SIMPLIFIED DC PREDICTOR IMPROVEMENT FOR DEPTH INTRA MODES

JCT3V-F0157

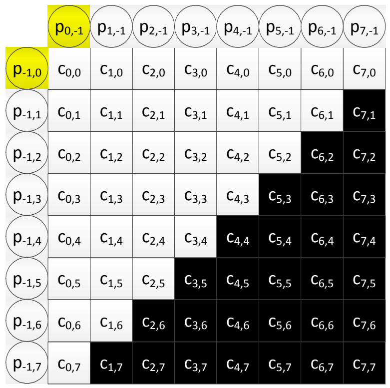
Zhouye Gu (Santa Clara University)
Jianhua Zheng (HiSilicon)
Nam Ling (Santa Clara University)
Philipp Zhang (HiSilicon)

Introduction

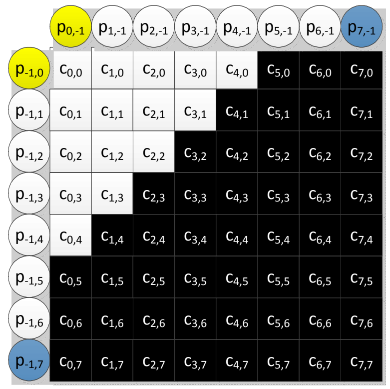
- Use additional partition patterns and reference pixels to improve DC prediction
- Implementation based on HTM-8.0:
- 0.1% gain (0.2% gain for Shark sequence) for synthesized view under All Intra case
- No impact under CTC

Thanks HHI for crosscheck (JCT3V-F0224)

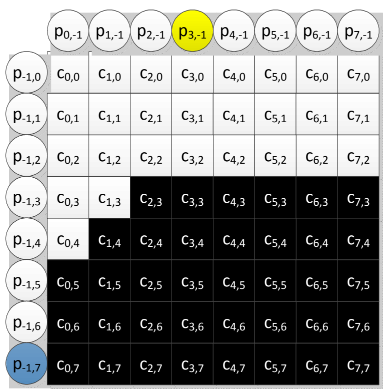
Current DC Prediction in HTM-8.0



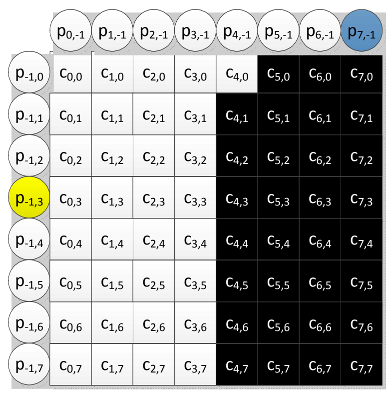
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Figure 1: Selection of reference samples for difference partition pattern cases

The DC value of each partition is predicted using one or two reconstructed neighboring reference samples depending on the partition pattern as shown in Figure 1

Set bT = ($bPattern_{0,0}$!= $bPattern_{N-1,0}$)? 1 : 0;

Set bL = ($bPattern_{0,0}$!= $bPattern_{0,N-1}$)? 1 : 0;

If bT equals bL

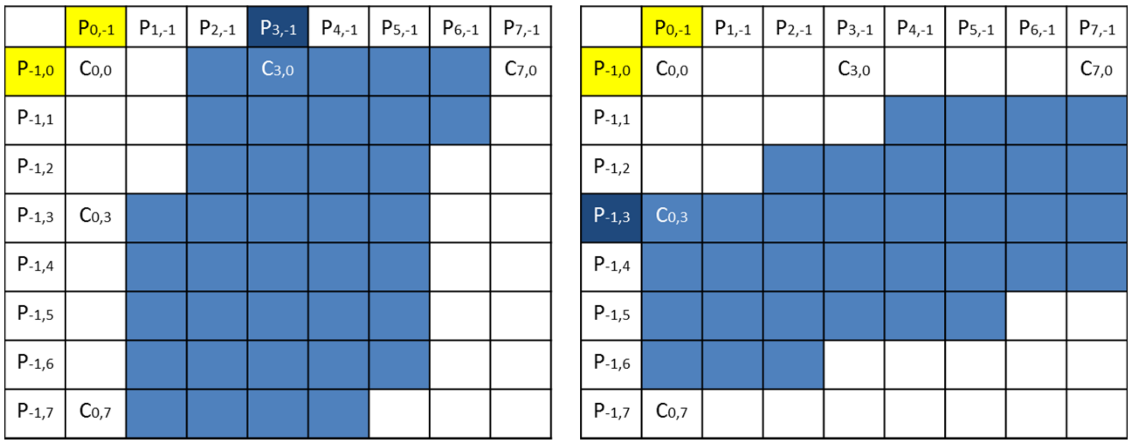
- $DC_X = (p_{-1,0} + p_{0,-1}) >> 1$
- $DC_{1-X} = bL \; ? \; (p_{-1,N-1} + p_{N-1,-1}) >> 1 : 2^{B-1}$

Otherwise

- $DC_X = bL \; ? \; p_{(N-1)>>1,-1} : p_{-1,(N-1)>>1}$
- $DC_{1-X} = bL \; ? \; p_{-1,N-1} : p_{N-1,-1}$

Proposed DC Prediction

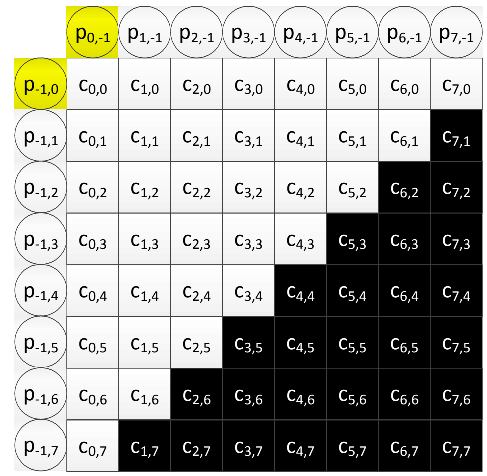
- Additional Partition Patterns



(a) Case 1a: ($c_{0,0}$), ($c_{7,0}$) and ($c_{0,7}$) have the same partition value which is different from the partition value of ($c_{3,0}$)

(b) Case 1b: ($c_{0,0}$), ($c_{7,0}$) and ($c_{0,7}$) have the same partition value which is different from the partition value of ($c_{0,3}$)

Figure 2. Selection of reference samples for additional partition pattern cases



Case 1 in Figure 1

Under case 1, we check the partition value of ($c_{0, (N-1)>>1}$) and ($c_{(N-1)>>1,0}$) to decide if it is a contour partition in DMM4 as shown in Figure 2.

If the partition is Case 1

Set bTM = ($bPattern_{0,0}$!= $bPattern_{(N-1)>>1,0}$)? 1 : 0;

Set bLM = ($bPattern_{0,0}$!= $bPattern_{0, (N-1)>>1}$)? 1 : 0;

If bTM does not equal to bLM

DC_{1-X} = bLM ? $p_{-1, (N-1)>>1}$: $p_{(N-1)>>1,-1}$;

Otherwise

DC_{1-X} = ($p_{-1, (N-1)>>1}$ + $p_{(N-1)>>1,-1}$) >> 1;

Proposed DC Prediction

- Additional Reference Pixels

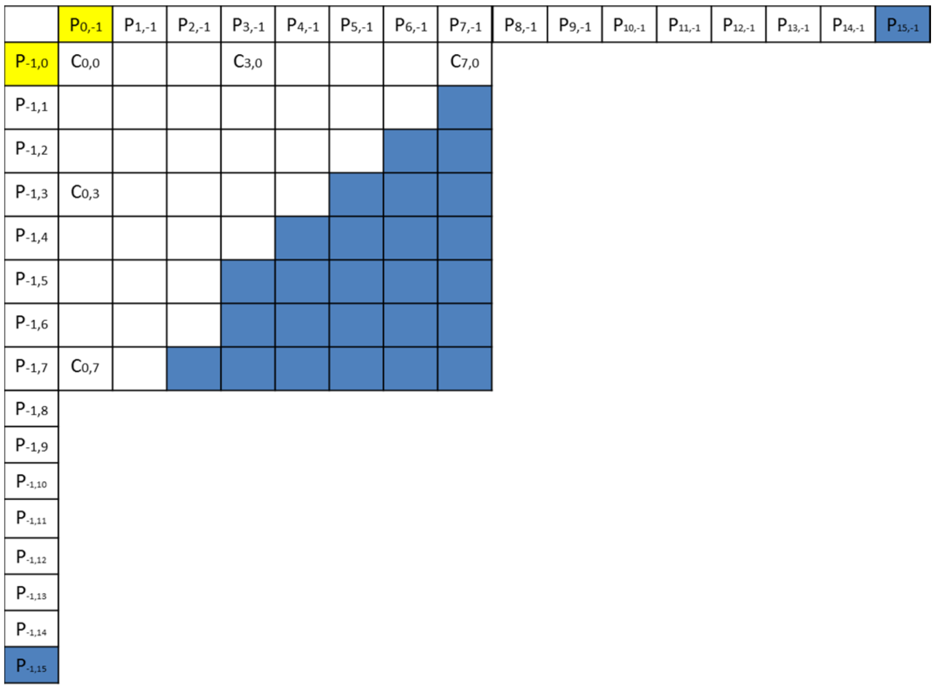
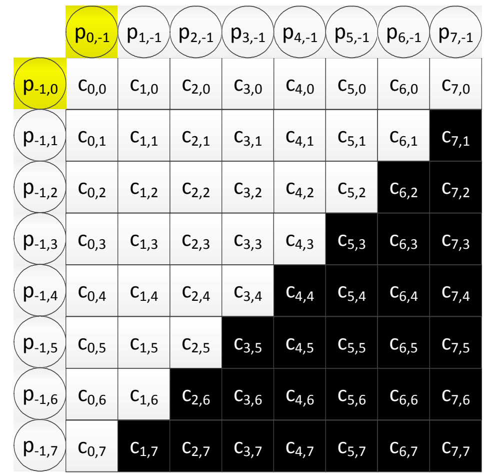


Figure 3. Selection of reference samples from neighboring

Under case 1, if the partition value of ($c_{0,\ (N-1)>>1}$) and ($c_{(N-1)>>1,0}$) are the same as ($c_{0,\ 0}$), we will check the availability of pixels at ($p_{-1,2*N-1}$) and ($p_{2*N-1,-1}$). If both ($p_{-1,2*N-1}$) and ($p_{2*N-1,-1}$) are available, we will use them as DC predictor



Case 1 in Figure 1

If $p_{-1,2*N-1}$ and $p_{2*N-1,-1}$ are both available

$DC_{1-X} = abs(p_{-1,2*N-1} - p_{-1,0}) > abs(p_{2*N-1,-1} - p_{0,-1})$? $p_{-1,2*N-1} : p_{2*N-1,-1}$

HTM-8.0 All Intra test result

	video 0	video 1	video 2	video PSNR / video bitrate	video PSNR / total bitrate	synth PSNR / total bitrate	enc time	dec time	ren time
Balloons	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	98.7%	88.9%	90.9%
Kendo	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	99.5%	85.1%	95.7%
Newspaper_CC	0.0%	0.0%	0.0%	0.0%	0.0%	-0.1%	98.5%	89.8%	97.9%
GT_Fly	0.0%	0.0%	0.0%	0.0%	-0.1%	-0.1%	99.3%	104.0%	102.5%
Poznan_Hall2	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	99.2%	94.5%	100.8%
Poznan_Street	0.0%	0.0%	0.0%	0.0%	0.0%	-0.1%	99.1%	98.9%	99.4%
Undo_Dancer	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	99.7%	105.1%	101.7%
1024x768	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	98.9%	87.9%	94.8%
1920x1088	0.0%	0.0%	0.0%	0.0%	0.0%	-0.1%	99.3%	100.6%	101.1%
average	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**-0.1%**	**99.1%**	**95.2%**	**98.4%**

Shark	0.00%	0.00%	0.00%	0.00%	-0.10%	-0.22%	99.3%	87.6%	96.0%

HTM-8.0 Random Access (CTC) test result

	video 0	video 1	video 2	video PSNR / video bitrate	video PSNR / total bitrate	synth PSNR / total bitrate	enc time	dec time	ren time
Balloons	0.0%	-0.1%	0.1%	0.0%	0.0%	0.0%	100.0%	96.0%	95.6%
Kendo	0.0%	-0.1%	-0.2%	0.0%	0.0%	-0.1%	99.7%	98.3%	96.0%
Newspaper_CC	0.0%	0.0%	0.2%	0.0%	0.0%	0.0%	99.9%	96.6%	101.2%
GT_Fly	0.0%	0.1%	0.0%	0.0%	0.0%	0.1%	97.9%	76.5%	95.2%
Poznan_Hall2	0.0%	0.0%	0.1%	0.0%	0.0%	-0.1%	99.8%	113.1%	103.6%
Poznan_Street	0.0%	-0.1%	0.1%	0.0%	0.0%	-0.1%	99.1%	91.8%	91.3%
Undo_Dancer	0.0%	0.1%	0.0%	0.0%	0.0%	-0.1%	97.8%	106.8%	99.5%
1024x768	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	99.8%	97.0%	97.6%
1920x1088	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	98.7%	97.0%	97.4%
average	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**99.2%**	**97.0%**	**97.5%**

Shark	0.00%	-0.15%	0.08%	-0.01%	-0.04%	-0.08%	98.4%	100.1%	95.0%

Change in WD

H.8.4.4.2.11 Depth partition value derivation and assignment process

– the neighbouring samples p[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,
– a binary array partitionPattern[x][y], with x, y =0..nTbS − 1, specifying a partitioning of the prediction block in a partition 0 and a partition 1.
– a variable nTbS specifying the transform block size,
– a flag dcOffsetAvailFlag, specifying whether DC Offset values are available,
– a flag intraChainFlag, specifying whether the current intra prediction mode is equal to INTRA_CHAIN
– the variables dcOffsetP0 and dcOffsetP1, specifying the DC offsets for the block partitions

Output of this process is:

– the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

The variables vertEdgeFlag and horEdgeFlag are derived as specified in the following:

vertEdgeFlag = (partitionPattern[0][0] != partitionPattern[nTbS − 1][0]) ? 1 : 0 (H 45)

horEdgeFlag = (partitionPattern[0][0] != partitionPattern[0][nTbS − 1]) ? 1 : 0 (H 46)

The variables dcVal0 and dcVal1 are derived as specified in the following:

– If vertEdgeFlag is equal to horEdgeFlag, the following applies:

  dcValBR = horEdgeFlag ?
    ((p[−1][nTbS − 1] + p[nTbS − 1][−1]) >> 1) : (1 << (BitDepthY − 1)) (H 47)

  dcValLT = (p[−1][0] + p[0][−1]) >> 1 (H-48)

  – If horEdgeFlag is equal to 0

    vertMidEdgeFlag = (partitionPattern[0][0] != partitionPattern[(nTbS – 1) >> 1][0]) ? 1 : 0

    horMidEdgeFlag = (partitionPattern[0][0] != partitionPattern[0][(nTbS – 1) >> 1]) ? 1 : 0

    – If vertMidEdgeFlag is not equal to horMidEdgeFlag

      dcValBR = horMidEdgeFlag ? p[-1][(nTbS – 1) >> 1] : p[(nTbS – 1) >> 1][-1]

    – Otherwise,
      dcValBR = (p[-1][(nTbS – 1) >> 1] + p[(nTbS – 1) >> 1][-1]) >> 1

    – If vertMidEdgeFlag is equal to 0 and horMidEdgeFlag is equal to 0

      –if p [nTbS*2 − 1][-1] is available and p [-1][nTbS*2 − 1] is available

        dcValBR = abs(p [0][-1] - p [nTbS*2 − 1][-1]) > abs(p [-1][0] - p [-1][nTbS*2 − 1]) ?
          p [nTbS*2 − 1][-1] : p [-1][nTbS*2 − 1]

– Otherwise (horEdgeFlag is not equal to vertEdgeFlag), the following applies:

  dcValBR = horEdgeFlag ? p[−1][nTbS − 1] : p[nTbS − 1][−1] (H-49)

  dcValLT = horEdgeFlag ? p[(nTbS – 1) >> 1][−1] : p[−1][(nTbS – 1) >> 1] (H-50)

Summary

- Additional partitions and reference pixels are considered to improve DC prediction
- Achieves 0.1% gain (0.2% gain for Shark sequence) for synthesized view under All Intra case
- No impact for CTC
- Suggest to adopt the proposal into the working draft