



Non-CE4: On Affine Motion Vector Restriction

JVET-P0261

Xuwei Meng(Peking University), Xiaozhen Zheng(DJI),
Shanshe Wang, Siwei Ma



Overview

- Two methods for Affine memory bandwidth restriction
 - 6-tap interpolation filter (JVET-N0196)
 - MV spread area restriction (JVET-N0068)
 - 8x8 based MV spread area limitation for Bi-prediction
 - 4x8/8x4 based MV spread area limitation for Uni-prediction
- Proposed
 - Remove the MV spread limitation for Affine uni-prediction
- 4 multiples, 20 additions, 12 shifts, 4 max functions and 4 functions are saved at most for each uni-prediction Affine CU in encoder and decoder
- 0.00% and -0.02% coding performance change compared to VTM-6.0 with RA and LDB configuration.

Introduction

□ Affine memory bandwidth constraint methods

■ 6-tap interpolation filter (JVET-N0196)

■ MV spread area restriction (JVET-N0068)

□ Bi-prediction CU

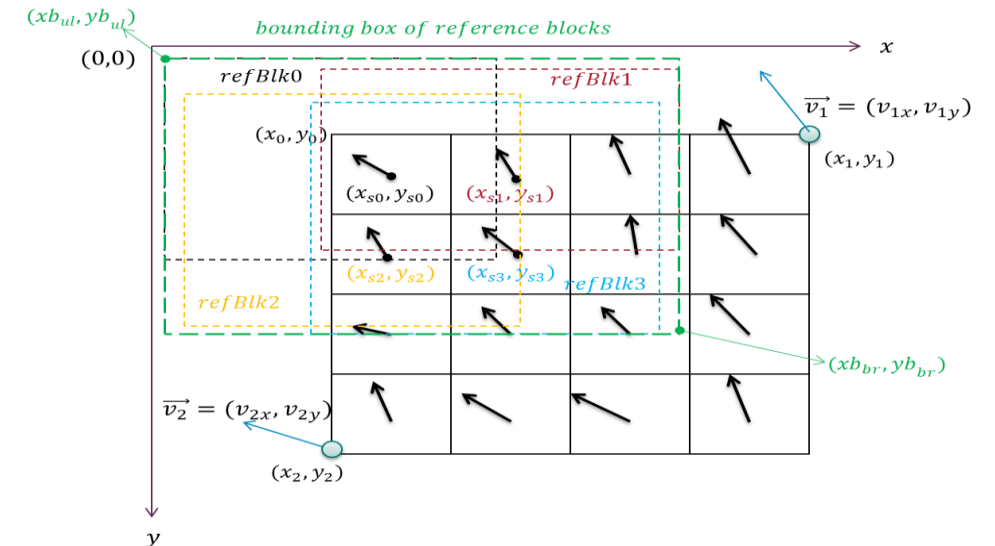
$$\begin{cases} bxW4 = \max(0, 4(1+a), 4c, 4(1+a)+4c) - \min(0, 4(1+a), 4c, 4(1+a)+4c) + 11 \\ bxH4 = \max(0, 4b, 4(1+d), 4b+4(1+d)) - \min(0, 4b, 4(1+d), 4b+4(1+d)) + 11 \end{cases}$$

□ Uni-prediction CU

$$\begin{cases} bxW_h = \max(0, 4(1+a)) - \min(0, 4(1+a)) + 11 \\ bxH_h = \max(0, 4b) - \min(0, 4b) + 11 \end{cases}$$

$$\begin{cases} bxW_v = \max(0, 4c) - \min(0, 4c) + 11 \\ bxH_v = \max(0, 4(1+d)) - \min(0, 4(1+d)) + 11 \end{cases}$$

$$\begin{cases} Thred4 = (15 + \delta_x) * (15 + \delta_y) \\ Thred_h = (15 + \delta_x) * (11 + \delta_y) \\ Thred_v = (11 + \delta_x) * (15 + \delta_y) \end{cases}$$



Proposed method

□ Current design

■ Memory bandwidth consumption in current design

Table 1. Memory bandwidth consumption (per-pixel) of normal inter mode

Block Size	Prediction mode	Luma (8 tap)	Chroma (4 tap)	All
4x8/8x4	Uni	5.16	2.18	7.34
4x16/16x4	Bi/Triangle	7.9	3.44	11.34
8x8	Bi/Triangle	7.03	3.06	10.09
4x32/32x4	Bi/Triangle	6.70	2.97	9.67

Table 2. Memory bandwidth consumption (per-pixel) of Affine mode **without MV Spread Limitation**

Block Size	Prediction mode	Luma (6 tap)	Chroma (4 tap)	All
MxN	Affine Uni	5.06	2.18	7.24
MxN	Affine Bi	10.13	3.06	13.19

Table 3. Memory bandwidth consumption (per-pixel) of Affine mode **with MV Spread Limitation**

Block Size	Prediction mode	δ_x	δ_y	$Thred_4$	Luma (6 tap)	Chroma (4 tap)	All
MxN	Affine Uni	2	2	15x11	5.06	2.18	7.24
MxN	Affine Bi	2	2	15x15	7.03	3.06	10.09

Proposed method

□ Current design

■ Computation complexity (per-CU) introduced by MV Spread Limitation

Block Size	Prediction mode	Mult.	Add.	Shift	Max	Min
MxN	Affine Bi	2	20	18	6	6
MxN	Affine Uni	4	20	12	4	4

□ Proposed method

■ All the operations for Affine Uni-prediction CU are removed

Block Size	Prediction mode	Mult.	Add.	Shift	Max	Min
MxN	Affine Bi	2	20	18	6	6
MxN	Affine Uni	-	-	-	-	-

Performance

- Anchor: VTM-6.0
- Test: VTM-6.0 with proposed method (CTC)

	Random access Main10				
	Over VTM-6.0				
	Y	U	V	EncT	DecT
Class A1	0.01%	-0.01%	-0.01%	103%	100%
Class A2	0.02%	0.02%	0.04%	110%	100%
Class B	-0.02%	0.04%	0.08%	100%	100%
Class C	0.00%	0.07%	0.17%	104%	100%
Class E					
Overall	0.00%	0.03%	0.08%	104%	100%
Class D	-0.03%	0.06%	0.11%	99%	98%
Class F	0.01%	-0.05%	-0.03%	99%	99%

	Low delay B Main10				
	Over VTM-6.0				
	Y	U	V	EncT	DecT
Class A1					
Class A2					
Class B	0.00%	0.06%	-0.34%	91%	100%
Class C	-0.03%	-0.05%	0.07%	100%	99%
Class E	-0.03%	0.29%	-0.22%	82%	99%
Overall	-0.02%	0.08%	-0.17%	92%	100%
Class D	-0.04%	0.18%	0.09%	100%	99%
Class F	0.06%	0.15%	1.14%	108%	99%

Performance

- Anchor: VTM-6.0 with JVET_J0090_MEMORY_BANDWIDTH_MEASURE on
- Test: VTM-6.0 with proposed method

	Random access Main10	Low delay B Main10	Low delay P Main10
	cache_1d.cfg	cache_1d.cfg	cache_1d.cfg
Class A1	100%		
Class A2	100%		
Class B	100%	100%	100%
Class C	99%	100%	100%
Class E		100%	100%
Overall	100%	100%	100%
Class D	100%	101%	100%
Class F	100%	101%	100%

Conclusion

- This proposal suggests to remove the unnecessary MV spread limitation method for Affine Uni-prediction.

- Compared to VTM-6.0
 - 4 multiples, 20 additions, 12 shifts, 4 max functions and 4 functions are saved at most for each uni-prediction Affine CU in encoder and decoder
 - 0.00% and -0.02% coding performance change compared to VTM-6.0 with RA and LDB configuration.

- It is recommended to adopt it to VVC.

Thanks!

