Fast Still Picture Coding for VVC

Kei Kawamura* and Kyohei Unno* and Yoshitaka Kidani*
* KDDI Research, Inc., Saitama, Japan
E-mail: kei@kddi-research.jp Tel/Fax: +81-80-5066-9075

***Abstract*—State-of-the-art video coding technology, called Versatile Video Coding (VVC), provides double performance compared with the High Efficiency Video Coding (HEVC) for video sequences. Such video codecs can also provide better performance for still pictures. The complexity of VVC for still pictures, however, is very high for practical use. In order to reduce the encoding runtime, in this paper, we propose a combination of a fast coding configuration, an optimized implementation with parallel processing, and a fast algorithm for an intra mode decision. We also investigate the speedup factor of each proposed step. Finally, an average speedup factor of 710 times over default VTM is realized.**

I. Introduction

A still picture is one of the most consumed digital media on the Internet. The most successive compression technology is JPEG, which is standardized in 1992. The latest compression technology is a part of the video coding one, called Versatile Video Coding (VVC) [1]. VVC is standardized in July 2020, which is state-of-the-art. This trend is recognized from HEVC Still Picture Profile in 2012. The file format with HEVC for still pictures is also defined and widely used as a High Efficiency Image File Format (HEIF).

Coding performance and complexity is a trade-off relationship. JPEG provides roughly 1/10 compression while the complexity is very low. HEVC and VVC realize much higher compression while the complexity is huge. From the encoder point-of-view, the main reason for the huge complexity is rate-distortion optimization-based search for various partitioning types and many intra-prediction modes. For instance in VVC, partitioning candidates are quad-tree, binary tree, and ternary tree with hierarchical structure while the number of the intra prediction modes is up to 65. Furthermore, several transform matrixes are defined such as DST/DCT-II/DCT-VII and low-frequency non-separable transform [2].

This paper presents a fast still picture coding method compliant with VVC. VVC is efficient for both motion pictures and still picture. The coding performance of still picture by VVC is roughly -25% compared with that by HEVC while the encoding runtime increases by approximately 25 times when single instruction multiple data (SIMD) implementations are used [3]. To overcome such huge complexity, we propose a fast coding method for still pictures. Our key idea is a combination of fast coding configuration, optimized implementation with coding-tree-unit-based parallel processing, and fast algorithms on an intra mode search. We also investigate the speedup factor of each step based on the content characteristics.

In Section II, we briefly explain the existing fast implementation of both encoder and decoder compliant with VVC. In Section III, we describe the proposed method. In Section IV, we present the detail of the simulation results and extensive observation. Finally, we conclude in Section IV with a brief summary.

II. Related works

Several VVC encoders and decoders, which are compliant with the international standard, are already studied so far. The most used encoder/decoder is the VVC test model, called VTM, as the reference software for the development of the VVC standard in JVET [4]. At the later stage of the VVC development, a fast VVC encoder/decoder implementation, called VVenC/VVdeC, is proposed and both VTM and VVenC are used for verification test in the subjective evaluation [5]. Alternatively, S. Fang and J. Cui propose optimized VVC software encoder implementation [6], [7].

Fast decoder implementation is proposed and demonstrated. F. Bossen initially proposes a reasonably fast VVC software decoder [8]. Later, Y. Li, W.-L. Feng, and Y. He propose VVC software decoders for mobile platform [9]–[11]. F. Hiron proposes VTM based decoder supporting multi-thread [12]. In this paper, fast/optimized decoder implementation is out of scope while the generated bitstream by the proposed method is decodable by the third party decoder since the bitstream is compliant with the VVC standard.

Not only encoder or decoder packages but also each coding tool is also studied so far. In another word, performance optimization and complexity reduction in video coding is an extensive research topic for both software and hardware implementation. N. Tang proposes a fast block partitioning algorithm for intra and inter block [13]. For intra block, a Canny edge detector is used to find a flat region or one-directional edges region. T. Amestoy proposes machine learning-based split decision classifiers [14]. M. Aklouf presents the analysis to identify a subset of VVC coding tools that optimizes encoding time-saving with regard to a maximum acceptable BD-rate loss compared with the default VTM configuration [15]. A. Wieckowski reviews fast partitioning search algorithms available in the VTM reference software [16]. Compared with an extensive partitioning search, complexity can be reduced from the results by the default configuration and encoder speedup of 7 times while BD-rate increase of 1.1% [5].

To the best of our knowledge, the fast still picture encoding studies have been not yet reported, even though the above studies contain both still and motion pictures coding. In this paper, we present room for the faster VVC encoding method for still pictures. Additionally, we investigate the relationship

between the speedup factor and content characteristics. Finally, we describe the encoding time for several still pictures.

III. Overview of the Proposed Faster Encoding

A. Fast Encoding Configuration

A fast encoding setting is introduced by disabling heavy complexity tools. For instance, DualTree, BinaryTree, TernaryTree, ALF, SAO, DQ, LMCS, MTS, TransforSkip, RDOQ, LFNST, ISP, MIP, JCC, MRL, and IBC are disabled. Each abbreviation can be confirmed in the VVC algorithm description [2]. An approach of this configuration is similar to the fast configuration of VVenC [5].

B. Optimized Implementation and Parallel Processing

For the optimized implementation, some scalar operation is replaced by single-instruction-multiple-data (SIMD). In VTM, SIMD implementation is provided for heavy computation parts like sum-of-absolute-difference (SAD), sum-of-square-difference (SSD), sum-of-transformed-difference (SATD), interpolation filter, adaptive-loop filter, and so on. We further implemented by SIMD technique for discrete transform, inverse transform, quantization, inverse quantization, intra prediction, deblocking filter, and sample adaptive offset filter.

For parallel processing, the minimum operation unit is the coding-tree-unit. Consequently, wave-front parallel processing (WPP) is realized. Cabac initialization for WPP is also enabled. When slice structure is utilized, each slice is also parallelly processed.

Another operation unit is in-loop filters. Horizontal and vertical deblocking filters are separately processed by different threads.

C. Fast Encoding Algorithm

For the fast intra coding, well-known search restriction techniques are introduced.

Intra coding consists of two factors; one is the intra mode decision and the other is block partitioning. The intra mode decision means to select the best intra mode from the total 65 modes in VVC. In detail, the rate-distortion optimization approach is the basic strategy. The cost value is calculated from the distortion and the number of bits for each intra prediction mode. The mode with the lowest cost is selected as the best mode in terms of coding performance.

On the VTM approach, costs are calculated for planar, DC, and half the number of angular modes for both luma and chroma. When the best mode is an angular mode, two additional costs are calculated for neighboring directions. Compared with the full search, the VTM approach can reduce the complexity by roughly half.

In this paper, for the luma component, initially obtaining the costs of planar and DC, we use the greedy algorithm with coarse-to-fine search in angular direction granularity. To avoid the local minimum decision, we also compared it with the modes in MPM lists.

Intra mode is separately selected for luma and chroma. For the chroma component, the intra prediction mode is selected only from the linear prediction mode and the same mode of luma.

For block partitioning, we don't modify the partitioner algorithm. Compared with the common test configuration, additional early terminations are enabled.

IV. Results and Discussions

A. Simulation set-up

The proposed methods are implemented on the top of VTM-10.0 [4]. The coding conditions follow all intra settings in the common test conditions (CTC) [17]. The BD-rate metric is used for the coding performance evaluation based on the bitrate and distortion between two coding methods [18]. As mentioned in section II, a lot of coding tools are disabled and partitioning type is restricted to quadtree structure. To confirm the wide bitrate range, four QP values are used.

Test sequences are the same as in the CTC but the length of frames is shortened for reducing simulation time. Test sequences are categorized by the resolutions. The resolutions of Class A1/A2, B, and C are 3840 pixels $\times$ 2160 lines, 1920 pixel $\times$ 1080 lines, and 832 pixels $\times$ 480 lines, respectively. Input bit-depth of the sequence is 10bit per pel. Although original sequences have their own frame rate from 25 fps to 60 fps, they are treated as a sequence of still pictures in this paper.

Thumbnails of all pictures are listed in Figure 1. Class A1 mostly focuses on the natural scene taken by a camera while Class A2 contains unique characteristics for evaluating the coding behavior. Class B and C are natural scenes, too.

B. Results by Fast Encoding Configuration

By introducing fast configuration to VTM, 38.40% BD-rate loss in overall average is observed in Table I. Such large losses can be reduced by enabling each tool. Fast encoding algorithms for each tool will be studied or integrated in the future.

At the same time, encoding time is dramatically reduced in the range of 1.5% to 9.1%. It means 31.5 times faster than that by the default configuration. From the Table I, three points are derived.

Firstly, when the result of Class A1 and A2 are compared, speedup ratios are 10.9x and 21.6x. The reason is that the RDO search range by each tool for Class A2 is wider than that for Class A1. In detail, the encoding time of Class A2 is roughly double that of Class A1 by the default configuration. The encoding times in both Class A1 and A2 by the fast configuration are very similar. An actual encoding time for each picture will be discussed later.

Secondly, comparing the results among classes, the speedup ratio is proportional to the picture resolution. The reason is RDO search-range reduction in both tool level and partitioning. The tool level discussion is the same as the above-mentioned. For the partitioning, the RDO search range by the binary tree and ternary tree, which are effective for lower resolution, is disabled so that encoding runtime is reduced.

Fig. 1. Thumbnail of test sequences. From the top row to down, Class A1, A2, B, and C, respectively.

Finally, decoding runtime is reduced to 74% on the overall average. Since the decoder is the VTM in both anchor and test, reduction comes from the absence of complex coding tools. The reduction trend is consistent among classes.

C. Results by Fast Encoding Algorithm

By introducing a fast encoding algorithm into VTM with fast configuration, the encoding runtime of each class is summarized in Table II. It is noted that anchors in Table I and II are different. The encoding speedup of 5.4 times faster is realized while a small BD-rate loss of 7.45% in overall average is observed. In addition to the discussions of the previous section, two points are observed from Table II.

One point is that the BD-rate losses are similar among classes. The reason is that the proposed techniques are independent of the resolution and content characteristics.

The other point is that the higher the resolution, the larger the speed-up ratio. The reason is that the cost computation, which is optimized by the SIMD implementation, is proportional to a processing unit size where the unit size can be large in high resolution.

It is noted that the reduction of decoding runtime is very minor because the decoding process is not changed.

D. Multi-thread Performance Results

In this section, multithread performance is discussed. Our implementation can be scaled to a hundred threads but it is effective only for the motion picture content. For some realistic applications, we compare the single thread and four threads cases. Even the number of threads is small, the set-up and tear-down costs of the multithreading process are not negligible.

TABLE I
ENCODING PERFORMANCE WITH THE FAST VTM CONFIGURATION. ANCHOR IS THE DEFAULT VTM CONFIGURATION.

	BD_{YUV}	EncT	Speedup	DecT
Class A1	36.58%	9.1%	10.9x	71.7%
Class A2	41.01%	4.6%	21.6x	76.4%
Class B	36.95%	2.4%	41.4x	75.8%
Class C	39.61%	1.5%	65.9x	72.2%
Overall	38.40%	3.2%	31.5x	74.1%

TABLE II
ENCODING PERFORMANCE WITH THE PROPOSED FAST ENCODING ALGORITHM. ANCHOR IS THE FAST VTM CONFIGURATION.

	BD_{YUV}	EncT	Speedup	DecT
Class A1	7.60%	11.9%	8.4x	98.6%
Class A2	6.67%	12.8%	7.8x	93.0%
Class B	6.88%	23.1%	4.3x	98.0%
Class C	8.63%	25.7%	3.9x	92.9%
Overall	7.45%	18.5%	5.4x	95.7%

In this paper, we assumed that the small number of sets of pictures are sequentially encoded.

The encoding runtime of each sequence is summarized in Table III. Coding performance does not change regardless of the number of threads. From this table, three points are confirmed.

Firstly, encoding time is basically proportional to the image resolution. The reason is that resolution-specific search is almost disabled by both tool level and partitioning. In detail, remaining coding tools such as intra prediction, transform, and deblocking filter depends on only the number of pixels. The partitioning is also simplified to the quad-tree structure. As a result, encoding time depends on only the input image resolution.

Secondly, encoding time, however, depends on the content characteristics. Comparisons between Figure 1 and Table III suggest the following tendency. Encoding times of FoorMarket and RitualDance are relatively small since this content has a large region with a similar texture. On the other hand, the encoding time of Class A2 is relatively large since the content has strong edges and fine textures.

Finally, the speedup ratio is 3.6 times on the overall average. The ideal or upper limit ratio is four because of four-thread conditions, then the observed ratio is reasonable. It means that the overhead of multi-thread is the only 3.0% of the single-thread running time. The speedup ratio of Class A1/A2 is consistently larger than that of the remaining class. The main reason is that Class A1/A2 utilizes a four-slice structure due to the 4K resolution.

It is eventually realized an average speedup factor of 710 times over default VTM by using fast configuration, optimized implementation, and fast algorithm in still picture coding.

V. Conclusion

This paper proposed the fast still picture coding compliant with the VVC standard. Three steps are shown. Firstly, fast coding configuration is introduced by disabling heavy coding tools on the VTM configuration setting. Secondly, additional SIMD implementation and multi-threading framework are introduced into VTM. Finally, well-known but fast algorithms are implemented by our own optimization. Simulation results show that 4K and full HD resolutions are compressed by less than 2 seconds and 0.7 seconds, respectively. In our simulation environment with four multi-threading configurations, a speedup factor of 710 times over VTM was reached. In this paper, although a coding performance is sacrificed for the ultra-fast coding, such performance will be repaired by introducing a fast algorithm of each coding tool.

TABLE III
Encdoing time per one picture for each Sequence.

Class	Sequence	Proposed (single thread)	Proposed (four threads)	Speedup
A1 (4K)	Tango	6.4s	1.7s	3.8x
	FoodMarket	4.9s	1.3s	3.8x
	Campfire	6.7s	1.8s	3.6x
A2 (4K)	CatRobot	7.5s	2.0s	3.8x
	DaylightRoad	8.1s	2.1s	3.8x
	ParkRunning	9.0s	2.4s	3.7x
B (HD)	MarketPlace	1.9s	0.6s	3.4x
	RitualDance	1.5s	0.4s	3.4x
	Cactus	2.3s	0.7s	3.4x
	BasketballDrive	2.1s	0.6s	3.5x
	BQTerrace	2.4s	0.7s	3.3x
C (WVGA)	BasketballDrill	0.5s	0.1s	3.5x
	BQMall	0.5s	0.1s	3.6x
	PartyScene	0.6s	0.2s	3.5x
	RaceHorses	0.5s	0.2s	3.2x

Acknowledgment

This work was supported by Ministry of Internal Affairs and Communications (MIC) of Japan (Grant no. JPJ000595).

References

[1] ITU-T "Recommendation H.266: Versatile video coding," 2020.
[2] J. Chen, Y. Ye, and S. Kim, "Algorithm description for Versatile Video Coding and Test Model 12 (VTM 12)," JVET-U2002, Joint Video Experts Team (JVET), 2021.
[3] F. Bossen, X. Li, K. Sühring, K. Sharman, and V. Seregin, "JVET AHG report: Test model software development (AHG3)," JVET-V0003, Joint Video Experts Team (JVET), 2021.
[4] VTM software repository, vesion VTM-10.0. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-10.0
[5] J. Brandenburg, A. Wieckowski, T. Hinz , A. Henkel , V. George, I. Zupancic, C. Stoffers, B. Bross, H. Schwarz, and D. Marpe, "Towards Fast and Efficient VVC Encoding," IEEE 22nd Workshop on Multimedia Signal Processing (MMSP), 2020.
[6] J. Cui, Y. Fan, Y. He, X. Jiang, H. Liu, H. Shi, H. Yang, H. Yang, H. Yin, J. Zhang, and L. Zhang, "An optimized VVC encoder implementation," JVET-V0127, Joint Video Experts Team (JVET), 2021.
[7] S. Fang, J. Guo, Z. Huang, J. Liu, S. Xu, L. Yu, J. Chen, R.-L. Liao, and Y. Ye, "Ali266: an optimized VVC software encoder implementation," JVET-W0127, Joint Video Experts Team (JVET), 2021.
[8] F. Bossen, "AHG16: Performance of a reasonably fast VVC software decoder," JVET-S0224, Joint Video Experts Team (JVET), 2020.
[9] Y. Li, S. Liu, Y. Chen, Y. Zheng, S. Chen, B. Zhu, and J. Lou, "Performance of a VVC Software Decoder on Mobile Platform," JVET-U0071, Joint Video Experts Team (JVET), 2021.
[10] W.-L. Feng, F.-L. Luo, Y.-S. He, Z.-H. Liu, S.-M. Meng, and X. Wen, "VVC Software Decoder for Mobile Platforms," JVET-V0070, Joint Video Experts Team (JVET), 2021.
[11] Y. He, L. Li, Y. Li, H. Yin, J. Zhang, L. Zhang, and Y. Zhang, "Performance of a VVC software decoder - BVC," JVET-V0128, Joint Video Experts Team (JVET), 2021.
[12] F. Hiron, R. Jullian, F. Urban, and P. de Lagrange, "Multi-thread VTM decoder: information update," JVET-V0088, Joint Video Experts Team (JVET), 2021.
[13] N. Tang, J. Cao, F. Liang, J. Wang, H. Liu, X. Wang, and X. Du, "Fast CTU Partition Decision Algorithm for VVC Intra and Inter Coding," 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2019.
[14] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard, and C. Bergeron, "Tunable VVC Frame Partitioning Based on Lightweight Machine Learning," IEEE Transactions on Image Processing, 2020.
[15] M. Aklouf, M. Leny, F. Dufaux, and M. Kieffer, "Low Complexity Versatile Video Coding (VVC) for Low Bitrate Applications," 8th European Workshop on Visual Information Processing (EUVIP), 2019.
[16] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, and T. Wiegand, "Fast Partitioning Decision Strategies for The Upcoming Versatile Video Coding (VVC) Standard," IEEE International Conference on Image Processing (ICIP), 2019.
[17] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, "JVET common test conditions and software reference configurations for SDR video," JVET-N1010, Joint Video Experts Team (JVET), 2019.
[18] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," Technical Report VCEG-M33, ITU-T SG16/Q6, Austin, Texas, USA, 2001.