Fast partitioning strategies for VVC and their implementation in an Open Optimized Encoder

Adam Wieckowski, Benjamin Bross, Detlev Marpe
Video Communication & Applications Department
Fraunhofer HHI, Berlin, Germany
{firstname.lastname}@hhi.fraunhofer.de

***Abstract*— In July 2020 the new video coding standard Versatile Video Coding (VVC) was released. VVC delivers bit-rate savings of up to 50% relative to its predecessor High Efficiency Video Coding (HEVC) without compromising subjective quality. It has been designed for a broad range of applications and input material ranging from standard dynamic range to high dynamic range camera-captured content, screen content, and immersive applications. One of the key technologies included in the new standard is the flexible block partitioning using a quad-tree with multi-type tree structures (QT+MTT). Based on the VVC test model VTM, the VVenC project provides an open optimized VVC encoder. VVenC inherits and utilizes all of VTMs in-built partitioning search strategies. The QT+MTT partitioning search still poses a challenge for fast encoder implementations. After defining alternative low-depth partitioning configurations to VTM, it was observed that the initial search algorithm performs less optimal. We analyze and propose in this paper a set of additional fast search strategies for partitioning within VVenC, optimized for low-depth partitioning configurations. As a result, an average of 38% runtime reduction is achieved at the managable cost of less than 2% BD-rate loss.**

Keywords—VVC, Partitioning, Speedup, Quad-tree, Binary-tree.

I. INTRODUCTION

Versatile Video Coding (VVC) [1] is a new video coding standard, recently finalized by the Joint Video Experts Team (JVET), a shared effort of the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG). The new codec delivers an average of 50% bit-rate reduction [2] over its predecessor H.265/HEVC [3] at the same subjective quality. In addition to the support of natural camera-captured content using standard dynamic range (SDR), VVC offers a wide range of specific coding tools supporting different applications such as, e.g., screen content, high dynamic range (HDR), adaptive streaming, and immersive applications like 360-degree video or augmented reality.

Two months after the completion of the standard in July 2020, the first open and optimized encoder and decoder implementation, VVenC [4][5] and VVdeC [6], were released. The goal of the VVenC encoder is to deliver as much of the VVC compression capability as possible at a reasonable runtime. VVenC 0.3.1 supports five presets: “faster”, “fast”, “medium”, “slow”, and “slower”, with “slower” achieving all the gains of the VVC reference software VTM [7], however, at around half the runtime of VTM, or around 3.5× runtime of HM reference software HM [8].

VVC is designed as a block-based hybrid video codec, similar to HEVC and other state of the art video codecs. To deliver the promised bit-rate reduction and flexibility, the new standard supports a wide range of new high-level extensions, and block-level coding tools allowing for better prediction and residual coding. For encoding, an input picture is first partitioned into square coding tree units (CTU) of fixed size (up to 128x128 luma samples). High-level extensions allow further picture partitioning based on this basic subdivision to define logical sub-picture areas such as slices, tiles, or sub-pictures. Each CTU is further subdivided into rectangular coding units (CU) using a flexible partitioning scheme. The prediction and residual coding tools operate on the final CU blocks.

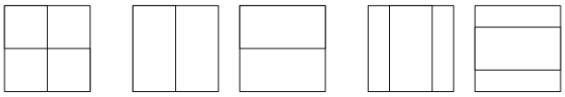

Fig. 1. Partitioning split types in VVC. From left to right: Quad split, binary splits (vertical and horinzontal), and ternary splits (vertical and horizontal).

An important improvement over HEVC is the flexible block partitioning scheme for a fine-tuned CTU-subdivision using a quadtree with multi-type tree (QT+MTT) [9]. In comparison to HEVC, which only allows recursive quad-tree (QT) splits, VVC allows four additional recursive splits by horizontally or vertically oriented binary and ternary splits. All split types available in VVC are shown in Fig. 1. A CTU is first partitioned using quad-splits, until it is signaled that no further quad-split is to be performed. The QT leaves are then further subdivided using a nested multi-type tree with binary and ternary splits. All of the splits are subject to some basic constraints, like the minimum block size or high-level parameter limits, like the maximum tree depth.

To find the optimal partitioning, the encoder will evaluate the available partitioning options (incl. no split) and choose the best one in terms of Lagrangian rate-distortion (RD) cost [10],

$$J = D + \lambda \cdot R, \tag{1}$$

with D being the distortion resulting from a particular encoding and R representing the rate, i.e., the number of bits required to signal the encoding. The Lagrange multiplier λ is a constant defining the desired tradeoff between quality and rate, usually derived from the quantization parameter QP [10]. For a split mode with k sub-blocks the cost (1) becomes

$$J_s = \lambda \cdot R + \sum_k J_k. \tag{2}$$

For recursive partitioning search, each of the optimal sub-block costs J_k might itself represent a split search result, increasing the search space in an exponential manner. The top-down generalized Breiman, Friedman, Olshen and Stone (G-BFOS) search strategy [11] employed in the VVC as well as in the HEVC reference software [7][8] already reduces the number of tested partitioning configurations. Still, the recursive partitioning introduces a very large search space, which requires further optimized search strategies.

In [12], we discussed heuristic speedup rules already available with the VVC reference software VTM and showed

2021 Picture Coding Symposium (PCS) | 978-1-6654-2545-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/PCS50896.2021.9477510

TABLE I. ENCODER RUNTIME-BD-RATE TRADEOFFS OF VVENC COMPARED TO "MEDIUM" PRESET WITH THE VTM PARTITONING ALGORITHM WITH AND WITHOUT PARTITIONING SPEEDUPS. ALTERNATIVE CONFIGURATIONS FOR THE VTM ALGORITHM ARE COMPARED TO THE MEDIUM CONFIGURATION. CORRESPONDING NO SPEEDUP RESULTS ARE RELATIVE TO THE RESPECTIVE CONFIGURATION WITH VTM ALGORITHM.

Max BTT Depth		Name	VTM Alg.		No Speedups	
Intra	Inter		BD_{YUV}	EncT	BD_{YUV}	EncT
0	0	BTT00	9.96%	38%	-0.27%	155%
VVenC "faster"						
1	0	BTT10	8.38%	40%	-0.55%	156%
VVenC "fast"						
1	1	BTT11	0.62%	97%	-0.76%	251%
2	1	BTT21	0.00%	100%	-0.79%	255%
VVenC "medium"						
3	1	BTT31	-0.37%	107%	-0.72%	259%
3	2	BTT32	-2.77%	172%	-1.39%	454%
VVenC "slow"						
3	3	BTT33	-4.23%	325%	-1.73%	692%
VTM-11 CTC						
VVenC "slower"						
4	3	BTT43	-4.38%	336%	-1.74%	685%
4	4	BTT44	-4.98%	588%	-2.17%	1059%

that, overall, over 85% of the encoding runtime can be saved at a cost of around 1% BD-rate loss [13]. Other approaches discuss learning-based heuristics to derive the split decisions with varying results [14-16]. Our previous work [12] is based on the JVET common test conditions (CTC) [17] and the corresponding partitioning configuration, i.e., allowing up to three consecutive binary and ternary splits.

In this work, we will discuss the high-level partitioning options available in the VVC standard and how they can be used to achieve different quality-speedup tradeoffs. Based on Pareto optimal partitioning configurations for an early version of VVenC [4] we will discuss specific challenges associated with fast partitioning schemes. Finally, we will discuss how some of the rules can be adapted for low-depth configurations and we will show how the convex hull of partitioning configurations can be extended.

All of the results presented in this paper were obtained using the VVenC software (version 0.3.1.0) running in "medium" preset for random-access configuration with the JVET CTC HD and UHD test sequences [17] (see Table III). Being based on VTM, the software can reproduce the VTM partitioning algorithm, which comprises the baseline for the discussion. To simplify the comparison, the Y, U, and V per component bit-rate savings in terms of Bjøntegaard delta (BD) rate are weighted to derive a single YUV BD rate

$$BD_{\mathrm{YUV}} = (8 \cdot BD_{\mathrm{Y}} + BD_{\mathrm{U}} + BD_{\mathrm{V}}) / 10. \quad (3)$$

The impact on encoder runtime is measured by EncT, which is calculated using the geometric mean of the encoding times relative to the runtime of the anchor ($100\% \cdot T_{test}/T_{ref}$) over all points in the current test, i.e. the four QP points tested per sequence for all considered sequences (as per [17]).

II. PARTITIONING IN VVC

The VVC standard allows to define the maximum depth as well as the maximum and minimum size of the QT and the embedded MTT, i.e. binary-and-ternary tree (BTT) structures. Specific configurations can be defined for intra-slices (for both luma and chroma trees in case of chroma separate tree) and the P/B (motion compensated) slices. Because of the recursive search performed by the G-BFOS algorithm, the search space grows exponentially with respect to available split depth – see (2). Most notably, the embedded MTT depth has a big influence on the search space.

In [4], a Pareto optimal set of partitioning configurations was estimated for an early version of VVenC over a broad range of options, including high-level partitioning configuration as well as a set of additional partitioning search strategies. Based on this analysis, the "medium" preset was configured to allow up to two recursive BTT splits for intra slices (both luma and chroma) and only one BTT split in inter-slices (i.e. no BTT recursion). Equivalent to VTM CTC, the maximum QT depth is set to 4 for all slice and tree types. Similarly, the "slow" preset has been chosen in a Pareto-optimal sense to allow 3 and 2 recursive BTT splits, for intra and non-intra slices, respectively. In this configuration, the maximum BTT depth for non-intra slices is lower than in VTM-11 CTC (where maximum BTT depth is set to 3 for all slice and tree types [18]). Only the "slower" preset uses the same partitioning depth configuration as VTM, while also not utilizing all of the methods presented within this paper (as will be discussed further).

In VVenC version 0.3.1, the "faster" and "fast" presets employ partitioning configurations with even less splitting allowed – i.e. the BTT splits are disabled in the "faster" configuration, and disabled for inter-frames in the "fast" preset [19][20]. For those two configurations, smaller CTU size of 64×64 is used. The CTU size does not have influence on the efficiency of the presented methods.

Table I shows different tradeoffs achievable on top of the VVenC "medium" configuration when using the VTM partitioning search algorithm. Additionally, for each configuration, the effect of disabling all the partitioning speedups at this point is discussed. The results roughly resemble the findings from [12]. At the same time it can be seen that the speedups introduced in VTM are generally higher the deeper the partitioning depths become.

The fast partitioning strategies in VTM were designed around its respective working point, with many recursion levels available for all slice- and tree-types. In the shallow VVenC "medium" partitioning configuration, some of those fast partitioning strategies cannot be applied. For example, some of the strategies are applied when exactly the same block (i.e. with the same size and at the same position) is revisited within the CU search loop in different partitioning configurations. Without BTT recursion, this case never occurs, rendering the methods inactive (for the non-intra slices). Similarly, some of the methods are only active at predefined non-zero depths of the BTT tree. Without recursion, a split/no-split decision will only be applied at BTT depth 0, rendering those rules inactive as well.

III. VVC PARTITIONING STRATEGIES

To overcome the limitations described in the previous section, we adapted four of the 12 search strategies described in [12] and created an additional new speedup rule, all of which provide good tradeoffs as long as BTT splitting is active, i.e. the maximum BTT depth is larger than zero. One rule introduces an attractive speed-up even if no BTT splitting is allowed at all.

TABLE II. ENCODER RUNTIME-BD-RATE TRADEOFFS FOR DIFFERENT VVENC PARTITIONING SEARCH STRATEGIES AND DIFFERENT PARTITIONING CONFIGURATIONS COMPARED TO A VVENC ANCHOR BASED ON "MEDIUM" PRESET WITH VTM PARTITIONING SEARCH IN THE RESPECTIVE PARTITIONING CONFIGURATIONS.

BTT Depth		Gradient Rule		Section 3.A.		Section 3.B.		Section 3.C.		Section 3.D.		Section 3.E.		All	
Intra	**Inter**	**BD_{YUV}**	**EncT**	**BD_{YUV}**	**EncT**	**EncT**	**EncT**	**BD_{YUV}**	**EncT**	**BD_{YUV}**	**EncT**	**BD_{YUV}**	**EncT**	**BD_{YUV}**	**EncT**
0	0	0.00%	100%	0.00%	100%	0.00%	100%	0.17%	93%	0.00%	100%	0.00%	100%	0.17%	94%
1	0	0.02%	101%	0.00%	100%	0.01%	99%	-0.05%	94%	0.08%	99%	0.05%	100%	0.07%	93%
1	1	0.49%	86%	0.21%	93%	0.69%	80%	-0.18%	97%	0.51%	90%	0.22%	98%	1.74%	61%
2	**1**	0.47%	87%	0.23%	93%	0.71%	81%	-0.12%	98%	0.52%	90%	0.22%	99%	1.80%	62%
3	1	0.56%	87%	0.23%	94%	0.72%	82%	-0.12%	98%	0.57%	89%	0.23%	99%	1.85%	63%
3	2	0.76%	82%	0.12%	96%	0.43%	87%	-0.15%	97%	0.67%	85%	0.00%	100%	1.51%	61%
3	**3**	0.87%	75%	0.00%	100%	0.30%	89%	-0.15%	96%	0.76%	78%	0.00%	100%	1.49%	55%
4	3	0.90%	76%	0.00%	100%	0.35%	90%	-0.16%	96%	0.79%	78%	0.00%	100%	1.51%	56%
4	4	0.97%	71%	0.00%	100%	0.31%	93%	-0.16%	95%	0.76%	73%	0.00%	100%	1.55%	50%

A. Skip condition with remaining depth restriction

In VTM, there exists a speedup rule allowing for skipping all of the BTT splits, if the non-split search resulted in a Skip mode was determined as having the lowest RD costs for that block (SKIPD in [12] and Section 3.A in Table II). The Skip mode indicates inter prediction with motion information copied from a merge candidate without residual. However, this rule is only applied at BTT depth 2 and higher. We modified the rule to be conditioned on the remaining available BTT depth (i.e. maximal depth minus current depth) not being larger than 1. This way the idea behind the constraint, i.e. only to skip testing of a limited number of partitioning combination, is preserved. For the VTM partitioning configuration, this has no effect as the rule is equivalent to its original in this setting, but still provides speedup with depth-reduced configurations as in the VVenC "medium" preset.

B. Skip with Sub-Split Skip

Similar to the previous speedup, this rule is also conditioned on a non-split block being chosen to be coded using the Skip mode in the CU-search. As a side condition, for at least any two further splits, all sub-blocks resulting from those splits must also be coded using the Skip mode. If the condition is satisfied, no further splits are tested at that level. This way, testing of all four BTT splits can be spared if both the QT split and non-split search results in all-Skip encoding (with QT having a special meaning and accounting for two splits). Otherwise up to two out of the four BTT splits and possibly the QT split will not be tested if two BTT splits and the non-split search result in all-Skip encoding.

C. Split cost prediction adaptation

As described in [12] and further detailed in [21], split cost prediction is used to determine if a split is probable to improve the best coding cost J_0. The predictor used in [12][21] is

$$J_P = \lambda \cdot R_S + J_0 / f. \quad (4)$$

The predicted cost J_p is derived from the rate R_s for signaling the split flag, the current optimal cost J_0, and a factor f describing the anticipated cost reduction with the split. In VTM, f is defined as being 1.1 for QPs larger than 30 and 1.075 otherwise.

We extended the prediction of (2) in (4) by the number of split subblocks k, resulting in the predictor

$$J_P = \lambda \cdot R_S + J_0 / f + k. \quad (5)$$

The factor f is increased to be 1.11 for QPs larger than 30, and 1.085 otherwise, which per [21] implies that the expected improvement from the split is larger. On the other hand, the added factor k, representing the number of subblocks in (2), heuristically describes the added cost of coding multiple blocks instead of one. Overall, compared to the formulation in VTM, the changes imply larger gains are expected, but at a higher overhead.

Additionally, when coding the chroma component separately from luma (i.e. for the chroma separate tree case), f is increased by 0.1, implying the split is expected to have high impact on the block coding costs. The last adaptation causes the method to reduce speedup loss (see Section 3.C. in Table II) resulting overall in minor speedup with a bit of gain.

D. Test ternary split parallel to better binary split

Further, a new rule not based on any of the preexisting rules was introduced. After both binary splits have been tested, only the ternary split parallel to the binary split resulting in lower coding cost J_s will be considered. This rule is applied independent of any side-conditions. Here, the encoder search for the binary splits is used as a heuristic to determine the optimal ternary split.

E. Inverse split order in low-depth configurations

In VTM there are two competing rules, allowing either to infere the decision on the QT-split from the result of the BT-splits or the BT/TT-decision from the QT-split. The former of the two rules is only applied if the maximal used BT-depth is higher or equal to 2 for inter frames, and 3 for intra-frames. The split order is inverted using a heuristic method to allow the application of the latter rule. As the condition for the first is not fullfilled in low-depth configurations, the split order can always be inverted allowing broader application of the BT/TT split decision inference from the results of the QT-split.

IV. DISCUSSION

Relative encoding time and BD-rate results for the additional partitioning search rules used in VVenC are summarized in Table II. Besides the rules described in the previous section, a rule available in VTM, but not enabled in VTM CTC is used in VVenC (a rule based on signaling gradients enabled by the option "ContentBasedFastQtbt", as described in [12]).

Overall, it can be observed that compared to the original VTM partitioning search strategy, the rules used in VVenC introduce more loss, except for rule 3.C which compensates some chroma loss. This offers different working points for a wide range of practical applications of the VVenC software. In the VVenC Pareto optimization, the rules from sections

TABLE III. PER-SEQUENCE RESULTS OF ALL PRESENTED SPEEDUPS AND TWO STATE OF THE ART METHODS.

	VVenC with all speedups (vs VTM Alg.)					Ref [14]		Ref [15]	
	for BTT21 ("medium")			for BTT33		for BTT33		for BTT33	
Sequence	**EncT**	**max EncT**	**BD_{YUV}**	**EncT**	**BD_{YUV}**	**EncT**	**BD-Rate**	**EncT**	**BD-Rate**
UHD (Classes A1 and A2)									
Tango2	63%	67%	2.15%	53%	2.10%	-	-	-	-
FoodMarket4	65%	66%	1.14%	54%	1.21%	-	-	-	-
Campfire	66%	70%	0.90%	54%	1.16%	-	-	*65%*	*0.36%*
CatRobot1	60%	63%	2.69%	56%	1.85%	-	-	-	-
DaylightRoad2	61%	65%	3.19%	55%	2.43%	-	-	-	-
ParkRunnig3	62%	65%	1.25%	56%	0.85%	-	-	*79%*	*0.49%*
HD (Class B)									
MarketPlace	62%	64%	1.54%	57%	1.20%	-	-	-	-
RitualDance	63%	65%	1.49%	55%	1.44%	-	-	*68%*	*0.90%*
Cactus	61%	65%	2.02%	55%	1.61%	67%	1.51%	*66%*	*1.18%*
BasketballDrive	62%	66%	1.83%	54%	1.43%	58%	2.69%	-	-
BQTerrace	60%	65%	1.60%	59%	1.13%	71%	1.19%	-	-
Average	**62%**	**70%**	**1.80%**	**58%**	**1.32%**	-	-	-	-

3.A., 3.B, 3.D., and 3.E. are bundled as a single option for the Pareto-Set estimation, but the gradient rule is tested separately. The latter is deactivated for the "slow" preset (using the BTT32 partitioning configuration, see Table. I), as the loss is considered too high for the given speedup at this point in the Pareto set. The remaining rules are only found to provide sub-optimal results for the VTM-like configuration with maximal depth of a BTT-tree set to 3 (i.e. "slower" preset partitioning configuration). Rule 3.C. is always applied.

In Table III, per sequence results for VVenC with all presented speedups for the setting with max. BTT depth Intra/Inter equal to 2/1 (BTT21 or "medium" preset) and for the setting with max. BT depth Intra/Inter equal to 3/3 (BTT33) are reported. Most notably, it can be observed, that the achieved speedup is relatively constant over different QPs (i.e. the maximum runtime, "max EncT", only differs slightly from the average runtime "EncT"). The variation of BD-rate losses introduced by the speedups over the sequences is also low, ranging from 0.90% to 2.69%.

Additionally, Table III shows a comparison with the state of the art methods [14] and [15]. Our proposed method outperforms [14] for the common HD sequences by providing a higher speedup with less BD-rate increase. The results presented in [15] for sequences common between the works are comparable, while the presented method generally shows more speedup and more loss for the BTT33 configuration. It should be noted that considering Pareto optimality not all speedups would be enabled for this configuration, as can be seen in the Pareto-Set visualization in Fig. 2. The results demonstrate that fine-tuned speedup heuristics can still compete with complex state of the art AI based approaches.

In Fig. 2, the Pareto set of partitioning options is shown for different search algorithms (relative to VVenC "medium"

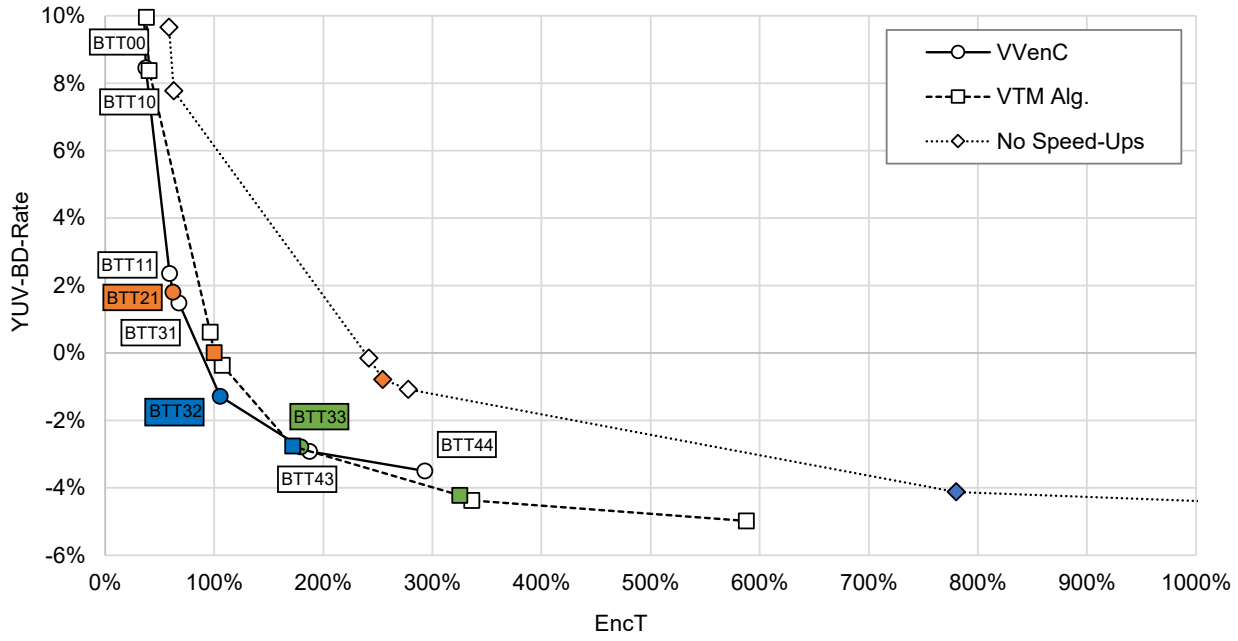

Fig. 2. The Pareto-Set of partitioning configurations as in Tables I and II for different search algorithms. The described fast options extend the Pareto-Set around the configuration used in VVenC "medium", while providing a worse tradeoff at high-depth partitioning configurations.

preset with VTM algorithm). It can be observed that the implemented speedup methods extend the Pareto set of partitioning options around the working point of the "medium" preset (100% runtime at 0% gain). This is consistent with the behavior of the overall configuration space Pareto-Set as observed in [4]. Additionally, for each partitioning search algorithm, the "medium" partitioning configuration (BTT21) is marked orange, as well as the BTT32 (as used in VVenC "slow") and BTT33 (as used in VTM and VVenC "slower") partitioning configurations – blue and green, respectively.

Fig. 2 demonstrates how the performance of the fast partitioning rules available in VTM degrades with reduced partitioning complexity, as previously discussed in Table I. The presented methods provide a consistent speedup across the configurations, as shown in Table II. The speedups of rule 3.A. and 3.B. are reduced with increasing available recursion depth (with more depth, the rules will compete with other rules from VTM). This behavior is compensated by rule 3.D., which increases the speedup with increasing depth, because the skipped search will include more and more subsequent decisions. It is shown how the additional rules extend the Pareto-Set around the working point of the "medium" presets and its partitioning configuration, but fall short at different working points – i.e. the last presented point on the "VVenC" curve (BTT44) does not provide an improvement (in the Pareto-optimal sense [22]) over the working points available within the VTM-Algorithm by varying high-level configuration parameters. The BTT33 working point with the proposed algorithm provides a very similar tradeoff to the BTT32 working point with VTM algorithm. For higher available recursion depths, the VTM algorithm is more optimal.

The analysis in [4] shows that the proposed methods also perform very well in context of the overall Pareto-Set, beyound the restriction to only partitioning options as presented for the purpose of this paper.

In future works it will be of interest to consider each of the speedup methods separately for the Pareto-Set optimiztion, both for the proposed methods as well as the methods already available in VTM and showing up in the state of the art.

V. Conclusion

In this paper we discussed the fast partitioning methods as incorporated into VVenC, an open and optimized VVC encoder implementation. The proposed methods show improved coding efficiency (BD-rate) vs. runtime tradeoffs and extend the Pareto-Set around the "medium" preset of the VVenC encoder when compared to the speedups implemented in the VVC reference software (VTM). Some of the methods also provide attractive tradeoffs for more complex configurations, e.g. as used in JVET CTC. The methods measure up well to more complex, e.g. machine learning based, approaches providing 42% speedup at around 1.3% BD-rate loss for the VTM CTC partitioning configuration.

With the algorithmic improvements of the presented speedups, a Pareto-Set extension of the VVenC "medium" low-depth working point can be achieved. Here, the proposed methods achieve 38% speedup at around 1.8% BD-rate loss.

References

[1] ITU-T and ISO/IEC JTC 1, "Versatile video coding", Rec. ITU-T H.266 and ISO/IEC 23090-3 (VVC), August 2020.

[2] M. Wien, V. Baroncini, "Report on VVC compression performance verification testing in the SDR UHD Random Access Category," document JVET-T0097, Joint Video Experts Team (JVET), Okt. 2020.

[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649–1668, 2012.

[4] J. Brandenburg et al., "Towards Fast and Efficient VVC Encoding," 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP), Tampere, Finland, 2020, pp. 1-6.

[5] VVenC software repository, vesion VVenC-0.3.1.0. Available online: https://github.com/fraunhoferhhi/vvenc.

[6] A. Wieckowski et al., "Towards A Live Software Decoder Implementation For The Upcoming Versatile Video Coding (VVC) Codec," 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 2020, pp. 3124-3128.

[7] F. Bossen, X. Li, K. Sühring, K. Sharman, V. Seregin, A. Tourapis, "JVET AHG report: Test model software development (AHG3)," document JVET-U0003, Joint Video Experts Team (JVET), Jan. 2021.

[8] K. Suehring et al., "JCT-VC AHG report: Software development and software technical evaluation (AHG3)," document JCTVC-AN0003, Joint Collaborative Team on Video Coding (JCT-VC), Jun. 2020.

[9] X. Li, H.-C. Chuang, J. Chen, M. Karczewicz, L. Zhang, X. Zhao, and A. Said, "Mult-Type-Tree," document D0117, Joint Video Exploration Team (JVET), Oct. 2016.

[10] G. J. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," IEEE Signal Processing Magazine, vol. 15, no. 6, pp. 74–90, 1998.

[11] D. Marpe et al., "Video Compression Using Nested Quadtree Structures, Leaf Merging, and Improved Techniques for Motion Representation and Entropy Coding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 12, pp. 1676–1687, 2010.

[12] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe and T. Wiegand, "Fast Partitioning Decision Strategies for The Upcoming Versatile Video Coding (VVC) Standard," *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, pp. 4130-4134.

[13] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," Technical Report VCEG-M33, ITU-T SG16/Q6, Austin, Texas, USA, 2001.

[14] N. Tang *et al*., "Fast CTU Partition Decision Algorithm for VVC Intra and Inter Coding," *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Bangkok, Thailand, 2019, pp. 361-364.

[15] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard and C. Bergeron, "Tunable VVC Frame Partitioning Based on Lightweight Machine Learning," in *IEEE Transactions on Image Processing*, vol. 29, pp. 1313-1328, 2020.

[16] H. Yang, L. Shen, X. Dong, Q. Ding, P. An and G. Jiang, "Low Complexity CTU Partition Structure Decision and Fast Intra Mode Decision for Versatile Video Coding," in *IEEE Transactions on Circuits and Systems for Video Technology*.

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

[18] K. Choi, "Report of BoG on QTBT configuration setting," document JVET-C0101, Joint Video Exploration Team (JVET), May 2016.

[19] J. Brandenburg, A. Wieckowski, T. Hinz, B. Bross, "VVenC Fraunhofer Versatile Video Encoder", Fraunhofer HHI, Version v0.3.1-v1, 2021 [Online]. Available: https://www.hhi.fraunhofer.de/en/departments/vca/technologies-and-solutions/h266-vvc/fraunhofer-versatile-video-encoder-vvenc.html.

[20] A. Wieckowski et al., "Update on open, optimized VVC implementations VVenC and VVdeC", document JVET-U0135, Joint Video Experts Team (JVET), Jan. 2021.

[21] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, and T. Wiegand, "Recursive partitioning search space pruning using split cost prediction," in 2019 Data Compression Conference (DCC), Snowbird, UT, 2019.

[22] P. Czyżżak and A. Jaszkiewicz, "Pareto simulated annealing—a metaheuristic technique for multiple - objective combinatorial optimization," Journal of Multi-Criteria Decision Analysis, vol. 7, pp. 34 - 47, 1998.