<5th JCT-VC meeting @Geneva, CH, March 2011>

[JCTVC-E223]

Single Interpolation for Multi-sample Prediction for Intra Coding

Jinho Lee

jinosoul@etri.re.kr

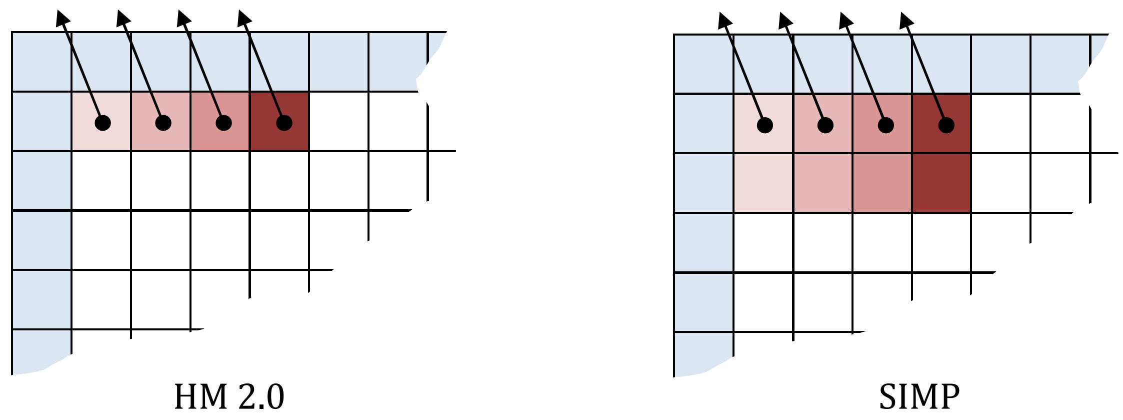
Realistic Media Research Team

ETRI

Contents

Summary

- ❑ **Purpose: Reduce computational complexity in Intra prediction**
- ❑ **Proposal: Single Interpolation for Multi-sample Prediction (SIMP)**

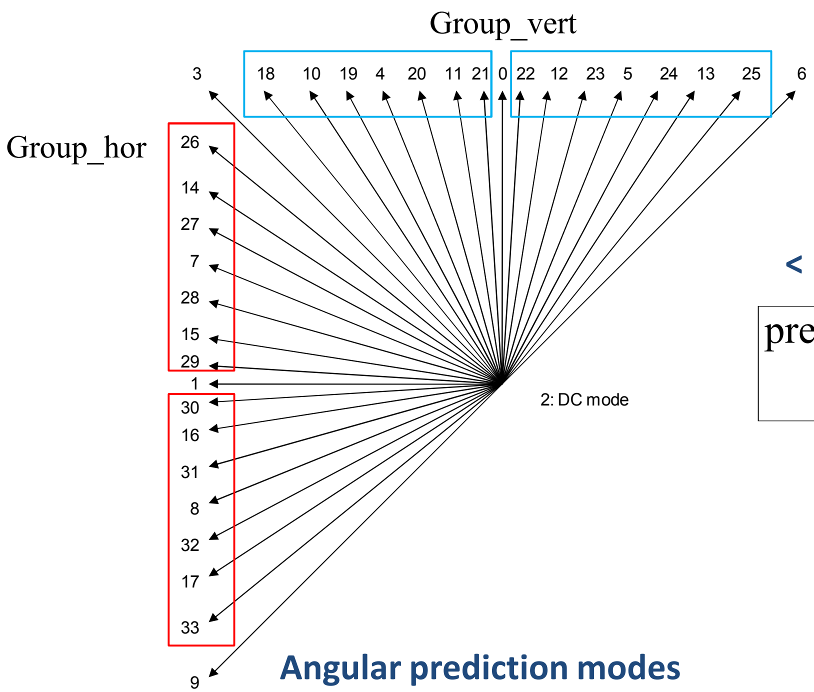


- ❑ **Introduce 4 methods of SIMP according to**
  - ❖ number of predicted samples
  - ❖ position for interpolation process
- ❑ **Results**
  - ❖ Number of interpolation process in PU_32x32 can be reduced nearly ½ or ¼ compared to HM
  - ❖ Coding loss caused by SIMP is 0.1% or 0.2%.

Introduction

❑ Intra prediction in HM2.0

- ❖ Vertical, Horizontal, DC, and Angular prediction
- ❖ Modes of 'Group_vert' and 'Group_hor' include the interpolation process



Angular prediction modes including the interpolation process

< Interpolation equation >

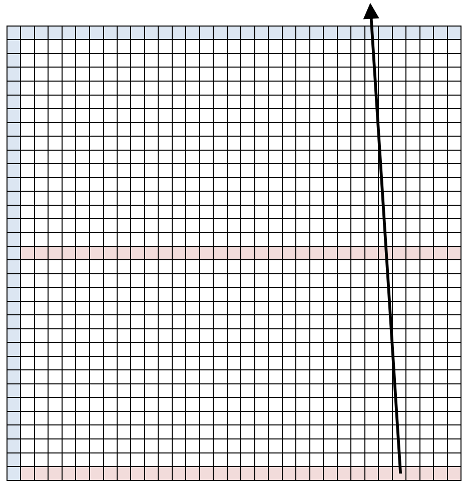
$$predSamples[x, y] = ((32 - iFact) * refMain[refMainIndex] + iFact * refMain[refMainIndex + 1] + 16) >> 5$$

Interpolation process in HM

❑ The number of interpolation process in PU_32x32

- ❖ 1 or 2 lines are predicted by copy of corresponding reference samples located in the integer position.
- ❖ Number of samples predicted by the interpolation process
  - ➢ 32*31 = 992
  - ➢ 32*30 = 960

$$predSamples[x, y] = ((32 - iFact) * refMain[refMainIndex] + iFact * refMain[refMainIndex + 1] + 16) >> 5$$



< ex) Mode 21 in PU_32x32 >

< The required number of operations to perform interpolation process in PU_32x32 (HM) >

Operation	Number of operations for one interpolation (A)	Number of interpolation for one mode (B)	Number of operations for one mode (A*B)
+, -	4	992 or 960	3968 or 3840
*	2		1984 or 1920
>>	1		992 or 960

Single Interpolation for Multi-sample Prediction

❑ Motivation & Approach

- ❖ Correlation of neighboring sample values in large PU size is higher than small PU size.
- ❖ Predicting multi-sample through single interpolation can reduce the computational complexity without significant coding loss.

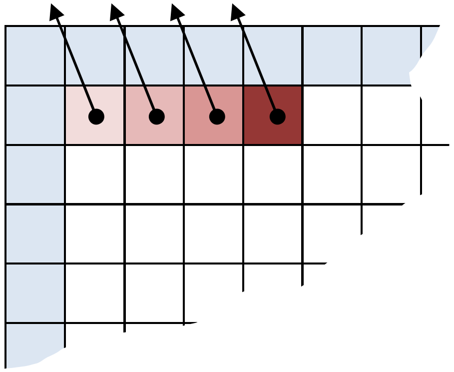
❑ Single Interpolation for Multi-sample Prediction (SIMP)

- ❖ Single interpolation for 2 samples prediction (SIMP_2)
  - ➢ Method 1 (M1): Interpolation at 'above' or 'left' sample
  - ➢ Method 2 (M2): Interpolation at the center of the 2 samples
- ❖ Single interpolation for 4 samples prediction (SIMP_4)
  - ➢ Method 3 (M3): Interpolation at 'left-above' sample
  - ➢ Method 4 (M4): Interpolation at a sample closed to reference samples
- ❖ Apply the proposed methods to PU_32x32.

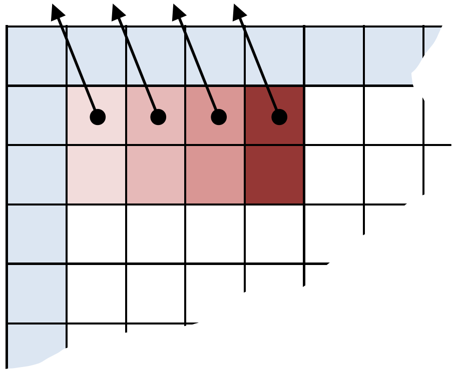
Single Interpolation for 2 samples Prediction (SIMP_2)

❑ Method 1 (M1)

- ❖ Position for single interpolation: a sample closed to reference samples



HM 2.0



M1

predSamples[x, y] =
predSamples[x, y + 1] = ((32 - iFact) * refMain[refMainIndex] +
iFact * refMain[refMainIndex + 1] + 16) >> 5

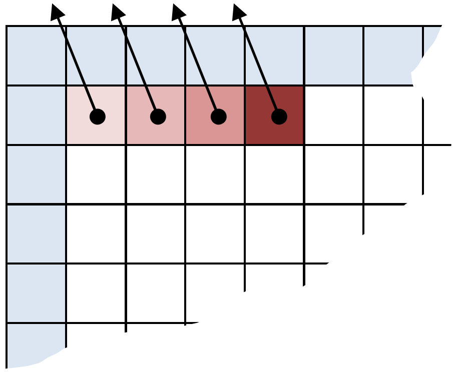
< The required number of operations to perform interpolation process in PU_32x32 (M1) >

Operation	Number of operations for one interpolation (A)	Number of interpolation for one mode (B)	Number of operations for one mode (A*B)		Number of operations for one mode [HM]
+, -	4	512 (32*16)	2048	Nearly ½	3968 or 3840
*	2		1024		1984 or 1920
>>	1		512		992 or 960

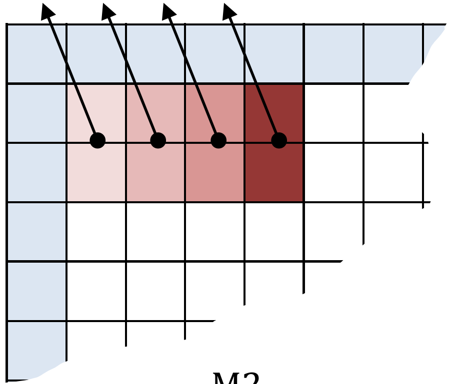
Single Interpolation for 2 samples Prediction (SIMP_2)

❑ Method 2 (M2)

- ❖ Position for single interpolation: center of the 2 samples



HM 2.0



M2

$$
\begin{aligned}
&predSamples[x, y] = \\
&predSamples[x, y+1] = ((32 - iFact) * refMain[refMainIndex] + \\
&\qquad\qquad iFact * refMain[refMainIndex+1] + 16) >> 5
\end{aligned}
$$

< The required number of operations to perform interpolation process in PU_32x32 (M2) >

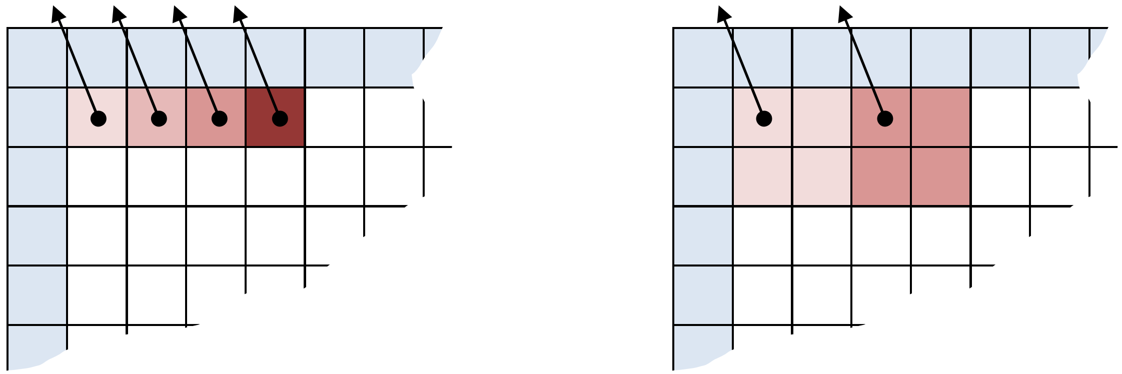
Operation	Number of operations for one interpolation (A)	Number of interpolation for one mode (B)	Number of operations for one mode (A*B)	Number of operations for one mode [HM]
+, -	4	512 (32*16)	2048	3968 or 3840
*	2		1024	1984 or 1920
>>	1		512	992 or 960

Nearly ½

Single Interpolation for 4 samples Prediction (SIMP_4)

❑ Method 3 (M3)

❖ Position for single interpolation: a left-above sample



HM 2.0 M3

$$predSamples[x, y] = predSamples[x, y+1] = predSamples[x+1, y] = predSamples[x+1, y+1] = ((32 - iFact) * refMain[refMainIndex] + iFact * refMain[refMainIndex+1] + 16) >> 5$$

< The required number of operations to perform interpolation process in PU_32x32 (M3) >

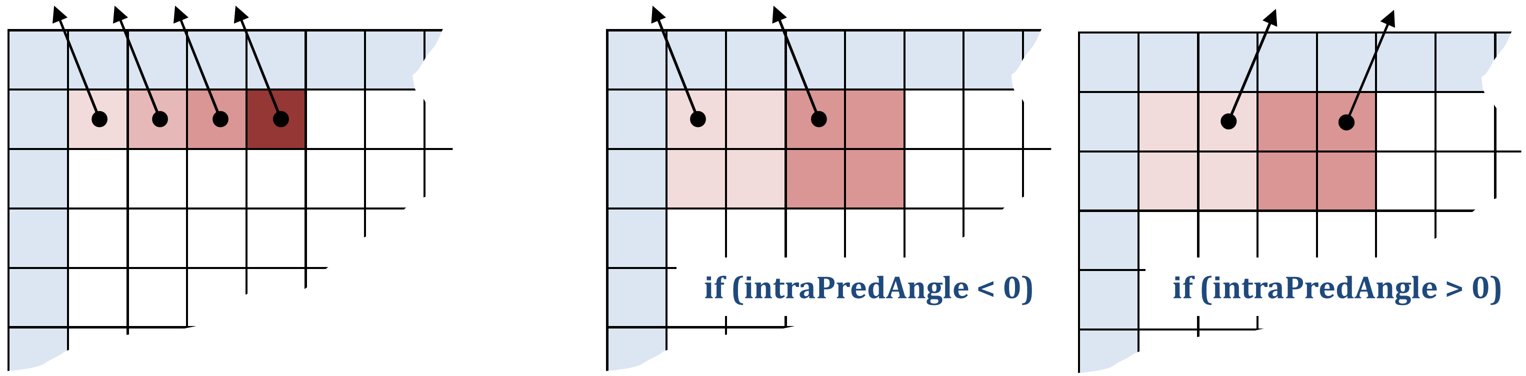
Operation	Number of operations for one interpolation (A)	Number of interpolation for one mode (B)	Number of operations for one mode (A*B)	Number of operations for one mode [HM]
+, -	4	256 (16*16)	1024	3968 or 3840
*	2		512	1984 or 1920
>>	1		256	992 or 960

Nearly ¼

Single Interpolation for 4 samples Prediction (SIMP_4)

Method 4 (M4)

- Position for single interpolation: a sample closed to reference samples according to prediction direction



HM 2.0 M4

< The required number of operations to perform interpolation process in PU_32x32 (M4) >

Operation	Number of operations for one interpolation (A)	Number of interpolation for one mode (B)	Number of operations for one mode (A*B)	Number of operations for one mode [HM]
+, -	4	256 (16*16)	1024	3968 or 3840
*	2		512	1984 or 1920
>>	1		256	992 or 960

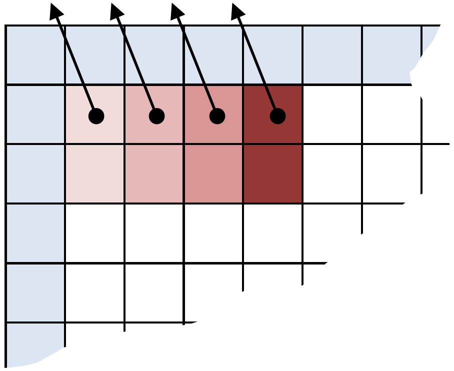
Nearly ¼

Experimental results

❑ Anchor: HM 2.0

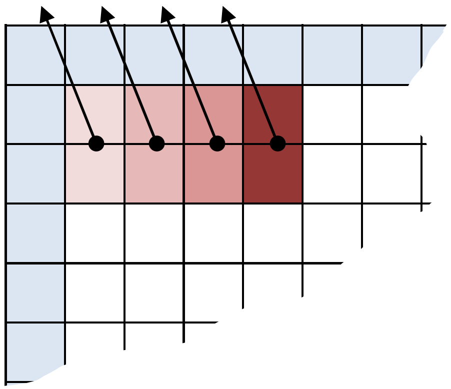
❖ M1

	Intra			Intra LoCo		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.1	0.0	0.1	0.0	0.0	0.0
Class B	0.0	0.0	0.0	0.0	0.0	0.0
Class C	0.0	0.0	0.0	0.0	0.0	0.0
Class D	0.0	0.0	0.0	0.0	0.0	0.0
Class E	0.1	0.1	0.1	0.0	0.0	0.1
All	0.1	0.0	0.1	0.0	0.0	0.0
Enc Time[%]	100%			99%		
Dec Time[%]	100%			99%		



❖ M2

	Intra			Intra LoCo		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.1	0.0	0.1	0.0	0.1	0.0
Class B	0.0	0.1	0.0	0.0	0.0	0.0
Class C	0.0	0.0	0.0	0.0	0.0	0.0
Class D	0.0	0.0	0.0	0.0	0.0	0.0
Class E	0.1	0.2	0.1	0.0	0.0	0.1
All	0.1	0.1	0.1	0.0	0.0	0.0
Enc Time[%]	#NUM!			#NUM!		
Dec Time[%]	99%			100%		

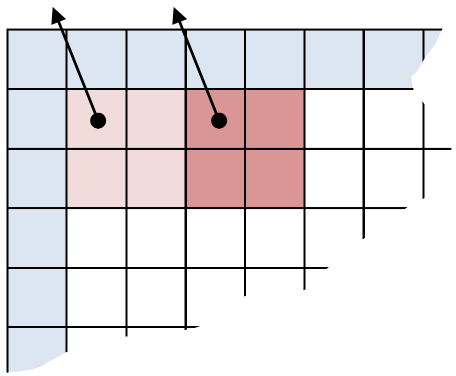


Experimental results

❑ Anchor: HM 2.0

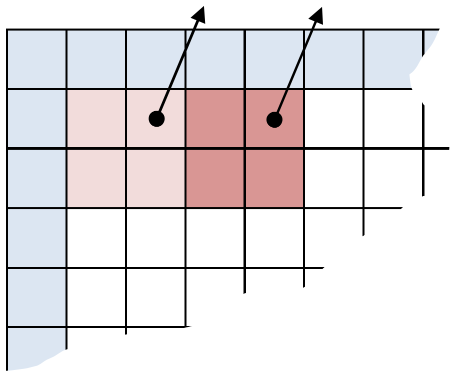
❖ M3

	Intra			Intra LoCo		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.4	0.5	0.6	0.1	0.3	0.3
Class B	0.2	0.4	0.4	0.2	0.2	0.3
Class C	0.1	0.1	0.1	0.1	0.1	0.1
Class D	0.0	0.1	0.0	0.0	0.0	0.0
Class E	0.5	0.8	0.7	0.3	0.4	0.5
All	0.2	0.3	0.4	0.2	0.2	0.2
Enc Time[%]	#NUM!			#NUM!		
Dec Time[%]	100%			100%		



❖ M4

	Intra			Intra LoCo		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.4	0.4	0.6	0.1	0.3	0.3
Class B	0.2	0.4	0.4	0.2	0.2	0.3
Class C	0.1	0.1	0.1	0.1	0.1	0.0
Class D	0.0	0.1	0.0	0.0	0.0	0.0
Class E	0.4	0.7	0.7	0.3	0.4	0.5
All	0.2	0.3	0.4	0.1	0.2	0.2
Enc Time[%]	#NUM!			#NUM!		
Dec Time[%]	99%			99%		



Conclusions

	SIMP_2		SIMP_4	
Operations	nearly half of HM		nearly one-fourth of HM	
	M1	M2	M3	M4
Luma BD-rate loss (HE/LC)	0.1/0.0 %	0.1/0.0 %	0.2/0.2 %	0.2/0.1 %

- **Changing interpolation position within the multi-sample**
  - It's not critical issue.
    → The correlation of neighboring samples in PU_32x32 is high.
- **Thank Qualcomm for cross-check of M1. (JCTVC-E360)**
- **Suggest M1 of this contribution to be adopted into the HM.**

Thank You Very Much !

www.etri.re.kr

Implementation in S/W

❖ M1

```
#ifdef ETRI_SIMP
 if (blkSize >= 32) {
  for (k=0;k<blkSize;k++) {
   deltaPos += intraPredAngle;
   deltaInt = deltaPos >> 5;
   deltaFract = deltaPos & (32 - 1);
   if (deltaFract) {
    // Do linear filtering
    for (l=0;l<blkSize;l++) {
     refMainIndex = l+deltaInt+1;
     pDst[k*dstStride+l]=
     pDst[(k+1)*dstStride+l]=(Pel) ( ((32-deltaFract)*refMain[refMainIndex]
                      +deltaFract*refMain[refMainIndex+1]+16) >> 5 );
    }
    k += 1;
    deltaPos += intraPredAngle;
   }
   else {
    // Just copy the integer samples
    for (l=0;l<blkSize;l++) {
     pDst[k*dstStride+l] = refMain[l+deltaInt+1];
    }
   }
  }
 }
 else {
#endif
```

❖ M4

```
#ifdef ETRI_SIMP
 if (blkSize >= 32) {
   Int step;
   if (intraPredAngle < 0) step=1;
   else step = 2;
   for (k=0;k<blkSize;k++) {
   deltaPos += intraPredAngle;
   deltaInt = deltaPos >> 5;
   deltaFract = deltaPos & (32 - 1);
   if (deltaFract) {
    // Do linear filtering
    for (l=0;l<blkSize;l++) {
     refMainIndex = l+deltaInt+step;
     pDst[k*dstStride+l]=
     pDst[k*dstStride+l+1]=
     pDst[(k+1)*dstStride+l+1]=
     pDst[(k+1)*dstStride+l]=(Pel) ( ((32-deltaFract)*refMain[refMainIndex]
                      +deltaFract*refMain[refMainIndex+1]+16) >> 5 );
    }
    k += 1;
    deltaPos += intraPredAngle;
   }
   else {
    // Just copy the integer samples
    for (l=0;l<blkSize;l++) {
     pDst[k*dstStride+l] = refMain[l+deltaInt+1];
    }
   }
  }
 }
 else {
#endif
```