JCTVC-M0263

AHG13: SHVC Upsampling with phase offset adjustment

David Baylon, Ajay Luthra and Koohyar Minoo

Problem(s) Being Addressed

In the current SHM the interpolation filters are fixed. This will have the following implications:

1. The down-sampling phase offset should be known during the up-sampling process. Currently these phase offsets are assumed to be known.
2. To cover a reasonable number of sub-pixel positions at the base layer SHVC needs a large number of filters (currently 12 or 16 for each Luma and Chroma) to be designed (and tested!).

Compensating for the down-sampling phase offset

With the exception of mandating the phase offset used during down-sampling, the following two solutions are proposed.

1. The phase offset is signaled for vertical and horizontal directions per Luma and Chroma.
2. Encoder signals the filter coefficients for each sub-pixel filter index (Indices are calculated assuming zero phase offset). The filter coefficients will make sure the interpolation for each index has the proper phase shift.

Solution #1: Syntax for signaling the phase offset

pic_parameter_set_rbsp( ) {	**Descriptor**
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
if(nuh_layer_id > 0 && InterLayerTextureRlEnableFlag) {	
luma_phase_offset[0]	se(v)
luma_phase_offset[1]	se(v)
chroma_phase_offset[0]	se(v)
chroma_phase_offset[1]	se(v)
}	
. . .	
}	

Solution #1: Signaling the phase offset

- ~~The variable xRef16 is derived as follow:~~
- ~~xRef16 = (xP * PicWRL * 16 + ScaledW / 2) / ScaledW (G-3)~~
- ~~The variable yRef16 is derived as follows:~~
- – If cIdx is equal to 0, the variables xRef16 and yRef16 ~~is~~are derived as follows:
- xRef16 = (xP * PicWRL * 16 + ScaledW / 2) / ScaledW + **luma_phase_offset**[0] (G-3)
- yRef16 = (yP * PicHRL * 16 + ScaledH / 2) / ScaledH + **luma_phase_offset**[1] (G-4)
- – Otherwise, the variables xRef16 and yRef16 ~~is~~are derived as follows:
- xRef16 = (xP * PicWRL * 16 + ScaledW / 2) / ScaledW + **chroma_phase_offset**[0] (G-5)
- yRef16 = (yP * PicHRL * 16 + ScaledH / 2) / ScaledH ~~– offset~~ + **chroma_phase_offset**[1] (G-6)
- ~~where the value of offset is derived as follows:~~
- ~~if (ScaledH is equal to PicHRL)~~
  ~~offset = 0~~
  ~~otherwise if (ScaledH is equal to 1.5 * PicHRL)~~
  ~~offset = 1~~
  ~~otherwise if (ScaledH is equal to 2.0 * PicHRL)~~
  ~~offset = 2~~

Solution #2: Signaling the filter coefficients

- Number of sub-pixel positions and filter coefficients for each position are signaled.
- The filter indices are calculated based on scaling factor while assuming a zero-phase shift.
- The coefficients at each index accommodate for the proper sub-pixel position.
- Example for 2X spatial scaling with 0.25 phase offset:
  - Number of filters: 2
  - f(0) = { -1, 4, -10, 58, 17, -5, 1, 0 }
  - f(1) = { 0, 1, -5, 17, 58, -10, 4, -1 }

Solution #2: Syntax for signaling the filter coefficients

pic_parameter_set_rbsp( ) {	**Descriptor**
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
if(nuh_layer_id > 0 && InterLayerTextureRlEnableFlag) {	
for(i = 0; i < 2; i++) {	
num_ phase_offsets_minus1[i]	ue(v)
for(j = 0; j <= num_ phase_offsets_minus1[i]; j++) {	
luma_pixel_shift_flag[i][j]	u(1)
ref_luma_filter_indx[i][j]	ue(v)
for(k = 0; k < num_luma_taps; k++) {	
delta_luma_filter_coef[i][j][k]	se(v)
}	
chroma_pixel_shift_flag[i][j]	u(1)
ref_chroma_filter_indx[i][j]	ue(v)
for(k = 0; k < num_chroma_taps; k++) {	
delta_chroma_filter_coef[i][j][k]	se(v)
}	
}	
}	
}	
. . .	
}	

Solution #2: Signaling the filter coefficients

- ~~The variable xRef16 is derived as follows:~~
- ~~xRef16 = (xP * PicWRL * 16 + ScaledW / 2) / ScaledW (G-3)~~
- ~~The variable yRef16 is derived as follows:~~
- – ~~If cIdx is equal to 0, the~~ The variables xRefphase and yRefphase ~~yRef16 is~~are derived as follows:
- ~~xRef16~~ xRefphase = (xP * PicWRL * (num_ phase_offsets_minus1[0] + 1) ~~16 + ScaledW / 2~~) / ScaledW (G-3)
- ~~yRef16~~ yRefphase = (yP * PicHRL * (num_ phase_offsets_minus1[1] + 1) ~~16 + ScaledH / 2~~) / ScaledH (G-4)
- ~~– Otherwise, the variable yRef16 is derived as follow:~~
- ~~yRef16 = (yP * PicHRL * 16 + ScaledH / 2) / ScaledH – offset (G-5)~~
- ~~where the value of offset is derived as follows:~~
- ~~if (ScaledH is equal to PicHRL)~~
  ~~offset = 0~~
  ~~otherwise if (ScaledH is equal to 1.5 * PicHRL)~~
  ~~offset = 1~~
  ~~otherwise if (ScaledH is equal to 2.0 * PicHRL)~~
  ~~offset = 2~~
- The variables xRef and xPhase are derived by
- xRef = ~~(xRef16 >> 4)~~ (xRefphase / (num_ phase_offsets_minus1[0] + 1)) (G-7),(G-15)
- xPhase = ~~(xRef16) % 16~~ (xRefphase – xRef * (num_ phase_offsets_minus1[0] + 1)) (G-8),(G-16)
- The variables yRef and yPhase are derived by
- yRef = ~~(yRef16 >> 4)~~ (yRefphase / (num_ phase_offsets_minus1[1] + 1)) (G-9),(G-17)
- yPhase = ~~(yRef16) % 16~~ (yRefphase – yRef * (num_ phase_offsets_minus1[1] + 1)) (G-10),(G-18)

Note: In this case there is no need for rounding operation to find the closest sub-pel position (just truncation).

Summary

- Need to signal the phase offset
- Advantages of signaling coefficients:
  - Compensating for arbitrary phase shift due to down-sampling with a small number of filters
  - No need to design and verify a large set of fixed of filters, given the limited test conditions available and affordable.
  - No need for rounding, during phase index derivation
  - Propose further study of Adaptive Upsampling Filters in regards to coding efficiency and possible new applications possible by supporting adaptive filter coefficients.