Fast Prediction Unit Selection and Mode Selection for HEVC Intra Prediction

Heming SUN†,††, Dajiang ZHOU††, Peilin LIU†, Nonmembers, and Satoshi GOTO††, Fellow

1. Introduction

High Efficiency Video Coding (HEVC) [1] is a new video compression standard and it has already been formally published by ITU-T. The new standard aims at 50% bit rate reduction relative to the previous H.264/AVC standard. To achieve this target, many advanced coding tools are adopted [2],[3]. Since H.264/AVC, intra prediction has significant contribution to the overall coding performance, while it still plays an important role in HEVC. To achieve higher coding efficiency for HEVC intra coding, many new features are adopted with two major differences from H.264. One is that the number of prediction unit types supported is increased to 5 (from 64 × 64 down to 4 × 4), whereas only 3 kinds of prediction blocks (from 16 × 16 to 4 × 4) are conventionally used in the H.264 high profile. The other is that up to 35 angular intra prediction modes are defined in HEVC, much more than the 9 modes supported in H.264. With coding efficiency improved with these new features, they also result in significant increase in the computational complexity of intra prediction. Therefore, efficient algorithms and architectures become more essential and desirable in implementing support for the new standard.

To reduce the complexity of H.264 intra prediction, extensive research has been conducted. A directional field based approach was reported by Pan, et al.[4] in which prediction modes are reduced according to an edge direction texture histogram. Based on that, Wei, et al.[5] further proposed a method to reduce the computation overhead for edge detection. Kim, et al.[6] suggested a method based on a multi-stage sequential mode decision process to filter out unlikely candidate modes. Huang, et al.[7] proposed a variance-based algorithm for block size decision and an improved filter-based algorithm for prediction mode decision. Tian, et al.[8] proposed a fast block type decision algorithm based on the entropy feature.

However, most of the above algorithms for fast mode and block size decision were designed for H.264/AVC, which cannot be directly applied in HEVC due to the distinctions in block types and angular prediction modes. Several latest proposals have been reported to reduce the complexity of HEVC intra prediction. Chen et al.[9] executed chroma prediction in advance to generate a depth map, and thereby reduced the complexity of luma prediction. Wallen-dael et al.[10] exploited the neighboring intra modes to obtain a prioritization of the different modes. Similarly, Zhao et al.[11] made use of neighboring prediction information to eliminate unlikely angular modes. Zou et al.[12] presented a rate distortion optimized transform scheme to improve the coding efficiency. Jiang et al.[13] calculated the gradient directions and generated a gradient-mode histogram to do the fast mode decision. Motra et al.[14] used direction information of the co-located neighboring block of previous frame along with neighboring blocks of current frame to speed up intra mode decision. Silva et al.[15] calculated the predominant orientation of the edge of the current PU pixels to reduce the number of evaluated angular modes. Zhang et al.[16] presented a gradient-based fast decision algorithm for prediction unit size selection and mode selection. Xiong et al.[17] used the non-normalized histogram of oriented gradient (n-HOG) to select CU size. [18] can reduce the computational complexity by using the signaling information coded block flag (cbf) in the HEVC syntax. If cbf is zero after encoding a PU partition, the next PU process of...
encoding the CU can be eliminated. [18] can only be used in P or B slices. And it has already been included in HEVC Test Model (HM) 7.0.

Some these proposals, particularly [11] and [12], have already been incorporated into the latest versions of the HEVC Test Model (HM). [10] did not aim at the fast intra algorithm. The performance of [9] is highly dependent on sequence characteristics, so they cannot ensure a stable complexity for the sequences with various video properties. [13]–[15] only proposed the algorithms for the fast intra mode decision without prediction unit size selection, the huge complexity for the remained prediction units is still very critical.

Our fast intra prediction algorithm is designed based on HM 7.0. A low-overhead preprocessing stage employing a simplified cost model is first proposed. According to its results, a prediction unit size selection scheme is adopted to reduce the number of prediction unit levels that requires fine processing from 5 to 2. To supply the level filtering decision with appropriate thresholds, a fast training method is also designed. Still using the preprocessing results, a fast mode selection scheme reduces the maximum number of angular modes to evaluate from 35 to 8. This eliminates the necessity for fine Hadamard cost calculation and thereby further reduces complexity. To alleviate the mismatch between R-D cost function and the proposed low-complexity cost model observed from some high-resolution sequences, a $32 \times 32$ PU compensation scheme is further proposed, which is effective in preserving coding efficiency.

In comparison with original HM 7.0, the proposed algorithm achieves over 50% complexity reduction in terms of encoding time, with the corresponding bit rate increase lower than 2.0%. The complexity reduction is relatively stable and independent to sequence characteristics.

The rest of this paper is organized as follows. Section 2 gives a brief introduction of HEVC intra prediction. Section 3 describes the fast pre-processing stage with the low-complexity cost model and gives the summary for the prediction unit selection algorithm. Section 4 presents the proposed algorithms for the mode selection. Section 5 explains the $32 \times 32$ PU compensation scheme. Section 6 shows the experimental results, followed by the conclusions in Sect. 7.

2. Intra Prediction in HEVC

2.1 Coding Unit Tree Structure

HEVC adopts a highly flexible hierarchical structure based on three types 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, synonymous with coding tree block (CTB)). Each $2N \times 2N$ CU can be divided to four $N \times N$ 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 $2N \times 2N$ CU. For intra prediction, two possible splitting patterns of $2N \times 2N$ and $N \times N$ are supported. TU is the third basic unit defined for transform and quantization. TU size should be less than or equal to PU size in the HEVC intra prediction. The maximum transform size can be extended to $32 \times 32$ while the minimum size for TU coding is $4 \times 4$.

In HEVC Test Model (HM) [19], LCU size and maximum coding unit depth are by default set as $64 \times 64$ and 4. In addition to the four types of CUs, PU of size $4 \times 4$ is also possible as a splitting strategy for an $8 \times 8$ CU. Therefore, there are altogether 5 types (levels) of blocks to perform the intra prediction, as shown in Fig. 1. Based on this hierarchical structure, the HM encoder recursively traverses all the combinations for CU, PU and TU to find the optimal structure.

2.2 Intra Prediction in HEVC

As can be seen in Fig. 2, there are at most 36 HEVC intra prediction modes, including one dedicated to chroma prediction. The number of support modes is 35 for all the prediction unit size, as shown in the second column of Table 1.

R-D cost can be adopted as an optimal cost criterion for intra mode decision making. The cost function can be given as:

$$J_{\text{RDO}} = SSD + \lambda \cdot R$$  \hspace{1cm} (1)$$

where $SSD$ is the distortion given by the sum of the squared differences between the current and reconstructed blocks, $R$ is the encoded bit rate, and $\lambda$ is a coefficient related to QP, which indicates the ratio in significance between the distortion and rate.

Because the calculation of $J_{\text{RDO}}$ is computationally intensive, HM first uses a simpler Hadamard to traverse all of
the supported modes. The Hadamard cost function is given by:

$$J_{\text{HAD}} = \text{SATD} + \lambda \cdot R$$  \hspace{1cm} (2)$$

where SATD is the sum of absolute transformed differences, which represents distortion in the Hadamard cost.

In original HM, we have to go through all the 35 intra modes by calculating Hadamard-based cost to select 3 or 8 candidate modes at first. The number of selected candidate modes for various PU sizes is listed in the third column of Table 1. In addition, the most probable mode (MPM) derived from the intra modes of the above and left neighboring blocks will also be added into the candidate modes. These candidate modes are further evaluated in terms of their R-D cost to obtain a final best mode.

3. Proposed Algorithm for PU Size Selection

3.1 Fast Pre-Processing Stage

Although the R-D cost is nearly optimal, intra mode decision involve huge computational complexity, making it unsuitable for pre-processing. It is therefore useful to start with a simpler Hadamard cost function and reduce its complexity by using two schemes.

Firstly, original pixels rather than reconstructed pixels can be utilized to compute the sum of the absolute transformed differences (SATD). In the pre-processing part, the mode decision part has not been finished, so the reconstructed pixels cannot be used. By using the original pixels, the estimated SATD cost can be calculated to support the proposed PU size and mode selection algorithm, which is beneficial to the complexity reduction. Using original pixels can also eliminate the reconstruction process and reduces critical data dependency in the reconstruction loop, which makes implementation of parallel design for hardware or software easier.

The HEVC behavior uses the reconstructed pixels of the neighboring PUs as the neighboring pixels to compute the SATD for the current PU. In our proposal, after the pre-processing, some PUs (levels) and modes can be filtered. For the remained levels and modes, we also use the reconstructed pixels to do the fine-processing including the R-D cost calculation.

An SATD cost vector (CV_{SATD}) is employed to calculate the SATD of each unit. Each CV_{SATD} is composed of at most 35 elements and each element is the estimated SATD for each mode. After it obtains the CV_{SATD} of a PU, the minimum element estimates that unit’s minimum SATD.

The second scheme involves reducing the number of SATD calculations. The CV_{SATD} of only the 8 × 8 PU units are obtained by means of complete calculations. After this, rather than calculating the CV_{SATD} for the larger PUs, the existing 8 × 8 CV_{SATD} is used to estimate these values. For instance, the CV_{SATD} of a 2N × 2N (N = 8, 16) PU can be estimated by summation of the CV_{SATD} of the leaf 8 × 8 PUs. (3) and (4) show the estimation method, j means the specific larger 2N × 2N (N = 8, 16) PU while i represents the corresponding specific leaf 8 × 8 PU. Considering that the 64 × 64 level cannot bring any gain, this level is split without any processing. So there is no need to compute CV_{SATD} for PU 64 × 64.

$$\text{CV}_{\text{SATD}_{32}} = \sum_{i=0}^{15} \text{CV}_{\text{SATD}_{8}} \quad (j = 0, 1, 2, 3)$$  \hspace{1cm} (3)$$

$$\text{CV}_{\text{SATD}_{16}} = \sum_{i=0}^{8} \text{CV}_{\text{SATD}_{8}} \quad (j = 0, 1, 2, \ldots, 15)$$  \hspace{1cm} (4)$$

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

$$J_{\text{HAD}} = \text{SATD}' + \Delta \text{E} + \lambda \cdot R$$  \hspace{1cm} (5)$$

Here SATD’ is the estimation to SATD with ΔE 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 more likely to be closer to the current pixels. As a result, ΔE tends to be positive. In additional, ΔE increases with prediction unit size 2N, since the estimation of SATD’ is performed from lower 8 × 8 level to higher 2N × 2N levels, while the error also accumulates.

\begin{table}
\centering
\begin{tabular}{|c|c|c|}
\hline
PU size & # of intra modes & # of candidate modes \\
\hline
4x4 & 35 & 8 \\
8x8 & 35 & 8 \\
16x16 & 35 & 3 \\
32x32 & 35 & 3 \\
64x64 & 35 & 3 \\
\hline
\end{tabular}
\caption{Number of intra modes and candidate modes supported for various prediction units.}
\end{table}
In (5), $\lambda$ is closely related to $QP$. In deciding the mode between prediction levels, $R$ indicates coding bits that involve information for PUs of size $2N$, as larger PUs involve less mode information to be written to the bit rate.

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

$$J_{\text{HAD}} \approx SATD' + f(QP, 2N).$$

In HM, whether a $2N \times 2N$ unit is divided into four $N \times N$ units or not is determined by the cost of adjacent levels recursively. Based on approximate Hadamard based cost shown in Eq. (6), the cost of split or not for a $2N \times 2N$ unit is shown as follows:

$$J_{\text{non-split}} \approx SATD_{2N'} + f_{2N}(QP, 2N)$$
$$J_{\text{split}} \approx \sum (SATD_{N'} + f_N(QP, N))$$

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

$$\begin{align*}
\text{nonsplit} &\quad 2N \times 2N, \\
\Delta C_{2N} &\quad < \sum f_N(QP, N) - f_{2N}(QP, 2N) \triangleq T(QP, 2N) \\
\text{split} &\quad 2N \times 2N, \\
\Delta C_{2N} &\quad > \sum f_N(QP, N) - f_{2N}(QP, 2N) \triangleq T(QP, 2N)
\end{align*}$$

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 2 thresholds are involved, which results in huge number of combinations of thresholds. This may not be necessary 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 $2N \times 2N$ 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 $2N \times 2N$ unit, and the corresponding determination results. Figure 4 plots the normalized probabilities (NP) of split and non-split PUs under different $\Delta C_{2N}$. Normalized probability (NP) is plot for each $\Delta C_{2N}$ value by using:

$$\text{NP}_{\text{split}}(\text{delta cost}) = \frac{M_{\text{split}}(\text{delta cost})}{\sum_{i=0}^{\infty} M_{\text{split}}(i)}$$
$$\text{NP}_{\text{nonsplit}}(\text{delta cost}) = \frac{M_{\text{nonsplit}}(\text{delta cost})}{\sum_{i=0}^{\infty} M_{\text{nonsplit}}(i)}$$

From Fig. 4, it can be seen that CU is more likely to be non-split for small $\Delta C_{2N}$, while the NP of being split increases as $\Delta C_{2N}$ increases. These trends conform to the assumed decision mechanism described in (9). The combined error rate of the conditional probabilities $P(\text{split}|\Delta C_{2N})$ and $P(\text{non-split}|\Delta C_{2N})$ can be defined as:

$$E(T) = \int_{\Delta C_{2N}=0}^{T} P(\text{split}|\Delta C_{2N}) + \int_{\Delta C_{2N}=T}^{\infty} P(\text{non-split}|\Delta C_{2N})$$

where $E$ denotes error rate and $T$ is the threshold.

Therefore, the threshold which can satisfy 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 $2N \times 2N$ unit, if $\Delta C_{2N}$ is larger than the off-lined trained threshold, it is considered that it should split to $N \times N$s. Otherwise, $2N \times 2N$ prediction unit size.
is suitable for encoding this $2N \times 2N$ unit.

Following this approach, for each sequence and QP 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 Prediction Unit Selection Algorithm

Generally, there are 5 levels (PUs) from $64 \times 64$ to $4 \times 4$. The filtering judgment for each unit of each PU can be obtained according to the decision mechanism in (9), based on $\Delta C_{2N}$ acquired in the fast pre-processing (FP) stage together with already offline trained $T(QP, 2N)$. However, the FP system is based on fast Hadamard 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.

If $2N \times 2N$ level is not filtered in FP, $2N \times 2N$ 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 $2N \times 2N$ 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 $2N \times 2N$ PU. The R-D cost by the lower levels is defined as the sum of R-D costs of four split $N \times N$ PUs.

By the comparison of the R-D costs of the neighboring 2 levels, the branch structure of the current $2N \times 2N$ unit is constructed as either itself or the four splits of it. This situation is shown in the right case in the Fig. 5.

Otherwise, $2N \times 2N$ 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 $N \times N$ level and process four $N \times N$ units recursively, which is shown in the left case of the Fig. 5.

For one LCU, all the filter-flag for each unit can be achieved by the fast pre-processing. Then we process from higher level to lower level. Starting from LCU ($64 \times 64$), each $2N \times 2N$ unit is processed recursively as Fig. 5 shows. The final structure by this PU size selection scheme is still based on the generic quad-tree. However, only two neighboring levels need detail fine processing to get each leaf, while we have to go through all the five levels in HM.

### 4. Proposed Algorithm for Mode Selection

After doing the PU size selection, we can filter some unlikely levels. However, for some remaining levels as $32 \times 32$, $16 \times 16$ and $8 \times 8$, we have to go through all the 35 intra modes by calculating Hadamard-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 Hadamard-based cost computation.

In the process of the pre-processing, we can get $CVSATD$ for each $2N \times 2N$ unit ($N=4, 8, 16$) according to the Eqs. (3) and (4). Despite the element of $CVSATD$ represents the SATD' by original pixels quickly, it can still be utilized to reflect the relative value of precise Hadamard cost in the case of different modes. The modes having a smaller SATD’ are mostly likely to incur smaller Hadamard cost. In order to reduce the complexity of this approach, the modes with the smallest SATD’ are used as candidates. In this way, the precise Hadamard calculation stage can be eliminated. The number of candidate modes left by the mode selection is 8, 3 and 3 for $8 \times 8$, $16 \times 16$ and $32 \times 32$ PU, respectively. Since no CV is calculated for $4 \times 4$ PU in preprocessing, the mode selection process does not apply to $4 \times 4$ PUs.

It has to be noted that we did not ignore the most probable mode (MPM) in our proposals. After 3 or 8 modes are selected at first by the mode filtering algorithm, MPMs are also added for the final R-D cost evaluation. The flowchart of this proposed method is shown in Fig. 6.
5. 32 × 32 PU Compensation Strategy

As mentioned in Sect. 2.1, the largest TU size supported in HEVC can be 32 × 32. By using large TUs, a large PU (32 × 32 PU) can benefit a lot in terms of R-D cost, especially in high resolution sequences. However, this advantage will not be reflected in fast pre-processing (FP), which always estimates the 8 × 8 SATD costs, since the largest SATD supported by HM is 8 × 8. Therefore, we propose this 32 × 32 PU compensation strategy to compensate some 32 × 32 PUs.

One LCU (CTB) can be divided to four 32 × 32 CUs as shown in Fig. 7. To solve the above problem, a 32 × 32 PU compensation strategy can be applied to b0 and b3. For the b1 and b2, some filtered 32 × 32 levels in FP can then be selectively compensated based on the spatial features of adjacent b0 and b3.

Figure 8 shows the 32 × 32 PU compensation scheme for b0 and b3. Even if the 32 × 32 level is filtered by the FP system, this level will be fine-processed additionally. In this manner, it can be always determined whether 32 × 32 PU is the best coding structure for each b0 and b3.

For b1 or b2, if its 32 × 32 PU is filtered in pre-processing, the compensation strategy will be taken into account. The compensation strategies for b1 and b2 are the same, as shown in Fig. 9.

When processing each b1 or b2, their left and top neighboring 32 × 32 CUs have already been processed. So it can refer to the best coding tree of the left and top neighboring CU. If one of its neighboring CU (left or top) is partitioned as 32 × 32 PU, then the 32 × 32 PU should be considered into PU size selection for this current CU (b1 or b2). The reason is that the current CU is more likely to be partitioned into 32 × 32 PU if its neighboring CU is also partitioned as 32 × 32 PU. This case is shown in the path b of Fig. 9. In the other cases (path a or c), it will follow the PU size selection method in Sect. 3.3.

6. Experimental Results

6.1 Threshold Training

Based on test conditions recommended for HM, the test sequences were divided into five classes according to the resolution. For each class, one sequence was selected to perform the fast threshold training detailed in Sect. 3.2. These training sequences are listed in Table 2, and the rest sequences are used as testing sequences. For each sequence, four QP values were used: 22, 27, 32 and 37. As shown in (9), the threshold was based on both the CU size (2N) and the QP value. After obtaining the T(QP,2N) combinations from all the training sequences, we calculate the average T(QP,2N) of two low resolution (LR) training sequences.
Table 2  Training and testing sequences.

<table>
<thead>
<tr>
<th>Training sequences</th>
<th>Traffic, Cactus, BQMall, RaceHorses, KristenAndSara</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing sequences</td>
<td>PeopleOnStreet, Nebuta, StreamLocomotive, Kimono, ParkScene, BQTerrace, BasketballDrive, RaceHorses, PartyScene, BasketballDrill, BQSquare, BlowingBubbles, BasketballPass, FourPeople, Johnny</td>
</tr>
</tbody>
</table>

Table 3  T(QP,32) finally adopted with various QPs.

<table>
<thead>
<tr>
<th>QP</th>
<th>High Resolution</th>
<th>Low Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>285</td>
<td>102</td>
</tr>
<tr>
<td>27</td>
<td>293</td>
<td>111</td>
</tr>
<tr>
<td>32</td>
<td>301</td>
<td>119</td>
</tr>
<tr>
<td>37</td>
<td>309</td>
<td>128</td>
</tr>
</tbody>
</table>

Table 4  T(QP,16) finally adopted with various QPs.

<table>
<thead>
<tr>
<th>QP</th>
<th>High Resolution</th>
<th>Low Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>200</td>
<td>101</td>
</tr>
<tr>
<td>27</td>
<td>213</td>
<td>134</td>
</tr>
<tr>
<td>32</td>
<td>226</td>
<td>167</td>
</tr>
<tr>
<td>37</td>
<td>240</td>
<td>200</td>
</tr>
</tbody>
</table>

(BQMall and RaceHorses) according to the corresponding QP and 2N. After that, we do the polynomial fitting (degree is one) for all the QP values by MATLAB to generate the threshold combinations for LR (Class C&D). The method to gain the threshold combination for HR (Class A&B&E) is the same while the training sequences are Traffic, Cactus and KristenAndSara. T(QP,32) and T(QP,16) finally adopted are listed in Table 3 and Table 4, respectively. We can observe that the T(QP,32) adopted for low resolution is small. Thus, with smaller T(QP,32), more 32 × 32 levels will be split for the low resolution. This training result is quite reasonable considering that the 32 × 32 level is always split for the low resolution in original HM.

6.2 The Performance of Proposed Algorithm

The proposed algorithms were integrated with HM 7.0. We adopted the common test condition “Intra, main” recommended in [20]. The simulation was performed on a six-core, 3-GHz Intel Xeon-based server. In the experiment, we coded entire sequences for all five classes. The results of these assessments, in which the bit-rate were measured for QPs of 22, 27, 32, and 37 using Bjontegaard’s method [21], are listed in Table 5. The time reduction is evaluated as follows:

\[ \Delta \text{Time} = \frac{\text{Encoding time}_{\text{proposed}} - \text{Encoding time}_{\text{HM}}}{\text{Encoding time}_{\text{HM}}} \times 100\% \]

As shown in Table 5, the proposed algorithm achieves an average of 52% encoding time saving with the performance loss being less than 1.9%, which can be considered acceptable. For different sequences, we can achieve stable complexity reduction. We can see that at least 41% complexity can be reduced, with sequence Nebuta. In addition, a quite stable complexity reduction can be achieved for different sequences within each resolution.

The effect of mode selection scheme is shown in Table 6. Without the mode selection, about 48% time reduction can be achieved with BD-rate about 1.8%. Mode filtering is not as effective as level filtering (PU size selection) in terms of the complexity reduction. The reason is that mode filtering can reduce the complexity of Hadamard cost while
the level filtering can reduce the number of levels with R-D cost calculation. R-D cost takes the majority of encoding complexity. Even so, the mode filtering can still achieve 5% more complexity reduction with almost no performance loss.

By using the PU size selection method presented in Sect. 3.3, only two levels need fine-processing. After adopting the 32 × 32 compensation strategy proposed in Sect. 5, only two or three levels need fine-processing including R-D cost calculation. To further verify the effectiveness of our algorithms, we compare our proposals with the fixed two or three levels. The comparison result is shown in Table 7. Almost all the combination of fixed two or three levels cannot get the better coding efficiency than our algorithms. The fixed three levels (16&8&4) can achieve almost the same coding efficiency as our algorithms and it can achieve the BD-rate of 2.12%. However, our algorithm is much more attractive in terms of encoding time reduction (−52.01% vs. −16.49%).

Table 8 shows the performance comparison with two previous researches [16] and [17]. These two works also made a fast decision on the prediction unit size selection. [16] did achieve more complexity reduction than our proposals. However, the performance loss of [16] is quite large and the average BD-rate is 6.6%. For some specific sequence such as PeopleOnStreet, the corresponding BD-rate is up to 9.0%, which is not slight and acceptable. Compared with [17], our proposals can achieve almost the same good coding quality with about 15% more complexity reduction.

As mentioned before, our proposals can achieve a stable reduction for various sequences. Moreover, we can also ensure a stable time reduction for different resolutions. Table 9 shows the time reduction of different QPs, [17] could gain more time reduction under larger QPs. By using our proposals, a stable and considerable time reduction can be achieved under different QPs, which indicates the stability of our algorithms. In Table 10, we can observe that [17] can achieve about 52% reduction for Class E while only 27% reduction can be achieved for Class D. However, we can ensure a stable reduction for various resolutions (Class).

To compare with [18] (HM7.0), we have done the experiments based on the common test condition “lowdelay, main” and “randomaccess, main” which is recommended in [20]. And the function (Cbf fast mode setting) for [18] is turned on in HM7.0 during this experiment. Since our algorithm aims to the early termination for intra prediction, we compare with HM7.0 in terms of encoding time for intra prediction and bit-rates.

The results are shown in Table 11. By using the common condition “lowdelay, main”, average 56% intra encoding time can be reduced and the average BD-rate is only 0.88%. Under the common condition “randomaccess, main”, about 51 intra encoding time can be saved while the performance loss is only 1.44% in average.

6.3 The Effect of 32 × 32 PU Compensation

For the common test sequences in HM, the effect to high resolution (Class A&B&E) is shown in Table 12. The sequence Nebuta, SteamLocomotive, Kimono and Johnny can benefit a lot from this compensation. This was because using a large PU (32 × 32 PU) contributed significantly to the coding efficiency of these sequences. It can be concluded
that the proposed strategy is really effective to improve performance considerably at high resolution.

7. Conclusion

We have several proposals to realize a low-complexity algorithm for intra prediction in HEVC. By using the fast prediction unit selection scheme, the number of prediction unit levels that requires fine processing is reduced from five to two. To supply this PU size decisions, a fast off-line training method was also designed. Based on the benefit of using a large size transform, the 32×32 PU was selectively compensated and processed. For the remaining levels the proposed fast mode selection scheme used the information in the pre-processing in order to avoid the precise Hadamard cost calculation stage. The intra prediction proposed here achieves a greater than 50% reduction in encoding time relative to HM 7.0, while its corresponding bit rate increase is less than 2.0%.

Acknowledgments

This research was partly supported by Grand-in-Aid for Scientific Research (MEXT KAKENHI) and STARC.

References

[19] https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/ 
Heming Sun received the B.E. degree from Shanghai Jiao Tong University, Shanghai, China, in 2010 and the M.E. degree in engineering from Waseda University, Kitakyushu, Japan, through a double-degree program, in 2012. He is currently working toward the M.E. degree at Shanghai Jiao Tong University. His interests are in algorithms and VLSI architectures for multimedia signal processing.

Dajiang Zhou received B.E. and M.E. degrees in electronic engineering from Shanghai Jiao Tong University, China, in 2005 and 2008, respectively. In 2010 he earned his Ph.D. degree in engineering from Waseda University, where he is currently an assistant professor. Dr. Zhou’s major interests are in VLSI architectures and algorithms for multimedia signal processing. He has published over 50 technical papers on journals and conferences including IEEE JSSC, IEEE and IEICE Transactions, ISSCC and VLSI Circuits Symposium, etc. He holds 9 patents in Japan and China. He was a recipient of several awards including the best student paper award of VLSI Circuits Symposium 2010, the design contest award of ISLPED 2010, and the 2010 Chinese Government Award for Excellent Students Abroad.

Peilin Liu received the D.Eng. degree from the University of Tokyo, Tokyo, Japan, in 1998. She was a Researcher with the University of Tokyo, Tokyo, Japan, in 1999. From 1999 to 2003, she was a Senior Researcher with the Central Research Institute of Fujitsu, Tokyo, Japan, where her research topics included multimedia processing, IC design, and high-performance processor architecture. She joined Shanghai Jiao Tong University, Shanghai, China, in 2003, and is currently a Professor with the Department of Electronic Engineering. She directs the MediaSoc Lab and is responsible for a series of projects, such as HDTV SoC platform development and high-performance portable audio/video systems.

Satoshi Goto received the B.E. and M.E. degrees in electronics and communications engineering from Waseda University in 1968 and 1970, respectively. He received the Doctor of Engineering degree from Waseda University in 1981. He joined NEC laboratories in 1970, where he worked on LSI design, multimedia system and software as GM and VP. Since 2003, he has been a professor at the Graduate School of Information, Production and System, Waseda University, Kitakyushu, Japan. His main interests are in VLSI design methodologies for multimedia and mobile applications. He has published 7 books and more than 200 technical papers in international journals and conferences. Dr. Goto served as GC of ICCAD and ASPDAC and was board member of IEEE CAS society. He is a life fellow of IEEE, and a member of the Academy Engineering Society of Japan.