolution Adaptation Coding (ARC) using single resolution in DPB** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Philippe Bordes, Pierre Andrivon 1, av. De Belle-Fontaine – CS17616 35576 Cesson-Sévigné Cedex - France | Tel: Email: | +33-2-99-27-32-42 [philippe.bordes@technicolor.com](mailto:philippe.bordes@technicolor.com) |
| *Source:* | Technicolor | | |

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

# Abstract

This contribution presents a method for implementing Adaptive Resolution Coding (ARC) using only one single reference pictures resolution for prediction. It is reported the encoding gains are between 0.1% to 1.8% in Luma and up to 4.9% in Chroma on average depending on configuration cases and scenarios (Low-to-High or High-to-Low). The DPB memory size is reduced at the decoder side, compared to previous ARC proposals.

At the encoder side, both resolutions are still necessary to compute motion estimation only.

# Introduction: problem statement

The concept of Adaptive Resolution Change (ARC) in HEVC – adapting resolution dynamically in response to circumstances, without inserting IDR – has been introduced in the context of HEVC by JCTVC-F158 [1]. One added value of the approach is to not insert IDR at resolution change point and to continue temporal predicting across the resolution change. Further studies reported potential gains between 3% to 30% depending on the original frame size and configuration, and using one empirical resolution selection method [2]. However, in the proposed implementation, both resolutions (low and high) are stored in the DPB for every reference pictures (Figure 1). This increases the decoder memory amount requirements. The prediction is built using the reference picture with same resolution as the current frame.

This contribution is intended to cope with this limitation and to answer one mandate of AHG 18 on Resolution Adaption [3]:

* *Determine the necessity of keeping both resolutions in buffer and suitable limitations on DPB for managing complexity.*

With this proposal, only one single resolution of the reconstructed frames is stored in the DPB for the decoder. There is no additional down-sampling (or up-sampling) of reconstructed reference frames at the decoder side. Both resolutions are still necessary at the encoder side for motion estimation.



Figure : In previous ARC implementation, the DPB contains both the reconstructed pictures marked as reference (in white) and the corresponding down-sampled (resp. up-sampled) resolutions (in grey).

# Approach

In HEVC, the interpolation filters used to build the Prediction with motion compensation supports fractional values (fracx,fracy) of motion vectors (1/4 pel precision for Luma and 1/8 pel precision for Chroma) (Figure 2). They are separable filters: the Horizontal filter first MCIH(fracx) and the Vertical filter next MCIV(fracy).



Figure : The position (fracx,fracy) of the interpolated samples (in grey) with respect to the original samples in the reference picture (in black).

In ARC proposals, a set of filters are used for up or down scaling the reference pictures that will be needed across a resolution switch point. They are Horizontal SCH(θx) and Vertical SCV(θy) separable filters too. The parameters (θx, θy) are the Horizontal and Vertical phase, that is the relative position of the pixel to interpolate in the original picture grid (Figure 3). These filters are applied on the reconstructed pictures in order to store in the DPB reference pictures at all possible resolutions.



Figure : The position (θx, θy) of the pixels to interpolate (in grey) in the original grid (in black).

Then the whole process for building Inter prediction is equivalent to the convolution of these 4 filters (Figure 4a):

Pred(xcur,ycur) = [ MCIH(fracx) o MCIV(fracy) o SCV(θy) o SCH(θx) ] (refPic(xref,yref)) (1)

One property of separable filters is they can be permuted. Then (1) is equivalent to (2):

Pred(xcur,ycur) = [MCIV(fracy) o SCV(θy) o MCIH(fracx) o SCH(θx) ] (refPic(xref,yref)) (2)

MCIV(fracy) o SCV(θy) and MCIH(fracx) o SCH(θx) are equivalent to two separable filters GV(fracy,θy) and GH(fracx,θx) as depicted in Figure 4b.



Figure : The convolution of 4 separable filters is equivalent to 2 separable (Horizontal and Vertical) filters.

For each sample values (xcur,ycur) of the Prediction Unit (PU), we determine the corresponding position in the reference picture grid (Pxref,Pyref) (in grey in Figure 3) with (4). Next, we compute θx,θy using (6).

The parameters (fracx,fracy) are computed with (3) as usual in HEVC. In HM5.0, at the Inter prediction stage, interpolation filters are replaced with the GH(fracx,θx) and GV(fracy,θy) filters operating directly on the samples (xref(xcur),yref(ycur)) of the reconstructed reference pictures (5).

(3)

(4)

(5)

(6)

In Figure 5 is depicted the computation of the Prediction in the case of Low-to-High resolution change (reference is Low resolution and current is High resolution). In the first step, GH(fracx,θxu) and GH(fracx,θxv) are applied to compute temporary samples *u* and *v* from sample *s* in reference. Next, GV(fracy,θyu) and GV(fracy,θyv) are applied with (*u,v*) to compute (*pa,pb,pc,pd*).



Figure : Computation of the PU in case of Low-to-High resolution change.

# Results

This method has been implemented into the ARC-HM5.0 software provided in AHG18.

The reported method (single reconstructed resolution pictures in the DPB) is compared with previous one (ARC-HM5.0 : both resolutions in DPB). Two resolution change scenarios are tested: High-to-Low and Low-to-High Resolution Change. Since the prediction of Low (or High) resolution frames from High (resp. Low) resolution pictures has been modified only, we consider the portion just around the resolution change point.

The first 4 coded pictures are High (normal) resolution (resp. Low), the next ones are Low resolution (reps. High). In LP and LB configurations, 9 frames are encoded (Figure 6). In RA configuration 16 frames are encoded (Figure 7).



Figure : In LP and LB configuration, 9 first frames are coded (High-to-Low scenario).



Figure : In RA configuration, 16 first frames are coded (High-to-Low scenario).

The detailed results are provided in separate excel files:

For LowToHigh resolution change, I gathered separately the first part (part1) and the second part (part2) of the bit-streams:

* **HM5.0\_ARC\_Filters\_LowToHigh\_part1.xls**: Low-to-High resolution change first part (part1) of the bit-streams.
* **HM5.0\_ARC\_Filters\_LowToHigh\_part2.xls**: Low-to-High resolution change second part (part2) of the bit-streams.

And for HighToLow resolution change, I gathered separately the second part (part2) of the bit-streams, since the first parts (part1) are identical:

* **HM5.0\_ARC\_Filters\_HighToLow\_part2.xls**: High-to-Low resolution change second part (part2) of the bit-streams.

The files **HM5.0\_ARC\_Filters\_LowToHigh.xls** and **HM5.0\_ARC\_Filters\_HighToLow.xls** correspond to the low and high resolution parts gathered together.

## High-to-Low Resolution Change :

The results of the part-2 of the bit-stream (Low resolution) are presented below (without class F). The part-1 of the bit-stream is unchanged compared to original ARC-HM5.0 implementation.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Random Access HE** | | | **Random Access LC** | | |
|  | **Y** | **U** | **V** | **Y** | **U** | **V** |
| **Class A** | -1,0% | -4,8% | -5,2% | -1,2% | -4,2% | -5,1% |
| **Class B** | -1,7% | -6,7% | -6,4% | -1,2% | -5,4% | -5,2% |
| **Class C** | -1,1% | -2,9% | -3,4% | -1,0% | -2,4% | -3,1% |
| **Class D** | -2,3% | -4,7% | -4,5% | -1,7% | -4,9% | -4,0% |
| **Class E** |  |  |  |  |  |  |
| **Class F** | -2,8% | -5,0% | -4,2% | -2,2% | -4,1% | -4,3% |
| **Overall** | -1,6% | -4,9% | -4,9% | -1,3% | -4,3% | -4,3% |
|  | -1,6% | -4,9% | -4,9% | -1,4% | -4,4% | -4,2% |
| **Enc. Time** | 128% | | | 137% | | |
| **Dec. Time** | 150% | | | 152% | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Low Delay B - HE** | | | **Low Delay B - LC** | | |
|  | **Y** | **U** | **V** | **Y** | **U** | **V** |
| **Class A** |  |  |  |  |  |  |
| **Class B** | -1,0% | -3,7% | -4,0% | -1,4% | -4,9% | -4,2% |
| **Class C** | -0,8% | -2,7% | -1,9% | -0,9% | -1,2% | -1,6% |
| **Class D** | -0,8% | -2,4% | -1,8% | -1,1% | -3,7% | -2,4% |
| **Class E** | -2,8% | -9,0% | -6,1% | -3,7% | -8,8% | -7,6% |
| **Class F** | -1,4% | -3,1% | -2,1% | -0,7% | -3,2% | -1,6% |
| **Overall** | -1,3% | -3,9% | -3,1% | -1,4% | -4,2% | -3,3% |
|  | -1,3% | -3,9% | -3,1% | -1,4% | -4,2% | -3,4% |
| **Enc. Time** | 117% | | | 121% | | |
| **Dec. Time** | 131% | | | 138% | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Low Delay P - HE** | | | **Low Delay P - LC** | | |
|  | **Y** | **U** | **V** | **Y** | **U** | **V** |
| **Class A** |  |  |  |  |  |  |
| **Class B** | -1,1% | -5,2% | -5,0% | -1,2% | -5,6% | -5,1% |
| **Class C** | -0,5% | -2,3% | -1,5% | -0,6% | -1,1% | -1,3% |
| **Class D** | -1,0% | -2,8% | -2,4% | -1,9% | -3,4% | -3,8% |
| **Class E** | -3,6% | -9,9% | -7,1% | -4,8% | -9,5% | -8,0% |
| **Class F** | -1,0% | -2,4% | -0,6% | -1,3% | -2,5% | -2,8% |
| **Overall** | -1,3% | -4,3% | -3,2% | -1,8% | -4,2% | -4,1% |
|  | -1,4% | -4,3% | -3,1% | -1,8% | -4,3% | -4,1% |
| **Enc. Time** | 112% | | | 115% | | |
| **Dec. Time** | 133% | | | 138% | | |

## Low-to-High Resolution Change :

The results of both parts (part-1 and part-2) of the bit-stream (Low and High resolution) are presented below (without class F). The part-1 of the bit-stream is slightly different from original ARC-HM5.0 implementation since we used same filtering functions than used for inter prediction, with rounding stage performed once at last.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Random Access HE** | | | **Random Access LC** | | |
|  | **Y** | **U** | **V** | **Y** | **U** | **V** |
| **Class A** | -0,1% | -0,2% | -0,6% | -0,1% | -0,9% | -0,8% |
| **Class B** | -0,3% | -1,2% | -1,0% | -0,5% | -0,9% | -0,5% |
| **Class C** | -0,2% | -0,7% | -0,2% | -0,1% | -1,1% | -1,2% |
| **Class D** | -0,2% | 0,3% | 1,4% | 0,0% | -0,1% | -1,0% |
| **Class E** |  |  |  |  |  |  |
| **Class F** | 0,0% | 0,0% | -0,9% | 0,2% | 0,6% | -0,9% |
| **Overall** | -0,2% | -0,5% | -0,1% | -0,2% | -0,7% | -0,9% |
|  | -0,2% | -0,9% | 0,3% | -0,2% | -0,7% | -0,7% |
| **Enc. Time** | 119% | | | 122% | | |
| **Dec. Time** | 126% | | | 133% | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Low Delay B - HE** | | | **Low Delay B - LC** | | |
|  | **Y** | **U** | **V** | **Y** | **U** | **V** |
| **Class A** |  |  |  |  |  |  |
| **Class B** | -0,1% | -0,7% | -0,9% | -0,4% | -1,0% | -0,9% |
| **Class C** | -0,2% | -0,5% | 0,7% | 0,0% | 0,2% | -0,2% |
| **Class D** | -0,2% | -0,7% | 0,4% | 0,2% | -2,1% | -1,1% |
| **Class E** | -0,7% | -1,9% | -1,7% | -0,7% | -0,9% | -2,0% |
| **Class F** | 0,2% | -0,1% | 0,0% | 0,3% | -0,2% | 0,3% |
| **Overall** | -0,2% | -0,7% | -0,3% | -0,1% | -0,8% | -0,7% |
|  | -0,2% | -0,8% | -0,1% | -0,1% | -0,8% | -0,6% |
| **Enc. Time** | 114% | | | 114% | | |
| **Dec. Time** | 125% | | | 128% | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Low Delay P - HE** | | | **Low Delay P - LC** | | |
|  | **Y** | **U** | **V** | **Y** | **U** | **V** |
| **Class A** |  |  |  |  |  |  |
| **Class B** | -0,2% | -0,7% | -0,9% | -0,4% | -0,7% | -0,7% |
| **Class C** | -0,3% | -0,5% | 0,6% | -0,2% | 0,1% | -0,4% |
| **Class D** | -0,2% | -1,0% | -0,1% | -0,2% | -1,4% | -1,1% |
| **Class E** | -0,6% | -2,2% | -2,0% | -0,5% | -1,3% | -2,1% |
| **Class F** | -0,1% | 0,6% | -0,4% | 0,2% | 0,2% | -15,2% |
| **Overall** | -0,3% | -0,7% | -0,5% | -0,2% | -0,6% | -3,8% |
|  | -0,3% | -0,8% | -0,3% | -0,2% | -0,8% | -0,7% |
| **Enc. Time** | 111% | | | 112% | | |
| **Dec. Time** | 124% | | | 128% | | |

The gains may be explained because with the proposed method, only one rounding stage is made, while with previous method two rounding stages are performed: one after the up/down scaling and one after the interpolation filtering.

# Conclusion

In this contribution, a method for implementing Adaptive Resolution Coding (ARC) using only one single resolution reference pictures for prediction is proposed. It is reported that encoding gains are between 0.1% to 1.8% in Luma and up to 4.9% in Chroma on average depending on configuration cases and scenarios (Low-to-High or High-to-Low). The DPB memory size is reduced at the decoder side, compared to previous ARC proposals.

At the encoder side, both resolutions are still necessary to compute motion estimation only.

It is recommended that Adaptive Resolution Coding using single resolution for prediction is adopted into the Working Draft and the HM.

# References

1. JCTVC-F158, “Resolution switching for coding efficiency and resilience,” T. Davies (Cisco), JCTVC 6th Meeting, Turin, IT, 14-22 July 2011.
2. JCTVC-G264, “AHG18: Adaptive Resolution Coding (ARC),” T. Davies (Cisco), P. Topiwala (FastVDO), JCT-VC 7th Meeting, Geneva, CH, 21-30 November, 2011.
3. JCTVC-G018, “JCTVC AHG report: Resolution adaption (AHG18),” T. Davies (Cisco), P. Topiwala (FastVDO) and P. Wu (ZTE), JCT-VC 7th Meeting, Geneva, CH, 21-30 November, 2011.

# Patent rights declaration(s)

**Technicolor 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).**

# Annex: Filters

## Down-Scale filters

downScaleFilter (Original pictures) :

{

4, 0, -12, 0, 40, 64, 40, 0, -12, 0, 4, 0 // sum = 128

}

downScaleFilterHalfPhase (Reference Pictures) :

{

1, 3, -7, -15, 33, 113, 113, 33, -15, -7, 3, 1 // sum = 256

};

//==========================================================================

// DOWN-SCALING(downScale2x2HalfPhase) 2x2 + MV Interpolation Filters

//==========================================================================

Luma\_downScale2x2\_mvInter\_filter :

{1, 3, -7, -15, 33, 113, 113, 33, -15, -7, 3, 1, 0 },

// frac=0 (sum=256)

{0, -1, -3, 11, 27, -71, -203, 147, 743, -252, -1810, 597, 5925, 7313, 3987, 849, -435, -613, -113, 239, 85, -30, -12, 3, 1, 0, 0 },

// frac=1 (sum=16384)

{0, -1, -3, 11, 27, -72, -206, 136, 704, -136, -1584, -277, 3495, 6098, 6098, 3495, -277, -1584, -136, 704, 136, -206, -72, 27, 11, -3, -1, },

// frac=2 (sum=16384)

{0, 0, 0, 1, 3, -12, -30, 85, 239, -113, -613, -435, 849, 3987, 7313, 5925, 597, -1810, -252, 743, 147, -203, -71, 27, 11, -3, -1 }

// frac=3 (sum=16384)

Chroma\_downScale2x2\_mvInter\_filter :

{1, 3, -7, -15, 33, 113, 113, 33, -15, -7, 3, 1, 0 },

// frac=0 (sum=256)

{0, -1, -3, 36, 102, -231, -533, 808, 3166, 3464, 1544, 94, -152, -101, -39, 30, 12, -3, -1 }, // frac=1 (sum=8192)

{0, -2, -6, 41, 111, -247, -607, 608, 2862, 3352, 1824, 460, -40, -152, -62, 39, 15, -3, -1 }, // frac=2 (sum=8192)

{0, -3, -9, 44, 114, -246, -642, 320, 2284, 3120, 2392, 1162, 72, -367, -141, 72, 28, -6, -2 }, // frac=3 (sum=8192)

{0, -1, -3, 16, 42, -87, -221, 120, 846, 1336, 1336, 846, 120, -221, -87, 42, 16, -3, -1 }, // frac=4 (sum=4096)

{0, -2, -6, 28, 72, -141, -367, 72, 1162, 2392, 3120, 2284, 320, -642, -246, 114, 44, -9, -3 }, // frac=5 (sum=8192)

{0, -1, -3, 15, 39, -62, -152, -40, 460, 1824, 3352, 2862, 608, -607, -247, 111, 41, -6, -2 }, // frac=6 (sum=8192)

{0, -1, -3, 12, 30, -39, -101, -152, 94, 1544, 3464, 3166, 808, -533, -231, 102, 36, -3, -1 } // frac=7 (sum=8192)

## Up-Scale filters

upScaleFilter (Original pictures) :

{ 0, 64, 0, 0}, // phase 0

{-8, 40, 40, -8}, // phase 1

upScaleFilterQuarterPhase (Reference Pictures) :

{ 1, -6, 25, 130, -33, 11, 0, 0 }, // phase 0 - sum = 128

{ 0, 11, -33, 130, 25, -6, 1, 0 } // phase 1 - sum = 128

//============================================================

// UP-SCALING(QuarterPhase) 2x2 + MV Interpolation Filters

//============================================================

Luma\_upScale2x2\_mvInter\_filter :

// phase 0 : -----------------------

{ 1, -6, 25, 130, -33, 11, 0, 0, 0,0,0,0,0 },

// frac=0 (sum=128)

{0, -7, -43, 134, 125, 9216, -1906, 716, -44, 1, 0, 0, 0 },

// frac=1 (sum=8192)

{0, -7, -72, 522, -1146, 9598, -1146, 522, -72, -7, 0, 0, 0 },

// frac=2 (sum=8192)

{0, 1, -44, 716, -1906, 9216, 125, 134, -43, -7, 0, 0, 0 }

// frac=3 (sum=8192)

// phase 1 : -----------------------

{0, 11, -33, 130, 25, -6, 1, 0, 0,0,0,0,0 },

// frac=0 (sum=128)

{0, -1, 40, 558, -1930, 6957, 3330, -1010, 242, 6, 0, 0, 0 },

// frac=1 (sum=8192)

{0, -1, 39, 389, -1562, 5231, 5231, -1562, 389, 39, -1, 0, 0 },

// frac=2 (sum=8192)

{0, 0, 6, 242, -1010, 3330, 6957, -1930, 558, 40, -1, 0, 0 }

// frac=3 (sum=8192)

}

Chroma\_upScale2x2\_mvInter\_filter :

// phase 0 : -------------------------

{1, -6, 25, 130, -33, 11, 0, 0, 0,0,0 },

// frac=0 (sum=128)

{0, 18, -87, 436, 4370, -956, 321, -6, 0, 0, 0 },

// frac=1 (sum=4096)

{0, 5, -9, 157, 4475, -809, 280, -3, 0, 0, 0 },

// frac=2 (sum=4096)

{0, -10, 113, -265, 4685, -651, 232, -8, 0, 0, 0 },

// frac=3 (sum=4096)

{0, -2, 77, -196, 2290, -196, 77, -2, 0, 0, 0 },

// frac=4 (sum=2048)

{0, -8, 232, -651, 4685, -265, 113, -10, 0, 0, 0 },

// frac=5 (sum=4096)

{0, -3, 280, -809, 4475, 157, -9, 5, 0, 0, 0 },

// frac=6 (sum=4096)

{0, -6, 321, -956, 4370, 436, -87, 18, 0, 0, 0 }

// frac=7 (sum=4096)

// phase 1 : ------------------------

{0, 11, -33, 130, 25, -6, 1, 0, 0,0,0 },

// frac=0 (sum=128)

{0, -1, 330, -1023, 3798, 1278, -375, 90, -1, 0, 0 },

// frac=1 (sum=4096)

{0, -2, 317, -1000, 3483, 1651, -473, 121, -1, 0, 0 },

// frac=2 (sum=4096)

{0, -3, 285, -940, 3016, 2234, -683, 189, -2, 0, 0 },

// frac=3 (sum=4096)

{0, -1, 114, -387, 1298, 1298, -387, 114, -1, 0, 0 },

// frac=4 (sum=2048)

{0, -2, 189, -683, 2234, 3016, -940, 285, -3, 0, 0 },

// frac=5 (sum=4096)

{0, -1, 121, -473, 1651, 3483, -1000, 317, -2, 0, 0 },

// frac=6 (sum=4096)

{0, -1, 90, -375, 1278, 3798, -1023, 330, -1, 0, 0 }

// frac=7 (sum=4096)