

JCTVC-R0300 NON-SCE1: IMPROVED RESIDUAL COEFFICIENT CODING FOR 3D LUT



invention | collaboration | contribution

Yan Ye, Yuwen He
InterDigital Communications, Inc.

18th JCT-VC meeting, July 2014

Introduction

- JCTVC-R0151 proposes the following changes to CGS in SHVC draft 6

	SHVC draft 6	R0151
3D LUT partition	Even partition of all color components	Adaptive (uneven) partition of U/V components
Model parameters for each octant	Vertex based model parameters	(A, B, C, D) parameters with simplified prediction method
Residual coeff coding	se(v)	ue(v) for MSBs + u(7) for LSBs + 1-bit sign

- JCTVC-R0151 performance vs. SHM6.0:

	1x			2x		
	Y	U	V	Y	U	V
AI	-1.2%	-2.0%	-1.3%	-0.9%	-1.2%	-1.1%
RA	-1.1%	-1.7%	-1.5%	-0.4%	-0.6%	-0.7%

3D LUT signaling overhead of R0151

- JCTVC-R0151 uses 14% more CGS bits compared to SHM-6.0
- Adaptive partition of U/V increases the effectiveness of 3D LUT
 - For the two BirthdayFlash sequences, increased effectiveness causes more 3D LUT being sent in PPS
 - Increased overhead bits

		Total # of 3D LUT bits		% increase
		SHM6.0	R0151	
AI 1x	All seq	3772360	4438948	18%
	"BirthdayFlash" seq	2931460	3694998	26%
AI 2x	All seq	3779510	4228036	12%
	"BirthdayFlash" seq	2901170	3509570	21%
RA 1x	All seq	1548418	1768084	14%
	"BirthdayFlash" seq	1069898	1316350	23%
RA 2x	All seq	1452478	1594332	10%
	"BirthdayFlash" seq	990120	1141750	15%
All cases	All seq	10552766	12029400	14%

- A small change on top of R0151 to reduce 3D LUT signaling overhead

Residual coefficients in R0151

- Denote CMResX = prediction residual of component X after prediction
- Denote M = magnitude of CMResX
- Precision of $M = (12 - \text{cm_res_quant_bits})$
 - $(12 - \text{nMappingShift})$ bits for the integer part (MSB)
 - $(\text{nMappingShift} - \text{cm_res_quant_bits})$ bits for the fractional part (LSB)
 - Where $\text{nMappingShift} = 10 + \text{inputBitDepth} - \text{outputBitDepth}$
 - Applies to 3 out of 4 model parameters, except D
- For LSB coding, replace $u(7)$ with $u(v)$, where $v = (\text{nMappingShift} - \text{cm_res_quant_bits} - \text{cm_delta_flc_bits})$
- cm_delta_flc_bits : chosen and signaled by the encoder

Proposed change based on R0151 (1): colour_mapping_table()

colour_mapping_table() {	Descriptor
cm_octant_depth	u(2)
cm_y_part_num_log2	u(2)
cm_input_luma_bit_depth_minus8	u(3)
cm_input_chroma_bit_depth_delta	se(v)
cm_output_luma_bit_depth_minus8	u(3)
cm_output_chroma_bit_depth_delta	se(v)
cm_res_quant_bits	u(2)
cm_delta_flc_bits_minus1	u(2)
if(cm_octant_depth == 1) {	
cm_adapt_threshold_u_delta	se(v)
cm_adapt_threshold_v_delta	se(v)
}	
colour_mapping_octants(0, 0, 0, 0, 1 << cm_octant_depth)	
}	

cm_delta_flc_bits_minus1 specifies the delta value used to derive the number of bits used to code res_coeff_r. The variable CMResLSBits is set equal to max (0, (10 + CMInputBitDepthY – CMOutputBitDepthY – cm_res_quant_bits – (cm_delta_flc_bits_minus1 + 1))).

Proposed change based on R0151 (2): colour_mapping_octants()

colour_mapping_octants(depth, yldx, uldx, vldx, length) { [Ed. (MH): Camel casing suggested for all input parameters, i.e. depth and length should be changed.]	Descriptor
if (depth < cm_octant_depth)	
split_octant_flag	u(1)
if (split_octant_flag)	
for(k = 0; k < 2; k++)	
for(m = 0; m < 2 ; m++)	
for(n = 0; n < 2; n++)	
colour_mapping_octants(depth + 1, yldx + YPartNum * k * length / 2, uldx + m * length / 2, vldx + n * length / 2, length / 2)	
else	
for(i = 0; i < YPartNum; i++)	
for(j = 0; j < 4; j++) {	
coded_res_flag [yldx + (i << (cm_octant_depth-depth))][uldx][vldx][j]	u(1)
if(coded_res_flag[yldx + (i << (cm_octant_depth-depth))][uldx][vldx][j])	
for(c = 0; c < 3; c++) {	
res_coeff_q [yldx + (i << (cm_octant_depth-depth))][uldx][vldx][j][c])	ue(v)
res_coeff_r [yldx + (i << (cm_octant_depth-depth))][uldx][vldx][j][c])	u(7) -u(v)
if(res_coeff_q[yldx + (i << (cm_octant_depth-depth))][uldx][vldx][j][c]	
res_coeff_r[yldx + (i << (cm_octant_depth-depth))][uldx][vldx][j][c])	
res_coeff_s [yldx + (i << (cm_octant_depth-depth))][uldx][vldx][j][c])	u(1)
}	
}	
}	
}	

Proposed change based on R0151 (3): semantics

res_coeff_q[yldx][uldx][vldx][i][c] specifies the quotient of the residual for the colour mapping coefficient with index [yldx][uldx][vldx][i][c]. When not present, the value of res_coeff_q is inferred to be equal to 0.

res_coeff_r[yldx][uldx][vldx][i][c] specifies the remainder of the residual for the colour mapping coefficient with index [yldx][uldx][vldx][i][c]. **The number of bits used to code res_coeff_r is equal to CMResLSBits. If CMResLSBits is equal to 0, then res_coeff_r is not present.** When not present, the value of res_coeff_r is inferred to be equal to 0.

res_coeff_s [yldx][uldx][vldx][pldx][cldx] specifies the sign of the residual for the colour mapping coefficient with index [yldx][uldx][vldx][i][c]. When not present, the value of res_coeff_s is inferred to be equal to 0.

The variables CMResY[yldx][uldx][vldx][i], CMResU[yldx][uldx][vldx][i] and CMResV[yldx][uldx][vldx][i] are derived as follows:

CMResY[yldx][uldx][vldx][i] =
(1 - 2 * res_coeff_s[yldx][uldx][vldx][i][0]) * ((res_coeff_q[yldx][uldx][vldx][i][0] << 7 CMResLSBits) + res_coeff_r[yldx][uldx][vldx][i][0])

CMResU[yldx][uldx][vldx][i] =
(1 - 2 * res_coeff_s[yldx][uldx][vldx][i][1]) * ((res_coeff_q [yldx][uldx][vldx][i][1] << 7 CMResLSBits) + res_coeff_r[yldx][uldx][vldx][i][1])

CMResV[yldx][uldx][vldx][i] =
(1 - 2 * res_coeff_s[yldx][uldx][vldx][i][2]) * ((res_coeff_q[yldx][uldx][vldx][i][2] << 7 CMResLSBits) + res_coeff_r[yldx][uldx][vldx][i][2])

Performance evaluation

	AI HEVC 1x 10-bit base			AI HEVC 2x 8-bit base		
	Y	U	V	Y	U	V
Class A+	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Overall (Test vs Ref)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Overall (Test vs single layer)	17.3%	8.3%	-10.1%	9.9%	10.0%	2.6%
Overall (Ref vs single layer)	17.4%	8.4%	-10.1%	9.9%	10.0%	2.6%
EL only (Test vs Ref)	-0.1%	-0.1%	-0.1%	0.0%	0.0%	0.0%
Overall (Test EL+BL vs single EL+BL)	-36.3%	-40.9%	-50.6%	-26.8%	-27.0%	-31.7%
Overall (Ref EL+BL vs single EL+BL)	-36.3%	-40.9%	-50.6%	-26.8%	-27.0%	-31.7%
CGS bits[%]	90.8%			91.2%		

	RA HEVC 1x 10-bit base			RA HEVC 2x 8-bit base		
	Y	U	V	Y	U	V
Class A+	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Overall (Test vs Ref)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Overall (Test vs single layer)	27.7%	20.9%	-1.8%	20.7%	21.7%	11.7%
Overall (Ref vs single layer)	27.7%	20.9%	-1.8%	20.7%	21.7%	11.7%
EL only (Test vs Ref)	-0.1%	-0.1%	-0.1%	0.0%	0.0%	0.0%
Overall (Test EL+BL vs single EL+BL)	-29.6%	-33.0%	-44.8%	-18.8%	-18.0%	-24.2%
Overall (Ref EL+BL vs single EL+BL)	-29.6%	-33.0%	-44.8%	-18.8%	-18.0%	-24.2%
CGS bits[%]	90.5%			90.9%		

3D LUT signaling overhead reduction

	Total # of 3D LUT bits		% reduction
	R0151	Proposed	
AI 1x	4438948	4048276	-9%
AI 2x	4228036	3905824	-8%
RA 1x	1768084	1610996	-9%
RA 2x	1594332	1466554	-8%
All cases	12029400	11031650	-8%

Conclusion

- A small change to residual coefficient entropy coding on top of R0151 to reduce signaling overhead
- Replace $u(7)$ with $u(v)$ for `res_quant_r` coding
- On average 8% of signaling overhead reduction
- Suggest to adopt together with R0151 entropy coding method

**Thanks Qualcomm for cross-checking!
(JCTVC-R0310)**