Optical Engineering

SPIEDigitalLibrary.org/oe

Fast coding unit decision method based on coding tree pruning for high efficiency video coding

Kiho Choi
Euee S. Jang

Fast coding unit decision method based on coding tree pruning for high efficiency video coding

Kiho Choi and Euee S. Jang
Hanyang University, Digital Media Laboratory, 512 R&D Building, Haengdang-dong Seongdong-gu, Seoul, South Korea
E-mail: esjang@hanyang.ac.kr

Abstract. A fast coding unit (CU) decision method is proposed for high efficiency video coding (HEVC) by determining early the CU sizes based on coding tree pruning. One of the most effective, a newly introduced concept in HEVC, is variable CU size. In determining the best CU size, the reference encoder of the HEVC tests every possible CU size in order to estimate the coding performance of each CU defined by the CU size. This causes major computational complexity within the encoding process, which should be overcome for the implementation of a fast encoder. A simple tree-pruning algorithm is proposed that exploits the observation where the subtree computations can be skipped if the coding mode of the current node is sufficient (e.g., SKIP mode). The experimental results show that the proposed method was able to achieve a 40% reduction in encoding time compared to the HEVC test model 3.0 encoder with only a negligible loss in coding performance. The proposed method was adopted in HEVC test model 4.0 encoder at JCT-VC 6th meeting.
 [DOI: 10.1117/1.OE.51.3.030502]

Subject terms: high-efficiency video coding; early coding unit determination; fast high-efficiency video coding; video coding.

Paper 111201L received Sep. 27, 2011; revised manuscript received Jan. 17, 2012; accepted for publication Jan. 30, 2012; published online Mar. 20, 2012.

1 Introduction

ISO/IEC Moving Picture Expert Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG) initiated a new video coding standardization called high efficiency video coding (HEVC) in 2010. This new standard is expected to satisfy the ever-increasing requirements, such as compression efficiency, video resolution, frame rates, and computational complexity. While the HEVC standardization is in progress, the compression efficiency of HEVC is two times better when compared to that of its preceding standard, MPEG-4 AVC/H.264.[1]

Although HEVC follows the traditional coding structure, including block-based motion compensation and transform coding, a substantial improvement comes from a new hierarchical coding concept based on a quadtree structure.[2] In HEVC, the video encoding and decoding process is composed of three units as follows: the coding unit coding unit (CU) for the root of the transform quadtree as well as the prediction mode for Inter/SKIP/Intra prediction, the prediction unit (PU) for coding the mode decision, including motion estimation and rate-distortion optimization, and the transform unit TU for transform coding and entropy coding.[3] Among the three units, CU is the most critical in relation to the compression efficiency because it determines the initial coding block size, which affects the performance of the remaining processing units (i.e., PU and TU).

When using CU, PU, and TU, improved compression efficiency is possible, but this dramatically increases the computational complexity. Typically, a quadtree is generated during the encoding process where the root node indicates the largest CU size and has four nodes corresponding to the next CU size. The tree is called the coding tree, and the maximum depth can be as deep as four. CUs with coding trees (e.g., CU0 for 64×64, CU1 for 32×32,..., CU3 for 8×8 when the largest CU size is set to 64 and the depth is set to 4) are included within the computation and selection of the optimal CU size, which causes exponential growth of the number of CUs (e.g., $1 \times \mathrm{CU0} + 4 \times \mathrm{CU1} + 16 \times \mathrm{CU2} + 64 \times \mathrm{CU3}$) to occur as shown in Fig. 1. Since each CU `is followed by a corresponding PU and TU computation, the growing complexity poses a serious challenge to the real-time encoder design. The HEVC reference encoder (i.e., Version 3.0) is approximately three times slower than the MPEG-4 AVC/H.264 reference encoder (i.e., JM 17.0 High profile). In order to overcome the computational complexity in HEVC encoder, we exploited the coding tree pruning based CU early termination and introduced the method at the JCT-VC 6th meeting.[4]

2 Fast CU Decision Method Based on Coding Tree Pruning

The computational complexity based on variable CU sizes can be described as follows:

$$f(n) = f(n-1) + 4^n * M_n, f(0) = M_0 \quad \text{and}$$
$$M_i = \left(\frac{1}{4}\right)^i * M_{i-1}, \tag{1}$$

where $f(n)$ is the total number of operations when the maximum depth of the coding tree is set to n, and M_i is the number of operations required for the given CU size at the i'th level. In Eq. (1), we assume that M_i is one-fourth of M_{i-1} because the CU size decreases by one-fourth from the previous level. The total number of operations can be combined as follows:

$$f(n) = \sum_{i=0}^{n} 4^i * M_i \cong O(M_0 * n). \tag{2}$$

As shown in Eq. (2), the total computational complexity increases monotonically with respect to the maximum CU depth. When considering the fact that M_i represents all the encoding processes, including motion estimation at each CU size, the maximum CU depth is a dominant factor that determines the encoding time.

This letter investigates the strategy for a fast CU depth decision by determining the CU depth early. We exploit the fact that no further processing of subtrees is necessary when the cost of the current CU node is lower than those of the CU nodes belonging to the subtrees of the current CU node [i.e. $\mathrm{RDCost}(CU_t) < \sum_{i=0}^{3} \mathrm{RDCost}(CU_{t-1}^i)$]. The only problem with the previous condition is that the cost of subtrees must be known, and this requirement hinders the efficient reduction of the computational complexity of the

0091-3286/2012/$25.00 © 2012 SPIE

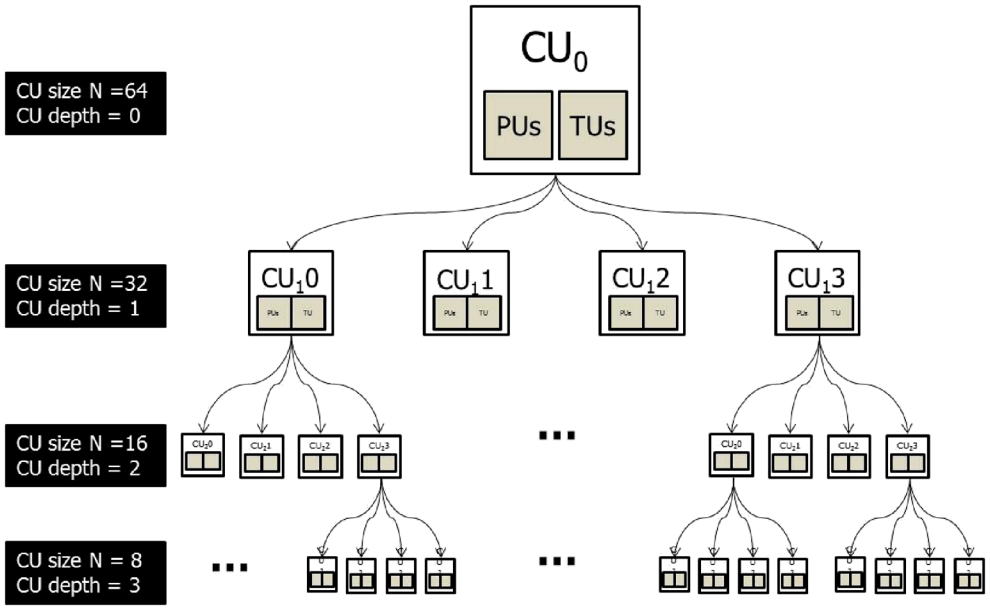

Fig. 1 Structural example of HEVC where the CU size is 64×64 and the depth is 4 (coding tree).

subtrees. We can avoid computing the subtree cost if the cost of the current node is the minimum (i.e., SKIP mode); this is the major concept of the proposed method. In order to further reduce the computational complexity with some loss in compression efficiency, it is possible to define the condition to pruned subtrees, as follows:

Pruning condition :

$$(m' = \underset{m \in \text{Mode}}{\text{argmin}}\, \text{RDCost}(CU_t|PU = m)) \leq \text{Threshold}$$
$$\text{Mode} = \{\text{SKIP}, \text{Inter}2N \times 2N, \text{Inter}2N \times N, \text{Inter}N \times 2N, \text{Inter}N \times N\}, \quad (3)$$

where Mode indicates the ordered set of prediction modes according to the complexity, m' is a selected prediction mode for the current CU depth, and Threshold may be chosen from Mode. In Eq. (3), the likelihood of the pruning process is determined by varying the Threshold value from SKIP to Inter $N \times N$. If the Threshold is set to SKIP, the

Table 1 The procedure for the proposed CU decision method based on coding tree pruning

Algorithm proposed CU decision

```
Recursive_CU_Processing(depth,index){
  parent_cost = CU_processing(depth,index)
  if (selected prediction_mode ≤ Threshold
      Best_CU = CU(depth)
    pruning remaining processes
  else
    for from index = 0 to to index = 3 do
      children_cost+ = Recursive_CU_Processsing(depth + 1,
index)
    end
    if(parent_Cost ≤ children_cost)
      Best_CU = CU(depth)
    else
      Best_CU = CU(depth+1)
    if(leaf node)
      return
}
```

Table 2 Test conditions and software reference configurations

Test Sequences	• Class A (2560×1600): Traffic and People On Street • Class B (1920×1080): Kimono, ParkScene, Cactus, and BQTerrace • Class C (832×480): BasketballDrill, BQMall, PartyScene, and RaceHorsesC • Class E (416×240): BasketballPass, BQSquare, BlowingBubbles, RaceHorses • Class E (1280×720): Vidyo1, Vidyo3, and Vidyo4
Total Frames to be Coded	• Class A: 5 seconds of video duration • Other Classes: 10 seconds of video duration
Software	• HM 3.0
Quantization Parameter	• 22, 27, 32 and 37
Others	• High efficiency setting

Table 3 Results of the comparison between HM 3.0 and the proposed method with respect to total encoding time: (a) low-delay scenario and (b) random-access scenario (kb/s: kilobit per second, dB: decibel, and ms: millisecond)

(a)	HM 3.0			Proposed Method			Comparison		
Class	**Bitrate (kb/s)**	**PSNR (dB)**	**Time (ms)**	**Bitrate (kb/s)**	**PSNR (dB)**	**Time (ms)**	**Bitrate (%)**	**PSNR (dB)**	**Time (%)**
Class B Average	10,587	36.74	697.29	10,553	36.72	450.76	−0.34	−0.02	−38
Class C Average	3,803	35.04	141.45	3,794	35.02	104.81	−0.33	−0.02	−29
Class D Average	1,031	34.55	34.58	1,027	34.52	25.86	−0.44	−0.03	−28
Class E Average	1,862	40.19	246.60	1,847	40.16	104.67	−0.78	−0.03	−58
		Average					−0.44	−0.03	−37
(b)	HM 3.0			Proposed Method			Comparison		
Class A Average	14,423	37.11	1,116.5	14,363	37.05	690.30	−0.61	−0.06	−41
Class B Average	9,620	36.74	548.42	9,564	36.70	300.70	−0.55	−0.04	−47
Class C Average	3,533	35.09	112.66	3,523	35.04	74.15	−0.51	−0.05	−37
Class D Average	942	34.78	27.46	940	34.74	18.51	−0.44	−0.04	−36
		Average					−0.52	−0.05	−41

pruning process does not influence the compression efficiency; otherwise, more subtrees will be pruned at the cost of decreased compression efficiency. Based on this analysis, we propose the subtree pruning process as shown in Table 1.

3 Experimental Results

For the performance evaluation, we assessed the total execution time of the proposed method in comparison to that of the HEVC test model (HM) 3.0 in order to confirm the reduction in computational complexity.[5] The coding performance was evaluated based on the $\Delta\text{Bitrate}[(B_{\text{PRO}} - B_{\text{REF}})/B_{\text{REF}} \times 100]$ and $\Delta\text{PSNR}(P_{\text{PRO}} - P_{\text{REF}})$ performance measures and the time reduction was evaluated based on $\Delta\text{Time}\,[(T_{\text{PRO}} - T_{\text{REF}})/T_{\text{REF}} \times 100]$. For the experiment, we set the Threshold value as SKIP mode. Additional details of the encoding environment are described in Table 2.

Table 3 shows the total encoding time of HM 3.0 and the proposed method in the low-delay scenario as well as the random-access scenario.[6] The average results for each class are presented in Table 3 even though we tested all sequences with different QP values described in Table 2. Table 3 shows that the total time of the proposed method was reduced to 63.34% of that of HM 3.0 in a low-delay scenario and was reduced to 59.43% of that of HM 3.0 in a random-access scenario, respectively. The coding gain, with a minor degradation in picture quality, is the natural outcome of the proposed method because the coding tree pruning process favors the efficiency of bit reduction over picture quality. By adjusting the Threshold value, a further reduction in computational complexity can be achieved while the degradation in picture quality will be more severe. So far, the proposed method, with the Threshold value set to SKIP mode, performed the best in terms of both computation time and in coding efficiency.

4 Conclusion

Herein we proposed a fast CU decision method for HEVC by early determination of the CU size based on coding tree pruning, which yielded a reduction in encoding time of approximately 40% when compared to the HM 3.0 encoder with only a negligible loss of BD-bitrate. The experimental results showed that the proposed coding tree pruning method should be considered when designing a fast HEVC encoder. The proposed method was adopted in HEVC test model 4.0 encoder at JCT-VC 6th meeting.

Acknowledgments

This work was supported by Seoul R&BD Program (PA100094), Korea.

References

1. T. Wiegand et al., "Special Section on the Joint Call for Proposals on High Efficiency Video Coding (HEVC) Standardization," *IEEE Trans. Circuits Syst. Video Technol.* **20**(12), 1661–1666 (2010).
2. W. J. Han et al., "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools," *IEEE Trans. Circuits Syst. Video Technol.* **20**(12), 1709–1720 (2010).
3. D. M. Marpe et al., "Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding," *IEEE Trans. Circuits Syst. Video Technol.* **20**(12), 1676–1687 (2010).
4. K. Choi, S. H. Park, and E. S. Jang, "Coding tree pruning based CU early termination" presented at *JCTVC-F092, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 6th Meeting*, JCTVC, Torino, Italy, 14–22 (July 2011).
5. High Efficiency Video Coding Test Model software 3.0, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware.
6. F. Bossen, "Common test conditions and software reference configurations" (presentation, *JCTVC-E700, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 5th Meeting*, JCTVC, Geneva, CH, March 16–23 (2011).