JCTVC-F073

MEDIATEK

Joint Luma-Chroma Adaptive Reference Picture Memory Compression

Shan Liu, Ximin Zhang and Shawmin Lei

Presented by Shan Liu
6^{th} JCT Meeting in Torino
14-22 July, 2011

Outline

- Prior arts
  - Block-based adaptive scaling (Toshiba, Zenverge)
- Proposed method
  - Block-based adaptive scaling
  - Jointly compress Luma and Chroma components
  - Version 1 and 2
- Results
- Conclusions

Prior Arts

- JCTVC-E133 and JCTVC-E432
  - 4x4 block-based adaptive scaling

RA HE	JCTVC-E133 Adaptive Scaling			JCTVC-E432 Unified Scaling		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.48	1.59	1.50	0.30	0.12	-0.06
Class B	0.57	1.24	1.12	0.25	0.04	0.02
Class C	0.36	0.61	0.48	0.22	0.08	0.05
Class D	0.31	0.41	0.14	0.30	0.19	0.01
Class E						
All	0.44	0.98	0.83	0.27	0.10	0.01
Enc Time[%]	100%			102%		
Dec Time[%]	105%			113%		

RA LD	JCTVC-E133 Adaptive Scaling			JCTVC-E432 Unified Scaling		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	1.02	1.62	1.61	0.73	0.03	0.22
Class C	0.62	0.70	0.87	0.56	0.48	0.45
Class D	0.34	0.57	0.27	0.27	0.32	0.57
Class E	3.37	2.69	3.88	5.00	0.43	0.64
All	1.19	1.33	1.52	1.37	0.29	0.44
Enc Time[%]	100%			102%		
Dec Time[%]	103%			113%		

Proposed Method

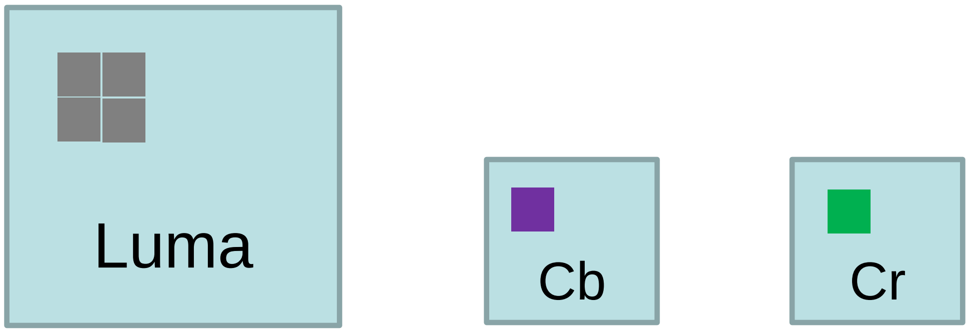
- Observations
  - Chroma pixel dynamic range tends to be narrower than Luma
  - Human vision system tends to tolerate Chorma loss better
- Basic idea
  - “Borrow” bits from Chroma for enhancing Luma
- Localized (block-based) adaptive scaling
  - 4x4 compression unit for both Luma and Chroma
  - One “scale” for each compression unit
  - Decoder memory access bandwidth same as prior arts

Proposed Method

- Joint Luma-Chroma compression
  - YUV420 format
    - One 4x4 Cb block
    - One 4x4 Cr block
    - Four 4x4 Luma block

    Total number of desired bits (bit budget) for 10-to-8 bit compression

    (8bit x 16) + (8bit x 16) + (8bit x 16) x4 = 768 bits



  - May also apply to YUV422 and YUV444 formats.

Proposed Method (version 1)

- Compress a 4x4 chroma block
  - Find max and min pixel value

    M = (min >> 1) << 1
  - Find scaling factor “S” s.t.

    (|max – M| >> S) < 128 S = {0, 1, 2, 3}
  - The position of pixel with “min” i.e. “min_pos”.
  - Store “S”, “M”, “min_pos” and 7bit representations for the rest 15 pixels → Total 120 bits.

S(2)	M (9)	min_pos(4)	7bit x 15

P_store = (P – M) >> S;

P_dec = (P_store << S) + M + 2^(S-1) when S>0.

Proposed Method (version 1)

- Compress a 4x4 luma block
  - Find max and min pixel value
  - Find scaling factor “S” s.t.

    $(|max - min| >> S) < 128$ $S = \{0, 1, 2, 3\}$
  - The position of pixel with “min” i.e. “min_pos”
  - When S < 3, store “S”, “M”, “min_pos”, 11 pixels are stored in 8 bits and 4 pixels are stored in 7bits → Total 132 bits

S(2)	min (10)	min_pos(4)	8bit x 11 + 7bit x 4

$S_8 = Max(S_7 -1, 0)$ $S_7 = S$

		P7	
	P7	m	P7
		P7	

  - When S=3, all 16 pixels are represented and stored in 8 bits.

S(2)	8bit x 16

Proposed Method (version 2)

- More aggressive bit savings from Chroma
  - Uniformed 7bit representation in version 1
  - Down to 5bit representation in version 2
- Compress a 4x4 chroma block
  - Find max and min pixel value
  - Find scaling factor “S” s.t.

    (|max – min| >> S) < 32 S = {0, 1, 2, 3, 4, 5}
  - When S=0, 1, 2, use 5, 6, 7 bits to store Chroma pixels (lossless)

00	min (10)	min_pos(4)	5bit x 15 pel	*Total 91 bits, save 37 bits*
01	min (10)	min_pos(4)	6bit x 15 pel	*Total 106 bits, save 22 bits*
10	min (10)	min_pos(4)	7bit x 15 pel	*Total 121 bits, save 7 bits*

  - When S=3 and 4, use 7 bits to store Chroma pixels (lossy)

11	0 1	min (9)	min_pos(4)	7bit x 15 pel	*Total 121 bits, save 7 bits*

  - When S=5,

11111111	7bit x 16 pel	*<121 bits*

Proposed Method (version 2)

- Compress a 4x4 luma block
  - Similar to version 1
  - With more possible bits “borrowed” from Chroma
  - One example:
    - 6 bit per pixel for Cb (save 22 bits) and Cr (save 22 bits)
    - Total 44 bit savings evenly distributed to four luma blocks → Each luma block may use up to (128+11=) 139 bits.

S(2)	min (10)	min_pos(4)	8bit x 12 + 9bit x 3

P9	**P9**	**m**	**P9**

Store and Reconstruct

- Store

  $$P\_str(i) = (p(i) - M) >> scale,$$

M = (min >> 1) << 1,	for chroma lossy compression
M = min,	otherwise

- Reconstruct

  $$P\_rec(i) = (P\_str(i) << scale) + 2^{(scale-1)}, \text{ when } scale > 0$$

Results – BD Rate (HM3.0)

	Random access – version 1			Random access – version 2			Random access – fix rounding		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.03	0.24	-0.01	0.01	0.10	0.27	0.74	6.87	8.88
Class B	0.07	0.04	0.02	0.01	-0.07	-0.01	1.05	3.54	4.86
Class C	0.08	0.16	0.21	0.02	0.07	0.21	0.66	2.15	2.16
Class D	0.08	0.06	0.00	0.01	-0.05	0.19	0.57	1.57	1.77
Class E									
All	0.07	0.12	0.05	0.01	0.01	0.16	0.77	3.53	4.44
Enc Time[%]	99%			99%			100%		
Dec Time[%]	105%			105%			107.5%		

	Low delay – version 1			Low delay – version 2			Low delay – fix rounding		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A									
Class B	0.36	0.18	0.19	0.19	0.07	0.13	2.35	5.71	8.11
Class C	0.24	0.43	0.65	0.14	0.33	0.47	1.26	4.14	4.46
Class D	0.07	0.01	0.83	-0.04	0.08	1.06	0.75	6.14	7.43
Class E	2.75	0.98	0.76	1.06	0.59	0.17	14.70	18.22	25.83
All	0.70	0.35	0.57	0.29	0.23	0.45	3.99	7.77	10.35
Enc Time[%]	100%			100%			102%		
Dec Time[%]	105%			106%			114.8%		

Results – MC Bandwidth

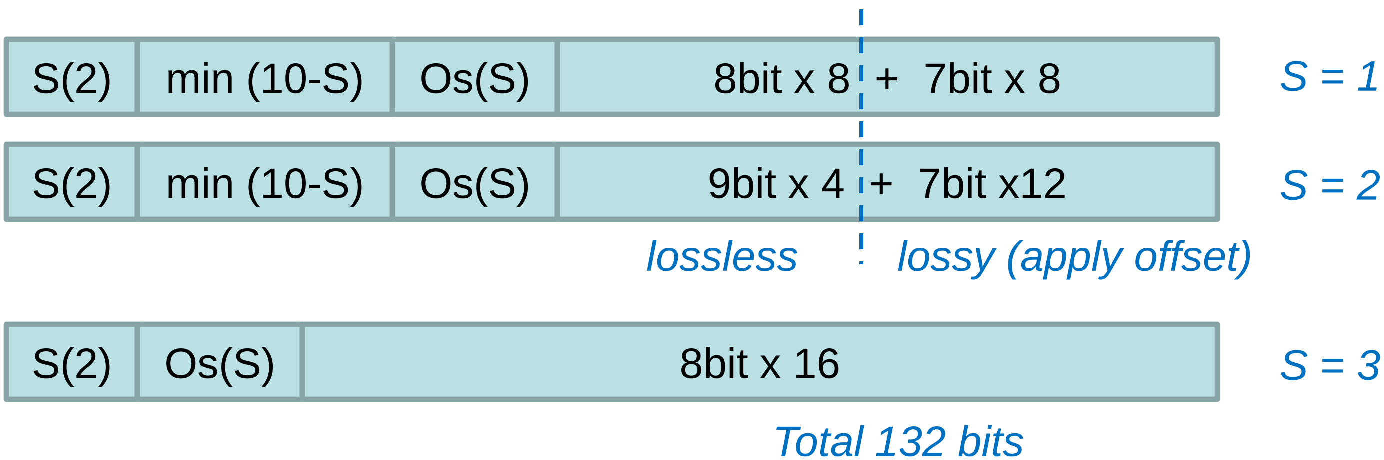
RA HE	Memory bandwith increase %							
	8bit/8bit	32bit/ 64bit	32bit/128bit	64bit/128bit	64bit/256bit	64bit/512bit	64bit/256bit FIFO	64bit/512bit FIFO
Class A	-2.65%	-13.12%	-20.20%	-22.76%	-26.86%	-39.52%	-28.36%	-30.70%
Class B	-2.74%	-12.69%	-19.21%	-21.68%	-25.42%	-40.20%	-26.77%	-30.24%
Class C	1.78%	-10.74%	-19.85%	-23.61%	-29.07%	-44.85%	-34.45%	-38.49%
Class D	8.03%	-7.21%	-18.46%	-23.23%	-29.67%	-46.92%	-35.76%	-40.03%
Class E								
All	0.82%	-11.10%	-19.50%	-22.82%	-27.76%	-42.86%	-31.15%	-34.68%

LD HE	Memory bandwith increase %							
	8bit/8bit	32bit/ 64bit	32bit/128bit	64bit/128bit	64bit/256bit	64bit/512bit	64bit/256bit FIFO	64bit/512bit FIFO
Class A								
Class B	0.31%	-11.11%	-19.01%	-22.13%	-26.58%	-41.58%	-29.97%	-33.37%
Class C	4.48%	-9.86%	-20.53%	-24.77%	-31.18%	-48.14%	-38.51%	-42.98%
Class D	12.47%	-5.95%	-18.96%	-24.12%	-31.73%	-50.35%	-39.45%	-44.66%
Class E	-12.20%	-17.00%	-21.87%	-23.41%	-33.36%	-48.63%	-31.52%	-34.09%
All	2.05%	-10.61%	-19.92%	-23.53%	-30.29%	-46.73%	-34.77%	-38.73%

Apply Offset to Version 1

- JCTVC-D035 by Zenverge
  - Use “offset” for each 4x4 block (both luma and chroma)

- A quick implementation
  - Use “offset” for each 4x4 luma block



MEDIATEK

Results – Version 1 with Luma Offset

	Random access – Zenverge			Random access – version1 + luma offset		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.00	-0.13	-0.26	-0.03	0.14	-0.12
Class B	0.10	-0.09	-0.04	0.04	-0.06	0.06
Class C	0.12	0.07	0.02	0.06	0.07	0.06
Class D	0.14	0.00	-0.03	0.06	0.06	-0.04
Class E						
All	0.09	-0.04	-0.07	0.03	0.05	0.00
Enc Time[%]	100%			100%		
Dec Time[%]	106.0%			103.3%		

	Low delay – Zenverge			Low delay – version1 + luma offset		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	0.28	0.23	0.15	0.16	-0.35	-0.16
Class C	0.21	0.22	0.10	0.10	-0.44	-0.61
Class D	0.17	0.38	0.64	0.08	0.19	0.20
Class E	1.61	0.24	0.66	0.98	0.31	1.34
All	0.48	0.27	0.36	0.28	-0.11	0.10
Enc Time[%]	100%			100%		
Dec Time[%]	106.2%			103.4%		

Conclusions

- Proposed a method for compressing reference picture
  - Block based adaptive scaling
  - Joint Luma-Chroma bit allocation
- Results
  - Version 1: average 0.07% and 0.70% BD-rate increase for RA-HE and LD-HE
  - Version 2: average 0.01% and 0.29% BD-rate increase for RA-HE and LD-HE
  - Decoder-side MC bandwidth consumption is same as (similar to) previous contributions e.g. JCTVC-E133 and JCTVC-E432.
- The proposed method can be easily combined with methods from other proponents for further improvements.
- Recommend for further study in AhG or CE.