Journal of Electronic Imaging

JElectronicImaging.org

Improve load balancing and coding efficiency of tiles in high efficiency video coding by adaptive tile boundary

Chia-Hsin Chan
Chun-Chuan Tu
Wen-Jiin Tsai

Chia-Hsin Chan, Chun-Chuan Tu, Wen-Jiin Tsai, "Improve load balancing and coding efficiency of tiles in high efficiency video coding by adaptive tile boundary," *J. Electron. Imaging* **26**(1), 013006 (2017), doi: 10.1117/1.JEI.26.1.013006.

Improve load balancing and coding efficiency of tiles in high efficiency video coding by adaptive tile boundary

Chia-Hsin Chan,* Chun-Chuan Tu, and Wen-Jiin Tsai
National Chiao Tung University, Department of Computer Science, 1001 University Road, Hsinchu 30010, Taiwan

Abstract. High efficiency video coding (HEVC) not only improves the coding efficiency drastically compared to the well-known H.264/AVC but also introduces coding tools for parallel processing, one of which is tiles. Tile partitioning is allowed to be arbitrary in HEVC, but how to decide tile boundaries remains an open issue. An adaptive tile boundary (ATB) method is proposed to select a better tile partitioning to improve load balancing (ATB-LoadB) and coding efficiency (ATB-Gain) with a unified scheme. Experimental results show that, compared to ordinary uniform-space partitioning, the proposed ATB can save up to 17.65% of encoding times in parallel encoding scenarios and can reduce up to 0.8% of total bit rates for coding efficiency. *© 2017 SPIE and IS&T* [DOI: 10.1117/1.JEI.26.1.013006]

Keywords: high efficiency video coding; tile; encoder optimization; video compression.

Paper 16626 received Jul. 15, 2016; accepted for publication Dec. 16, 2016; published online Jan. 12, 2017.

1 Introduction

As the successor of H.264/AVC,[1] high efficiency video coding (HEVC)[2,3] drives video compression into a new era by providing an approximately 50% bit-rate reduction at an equal perceptual quality.[2–4] Under the same block-based hybrid video coding framework as H.264/AVC, the improvement of HEVC comes from careful redesign of existing coding tools and adoption of several innovative coding tools. However, the increase in the number of coding tools and modes also means the increase of complexity. In addition, the demand of high resolution content in various applications, which is one of the reasons for triggering the project of HEVC,[5,6] also puts more challenges on video coding. Due to the growth of multicore and heterogeneous computation architectures in hardware, parallel processing seems to be a good solution to deal with the increasing coding complexity and the higher content resolution. Therefore, HEVC also considers parallel-friendly coding, with several parallel coding tools adopted.[7] There are three main parallel coding tools in HEVC: slice, wavefront parallel processing (WPP),[8] and tile.[9,10]

Slice is a data structure that consists of consecutive coded tree units (CTUs)[11] covering either the entire picture or a region of a picture, and a picture is, hence, a collection of one or more slices. Each slice has its own slice header and can be decoded independently; therefore, partitioning a picture into slices makes the parallel coding of the picture become possible. Slice is also one of the coding tools in H.264/AVC. In WPP, each row of CTUs is defined to be a unit for parallelism called thread. Each row has a two-CTU coded delay from the above row so that inter-/intracoding process will remain unchanged. At last, a picture can be partitioned into CTU-aligned, independently coded rectangular regions called tiles. An example illustrating the picture partition of tiles is given in Fig. 1. Compared to slice, tile obtains better coding efficiencies since its rectangular partitioning design can retain more spatial correlation for the prediction process. Additionally, tile also has the advantage of sharing little header information stored in the picture parameter set and requiring less line buffer.[12–15] Tile can also cooperate with slice in the way that a picture is partitioned into tiles with each tile containing multiple slices and such combination reveals the possibility of processing ultrahigh definition (UHD) contents by using multiple coders with limited processing capabilities.[16] Moreover, the rectangular shape nature of tile can facilitate the region of interest (ROI)-based coding.[9]

While having simple design and various applications, tile partitioning is controlled by the encoder parameters and remains an open issue. In order to increase the flexibility of tile coding, tile boundary positions can be made frame by frame changeable and such a concept is named as adaptive tile boundary (ATB) in this paper. Such a concept also appears in the two previous works: Ahn et al.[17] improved the load balancing by trying to partition tiles with the same coding complexities. The complexity is estimated via the analysis of the reference encoder, and each coding unit (CU)[11] has its own complexity cost based on its coding mode and size. Then an adaptive partitioning method is used to decide the tile boundaries of current frames by updating from the tile boundaries of the preceding coded frame with the aim of reaching the best load-balanced partitioning using estimated complexity costs. However, complexities have to be re-estimated for different encoder algorithms, implementations, and platforms. Blumenberg et al.[18] improved the tile coding efficiency by avoiding the high correlated CTUs which are identified as the CTUs having similar pixel variances, being divided by tile boundaries. Their method records the differences of CTU variances for the whole frame, and the boundaries with the least "variance of CTU variance difference" will be selected to be the tile boundaries. However, estimating correlation by pixel variance is imprecise for

*Address all correspondence to: Chia-Hsin Chan, E-mail: terry0201@gmail.com

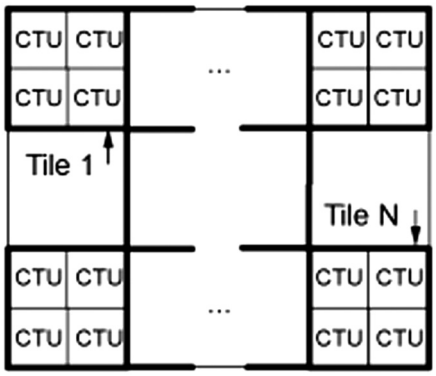


Fig. 1 Example illustrating picture partition using tiles.[2]

intercoded frames so this method works only for intracoded frames.

In this paper, first, the algorithm for tile coding time load balancing is proposed, which uses encoding time for complexity estimation. Then a multiple candidate approach with greedy strategy is used to ensure a better tile boundary selection. Second, the algorithm for tile coding efficiency is proposed for both intra- and interframes, which determines the tile boundaries based on heuristic costs of coding dependencies. The remainder of this paper is organized as follows. In Sec. 2, we discuss the reasons for imbalanced loading and degradation of coding efficiency in tiles. The proposed methods are described in Sec. 3. Experimental results are provided in Sec. 4, and Sec. 5 concludes this paper.

2 Unbalanced Loading and Coding Efficiency Degradation in Tiles

Tile is originally designed for parallel processing. A frame with tiles can be parallel encoded/decoded to shorten the overall processing time. HEVC in default recommends uniform-space tile partitioning with only one flag to be signaled. If tiles are not partitioned into uniform space, their boundary positions should be described explicitly and signaled one by one.

However, uniform-space partitioning cannot guarantee the best load balancing because the tiles having the same size does not mean they have equal complexities. Figure 2(a) shows the first frame of the sequence BQTerrace with uniform-space tile partitioning. Obviously, we can see that tile 2 is the most complex area in this frame and is expected to have the largest encoding time. Figure 2(b) is the actual encoding time of BQTerrace with uniform-space tile partitioning at QP 22. Even if having almost the same sizes, the encoding times are different for these four tiles and keeps varying along with the change of image contents. Although tile 2 and tile 0 have similar area sizes (not equal sizes because the sequence height is not an integer multiple of CTU size 64×64), the encoding time of tile 2 is about 1.5 times of that of tile 0 for the first frame, resulting in bad parallelism.

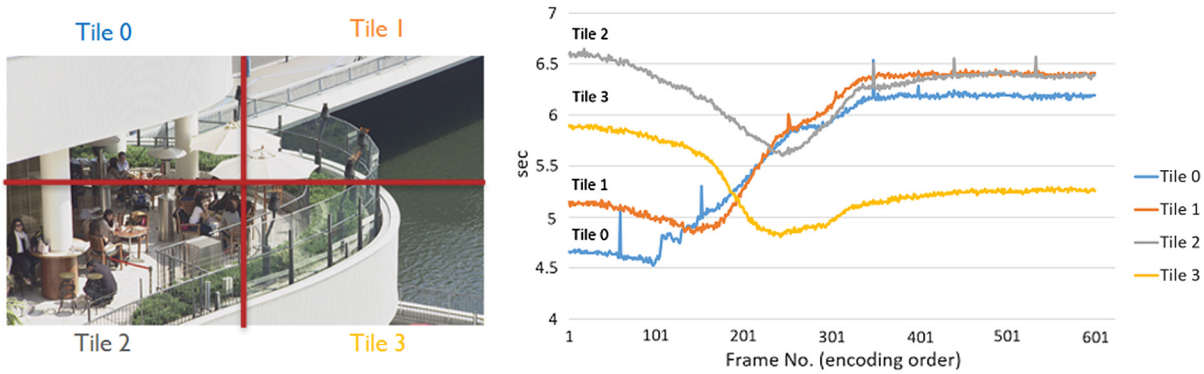


Fig. 2 (a) Uniform-spacing tiles in the first frame of sequence BQTerrace. (b) Encoding time of each tile in sequence BQTerrace under all-intra-QP22 setting.

Fig. 3 Examples of snapshots of (a) original sequence, (b) ordinary HEVC, and (c) HEVC with uniform-space tile partitioning (blue lines). (b) and (c) All intracoded at 1020 kbps.

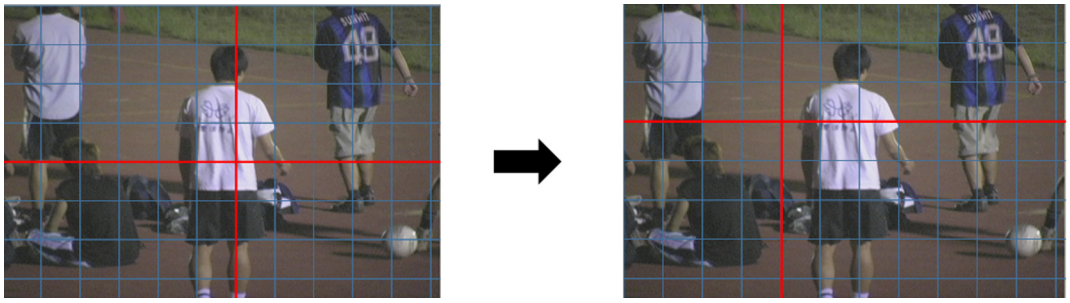


Fig. 4 HEVC tile boundaries will adapt to the video content to improve either coding time load balancing or coding efficiency by using the proposed ATB method. The above example shows that the uniform-spacing tile boundaries (left) separate objects in sequence "soccer," and the boundaries suggested by ATB locate at better positions and improve the coding efficiency.

Moreover, since tile boundary breaks the coding dependence including motion vector (MV) prediction, intraprediction and entropy coding, the coding efficiency is inevitably decreased. To be more precise: (1) for intraprediction, neighboring pixels across tile boundaries becomes unavailable and therefore restricts the number of available intraprediction modes; (2) for MV coding, tile boundary blocks the use of AMVP and merges candidates from the other side, which are usually highly preferable; (3) for entropy coding, probability state and context model estimation cannot cross tile boundaries. From Fig. 3, one can observe some artifacts caused by the tile boundaries when compared to ordinary HEVC coding. For example, the numbers on the player's back are more blurred and the black line on the side of the pants has aliasing distortion. These artifacts appear because the inappropriate tile boundary positions cut off the textures of an object, largely decreasing the prediction performances.

3 Proposed Adaptive Tile Boundary

In order to handle the unbalanced loading and the coding efficiency degradation when using tiles, we propose to increase the flexibility of tile coding by ATB. Figure 4 depicts the idea of ATB and Fig. 5. The core of the proposed ATB scheme is the cost map which is used to estimate the best tile boundary positions. Two branches of ATB including ATB for coding time load balancing (ATB-LoadB) and ATB for coding efficiency (ATB-Gain) are proposed in this paper and will be described as follows.

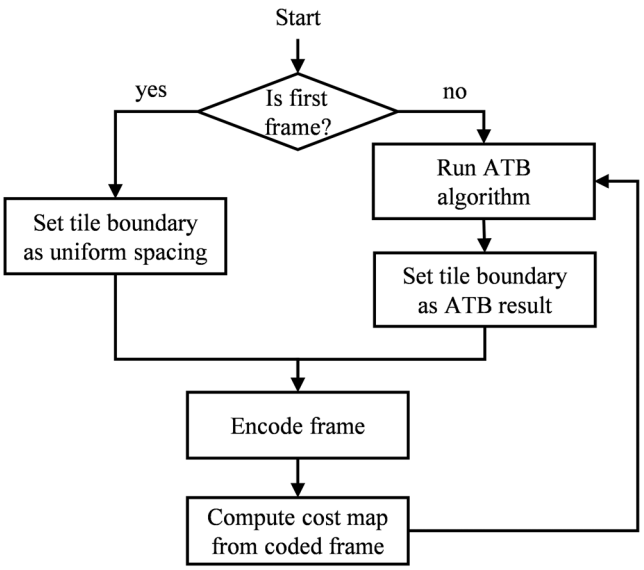


Fig. 5 Flowchart of the modified encoding process with ATB.

3.1 *Cost Map Construction*

In Ref. 17, the authors use constant numbers for the complexity measurement of different CTU coding modes. However, different CTUs with the same coding mode do not mean that they will have the same coding complexities. Therefore, in this paper, we propose to estimate the coding cost based on the on-the-fly coding information. Since tile boundary should be along with the CTU boundaries, a two-dimensional array of size $M \times N$ named "cost map" is created to give each CTU a heuristic cost, where M and N represent the numbers of rows and columns of CTU in a frame and the cost represents the estimate of performance degradation when a tile boundary is located at the right-hand side of that CTU.

However, since the tile partitioning should be decided before encoding but the actual costs for deciding tile boundaries are unknown, the colocated CTUs in a previous coded frame can be used for heuristic cost estimation. In order to select a suitable frame, we conducted an experiment to compare the preceding coded frame and the previous coded frame with the same quantization parameter to the current frame (such frame is named "co-QP frame" hereafter). Their CTU coding structure similarities to the current frame are measured and compared since frames with the same QP value will probably have similar coding complexities. To be more precise, first a frame is partitioned into 8×8 pixel regions $R_{8\times8}$ which are the minimal CU sizes defined in JCTVC common test conditions,[19] and depth of each region is assigned to be its CU depth. Then the CTU coding structure similarity is measured by the following mathematical expression:

$$\text{CU\_Depths\_Displacement\_Err} = 1 - \frac{\sum_{i=2}^{N} \sum_{(x,y)\in R_{8\times8}} |d_{x,y}^{i} - d_{x,y}^{f(i-1)}|}{(N-1) * |R_{8\times8}| * \text{max\_depth}} \quad (1)$$

where N is the total number of frames, $|R_{8\times8}|$ is the number of 8×8 pixel regions in one CTU, max_depth is the maximal possible CU depth, (x, y) is the position of each 8×8 pixel regions in $R_{8\times8}$, d_{xy}^{i}, $d_{xy}^{f(i-1)}$ is the depths at

3	3	2	2	1	1	1	1
3	3	2	2	1	1	1	1
2	2	2	2	1	1	1	1
2	2	2	2	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Fig. 6 An example of CTU coding structure depths. The numbers within the CTU block represent the depths of each 8×8 pixel regions.

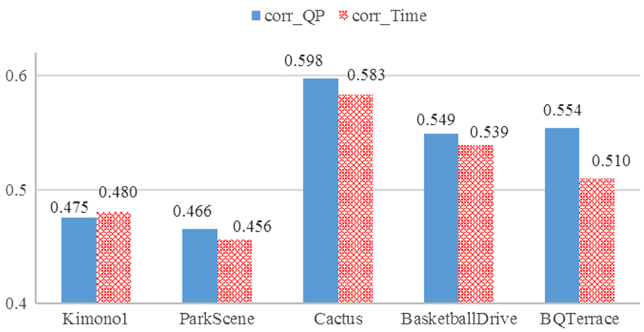


Fig. 7 The CU_Depths_Displacement_Err comparisons for HEVC class B test sequences.

(x, y) for the i'th and $f(i-1)$'th frame, where the frame number $f(i-1)$ can be either the preceding coded frame or the co-QP frame. According to Eq. (1), the value of CU_Depths_Displacement_Err is between 0 and 1 and a higher value represents a higher similarity. In the example of Fig. 6, the CU_Depths_Displacement_Err will be 0.4375 after computation when assuming max_depth is 3. In Fig. 7, the results of several sequences show that most of the time the co-QP frame has a higher CTU coding structure similarity to the current frame than the preceding coded frame. In Fig. 8, some example frames with their CTU coding structure are provided and it can be seen that the co-QP frame shows a more similar CTU coding structure to the current frame, even though it is eight frames away from the current frame in the display order. Note that if the co-QP frame cannot be found in previous frames, the preceding frame will be used instead for cost map construction. Finally, based on how the heuristic cost is defined, tile boundaries can be adjusted to improve load balancing or to avoid cutting off object textures with this unified scheme.

In addition, there might be scene switching within the video sequence in which frame characteristics will be quite different and will affect the cost map prediction. This problem can be solved by regarding the scene switching as the starting point of a new sequence, resetting and restarting the cost map construction for the ATB algorithm. In detail, first, the scene switching frames can be found via existing scene change detection methods.[20–22] Then, the scene switching frame is set to be uniform-spacing tile partitioning and cost maps of all previous frames are cleared.

3.2 *ATB for Load-Balancing*

The goal of load balancing is to allocate the same amount of works for all processing units. However, different encoding algorithms, such as fast mode decisions or motion estimation algorithms, may lead to different complexities for the same CTU. In this paper, the CTU encoding time is adopted for ATB-LoadB as the complexity measurement since it is more intuitive and simple for both software and hardware implementations. Moreover, it will become a general solution that is independent from encoding algorithms.

After building up the cost array from CTU encoding times, the best tile partitioning for maximizing load balancing is searched. Since tiles in one frame are supposed to be parallel executed, the bottleneck of the frame encoding time is the tile with longest encoding time. Therefore, the way of maximizing load balancing is to minimize the largest sum of costs (i.e., estimated encoding time) of this tile. Brute-forcing search is an intuitive solution that enumerates all combinations of tile partitions and searches for the best one. However, the combination grows exponentially with video resolution and tile numbers. For example, a video with resolution 2560×1600 samples and 4×4 tile partitioning will have $\binom{39}{3}\binom{24}{3} = 18{,}497{,}336$ combinations of tile partitioning, whose execution time is apparently not feasible. Therefore, a "greedy strategy" is adopted and tile boundaries in row and column are calculated separately.

An example of how greedy strategy is applied to decide two column tile boundaries for 3×5 CTUs is shown in Fig. 9. In Fig. 9(a), when searching from left to the CTU column B_P, the accumulation of cost is 9 and is closer to the expected cost $C_a = 30/3 = 10$ than the cost of the next column B_C, so B_P will be set as the first tile boundary.

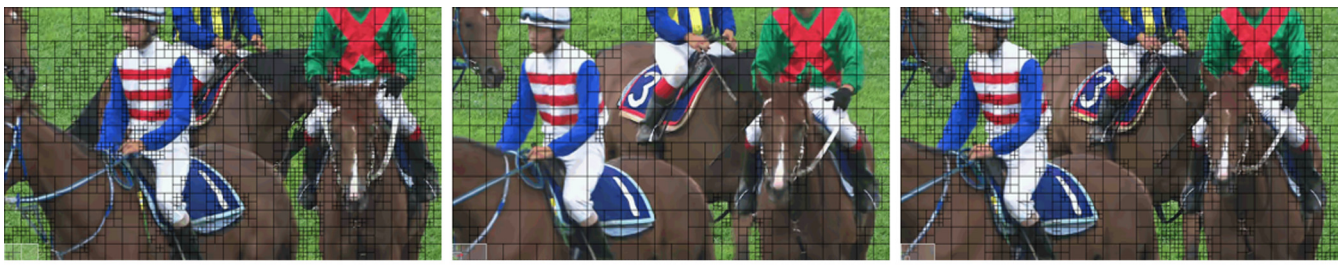


Fig. 8 CTU coding structure comparison of (a) the preceding coded frame (frame #15 w/QP = 39), (b) the current frame (frame #16 w/QP = 38), and (c) the co-QP frame (frame #8 w/QP = 38) for sequence RaceHorses under random access main setting with base QP 37.

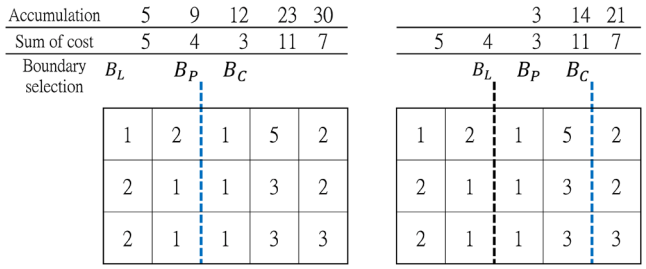


Fig. 9 A cost map example with the greedy strategy results for 3×5 CTUs with two column tile boundaries. Each square block represents a CTU and the number within stand for its encoding times. The result of (a) the first and (b) the second tile boundaries is depicted with blue dashed lines.

In Fig. 9(b), after the selection of the first tile boundary, start point B_L will move to the position of B_P and the greedy strategy proceeds to find the next boundary. The complete process of the proposed greedy algorithm is described as follows:

Step 1. Let C_a denote the expected cost for each column partition, which is defined as $\frac{\text{TotalCost}}{\text{ColumnBoundaryNum}+1}$. Set least partitioned boundary B_L as the left edge of the frame and set current boundary $B_c = B_L$.

Step 2. Check whether remaining boundaries can be further partitioned after moving B_c to the next CTU boundary. If yes, set previous boundary $B_P = B_c$, move B_c to the next CTU boundary and go to step 3. Otherwise, set final boundary $B_F = B_c$ and go to step 5.

Step 3. Calculate the sum of costs between B_L and B_c as S_C and the sum of costs between B_L and B_P as S_P. If $S_C > C_a$ then go to step 4. Otherwise, go to step 2.

Step 4. If $|S_C - C_a| \leq |S_P - C_a|$ then set B_F as B_P. Otherwise, set B_F as B_c.

Step 5. Let B_F be a tile boundary of the current frame. If no more tile boundaries are needed, the algorithm finishes. Otherwise, let $B_L = B_F$ and go to step 2.

Since the sum of cost with different ranges must be calculated, integral image technique is used to accelerate this repetitive computation process. As a result, the complexity of the greedy method is reduced from $O[M \times N \times (M+N)]$ to $O(M \times N)$.

However, greedy strategy cannot guarantee the optimal solution. Take Fig. 9 as an example; although the solution of selecting the second and the fourth columns is not bad, in fact the third and the fourth boundaries are actually the optimal solution in this case. Therefore, after both row and column tile boundaries are determined by the greedy strategy, a 3-candidate approach is applied to choose one of (a) the greedy strategy result, (b) uniform-space partitioning, or (c) the tile partitioning of the co-QP frame. The one with the minimal value of max-cost will become the final result, where the max-cost is the maximum cost among all tiles in a frame. The uniform-space partitioning is used as the candidate because the area size is still correlated to complexity and the result of the co-QP frame is used because its CTU structure usually resembles the current frame. Adding these two more candidates can avoid possibly local optimal solutions of the greedy strategy. An example of the 3-candidate approach is provided in Fig. 10.

3.3 *ATB for Coding Efficiency*

From Sec. 2, we know that the coding dependencies broken by tiles decrease the coding efficiency, and objects along tile boundaries are affected most. Therefore, the idea of ATB-Gain is to place the tile boundary at a position that breaks less coding dependencies. The coding mode of each PU[11] is checked and is assigned with a heuristic cost measuring the amount of coding dependencies. A higher cost is given when the tile boundary is expected to decrease more coding efficiency of this PU. Again, the coding modes of the co-QP frame are used to decide the heuristic costs of the current frame and the row/column tile boundaries are decided separately. Finally, the rows and columns with the lowest sum of costs are chosen to be the tile boundaries. An example is provided in Fig. 11.

Figure 12 shows the intraprediction modes in HEVC and their corresponding heuristic costs used in this paper. There are 33 modes with different prediction directions (modes 2 to 34) and 2 nondirectional modes (modes 0 and 1). The

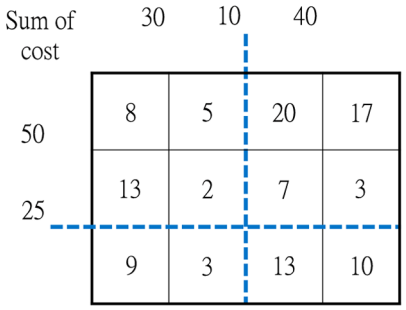


Fig. 11 A cost map example for a frame with 3×4 CTUs with two column tile boundaries. Each block represents a CTU and the numbers stand for their cost of coding dependencies.

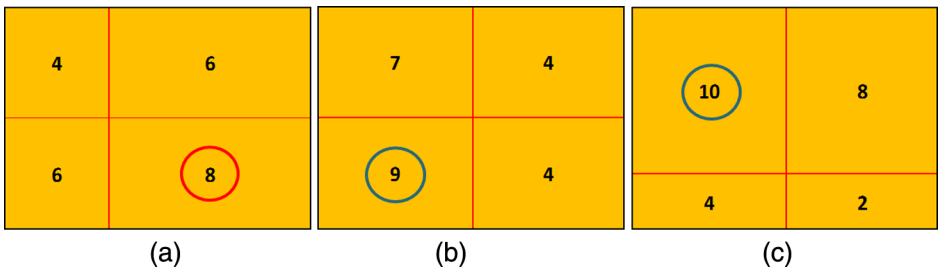
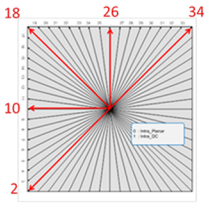


Fig. 10 An example of 3-candidate approach. (a)–(c) The partitions from the three candidates. Four tiles are used and the number in each tile represents the sum of heuristic costs, and the circles mark the max-cost for each candidate.

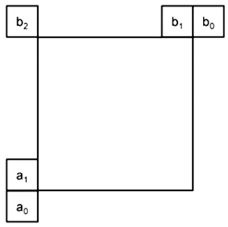


(a)

Angular a	Operation	
2≤a<10	RowCost[m+1]+=1	ColCost[n]+=1
a = 10		ColCost[n]+=2
11≤a<26	RowCost[m]+=1	ColCost[n]+=1
a = 26	RowCost[m]+=2	
27≤a≤34	RowCost[m]+=1	ColCost[n+1]+=1

(b)

Fig. 12 (a) Intraprediction modes and their directions in HEVC. (b) Table of heuristic costs for the corresponding intramodes.



(a)

MV ref.	Operation	
a_0	RowCost[m+1]+=2	ColCost[n]+=2
a_1		ColCost[n]+=2
b_0	RowCost[m]+=2	ColCost[n+1]+=2
b_1	RowCost[m]+=2	
b_2	RowCost[m]+=2	ColCost[n]+=2

(b)

Fig. 13 (a) MV references for intraprediction modes in HEVC. (b) Table of heuristic costs for the corresponding inter-MV references.

angular 10 and 26 represent horizontal and vertical prediction, which means there is strong reference in one direction. Therefore, their costs to the corresponding tile boundaries will be the largest. For directions other than 10 and 26, coding dependency can be broken by two sides of tile boundaries and we give both boundaries a smaller cost. For nondirectional modes since they use pixel information from all sides, we assign zero costs. The heuristic costs are summarized in the table of Fig. 12(b) where the indices m and n of the two cost arrays RowCost and ColCost indicate the costs of tile boundaries along with the PU's top and left edges. In a similar way, indices $m+1$ and $n+1$ indicate the tile boundary along the bottom edge and the right edge of that PU, respectively.

Figure 13 shows the MV references for interprediction in HEVC and their corresponding heuristic costs used in this paper. The principle of cost assignment is the same as that of intraprediction. The tile boundary that blocks the current PU's MV reference will be assigned with a heuristic cost. Last but not least, the heuristic cost will be normalized by the size of intra-/inter-PU and the normalized cost is given by

$$\text{original cost} \times \frac{\text{PU size}}{\text{CTU size}}. \tag{2}$$

4 Experiment

The proposed ATB algorithms are implemented into the HEVC test model HM 10.1.[23] The "random access main" setting is tested, which follows the JCTVC common test conditions[19] with QP values (22, 27, 32, 37) and CTU size 64×64 pixels. HEVC test sequences of classes A, B, and C are used to test the performance from high (2560×1600, 1920×1080) to low (832×480) resolutions. In order to simulate the real application of tiles, we parallelize the HM tile encoding process by using the OpenMP API[24] and the test sequences are encoded using 4, 8, 16 tiles with 2×2, 2×4, 4×4 tile partitioning, respectively. We use a desktop computer with an Intel i7 quad core processor and 16 GB RAM to evaluate the encoding of 4 tiles and use a server with an AMD Opteron 16-core processor and 32 GB RAM to evaluate the encoding of 8 and 16 tiles. All experiments followed the tile size restriction defined in HEVC Main Profile,[3] where the minimal tile width and height are 256 and 64 pixels, respectively. The coding performance is compared with uniform-space tile partitioning and is measured by (i) the average encoding time ratio denoting the load balancing improvement and (ii) the BD-rate measurement[25,26] with minus sign denoting the coding efficiency improvement (coding gains). The performances of each proposed ATB algorithm are evaluated and discussed in the following subsections.

4.1 *Coding Time Load-Balancing Improvement of ATB-LoadB*

The performance of the proposed ATB-LoadB for 2×2 tile partitioning is shown in Table 1. The encoding times that save more than 5% are marked as bold. The sequence with the best performance is BasketballDrive in class B, where the encoding time takes only 87.09% (12.91% saved) compared to uniform spacing. Specifically, for QP

Table 1 ATB-LoadB for 2×2 tile partitioning.

		ATB-LoadB	ATB-LoadB with brute force search
Class	Sequence	Y BD (%)/ Enc. time (%)	Y BD (%)/ Enc. time (%)
Class A	Traffic	0.0/97.75	0.0/**93.99**
	PeopleOnStreet	0.0/99.78	0.0/95.78
	Nebuta	−0.1/98.99	−0.1/95.09
	SteamLocomotive	0.0/97.08	0.0/**93.49**
Class B	Kimono	0.0/97.44	0.0/95.17
	ParkScene	0.0/96.87	0.0/***94.42***
	Cactus	−0.1/**94.12**	0.0/***93.95***
	BasketballDrive	0.0/***87.09***	0.1/***85.16***
	BQTerrace	0.0/98.76	0.0/95.85
Class C	BasketballDrill	0.1/97.11	0.1/***92.97***
	BQMall	0.0/99.01	0.0/95.47
	PartyScene	0.0/95.70	0.0/95.98
	RaceHorses	0.0/96.47	0.0/***93.68***
	Overall	*0.0/96.63*	0.0/***93.92***

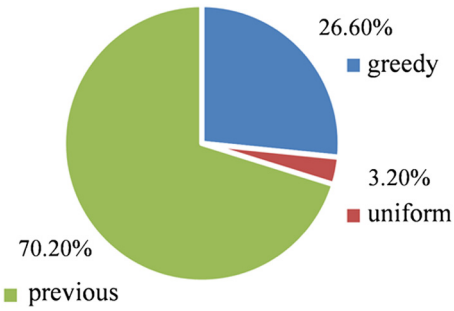


Fig. 14 The distribution of ATB-LB candidate selection for BasketballDrive at QP 22.

Table 2 ATB-LoadB encoding/decoding times for different tile partitioning.

Class	Sequence	2 × 2 (%)	2 × 4 (%)	4 × 4 (%)
Class A	Traffic	97.75/100.68	96.81/101.22	96.85/99.51
	People-OnStreet	99.78/99.58	98.77/100.03	99.23/106.09
	Nebuta	98.99/98.68	98.79/99.39	100.21/104.38
	Steam-Locomotive	97.08/99.62	96.47/96.98	**92.37/92.95**
	Overall	*98.43/99.64*	*97.71/99.40*	*97.71/100.73*
Class B	Kimono	97.44/100.22	97.76/97.33	97.66/99.42
	ParkScene	96.87/102.23	95.52/96.55	**92.45**/99.92
	Cactus	**94.12**/96.38	**90.53/91.72**	**90.71**/99.96
	Basketball-Drive	**87.09/93.37**	**86.91**/98.40	**88.09**/99.77
	BQTerrace	98.76/99.83	98.26/96.21	97.70/99.48
	Overall	***94.93***/*98.40*	***93.80***/*96.04*	***93.32***/*99.64*

22 in BasketballDrive the encoding time saving can achieve an even higher result of 16.52%. Figure 14 shows the distribution of tile boundary candidates selection for this test case. From Fig. 14, it can be seen that the two additional candidates other than uniform spacing take effect of suggesting better tile partitioning for coding time load balancing. Since there are 30×17 CTUs in each frame of BasketballDrive with the uniform spacing partitioned at (15, 8) in Fig. 14, it is observed that various boundary locations are selected to improve coding time load balancing, with large portions of the selected boundaries near the uniform-spacing locations and less boundaries being far away from uniform spacing.

Moreover, we additionally apply a brute force search of ATB-LoadB by trying all possible boundary locations (still under the main profile restriction) and select the one with maximal load balancing. Note that row and column boundaries are still separately decided. Such a solution can be seen as the best selection of the proposed ATB-LoadB; however, it is not applicable when using more tiles due to the growth of combinations of possible boundary positions. The result is also shown in Table 1 and it is observed that encoding time saving is about 2% to 5% higher than the proposed 3-candidate ATB-LoadB results.

The 8 (2×4) and 16 (4×4) tile partitioning are also tested and the results are shown in Table 2. Since we implement only the parallelized encoder, we try to measure the load balancing of the decoder by recording each tile's decoding time, and simulated the parallel decoding time by selecting the one with the largest tile decoding time among all tiles in the frame as the frame decoding time. It is observed that the estimated parallel decoding time is highly correlated to its encoding time, although the result is not as good as that of encoding. For the test case of BasketballDrive QP 22 using 2×4 tile partitioning, 17.65% of encoding time saving can be found, which has the highest encoding time saving among all test cases in this paper. The coding time saving trends from Table 2 show that for a reasonable number of tiles, when more tiles used more encoding/decoding time savings are prone to be achieved. Figure 15 shows the encoding time load balancing before and after applying the proposed ATB-LoadB algorithm. It can be seen that the encoding time of the four tiles is obviously getting closer to each other, which means that the encoding is more balanced.

To sum up, the proposed ATB-LoadB provides a comforting result that it is basically no encoding time loss for all test

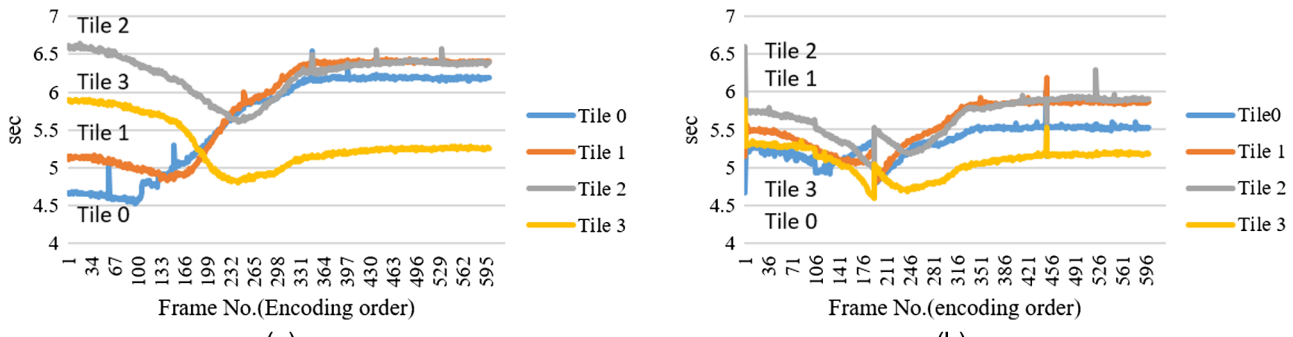


Fig. 15 Encoding time of each tile in sequence BQTerrace under all-intra-QP22 setting for (a) the default uniform-spacing tile partitioning and (b) the tile partitioning decided by the proposed ATB-LoadB.

cases and in most cases 2% to 3% of encoding time savings can be achieved. Moreover, the variations of BD-rate for all test cases are negligible, which means that the improvement of load balancing does not affect the coding efficiency.

4.2 *Coding Efficiency Improvement of ATB-Gain*

The left part of Table 4 shows the coding results of the proposed ATB-Gain. It is observed that the top three performers can achieve 0.5%, 0.4%, 0.3% of Y (luma component) BD-Rate savings, and the coding gains for U and V (chroma components) are even higher, with up to 1.7%. Also, almost all the test cases obtain BD-rate gains, except for two sequences where coding gains are found in one component but with coding loss in another component. Moreover, we performed an additional test of ATB-Gain for classes B and C by neglecting the tile size restrictions in the main profile, and thus the distances between tile boundaries can be as small as one CTU size. The results are shown in the right part of Table 3 and one can see that the BD-rate gains are further increased. The top performer makes a breakthrough to 1.0% of Y BD-rate savings and 0.8% of Y BD-rate savings is found for the second place. The coding gains are also increased with tile numbers, as shown in Table 4, with the highest coding gain of 0.8% achieved for SteamLocomotive with 16 tiles. We test ATB-Gain on several extra sequences and find that more coding gains can be achieved, as shown in Table 5. The highest coding gain is 1.6% of Y BD-rate savings.

Table 3 ATB-Gain for 2×2 tile partitioning.

		ATB-Gain		w/o tile size limitation	
Class	Sequence	Y BD (%)	U/V BD (%)	Y BD (%)	U/V BD (%)
Class A	Traffic	−0.1	0.0/ − 0.1	—	—
	People-OnStreet	−0.1	−0.3/ − 0.4	—	—
	Nebuta	0.0	−0.2/ − 0.2	—	—
	Steam-Locomotive	**−0.3**	**−0.6/ − 1.7**	—	—
Class B	Kimono	−0.2	−0.2/ − 0.2	−0.1	−0.1/ − 0.4
	ParkScene	0.0	0.2/ − 0.1	−0.2	−0.2/ − 0.2
	Cactus	−0.2	−0.2/ − 0.1	−0.4	−0.7/ − 0.7
	BasketballDrive	**−0.4**	**−0.8/ − 0.5**	**−1.0**	**−1.4/ − 1.4**
	BQTerrace	−0.1	−0.5/ − 0.4	−0.1	−0.1/ − 0.3
Class C	BasketballDrill	−0.1	−0.2/ − 0.5	−0.4	−0.7/ − 0.5
	BQMall	**−0.5**	**−0.4/ − 0.7**	**−0.8**	**−0.7/ − 0.6**
	PartyScene	−0.1	−0.1/ − 0.1	−0.1	−0.1/ − 0.1
	RaceHorses	−0.1	0.1/0.2	−0.1	−0.3/ − 0.2
	Overall	−0.2	−0.2/ − 0.4	−0.3	−0.4/ − 0.5

Table 4 ATB-Gain Y BD-Rate gain for different tile partitioning.

Class	Sequence	2×2 (%)	2×4 (%)	4×4 (%)
Class A	Traffic	−0.1	0.0	0.0
	PeopleOnStreet	−0.1	−0.2	**−0.4**
	Nebuta	0.0	0.0	0.1
	SteamLocomotive	**−0.3**	−0.2	**−0.8**
Class B	Kimono	−0.2	−0.2	**−0.5**
	ParkScene	0.0	0.0	0.0
	Cactus	−0.2	−0.1	**−0.5**
	BasketballDrive	**−0.4**	**−0.5**	**−0.8**
	BQTerrace	−0.1	0.0	−0.1
	Overall	−0.2	−0.1	−0.4

Table 5 ATB-Gain for extra test sequences of 2×2 tile partitioning.

		ATB-Gain	
Resolution	Sequence	Y BD (%)	U/V BD (%)
1280×720	SlideEditing	−1.0	−1.0/ − 1.1
	SlideShow	−1.6	−1.9/ − 2.4
720×480	rollingtomato	−0.9	−1.0/ − 1.6
	soccer	−1.1	−1.6/ − 2.5

4.3 *Comparison with Previous Works, Discussions, and Summary*

The comparison of the related works Refs. 17 and 18 with the proposed ATB-LoadB and ATB-Gain is shown in Tables 6 and 7. The results are obtained directly from the literature since we have not been able to reproduce it yet. But in Ref. 17, the complexity cost of different CTU coding modes is empirically assigned to be constant numbers. It is not guaranteed that these parameters are suitable for different testing environments. However, the proposed method uses on-the-fly information from the encoder which is more flexible and can adapt to any implementations and platforms.

When comparing with Ref. 17, the proposed ATB-LoadB works better for more sequences and has a fair average performance. Note that for sequence Kimono, it is surprising that the load balancing improvement of Ref. 17 is even better than our estimated upper bound for ATB-LoadB. When comparing with Ref. 18 for "all intramain" setting, in which all

Table 6 The proposed ATB-LoadB versus related work[17] for encoding time using four tiles.

Class	Sequence	Encoding time (%)	
		ATB-LoadB (%)	Ref. 17 (%)
Class B	Kimono	97.44	**90.29**
	ParkScene	**96.87**	97.57
	Cactus	**94.12**	98.30
	BasketballDrive	**87.09**	88.52
	BQTerrace	98.76	**93.69**
	Overall	*94.93*	***93.67***
Class C	BasketballDrill	**97.11**	100.67
	BQMall	99.01	**97.41**
	PartyScene	**95.70**	99.04
	RaceHorses	**96.47**	97.42
	Overall	***97.06***	*98.64*

Table 7 The proposed ATB-Gain versus related work[18] for coding gain using four tiles.

Class	Sequence	Y BD all intramain		Y BD random access main	
		ATB-Gain (%)	Ref. 18 (%)	ATB-Gain (%)	Ref. 18
Class A	PeopleOnStreet	**−0.091**	−0.049	−0.076	N/A
	Nebuta	**−0.004**	−0.002	0.005	N/A
	Overall	***−0.05***	−0.03	−0.04	N/A
Class B	Kimono	−0.026	**−0.067**	−0.172	N/A
	ParkScene	**−0.075**	−0.048	0.032	N/A
	Cactus	−0.068	**−0.275**	−0.219	N/A
	BasketballDrive	−0.267	**−0.348**	−0.435	N/A
	BQTerrace	−0.040	**−0.070**	−0.132	N/A
	Overall	−0.10	−0.16	−0.19	N/A

frames are intracoded frames, the proposed ATB-Gain works better for class A but improves less for class B, where the overall result still shows a comparable performance. However, the approach in Ref. 18 can be used only for intracoded frames since their analysis of frame variance characteristics is not suitable for intercoded frames (random access main setting in Table 7), where the proposed method can be applied to all coding scenarios.

Table 8 Average overhead of ATB encoding for different tile partitioning.

Class	Sequence	2 × 2 (%)	2 × 4 (%)	4 × 4 (%)
Class A	Traffic	0.0001	0.0001	0.0002
	PeopleOnStreet	0.0001	0.0000	0.0000
	Nebuta	0.0000	0.0000	0.0002
	SteamLocomotive	0.0001	0.0000	0.0002
	Overall	*0.0001*	*0.0000*	*0.0001*
Class B	Kimono	0.0003	0.0001	0.0000
	ParkScene	0.0003	0.0002	0.0003
	Cactus	0.0002	0.0001	0.0004
	BasketballDrive	0.0001	0.0000	0.0002
	BQTerrace	0.0001	0.0003	0.0002
	Overall	*0.0002*	*0.0002*	*0.0002*

For the computation overhead, Ref. 17 is the smallest since its complexity costs are constant numbers and the tile boundaries are decided through a mathematical updating formula. The overheads of the proposed ATB algorithms come from (1) saving encoding time of CTU and coding mode of PUs to arrays, (2) computing sum of cost along row/column boundaries (accelerated by integral image), (3) comparisons within the greedy algorithm, and (4) sorting of heuristic costs. Such computation loads are very low when comparing with the encoding process and are negligible, as shown in Table 8. For Ref. 18, it is more complex since it contains multiple times of variance computation thus the overheads are higher than ours, with average 0.083% of the total encoding time for 2 × 2 tile partitioning.

For rate control and other coding schemes that each CTU is allowed to change its QP value within a limited range based on the frame QP, the proposed method can still be used without modification by using the frame base QP for co-QP frame decision. A more complex but sophisticated solution might be to record the QP value for each CTU, and change the construction of the cost map from referencing the co-QP frame to referencing the CTU having the same QP value to the current CTU. It will be our future work.

5 Conclusion

In this paper, three kinds of ATB algorithms are proposed to improve the encoding performance of HEVC tile load balancing and coding efficiency. When considering only one aspect, apparent improvements can be achieved for all test sequences with large improvements for some test cases. Jointly consideration of improving both load balancing and coding efficiency is under study and will be our future work.

One more possible usage of ATB is that it can be applied to the ROI coding, in which the tile boundaries can be set to trace the desired object.

References

1. T. Wiegand et al., "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.* **13**(7), 560–576 (2003).
2. G. J. Sullivan et al., "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.* **22**(12), 1649–1668 (2012).
3. B. Bross et al., "High efficiency video coding (HEVC) text specification draft 9 (SoDIS)," Document of Joint Collaborative Team on Video Coding, JCTVC-K1003, Shanghai, China (2012).
4. J.-R. Ohm et al., "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.* **22**(12), 1668–1683 (2012).
5. J. Ostermann and M. Narroschke, "Draft requirements for next-generation video coding project," ITU-T Q.6/SG16, VCEG-AL96, London, GB (2009).
6. "Joint call for proposals on video compression technology," ISO/IEC JTC1/SC29/WG11, MPEG09/N11113 (2010).
7. C.-C. Chi et al., "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Trans. Circuits Syst. Video Technol.* **22**(12), 1827–1838 (2012).
8. F. Henry and S. Pateux, "Wavefront parallel processing," Document of Joint Collaborative Team on Video Coding (JCTVC), JCTVC-E196, Geneva, Switzerland (2011).
9. A. Fuldseth et al., "Tiles," Document of Joint Collaborative Team on Video Coding (JCTVC), JCTVC-F335, Torino, Italy (2011).
10. K. Misra et al., "An overview of tiles in HEVC," *IEEE J. Sel. Top. Signal Process.* **7**(99), 969–977 (2013).
11. I.-K. Kim et al., "Block partitioning structure in the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.* **22**(12), 1697–1706 (2012).
12. K. Misra and A. Segall, "Parallel decoding with tiles," Document of Joint Collaborative Team on Video Coding (JCTVC), JCT-VC F594, Torino, Italy (2011).
13. Hendry et al., "AHG4: harmonized method for signalling entry points of tiles and WPP substreams," Document of Joint Collaborative Team on Video Coding (JCTVC), JCTVC-H0556, San Jose, USA (2012).
14. Y.-K. Wang et al., "Text for tiles, WPP and entropy slices," Document of Joint Collaborative Team on Video Coding (JCTVC), JCTVC-H0737, San Jose, USA (2012).
15. Y. Wu et al., "Motion-constrained tile sets SEI message," Document of Joint Collaborative Team on Video Coding (JCTVC), JCTVC-M0235, Incheon, Korea (2013).
16. M. Zhou et al., "Parallel tools in HEVC for high-throughput processing," *Proc. SPIE* **8499**, 849910 (2012).
17. Y.-J. Ahn et al., "Complexity model based load-balancing algorithm for parallel tools of HEVC," in *Visual Communications and Image Processing (VCIP)*, pp. 1–5 (2013).
18. C. Blumenberg et al., "Adaptive content-based tile partitioning algorithm," in *Picture Coding Symp. (PCS)* (2013).
19. F. Bossen, "Common HM test conditions and software reference configurations," Document of Joint Collaborative Team on Video Coding (JCTVC), JCTVC-L1100, Geneva, Switzerland (2013).
20. C.-L. Huang and B.-Y. Liao, "A robust scene-change detection method for video segmentation," *IEEE Trans. Circuits Syst. Video Technol.* **11**(12), 1281–1288 (2001).
21. R. A. Joyce and B. Liu, "Temporal segmentation of video using frame and histogram space," *IEEE Trans. Multimedia* **8**(1), 130–140 (2006).
22. Z. Rasheed and M. Shah, "Detection and representation of scenes in videos," *IEEE Trans. Multimedia* **7**(6), 1097–1105 (2005).
23. "HEVC software repository," https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware (2016).
24. "OpenMP," http://www.openmp.org/ (2016).
25. G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," Document ITU-T SG16 Q.6, VCEG-M33, Austin, USA (2001).
26. G. Bjøntegaard, "Improvement of BD-PSNR model," Document ITU-T SG16 Q.6, VCEG-AI11, Berlin, Germany (2008).

Chia-Hsin Chan received his BS and MS degrees in computer science from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2007 and 2009, respectively. Currently, he is pursuing his PhD in computer science at NCTU. In 2009, he stood for NCTU to participate in the ISO/IEC MPEG and ITU-T VCEG joint call-for-proposals competition on a high efficiency video coding (HEVC) standard. His research interests include video compression and nighttime surveillance video processing.

Chun-Chuan Tu received his BS degree in computer science from National Taipei University, New Taipei City, Taiwan, in 2012, his MS degree in computer science from NCTU, Hsinchu, Taiwan, in 2014. His research interests include video compression, parallel processing, and reverse engineering.

Wen-Jiin Tsai received her PhD in computer science from NCTU, Hsinchu, Taiwan, in 1997. Currently, she is an associate professor in the Department of Computer Science, NCTU. Before joining NCTU in 2004, she was with Zinwell Corporation, Hsinchu, as a senior research and development manager for six years. Her current research interests include video coding, video streaming, error concealment, and error resilience techniques.