Encoding Complexity Analysis and Reduction for a Practically-Oriented VVC Encoder Implementation

Ivan Zupancic, Benjamin Bross, Tobias Hinz, Detlev Marpe
Video Communication and Applications
Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, Berlin, Germany
{name.surname}@hhi.fraunhofer.de

***Abstract*—The latest international Versatile Video Coding (VVC) standard was finalized in July 2020 by the Joint Video Experts Team (JVET) from ITU-T and ISO/IEC. Compared to the High Efficiency Video Coding (HEVC) standard, VVC offers up to 50% of bitrate savings. However, the encoder runtime of the reference software increases tenfold compared to the HEVC equivalent. This paper shows that VVC coding tools which do not consume much of the reference encoder time can have prohibitive computational cost for practical encoders, such as VVenC. VVenC is an optimized open-source real-world implementation with high relevance for practical applications. After analyzing the encoding tools in VTM and VVenC, Symmetric Motion Vector Difference (SMVD) was identified as one of the tools whose efficiency-complexity trade-off may be unfavorable for practical applications. For that reason, two complexity reduction methods for the SMVD search are proposed in this paper. The experimental evaluation on VVenC encoder at an optimized operation point confirms the SMVD encoding runtime reduction by over 72%, with minimal impact on the encoding efficiency.**

***Index Terms*—Symmetric Motion Vector Difference (SMVD), complexity reduction, Versatile Video Coding (VVC), VVenC**

I. Introduction

Recently finalized H.266/Versatile Video Coding (VVC) standard [1] offers substantial coding improvements compared to its predecessor, H.265/High Efficiency Video Coding (HEVC) standard [2]. The VVC was developed by the Joint Video Experts Team (JVET) as a response to an increased demand for a better compression efficiency of emerging video applications with exceptionally high data rate. Some of these applications include videos at 8K Ultra High Definition (UHD) resolution, High Dynamic Range (HDR), High Frame Rate (HFR), and 360° videos. Depending on the type of content and resolution, the VVC can achieve bitrate savings from 30% to over 50%, for the same level of visual quality compared to the HEVC [3]. The VVC also natively supports coding of screen content, reference picture rescaling for adaptive streaming with resolution switching, and independent sub-pictures for tiled streaming of 360° videos [4].

The superior coding performance of VVC was accomplished with many new and improved tools. The quad-tree splitting of HEVC [5] was extended in VVC by a nested multi-type tree using binary and ternary splits [6]. Many existing tools, such as Motion Vector (MV) prediction, merge mode, or Motion Vector Difference (MVD) coding, were generalized or extended. Moreover, new coding tools, such as affine motion model, Bi-prediction with Coding Unit (CU)-level Weights (BCW), Bi-Directional Optical Flow (BDOF), or new intra prediction tools were added [7].

All these improvements resulted in a high computational cost of the VVC encoder. The encoder runtime of the reference implementation VVC Test Model (VTM) [8] increased ten times [9], compared to the HEVC Test Model (HM) reference software [10]. Furthermore, the decoding time has almost doubled, compared to the HM. The most time consuming encoding tools were already identified during the VVC standardization process [11]. For obvious reasons, the computational complexity reduction of those tools is among the top priorities for any practical VVC encoder, such as VVenC [12]. VVenC is an optimized open-source encoder implementation, which is based on VTM software redesign to mitigate performance bottlenecks and includes extensive SIMD optimizations, improved encoder search algorithms and supports parallelism. However, when analyzing the VVC tools in VVenC, it becomes evident that many tools which do not account for much of the VTM encoding time can have unacceptable computational cost for practical VVC implementations.

One of the tools which does not consume much of VTM encoding runtime is Symmetric MVD (SMVD) mode for bi-prediction [13]. In the SMVD mode, motion information including reference picture indices of both list 0 and list 1, as well as MVD of list 1 are not signaled but derived [14]. At the encoder, optimal SMVD is obtained via SMVD search, spending roughly 3% of total encoding time in VTM-10.0. However, when measured in optimized VVenC encoder, the relative encoding time spent on SMVD search rises to over 23%. This paper proposes two methods for computational complexity reduction of the SMVD search, where one method favors the encoding efficiency, while the other method favors the speed. In both proposed methods, different stages of SMVD encoder search are optimized in terms of computational complexity, with a minor impact on the encoding efficiency.

The rest of this paper is organized as follows. A review of the related works is given in Section II. Analysis of the encoding time distribution in two different VVC implementations is presented in Section III. Section IV describes the proposed methods for computational complexity reduction of the SMVD motion search. Finally, the experimental results are reported in Section V, while Section VI concludes the paper.

2021 Picture Coding Symposium (PCS) | 978-1-6654-2545-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/PCS50896.2021.9477405

II. Related Works

Despite the very recent completion of the VVC standard in July 2020, several works from the literature target computational complexity reduction of various parts of the VTM reference encoder.

The works on fast block partitioning methods were among the first reported ones. A fast block partitioning algorithm for intra and inter predicted blocks was presented in [15]. The approach tries to detect flat areas in intra blocks, as well as areas dominated by edges in a single direction. In inter blocks, a three picture difference is computed to separate moving objects from the background, which would presumably require less partitioning. The resulting encoding time reduction was 31% for 1.3% of Bjøntegaard Delta (BD)-rate loss. Another fast block partitioning algorithm uses classifiers based on machine learning for split decisions [16]. The authors report encoding time savings in the range from 25% to 60%, for respective BD-rate losses of 0.4% to 2.2%. Furthermore, fast partitioning search algorithms available in one of the earlier versions of VTM reference software were reviewed in [17]. The proposed complexity reductions resulted in an encoding time reduction of 86% for 1.1% BD-rate loss, compared to an extensive partitioning search. A combination of fast block partitioning and fast mode estimation for intra blocks was reported in [18]. The method uses decision tree classifiers for split decisions and a gradient-based search algorithm for intra mode estimation. The reported average encoding runtime saving in All Intra (AI) configuration was 63% for 1.9% of BD-rate loss.

Other works from the literature focus on the analysis of the VVC tools and propose some optimizations. The analysis presented in [19] tries to identify a subset of VVC coding tools which optimizes the trade-off between the encoding performance and time. Using a branch-and-pruning search strategy, the optimal tool configuration was proposed which reduces the encoding time by 56% for 2% of BD-rate loss for low resolution videos. Another work provided an analysis of the most important intra prediction tools and calculated the theoretical upper limits for their complexity reduction [20]. According to the analysis, complexity reductions between 55% and 97% can be achieved for the most important intra prediction tools. Recently, the trade-off between the encoding efficiency and complexity was analyzed for different VVC tools [21]. Furthermore, software and algorithmic optimizations were combined to achieve the operating point 22 times faster than the VTM encoder in a single-threaded mode.

Finally, an encoder speed-up for SMVD was proposed and adopted in the context of VTM [22]. The approach limits the maximum number of initial MV candidates to 5 and modifies the distortion calculation for SMVD to align it with the regular bi-prediction search. The method reduced the total encoding time by 6% without affecting the encoding performance.

The complexity reductions proposed in this paper will be developed and tested using the optimized VVenC encoder [12] based on [21]. Moreover, the proposed complexity reductions will be implemented in addition to the speed-ups from [22].

III. Encoding Time Analysis of VVC Tools

In this section, encoding time of different VVC tools will be analyzed for two different encoders, the VTM-10.0 reference encoder [8] and the optimized VVenC encoder v0.2.0.0 [12].

The VTM reference software was developed during the VVC standardization process and was not designed for speed, but for demonstrating the capabilities of the VVC standard in a software codec. Due to the tenfold encoding time increase of VTM compared to HM [9], any practical VVC implementation would require a massive reduction of the encoder computational complexity. On the other side, VVenC was developed with the aim of reducing the computational complexity of the VTM encoder by means of software and algorithmic optimizations [12]. VVenC comes with five speed presets (slower, slow, medium, fast, faster), where each preset represents a different trade-off between the encoding efficiency and speed. In this paper, the faster preset will be used for analysis and encodings, as it includes a fast VVC partitioning scheme and some basic VVC tools, on top of the basic coding principles inherited from HEVC. When run single-threaded, its encoding runtime is almost 90 times faster than that of the VTM encoder, while even higher encoding time reductions can be achieved when run multi-threaded. This makes VVenC well suited as a basis for the development of an optimized VVC encoder which targets practical applications. Naturally, since many VVC tools are disabled in the faster preset, this results in significant BD-rate loss compared to VTM-10.0. However, that is still over 11% better in terms of BD-rate than HM-16.22.

To prove the necessity of complexity reductions of virtually all VVC tools, an encoding time saving (ΔT) of various VVC tools, along with the corresponding YUV BD-rate (BD_{YUV}) and gradient angle θ are shown in Table I. The ΔT is defined as:

$$\Delta T = \frac{ET_{test} - ET_{anchor}}{ET_{anchor}} \times 100\%, \tag{1}$$

where ET_{test} and ET_{anchor} are encoder runtimes of tested method and anchor, respectively. BD_{YUV} is calculated as a weighted average of the BD-rates for Y, U, and V components:

$$BD_{YUV} = \frac{6BD_Y + BD_U + BD_V}{8}. \tag{2}$$

The gradient angle θ is calculated for each tool in tool off configuration as:

$$\theta = \arctan\left(\frac{\Delta T}{BD_{YUV}}\right), \tag{3}$$

with lower absolute values of θ corresponding to better trade-offs between the encoding efficiency and complexity. The experiments were conducted on the sequences from Common Test Conditions (CTC) [23] under Random Access (RA) configuration. Two encoders with different configurations were used to obtain the results. The first experiment was conducted with the VTM-10.0 encoder in a tool off setting, i.e. a single tool disabled at the time, while the second experiment used

TABLE I
RELATIVE ENCODING TIME OF DIFFERENT VVC TOOLS IN VTM (TOOL OFF) AND OPTIMIZED VVENC IN FASTER PRESET (TOOL ON)

Tool	VTM 10.0 (tool off)			VVenC v0.2.0.0 (tool on)	
	BD_{YUV}	ΔT	θ	BD_{YUV}	ΔT
Affine	2.88%	-17.6%	-80.71°	-6.32%	232.7%
AMVR	1.63%	-15.1%	-83.84°	-3.17%	51.3%
MMVD	0.50%	-6.9%	-85.86°	-1.71%	40.4%
LFNST	0.76%	-5.6%	-82.27°	-1.37%	5.8%
MIP	0.33%	-4.6%	-85.90°	-1.01%	12.8%
ALF	8.05%	-3.4%	-22.90°	-4.85%	12.6%
SMVD	0.24%	-3.4%	-85.96°	-1.44%	23.1%
GPM	0.79%	-3.2%	-76.13°	-2.29%	29.8%
BDOF	0.65%	-2.2%	-73.54°	-1.27%	3.0%
CIIP	0.19%	-2.1%	-84.83°	-0.69%	7.6%
JCCR	0.47%	-1.4%	-71.44°	-0.34%	2.6%
DQ	2.07%	-1.2%	-30.10°	-2.22%	17.3%

VVenC v0.2.0.0 in faster preset in a tool on setting, i.e. a single tool enabled at the time. It should be mentioned that all the tools listed in Table I are disabled by default in the configuration of VVenC faster preset. The tools in Table I are sorted in descending order based on their relative encoding time savings in VTM. More information on tested tools and the corresponding abbreviations can be found in [6].

The results shown in Table I reveal that the relative encoding time, as well as the BD-rate gain of every tool is typically significantly higher when tested in tool on setting, compared to the tool off test. The relative encoding time increase is expected and can be explained with the fact that the absolute encoding times of the two tested encoders differ greatly, while at the same time, the absolute encoding time of each tool is generally independent of the tested encoder and configuration. The BD-rate gain increase for the tool on configuration can be justified with less tool interaction in this configuration. Finally, it should be noted that the implementations of the same tool in the two tested encoders are equivalent. Namely, the tested tool would produce bit exact results among the two encoders when tested on top of the aligned configurations of both encoders.

When analyzing the θ values for different tools, a high absolute value of θ can be an indication of a poor trade-off between the encoding efficiency and complexity. The SMVD, along with MIP and MMVD, is the tool with the highest absolute θ value, meaning that its trade-off between the encoding efficiency and complexity is probably undesirable for practical applications. In reality, this may result in disabling such tools in encoding configurations where computational resources are restricted, unless the tools are optimized.

As can be seen in Table I, the relative encoding times of various VVC tools in tool off setting range from 0.1% to almost 18%. The range becomes even higher for the tool on setting, with a maximum value of over 232%. When comparing the order of tools by their relative encoding times between the two encoders, there are some discrepancies. The SMVD tool again stands out, along with Affine and GPM, with over 23% of relative encoding time in VVenC, compared to just 3% in VTM. Hence, even though there are tools that account for more encoding time than SMVD, its unfavorable trade-off, indicated by a high absolute θ value, its higher relative encoding time in tool on configuration, and an increased BD-rate gain in tool on configuration, make it a good candidate for computational complexity reduction, in order to be become more suitable for practical encoder implementations.

IV. COMPLEXITY REDUCTION OF SMVD MOTION SEARCH

To get a better understanding of the proposed methods, the SMVD search process in VTM encoder will be described first in subsection IV-A. The proposed methods for computational complexity reduction of the SMVD mode will be described in detail in subsection IV-B.

A. SMVD in VTM Encoder

The SMVD motion estimation in the VTM encoder starts with initial MV evaluation. A set of up to 5 initial MV candidates comprises the MVs obtained from uni- and bi-prediction search, and the MVs from the AMVP list. The one with the lowest Rate-Distortion (RD) cost is chosen as the initial MV for the SMVD motion search.

The SMVD motion estimation in VTM reference encoder is performed in two stages. In the first stage, 8 neighboring positions in a large diamond pattern are examined around the initial position, as shown in orange Fig. 1 a). If any of the tested 8 points obtained better RD cost than the initial position, the search continues around the position with the lowest RD cost in up to 8 rounds of refinement, using the diamond refinement search pattern depending on the position, as denoted with green in Fig. 1 a). Otherwise, if none of the 8 tested points from the first search obtained better RD cost than the initial position, or if the refinement process of the first search is over, the second stage search is performed on a 4-point cross pattern around the position with the lowest RD cost, depicted with blue in Fig. 1 a). It has to be noted that the refinement rounds of the first stage are performed conditionally, i.e. the further refinement is terminated as soon as the current refinement round did not improve the RD cost.

B. Proposed SMVD Complexity Reductions

Two complexity reduction methods for SMVD are proposed, one that favors encoding efficiency, and the other one that favors speed. The proposed methods target different stages of the SMVD search and can be clustered into three sets.

The first set focuses on the initial MV position for the SMVD search. Experiments showed that up to 30% of the SMVD encoding time can be efficiently reduced with virtually no loss in encoding performance, when evaluating only 1 instead of 5 initial MV positions for the SMVD search. Hence, in both proposed methods, the number of evaluated initial MV positions for the SMVD motion search is limited to 1.

The second set of complexity reductions concentrates on the SMVD motion estimation. In both proposed methods, the testing order of the 8 positions in a large diamond pattern in the first search stage was modified. The 4 MV positions

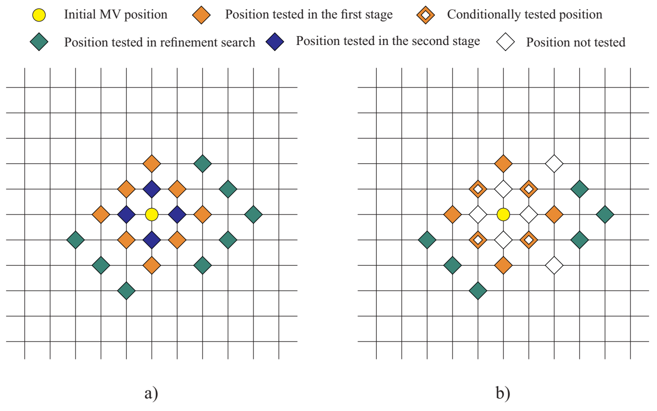
Initial MV position
Position tested in the first stage
Conditionally tested position
Position tested in refinement search
Position tested in the second stage
Position not tested
a)
b)

Fig. 1. Search patterns for SMVD motion estimation a) SMVD search patterns in VTM; b) proposed SMVD search patterns. All positions are on an integer sampling grid

(x_0, y_0+2), (x_0, y_0-2), (x_0+2, y_0), and (x_0-2, y_0) are tested first, where (x_0, y_0) are the coordinates of the initial MV position. These positions are marked with orange in Fig. 1 b). The remaining 4 positions of a large diamond pattern (x_0+1, y_0+1), (x_0+1, y_0-1), (x_0-1, y_0+1), and (x_0-1, y_0-1) (denoted with orange-white) are tested only if at least one of the four previously tested positions improves the RD cost. Otherwise, if none of the 4 tested points improved the RD cost, the first stage search is terminated, including also the refinement step. Moreover, the refinement pattern of the first search stage was also modified. When the optimal tested position in the first search stage is located at the edges of a large diamond pattern, only 3 instead of 5 positions are tested in the refinement step, as illustrated in Fig. 1 b). Both proposed methods include all the modifications described so far. Additionally, the second proposed method unconditionally skips the second stage of SMVD motion estimation.

The third set of modifications aims to conditionally skip the SMVD motion estimation in cases when it is unlikely that it would improve the overall RD cost. This set of modifications is used only with the second proposed method. The experimental analysis reported here was conducted on the first 8 non-I frames of class D sequences [23] encoded with the VTM-10.0 encoder according to CTC under RA configuration. In the first experiment, a relation between the RD costs of bi-prediction and SMVD mode was experimentally studied. The results showed that in over 70% of cases when the SMVD mode improved the overall RD cost, the bi-prediction mode had the lowest RD cost prior to testing the SMVD mode. Hence, the SMVD mode is conditionally tested in the proposed method only when the bi-prediction mode has the best RD cost before testing the SMVD mode. Another experiment was conducted to study the relations between the RD costs of the initial and final MV position for SMVD, as well as bi-prediction mode. The experiment showed that in more than 96% of cases when the SMVD mode improved the overall RD cost, the RD cost of the initial SMVD position was lower than that of the bi-prediction mode. Therefore, the SMVD motion estimation is skipped in the proposed method when the RD cost of the initial MV position for SMVD search is more than 2% greater than the RD cost of the bi-prediction mode.

The findings described in this subsection were used to develop two versions of complexity reduction algorithm for SMVD; the first one that favors the encoding efficiency, and the second one that favors the speed. The two proposed methods were implemented in VVenC and experimentally evaluated, as reported in the next section.

V. Experimental Results

The proposed methods were implemented in VVenC v0.2.0.0 and experimentally tested using VVenC full featured expert encoder application under faster preset with RA configuration. The simulations were conducted on an Intel Xeon E5-2697A v4 cluster with Linux OS and a GCC 7.2.1 compiler. The AVX2 optimizations were enabled, while hyper threading and turbo boost were disabled for the accurate runtime results. All the experiments were conducted on the sequences defined in CTC [23] and compared against a VVenC fast preset anchor. The experimental results are reported in terms of BD-rate (Y component) [24] and ΔT (defined in Eq. 1). The negative BD-rate values correspond to the encoding gain.

Two proposed complexity reduction methods were tested, one that favors the encoding efficiency (*Proposed 1*), and the second one that favors the speed (*Proposed 2*). Furthermore, the SMVD implementation from VTM-10.0 was also ported to VVenC v0.2.0.0 and tested along with the proposed methods. The results are shown in Table II. The SMVD implementation which was directly ported from the VTM software achieves 1.60% of BD-rate improvement for 23.1% of the encoding time increase. The first proposed method obtains 1.51% BD-rate gain for 14.8% encoding time increase. Finally, the second proposed algorithm increases the encoding time by only 6.3% while still achieving 1.17% BD-rate gains. Compared to the SMVD implementation from VTM, the first proposed complexity reduction method reduces the encoding time of SMVD mode by 36% with negligible loss in the encoding efficiency of just 0.09% BD-rate. The second proposed complexity optimization reduces the encoding time of SMVD mode by over 72%, while at the same time keeping over 73% of the encoding gains of the unoptimized SMVD implementation.

As the experimental results show, the proposed complexity reduction algorithms for SMVD mode provide much better trade-offs between the encoding efficiency and complexity than the SMVD algorithm ported from VTM. That can be quantified via the respective θ values of the proposed methods, which are $-84.75°$ and $-80.41°$, compared to $-86.43°$ for the unoptimized version in VVenC. Hence, the proposed SMVD complexity reduction methods are better suited for practical VVC encoding applications. As a consequence, the proposed method that favors the speed (*Proposed 2*) is enabled in medium and slow presets of VVenC v0.2.0.0.

TABLE II
EXPERIMENTAL RESULTS FOR THE PROPOSED COMPLEXITY REDUCTIONS FOR SMVD, ALONG WITH THE SMVD ALGORITHM PORTED FROM VTM 10.0 TO VVENC V0.2.0.0. ALL TESTS WERE CONDUCTED USING VVENC FULL FEATURED EXPERT ENCODER APPLICATION WITH FASTER PRESET UNDER RA CONFIGURATION

Sequence	SMVD ported from VTM 10.0		Proposed 1 (encoding efficiency)		Proposed 2 (speed)	
	BD_Y [%]	ΔT [%]	BD_Y [%]	ΔT [%]	BD_Y [%]	ΔT [%]
Tango2	-1.62	24.7	-1.52	15.3	-1.18	5.9
FoodMarket4	-3.17	26.4	-3.10	16.2	-2.63	7.6
Campfire	-0.44	19.9	-0.45	13.1	-0.34	4.2
Class A1 Avg.	**-1.74**	**23.6**	**-1.69**	**14.8**	**-1.38**	**5.9**
CatRobot1	-2.01	23.9	-1.87	14.4	-1.42	6.2
DaylightRoad2	-2.44	24.3	-2.33	14.6	-1.92	6.0
ParkRunning3	-1.80	22.3	-1.70	14.1	-1.36	7.5
Class A2 Avg.	**-2.08**	**23.5**	**-1.97**	**14.4**	**-1.57**	**6.6**
MarketPlace	-1.61	26.5	-1.58	16.4	-1.27	8.6
RitualDance	-1.38	25.9	-1.32	16.9	-0.95	7.5
Cactus	-2.17	25.0	-2.05	14.8	-1.59	7.1
BasketballDrive	-1.97	25.4	-1.81	16.1	-1.41	7.5
BQTerrace	-1.46	26.6	-1.34	16.3	-1.03	8.5
Class B Avg.	**-1.72**	**25.9**	**-1.62**	**16.1**	**-1.25**	**7.8**
BasketballDrill	-0.84	20.5	-0.79	14.0	-0.60	5.4
BQMall	-1.32	19.0	-1.25	13.5	-0.97	4.0
PartyScene	-0.95	17.6	-0.88	12.6	-0.64	4.8
RaceHorses	-0.75	18.5	-0.69	14.1	-0.27	4.0
Class C Avg.	**-0.96**	**18.9**	**-0.90**	**13.5**	**-0.62**	**4.6**
BasketballPass	-1.03	16.9	-0.93	13.0	-0.66	3.2
BQSquare	-1.04	18.5	-0.93	11.0	-0.62	5.9
BlowingBubbles	-0.92	15.7	-0.80	10.4	-0.55	3.4
RaceHorses	-0.63	17.8	-0.66	12.4	-0.45	4.0
Class D Avg.	**-0.91**	**17.2**	**-0.83**	**11.7**	**-0.57**	**4.1**
BasketballDrillText	-0.95	19.1	-0.87	13.3	-0.70	4.3
ArenaOfValor	-0.90	25.5	-0.83	12.9	-0.68	6.2
SlideEditing	-0.03	15.8	-0.09	8.7	0.03	1.1
SlideShow	-1.79	20.1	-1.75	12.2	-1.25	3.4
Class F Avg.	**-0.92**	**20.1**	**-0.89**	**11.7**	**-0.65**	**3.7**
CTC Avg.	**-1.60**	**23.1**	**-1.51**	**14.8**	**-1.17**	**6.3**

VI. CONCLUSIONS

Greatly improved encoding performance of the new VVC standard comes at a cost of very high encoder runtime increase. In this paper, it was shown that even though a specific encoding tool may not account for significant amount of encoding time in the VTM reference software, it may still have a prohibitive computational cost for a more practically-oriented VVenC encoder implementation. Based on this finding, two complexity reduction methods for the SMVD tool are proposed in this paper. The proposed methods utilize different strategies to minimize the number of tested motion vector positions in SMVD search. Experimental evaluations confirm the effectiveness of the proposed methods in reducing the encoder runtime of the SMVD mode by over 72% for minimal losses in encoding efficiency. Plans for the future work include developing methods for computational complexity reduction of other tools which may not be well suited for practical VVC encoder implementation, such as MMVD or GPM.

REFERENCES

[1] B. Bross, J. Chen, S. Liu, and Y. K. Wang, "Versatile video coding (Draft 10)," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-S2001, July 2020.

[2] V. Sze, M. Budagavi, and G. J. Sullivan, *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Springer, 2014.

[3] M. Wien and V. Baroncini, "Report on VVC compression performance verification testing in the SDR UHD random access category," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-T0097, October 2020.

[4] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC)," *Proceedings of the IEEE*, pp. 1–31, 2021.

[5] D. 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.*, vol. 20, no. 12, pp. 1676–1687, December 2010.

[6] J. Chen, Y. Ye, and S. Kim, "Algorithm description for versatile video coding and test model 10 (VTM 10)," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-S2002, July 2020.

[7] B. Bross *et al.*, "General video coding technology in responses to the joint call for proposals on video compression with capability beyond hevc," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1226–1240, May 2020.

[8] JVET, "VVC test model (VTM)," https://jvet.hhi.fraunhofer.de/, December 2020.

[9] F. Bossen, X. Li, and K. Suehring, "JVET AHG report: Test model software development (AHG3)," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-S0003, July 2020.

[10] JCT-VC, "HM reference software," https://hevc.hhi.fraunhofer.de/, December 2020.

[11] W. J. Chien *et al.*, "JVET AHG report: Tool reporting procedure and testing (AHG13)," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-S0013, July 2020.

[12] Fraunhofer HHI, "Fraunhofer versatile video encoder (VVenC)," https://github.com/fraunhoferhhi/vvenc, January 2020.

[13] H. Chen, H. Yang, and J. Chen, "CE4: Symmetrical MVD mode (test 4.5.1)," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-L0370, October 2018.

[14] J. Luo and Y. He, "CE4-related: Simplified symmetric MVD based on ce4.4.3," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-M0444, January 2019.

[15] N. Tang *et al.*, "Fast CTU partition decision algorithm for VVC intra and inter coding," in *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2019, pp. 361–364.

[16] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard, and C. Bergeron, "Tunable VVC frame partitioning based on lightweight machine learning," *IEEE Trans. Image Process.*, vol. 29, pp. 1313–1328, 2020.

[17] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, and T. Wiegand, "Fast partitioning decision strategies for the upcoming versatile video coding (VVC) standard," in *2019 IEEE International Conference on Image Processing (ICIP)*, September 2019, pp. 4130–4134.

[18] H. Yang *et al.*, "Low-complexity CTU partition structure decision and fast intra mode decision for versatile video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1668–1682, 2020.

[19] M. Aklouf, M. Leny, F. Dufaux, and M. Kieffer, "Low complexity versatile video coding (VVC) for low bitrate applications," in *2019 8th European Workshop on Visual Information Processing (EUVIP)*, 2019, pp. 22–27.

[20] A. Tissier *et al.*, "Complexity reduction opportunities in the future VVC intra encoder," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, September 2019, pp. 1–6.

[21] J. Brandenburg *et al.*, "Towards fast and efficient VVC encoding," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, September 2020.

[22] H. Chen and H. Yang, "AHG13: Encoder speed-up for SMVD," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-P0092, October 2019.

[23] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, "JVET common test conditions and software reference configurations for SDR video," Tech. Rep., Joint Video Experts Team (JVET), Doc. JVET-N1010, March 2019.

[24] G. Bjøntegaard, "Improvements of the BD-PSNR model," Tech. Rep., ITU-T SG16/Q6, Video Coding Experts Group (VCEG), Doc. VCEG-AI11, July 2008.