A LOW-COMPLEXITY HEVC INTRA PREDICTION ALGORITHM BASED ON LEVEL AND MODE FILTERING

Heming Sun, Dajiang Zhou and Satoshi Goto

Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Japan
Email: terrysun1989@akane.waseda.jp, zhou@fuji.waseda.jp, goto@waseda.jp

ABSTRACT

HEVC achieves a better coding efficiency relative to prior standards, but also involves increased complexity. For intra prediction, complexity is especially intensive due to a highly flexible coding unit structure and a large number of prediction modes. This paper presents a low-complexity intra prediction algorithm for HEVC. A fast preprocessing stage based on a simplified cost model is proposed. Based on its results, a level filtering scheme reduces the number of prediction unit levels that requires fine processing from 5 to 2. To supply level filtering decision with appropriate thresholds, a fast training method is also designed. A mode filtering scheme further reduces the maximum number of angular modes to be evaluated from 34 to 9. Complexity reduction from HM 3.0 is over 50% and stable for various sequences, which makes the proposed algorithm suitable for real-time applications. The corresponding bit rate increase is lower than 2.5%.

Index Terms— HEVC, intra prediction, video coding, real-time systems

1. INTRODUCTION

High Efficiency Video Coding (HEVC) [1] is a draft standard under development by Joint Collaborative Team on Video Coding (JCT-VC). It is aimed at 50% bit rate reduction relative to H.264. In H.264, intra prediction has been significant for video compression, while in HEVC, it still plays an important role. To improve the efficiency of HEVC intra coding, many new features are proposed, with two major differences from H.264. One is that the number of prediction block types in HEVC intra prediction can be at least 5 (from 64x64 down to 4x4), while the corresponding number for H.264 is only 3 (from 16x16 to 4x4). The other is that up to 34 angular prediction modes are defined in HEVC, much more than only 9 in H.264. These features results in significantly higher complexity for intra prediction. Therefore, efficient algorithms and architectures become even more essential.

There has been some related work aimed to reduce the complexity of H.264 intra prediction. A directional field based approach was reported by Pan, et al. [2] in which prediction modes are reduced according to an edge direction texture histogram. Based on that, Wei, et al. [3] further proposed a method to reduce the computation overhead for edge detection. Kim, et al. [4] suggested a method based on a multi-stage sequential mode decision process to filter out unlikely candidate modes. Huang, et al. [5] proposed a variance-based algorithm for block size decision and an improved filter-based algorithm for prediction mode decision. Tian, et al. [6] proposed a fast block type decision algorithm based on the entropy feature. Most of these proposals for H.264 cannot be directly applied in HEVC due to the differences in block types and prediction modes. Besides, the performance of some proposals such as [6] is highly dependent on sequence characteristics, so they cannot ensure a stable complexity for improving the worst-case performance.

The target of this work is to develop a fast intra prediction algorithm for HEVC which is suitable for real-time systems. Therefore it is necessary to consider the worst-case performance as first priority. This paper presents two proposals to achieve a stable complexity reduction while keeping video quality loss acceptable. Firstly, a level filtering scheme utilizes a fast pre-processing step to estimate the most likely prediction block types, so that the number of prediction levels that requires fine processing is reduced from 5 to 2. Moreover, a mode filtering scheme reuses the results from pre-processing to reduce the number of angular prediction modes to at most 9 for each remaining level. As a result, a steady complexity reduction of 50% or more is achieved in terms of prediction encoding time, while the bitrate increase is less than 2.5%.

The rest of this paper is organized as follows. Section 2 introduces intra prediction in HEVC. Section 3 and Section 4 present the proposed algorithms for level filtering and mode filtering, respectively. Section 5 shows the experiment results, followed by the conclusions in Section 6.

2. INTRA PREDICTION IN HEVC

2.1. Coding Unit Structure of HEVC

...
HEVC adopts a highly flexible hierarchical structure based on three kinds of basic units: coding unit (CU), prediction unit (PU) and transform unit (TU). A picture is divided into slices with each slice composed of a sequence of largest coding units (LCU). Each 2Nx2N CU can be divided to four NxN CUs recursively up to maximum coding unit depth. PU is provided as the basic unit for intra or inter prediction. There are different PU splitting strategies for a 2Nx2N CU. For intra prediction, two possible splitting patterns of 2Nx2N and NxN are supported. TU is the third basis unit defined for transform and quantization whose size is allowed to be independently from the PU. The maximum transform size can be extended to 64x64 while the minimum size for TU coding is 4x4.

2.2. Intra Prediction Algorithm in HM [7]

In HEVC Test Model (HM), LCU size is recommended as 64x64 and MaxPartitionDepth as 4. In this way, 4 types of CU sizes are possible. In addition, a PU of size 4x4 is also possible as a splitting strategy for an 8x8 CU. So there are altogether 5 types of prediction blocks, as shown in Fig. 1. From the LCU, each 2Nx2N (N=4, 8, 16, 32) unit is divided into four NxN units recursively, including 5 levels. Based on this hierarchy, the HM encoder recursively traverses all the combinations of CU, PU and TU to find the optimal structure.

2.3. Angular Intra Prediction in HEVC

As illustrated in Fig. 2, there are at most 34 intra prediction modes in HEVC. The number of support modes varies according to the prediction unit size, as shown in Table 1 [1]. HM takes two steps to get the best intra prediction mode for each unit. The first step is to pick up from all the angular directions a set of candidate modes with the minimum HAD (Hadamard transform Absolute Difference) costs. The number of candidate modes for each level is also listed in Table 1. Finally, from these candidates, the best prediction is selected as the mode with minimum R-D (Rate-Distortion) cost, denoted by \( J_{\text{RDO}} \):

\[
J_{\text{RDO}} = \text{SSD} + \lambda \cdot R
\]

where SSD denotes the sum of squared difference, and R denotes the total bits for encoding process. \( \lambda \) is related to quantization parameter(QP).

<table>
<thead>
<tr>
<th>The size of prediction unit</th>
<th>Number of angular intra direction modes</th>
<th>Number of candidate modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4x4</td>
<td>17</td>
<td>8</td>
</tr>
<tr>
<td>8x8</td>
<td>34</td>
<td>8</td>
</tr>
<tr>
<td>16x16</td>
<td>34</td>
<td>3</td>
</tr>
<tr>
<td>32x32</td>
<td>34</td>
<td>3</td>
</tr>
<tr>
<td>64x64</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

3. PROPOSED ALGORITHM FOR LEVEL FILTERING

3.1. Fast Pre-processing Stage

Although R-D cost is near optimal for intra mode decision, it involves large computational complexity, which makes it unsuitable for the fast pre-processing stage. Therefore, a low-complexity cost alternative to the R-D cost is desirable.

We start from the simpler HAD-based cost function:

\[
J_{\text{HAD}} = \text{SATD} + \lambda \cdot R
\]
where the sum of absolute transformed differences (SATD) provides an accurate cost evaluation. $\lambda \cdot R$ is related to QP and total bits during encoding.

To reduce the complexity of SATD, we calculate SATD based on the original, instead of reconstructed pixels. In the meanwhile, critical data dependency from the reconstruction loop can be eliminated, which is likely to ease the design of both hardware and parallel software. When doing the prediction for each unit, one SATD cost vector ($CV\_SATD$) is defined. Each $CV\_SATD$ has 34 elements with each element being the estimated SATD cost ($SATD'$) under the corresponding intra mode (from 0 to 33) by original pixels. After obtaining $CV\_SATD$, the minimum element of each $CV\_SATD$ indicates the $SATD'$ for this unit.

Firstly, the $CV\_SATD$ for 8x8 units are generated based on original pixels. Moreover, we prefer to use existing 8x8 $CV\_SATD$ to estimate $CV\_SATD$ for larger PUs rather than calculate them straightforwardly. Therefore, the $CV\_HAD$ for 2Nx2N (N=8, 16, 32) PU can be estimated by the sum of $CV\_HAD$ for individual 8x8 units that belong to the 2Nx2N unit, shown as followed:

$$ CV\_SATD_{8x8} = \sum_{i=0}^{3} CV\_SATD_{ij} $$

(3)

$$ CV\_SATD_{16x16} = \sum_{i=0}^{15} CV\_SATD_{ij} $$

(4)

$$ CV\_SATD_{64x64} = \sum_{i=0}^{63} CV\_SATD_{ij} $$

(5)

After applying the estimation, equation (2) can be rewritten as:

$$ J_{HAD} = SATD' + \Delta E + \lambda \cdot R $$

(6)

Here $SATD'$ is the estimation to SATD with $\Delta E$ being the estimation error. Note that $SATD'$ is usually smaller than SATD since the former is from the prediction based on small blocks which utilizes reference pixels nearer and therefore likely closer to the current pixels. As a result, $\Delta E$ tends to be positive. In additional, $\Delta E$ increases with prediction unit size 2N, since the estimation of $SATD'$ is performed from lower 8x8 level to higher 2Nx2N levels, while the error also accumulates.

$\lambda \cdot R$ is the remaining part of equation (2). Obviously, $\lambda$ is related to QP. In mode decision between prediction levels, $R$ can be regarded as mainly related to prediction unit size 2N, since larger PU involves significantly less mode information to be written to the bit rate.

Considering the strong relations between $\Delta E$ and 2N, and between $\lambda \cdot R$ and 2N together with QP, we can estimate $(\Delta E + \lambda \cdot R)$ as a function of 2N and QP, so that:

$$ J_{HAD} = SATD' + f(QP, 2N) $$

(7)

In HM, whether a 2Nx2N unit is divided into four NxN units or not is determined by the cost of adjacent levels recursively. Based on approximate HAD based cost shown in equation (7), the cost of split or not for a 2Nx2N unit is shown as follows:

$$ J_{non-split} = SATD_{2N} + f_{2N}(QP, 2N) $$

(8)

$$ J_{split} = \sum \{ SATD_{N} + f_{N}(QP, N) \} $$

(9)

where $\sum$ means the sum of $J_{HAD}$ for individual split NxN unit. We define $SATD_{2N}' \sim \sum SATD_{N}'$ as $\Delta C_{2N}$, and then the determination condition for splitting 2Nx2N is shown as follow:

$$ \begin{cases} 
\text{non-split 2Nx2N, } & \Delta C_{16} < \sum f_i(QP,N) - f_{16}(QP,2N) = T(QP,2N) \\
\text{split 2Nx2N, } & \Delta C_{16} > \sum f_i(QP,N) - f_{16}(QP,2N) = T(QP,2N) 
\end{cases} $$

(10)

where $T(QP,2N)$ is considered as a threshold, which is a function of QP and unit size 2N.

### 3.2. Training Method for Obtaining Thresholds

Theoretically, the optimal threshold can be obtained by training as shown in Fig. 3. The threshold combination with best coding efficiency should be the optimal one. However, for each round of the training process, we have to go through a complete encoding process which is computationally intensive. In addition, a total of 3 thresholds are involved, which results in huge number of combinations of thresholds. This may not be reasonable in practical use.
Here we present a fast training method. The original criterion for the optimal thresholds is to minimize the rate distortion. Alternatively, we aim at the closest decision results as HM. By taking HM’s decision on whether one 2Nx2N unit is split or not as the “correct” choice, we try to get a set of thresholds that minimize the “error” rate. For this purpose, we incorporate into HM a data statistic component for offline training of \( \Delta C_{2N} \) for the split determination of each 2Nx2N unit, and the corresponding determination results. Fig. 4 plots an example of the conditional probability distributions of \( P(\text{split} | \Delta C_{2N}) \) and \( P(\text{non-split} | \Delta C_{2N}) \). Then the error rate can be defined as follows:

\[
E(T) = \int_{\Delta C_{2N}=0}^{\infty} P(\text{split} | \Delta C_{2N}) + \int_{\Delta C_{2N}>T} P(\text{non-split} | \Delta C_{2N}) \tag{11}
\]

where \( E \) denotes error rate and \( T \) is the threshold.

Therefore, the threshold which can satisfied with \( \frac{dE}{dT} = 0 \) and \( \frac{d^2E}{dT^2} > 0 \) is the best threshold which can also be regarded as the intersection of two curves.

Then for each 2Nx2N unit, if \( \Delta C_{2N} \) is larger than the off-lined trained threshold, it is considered that it should split to NxsNs. Otherwise, 2Nx2N prediction unit size is suitable for encoding this 2Nx2N unit.

Following this approach, for each sequence andQP value in the training set, only one encoding iteration is required to get a whole set of thresholds. Although this may not ensure the optimum results as the method in Fig. 3 provides, the complexity is much more reasonable.

### 3.3. Summary of Proposed Level Filtering Algorithm

Generally, there are 5 levels from 64x64 to 4x4. The filtering judgment for each unit of each level can be obtained according to the decision mechanism in (10), based on \( \Delta C_{2N} \) acquired in the fast pre-processing (FP) stage together with already trained \( T(QP,2N) \). However, the FP system is based on fast HAD cost, while the optimal criterion is R-D cost. Therefore, only some unfiltered levels in FP system are required to calculate R-D cost to make the final decision of coding structure.

The pseudo codes in Algorithm 1 described the overall flow. If 2Nx2N level is decided to be filtered in FP, the fine processing including R-D cost for this level is eliminated and directly split to the lower level. Otherwise 2Nx2N PU is likely to be suitable for predicting this unit in R-D cost based criterion, so that it is required to calculate the R-D cost of this 2Nx2N PU. Considering the risk of missing the best coding structure generated by smaller PUs, the R-D cost generated by the lower levels is used to compare with the R-D cost of 2Nx2N PU. The R-D cost by the lower levels is defined as the sum of R-D cost of four split NxN PU as shown in Algorithm 2. By the comparison of the R-D costs of the neighboring 2 levels, the branch structure of the current 2Nx2N unit is constructed as either itself or the four splits of it.

### 4. PROPOSED ALGORITHM FOR MODE FILTERING

After doing the level filtering, we can filter some unlikely levels. However, for some remaining levels as 32x32, 16x16 and 8x8, we have to go through all the 34 intra modes by calculating HAD-based cost to select the candidate modes, as shown in Table 1. It will cost large computation. Therefore, we want to reduce the number of modes supported for HAD-based cost computation.

In the process of the pre-processing, we can get \( CV_{\text{SATD}} \) for each 2Nx2N unit (N=4, 8, 16) according to the equation (4) and (5). Although the element of \( CV_{\text{SATD}} \) represents the \( SATD' \) by original pixels quickly, it can still be utilized to reflect the relative value of precise HAD cost in the case of different modes. Thus, for each unit we select 9 intra modes with the minimum 9 \( SATD' \) to do the precise HAD calculation for the selection of candidate modes. The flow of the proposal is shown in Fig. 5.

### 5. EXPERIMENTAL RESULTS

#### 5.1. Threshold Training

---

**Algorithm 1 Level filtering**

```plaintext
1: LevelFiltering(k)
2: {
3:   if k is not filtered by FP then
4:     upper ← RDcost(k)
5:     lower ← do(k)
6:     return min(RDcost(upper,lower))
7:     else
8:       for each split sub-block do
9:         LevelFiltering(sub-block)
10:      end for
11:     end if
12:   }
```

**Algorithm 2**

```plaintext
1: do(k)
2: {
3:   for each split sub-block do
4:     lower ← RDcost(sub-block)
5:   end for
6:   return lower
7: }
```
For each resolution, we select one sequence to process the training defined in section 3.2. The five selected sequences are listed in Table 2. For each sequence, 4 QP values, 22, 27, 32 and 37, are used. After obtaining the threshold combinations from all the sequences, we calculate the average thresholds of five sequences. Finally, we do the polynomial fitting to generate the threshold combinations for all the QP values. The thresholds finally adopted are listed in Fig. 6. There are three curves corresponding to three unit sizes.

5.2. The Performance of Proposed Algorithms

The proposed algorithms are integrated with HM 3.0. To have a more realistic environment as for real-time systems, the number of candidate modes served for R-D cost is set as 1 for each PU. The simulation is performed on a 6-core, 3GHz Xeon-based server. In this experiment, all 20 frames are intra coded by setting the period of I-frames to 1. The results are listed in Table 3, in which PSNR and bit-rate are measured using Bjontegaard’s method [8] for QP of 22, 27, 32 and 37.

Table 3. Performance comparison by the proposed algorithms for level filtering and mode filtering.

<table>
<thead>
<tr>
<th>Class</th>
<th>Sequence</th>
<th>(\Delta\text{Pred Time} (%))</th>
<th>BD-psnr (dB)</th>
<th>BD-rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Traffic</td>
<td>-55%</td>
<td>-0.1781</td>
<td>3.39</td>
</tr>
<tr>
<td></td>
<td>PeopleOnStreet</td>
<td>-54%</td>
<td>-0.1276</td>
<td>2.33</td>
</tr>
<tr>
<td>B</td>
<td>ParkScene</td>
<td>-56%</td>
<td>-0.1288</td>
<td>3.02</td>
</tr>
<tr>
<td></td>
<td>Cactus</td>
<td>-56%</td>
<td>-0.1206</td>
<td>3.28</td>
</tr>
<tr>
<td>C</td>
<td>BasketballDrill</td>
<td>-56%</td>
<td>-0.1118</td>
<td>2.40</td>
</tr>
<tr>
<td></td>
<td>BQMall</td>
<td>-56%</td>
<td>-0.0728</td>
<td>1.18</td>
</tr>
<tr>
<td></td>
<td>RaceHorses</td>
<td>-56%</td>
<td>-0.0881</td>
<td>1.40</td>
</tr>
<tr>
<td>D</td>
<td>BasketballPass</td>
<td>-55%</td>
<td>-0.1262</td>
<td>2.23</td>
</tr>
<tr>
<td></td>
<td>BlowingBubbles</td>
<td>-52%</td>
<td>-0.0625</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>RaceHorses</td>
<td>-52%</td>
<td>-0.0805</td>
<td>1.15</td>
</tr>
<tr>
<td>E</td>
<td>Vidyo1</td>
<td>-60%</td>
<td>-0.1414</td>
<td>2.84</td>
</tr>
<tr>
<td></td>
<td>Vidyo4</td>
<td>-60%</td>
<td>-0.1822</td>
<td>4.16</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>-56%</td>
<td>-0.1184</td>
<td>2.37</td>
</tr>
</tbody>
</table>

As shown in Table 3, the proposed algorithm achieves an average of 56% encoding time saving for intra prediction part with the performance loss being less than 2.5%, which can be considered acceptable. Since our target is to realize a fast algorithm for HEVC intra prediction which is suitable for real-time systems, thus it is important to ensure a good worst-case performance. We can see that at least 52% computational complexity can be reduced, with sequences.
such as BlowingBubbles and RaceHorses as the worst or near-worst cases. In addition, almost the same complexity reduction can be achieved regardless of the resolution or the sequence characteristics, which indicates that potential to apply the proposed algorithm in real-time applications.

The combined effect of mode and level filtering is shown in Table 3. Table 4 shows the respective contribution from level filtering scheme. We can achieve on the average 50% prediction encoding time saving with BD-rate about 2.34%. Mode filtering can reduce the computational complexity of HAD cost while level filtering can reduce the number of levels with R-D cost calculation. Mode filtering is not as effective as level filtering, mainly because R-D cost calculation takes the majority of complexity in intra prediction in current HEVC Test Model. On one hand, the mode filtering can achieve further 6% complexity reduction. On the other hand, the quality degradation from mode filtering is negligible.

In the pre-processing stage, SATD' of larger units are estimated by those of leaf 8x8 units rather than achieved straightforwardly. While using the straightforward SATD' instead of the estimated one in the experiments, the average BD-rate increase is 1.92%. Compared to the 2.37% from estimated SATD', the 0.45% loss does not seem very significant. Therefore, the estimation strategy is adopted to get SATD' to ease the computation of pre-processing.

<table>
<thead>
<tr>
<th>Prediction Levels</th>
<th>△Pred Time (%)</th>
<th>BD-psnr (dB)</th>
<th>BD-rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64/32</td>
<td>-72%</td>
<td>-0.9696</td>
<td>18.61</td>
</tr>
<tr>
<td>32/16</td>
<td>-62%</td>
<td>-0.4968</td>
<td>9.11</td>
</tr>
<tr>
<td>16/8</td>
<td>-54%</td>
<td>-0.1994</td>
<td>3.77</td>
</tr>
<tr>
<td>8/4</td>
<td>-47%</td>
<td>-0.2469</td>
<td>5.03</td>
</tr>
<tr>
<td>Proposed</td>
<td>-56%</td>
<td>-0.1184</td>
<td>2.37</td>
</tr>
</tbody>
</table>

With our level filtering scheme, only two neighboring prediction levels need cost computation. We realize that our proposal is only meaningful when its coding efficiency can be better than algorithms using two fixed levels. Table 5 shows the comparison. Using only large prediction units (64/32 and 32/16) results in high encoding time reduction but they also involve significant video quality loss. The algorithms fixed on lower prediction levels (16/8 and 8/4) achieves comparable complexity as our proposal, but the quality is not as good.

<table>
<thead>
<tr>
<th>Prediction Levels</th>
<th>△Pred Time (%)</th>
<th>BD-psnr (dB)</th>
<th>BD-rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64/32</td>
<td>-72%</td>
<td>-0.9696</td>
<td>18.61</td>
</tr>
<tr>
<td>32/16</td>
<td>-62%</td>
<td>-0.4968</td>
<td>9.11</td>
</tr>
<tr>
<td>16/8</td>
<td>-54%</td>
<td>-0.1994</td>
<td>3.77</td>
</tr>
<tr>
<td>8/4</td>
<td>-47%</td>
<td>-0.2469</td>
<td>5.03</td>
</tr>
<tr>
<td>Proposed</td>
<td>-56%</td>
<td>-0.1184</td>
<td>2.37</td>
</tr>
</tbody>
</table>

It has to be noted that the reason why we set up the experiments as in the paper is to have a more realistic environment as for real-time systems since full RDO is hardly used in real-time design. To clarify the effect of the proposals more, we did the experiments once again with the original HM as the baseline. In the new setup (the encoder performs the full RDO for 3 or 8 candidates as shown in Table 1), the average encoding time reduction can achieve 55%. The worst-case reduction is 48% (for BlowingBubbles) which just drops a little bit from the previous 52%. BD-rate increase is 2.27%, close too.

6. CONCLUSION

We have several proposals to realize a low-complexity algorithm for intra prediction in HEVC. By using the results from fast pre-processing, the number of prediction unit levels that requires fine processing is reduced from 5 to 2. The time of offline training for obtaining thresholds for mode filtering decision is also made reasonable. Also based on the pre-processing, the proposed mode filtering scheme further reduces the maximum number of angular modes to be evaluated from 34 to 9. In the meanwhile, the overhead of pre-processing is small due to our low-complexity cost model. Our intra prediction algorithm achieves 50% encoding time reduction relative to HM 3.0, while the corresponding bit rate increase is lower than 2.5%.

7. ACKNOWLEDGEMENT

This research was supported by the Semiconductor Technology Academic Research Center (STARC), Japan.

8. REFERENCES