SPATIO-TEMPORAL PARALLELIZATION SCHEME FOR HEVC ENCODING ON MULTI-COMPUTER SYSTEMS

Alexandre Mercat, Sami Ahovainio, and Jarno Vanne

Ultra Video Group, Tampere University, Finland

ABSTRACT

High Efficiency Video Coding (HEVC) sets the scene for economic video transmission and storage, but its inherent computational complexity calls for efficient parallelization techniques. This paper introduces and compares three different parallelization strategies for HEVC encoding on multi-computer systems: 1) spatial parallelization scheme, where input video frames are divided into slices and distributed among available computers; 2) temporal parallelization scheme, where input video is distributed among computers in groups of consecutive frames; 3) spatio-temporal parallelization scheme that combines the proposed spatial and temporal approaches. All these three schemes were benchmarked as part of the practical Kvazaar open-source HEVC encoder. Our experimental results on 2–5 computer configurations show that using the spatial scheme gives 1.65×–2.90× speedup at the cost of 4.16%–13.09% bitrate loss over a single-computer setup. The respective speedup with temporal parallelization is 1.86×–3.26× without any coding overhead. The spatio-temporal scheme with 2 slices was shown to offer the best load-balancing with 1.81×–3.55× speedups and a constant coding loss of 4.16%.

***Index Terms*—**Video coding, High Efficiency Video Coding (HEVC), multi-computer systems, distributed HEVC encoding, HEVC parallelization strategies

1. INTRODUCTION

Digital video has become ubiquitous in our everyday life thanks to a myriad of media applications that strive for in-depth immersion through increasingly bandwidth-greedy video formats. Over the last three decades, ISO/IEC MPEG and ITU-T VCEG have addressed the snowballing growth of video transmission and storage requirements by releasing a series of international video coding standards. Currently, the landscape of these standards is dominated by the universal *Advanced Video Coding* (*AVC/H.264*) [1], well-established *High Efficiency Video Coding* (*HEVC*/*H.265*) [2], and emerging *Versatile Video Coding* (*VVC*/H.266) [3].

This work focuses on HEVC that is one of the most widespread video formats at the moment [4]. HEVC Main profile is shown to improve coding efficiency upon AVC by around 23% in *All-Intra* (*AI*) configuration and 40% in *Random Access* (*RA*) configuration. However, the reported coding gain comes at a cost of over 3.2× and 1.2× computational complexity [5], respectively. Complexity tends to act as a barrier for practical HEVC applications and tackling it calls for efficient techniques for HEVC codec parallelization.

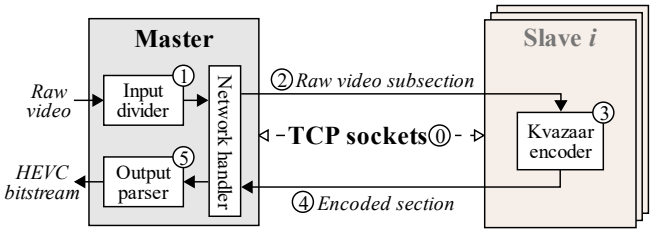


Fig. 1. Proposed framework for HEVC encoding parallelization.

HEVC encoding can be parallelized at three different levels: process, thread, and data levels. Table 1 characterizes the existing parallelization approaches for HEVC and AVC. In [6]–[8], process level parallelization was implemented by distributing the video temporally to different encoder instances in *group of pictures* (*GOPs*). Thread level parallelization was achieved with tiles and slices in [9], [10]. In [11], the encoding process was parallelized in all three parallelization levels simultaneously. A script enabling multi-computer parallelization for H.264 can be found in [12].

The first version of our encoding parallelization framework was introduced in [13], [14]. It supported process level parallelization. In addition, the encoding process was parallelized in thread- and data levels by the means of our Kvazaar open-source HEVC encoder [15], [16]. However, this proof-of-concept implementation was only able to support spatial parallelization in AI coding.

This work introduces a fully-fledged framework for parallel HEVC video encoding on multiple computers. The

Table 1. Comparison between the proposed and related works

Ref	Codec	Prediction mode	GOP	Tiles	Slices	WPP/ OWF	SIMD	Encoder
[6]	HEVC	Intra	×					HM 10.0
[7]	HEVC	Intra/Inter	×					HM
[8]	HEVC	Intra/Inter	×					HM 10.0
[9]	HEVC	Intra/Inter		×	×			HM 16.3
[10]	HEVC	Intra	×[1]	×	×			HM 16.3
[11]	HEVC	Intra/Inter	×		×		×	Private
[12]	AVC	Intra/Inter	×[2]				×	libx264
[13], [14]	HEVC	Intra		×	×	×	×	Kvazaar
Proposed	**HEVC**	**Intra/Inter**	×	×	×	×	×	**Kvazaar**

×[1]: Frame by frame for intra coding. ×[2]: Chunks.

This work was supported in part by the AI for situational Awareness (AISA) project led by Nokia and funded by Business Finland.

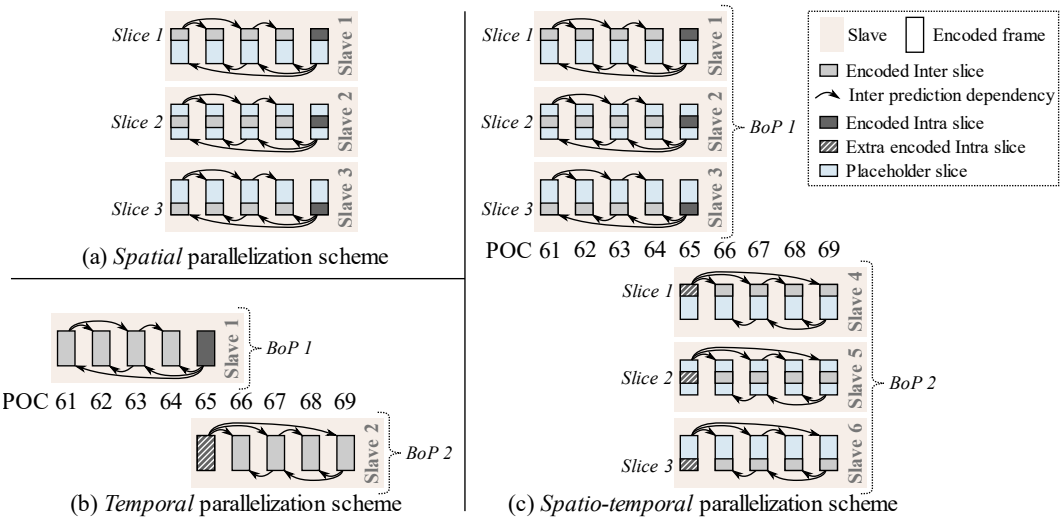


Fig. 2. Illustration of the proposed *spatial*, *temporal*, and *spatio-temporal* parallelization schemes.

designed framework is shown in Fig 1. It offers three parallelization schemes: *spatial*, *temporal*, and *spatio-temporal* parallelization. *Spatial* parallelization scheme divides input video into slices, which are allocated to the encoding slaves on the available computers. *Temporal* parallelization scheme processes input video in blocks of consecutive frames. *Spatio-temporal* parallelization scheme utilizes *spatial* and *temporal* parallelization simultaneously. It offers the best load-balancing of the three schemes.

The rest of the paper is outlined as follows. Section 2 gives an overview of the designed encoding parallelization framework. Our experimental setup and results are described in Section 3. Finally, Section 4 concludes the paper.

2. PROPOSED PARALLELIZATION SCHEMES

The proposed framework is principally designed for Kvazaar HEVC encoder [15], [16]. Kvazaar has been modified to write headers that include slice and *picture order count* (*POC*) offsets. Slice offset is needed for the *spatial* parallelization scheme and POC offset for the *temporal* one.

Fig. 1 presents the overview of our encoding framework that is divided into the following six processing stages:

0) During initialization, the connection between the master and encoding slaves is established.
1) The master reads the input video and divides it into sections. The number of sections equals that of slaves.
2) Raw video sections are sent to the slaves over TCP sockets.
3) Each slave encodes the assigned section of the video.
4) Each slave sends the encoded section back to the master.
5) The master keeps track of which section of the video is assigned to which slave and is thereby able to parse together the final HEVC bitstream.

The proposed framework supports three parallelization schemes: 1) *spatial*; 2) *temporal*; and 3) *spatio-temporal*. They are illustrated in Fig. 2 and further addressed in the following sections.

2.1. *Spatial* Parallelization Scheme

Spatial parallelization scheme utilizes slices to distribute partial frames to different encoder instances. Fig. 2(a) exemplifies how the raw video is divided into three slices for three encoding slaves.

Slices are used instead of tiles in order to provide support for the *wavefront parallel processing* (*WPP*) [17] and non-normative *overlapping wavefront* (*OWF*) mechanism [18], which offer massive parallelization in thread level. Kvazaar writes headers for the parallelized slices, which specify the offset of each slice in the frame. This optimizes the parsing of the final output.

2.2. *Temporal* Parallelization Scheme

Temporal parallelization scheme divides the raw input video into group of consecutive frames, called *Blocks of Pictures* (*BoP*), as illustrated in Fig. 2(b). A BoP can contain several GOPs and is surrounded by two intra frames.

Each BoP can be distributed to independent encoders. However, an extra intra frame is needed to be able to access all required reference frames in each BoP. This intra frame is a duplicate frame since successive BoPs, i.e., BoP_n and BoP_{n+1}, require it as reference. In practice, it is the last frame of BoP_n and the first frame of BoP_{n+1}, but it is dropped out from BoP_{n+1} for the final output. By choosing the first frame in BoP to be dropped and by introducing a POC offset into the header of each BoP, a valid output can be parsed effortlessly. The length of a BoP is chosen to match the selected intra period to avoid coding overhead.

2.3. *Spatio-Temporal* Parallelization Scheme

Spatio-temporal parallelization scheme combines *Spatial* and *temporal* ones. It virtually supports any number of slices and any BoPs lengths.

Table 2. Test sequences

Set	Resolution	Name(s)	Frames	Frame rate
CTC-A	2560×1600	*PeopleOnStreet, Traffic*	150	30
CTC-B	1920×1080	*BasketballDrive, Cactus*	500	50
		BQTerrace	600	60
		Kimono, ParkScene	240	24
UVG-4K	3840×2160	*SunBath*	300	50
		*ShakeNDry**	300	120
		CityAlley, FlowerFocus, FlowerKids, FlowerPan, RaceNight, RiverBank, Twilight	600	50
		Lips, Beauty, Bosphorus*, HoneyBee*, Jockey*, ReadySetGo*, YachtRide**	600	120
Long	4000×2250	*Big Buck Bunny 4K*	38074	60
	3840×1714	*Tears of Steel*	17616	24

* Also down-sampled to1920×1080 to form the ***UVG-FullHD*** set.

Fig. 3(c) exemplifies *spatio-temporal* parallelization scheme, in which the encoding process has been parallelized across six slaves using three slices and BoPs of 64 frames. Finer task granularity improves both scalability and load balancing. Compared with *temporal* scheme, these improvements come at the cost of coding loss introduced by slices. Instead, the coding efficiency equals that of *spatial* scheme with the same number of slices.

3. PERFORMANCE EVALUATION

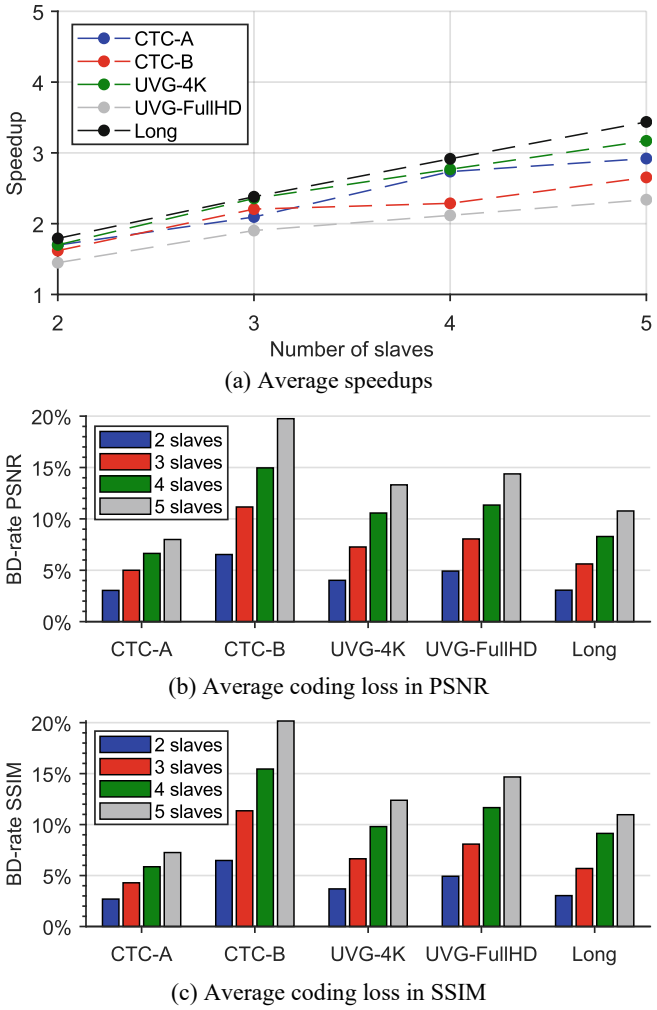
All our experiments were performed with Kvazaar and benchmarked with our open-source test automation framework called *uvgVenctester* [19] that is designed for performance and conformance testing of video encoders.

Table 2 details our test set that includes classes A and B from CTC dataset [20], all sequences from UVG dataset [21], and two longer sequences: *Big Buck Bunny 4K* [22], and *Tears of Steel* [23]. All schemes were tested with a set of five similar computers equipped with Intel i7 @ 3.40 GHz 4-core processor, 32 GB of RAM, and 10GbE links.

During the experiments, the number of computers was varied from one to five. One of the computers served as both a master and a slave since the master process is not computationally heavy.

The experiments were carried out using the RA configuration [20] with intra period of 64. The coding efficiency was benchmarked in terms of *Bjøntegaard Delta Bit Rate* (*BD-rate*) difference [24], [25] in percent for the same *Peak Signal-to-Noise Ratio* (*PSNR*) or *Structural SIMilarity* (*SSIM*) [26] using *Quantization Parameter* (*QP*) values of 22, 27, 32, and 37. The coding speeds of the framework were measured separately for each four QPs and the results were averaged. Each test run was repeated three times for the *Long* set and five times for the rest. The results of individual test runs were averaged for improved accuracy.

Kvazaar with *veryslow* preset was ran on the master computer and used as an anchor. All Kvazaar instances, i.e., the anchor and slaves, were configured to maximize the use of data-level and thread-level parallelization through SIMD optimizations and WPP+OWF parallelization strategies, respectively.



(a) Average speedups

(b) Average coding loss in PSNR

(c) Average coding loss in SSIM

Fig. 3. Experimental results of *spatial* parallelization scheme.

3.1. Experimental Results of *Spatial* Scheme

Fig. 3(a) shows the achieved speedups of *spatial* scheme. The results are reported for each slave count and sequence set used. The highest and the lowest speedups are obtained with *Long* and *UVG-FullHD* sets, respectively. The difference between sequence sets and visible fluctuations are caused by the uneven load-balancing, which is highly afflicted by the resolution and content of the sequence.

Fig. 3(b) and Fig. 3(c) plot the coding losses in BD-rate PSNR and SSIM for each sequence set and slave count, respectively. The more encoding slaves are used, the worse the BD-rate performance is. This is due to reference restrictions caused by the slice edges. Low-resolution sequences are the most vulnerable to these restrictions, so the average coding loss is the highest with them.

3.2. Experimental Results of *Temporal* Scheme

Temporal scheme exploits intra frames that naturally break the dependencies between frames. As the intra period

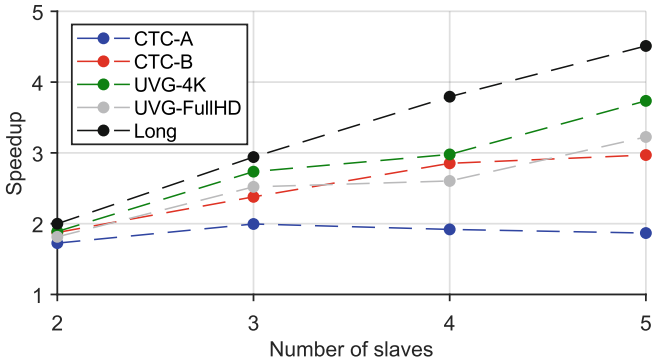


Fig. 4. Average speedup with the proposed *temporal* scheme.

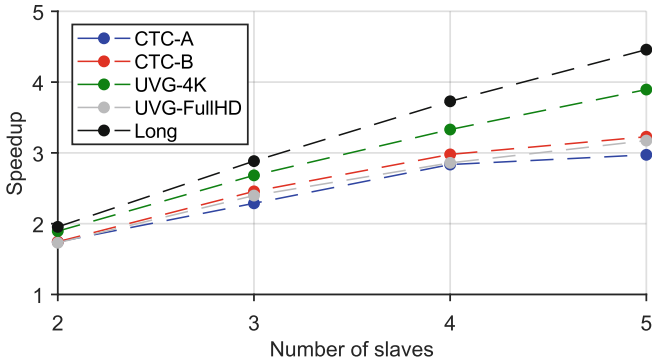


Fig. 5. Average speedup with the proposed *spatio-temporal* scheme.

matches the one used in the anchor configuration, *temporal* scheme introduces no BD-rate losses.

Fig. 4 illustrates the achieved speedups of *temporal* parallelization scheme for the sequence sets as a function of the number of slaves. The highest speedup is achieved with *Long* set, whereas the lowest speedup comes with *CTC-A* set. The speedup gap mainly stems from the BoP size of 64. Because the sequences in the *CTC-A* set only have 150 frames, this parallelization scheme cannot parallelize the encoding process across more than 3 slaves for those sequences. Conversely, the *Long*-set sequences are so long that this effect becomes negligible and that is why this set achieves the highest speedup.

3.3. Experimental Results of *Spatio-Temporal* Scheme

In the tested configuration, *spatio-temporal* scheme exploits *spatial* parallelization by dividing the video into 2 slices and *temporal* parallelization by distributing BoPs of size 64. Fig. 5 plots the achieved speedups for each sequence set according to the number of slaves. Contrary to *temporal* scheme, all test sets achieve similar speedups with the *spatio-temporal* scheme. The utilization of *spatial* parallelization on top of *temporal* one improves load-balancing of the solution, e.g., it allows to exploit more than three encoding slaves and hence improve the speedup for the *CTC-A* set.

Table 3 features the average coding losses of *spatio-temporal* scheme. Since it uses two slices regardless of the number of slaves, the BD-rate losses stay constant throughout the number of slaves. Specifically, the losses are the same as those of *spatial* scheme with two slaves, as shown in blue on Fig. 3(a) and Fig. 3(b).

Table 3. Coding loss with the proposed *spatio-temporal* scheme

	CTC-A	***CTC-B***	***UVG-4K***	***UVG-FullHD***	***Long***
BD-rate PSNR	3.04%	6.53%	4.02%	4.92%	3.03%
BD-rate SSIM	2.69%	6.48%	3.69%	4.93%	3.06%

4. CONCLUSION

This work proposed a new framework for HEVC encoding parallelization. It supports three different parallelization schemes: 1) *spatial*; 2) *temporal*; and 3) *spatio-temporal*. Their characteristics and suggested use cases are summarized in Table 4.

The selection of the parallelization scheme depends on the use case. For example, parallelizing *temporal* scheme across five slaves achieves massive 4.51× speedup without any coding loss with long test sequences, but it struggles with shorter sequences. Instead, *spatial* and *spatio-temporal* parallelization schemes are faster with shorter sequences but they introduce coding losses.

In the future, the proposed framework can also be upgraded to VVC encoding with minimal changes because VVC supports the same parallelization strategies as HEVC. Our framework could also be further developed for better scalability and load balancing.

Table 4. Comparative overview of the proposed parallelization schemes

Parallelization scheme	Number of slaves	Speedup	BD-rate PSNR	BD-rate SSIM	Strengths	Weaknesses	Potential use cases
Spatial	**2**	1.65×	4.16%	4.32%	• No additional throughput latency. • Not dependent on video length.	• BD-rate losses. • Dependent on video resolution.	• Online video encoding. • 360° video encoding.
	3	2.19×	7.21%	7.42%			
	4	2.56×	10.38%	10.36%			
	5	2.90×	13.09%	13.24%			
Temporal	**2**	1.86×	0.00%	0.00%	• No BD-rate losses. • Not dependent on video resolution.	• Additional throughput latency of one block of pictures. • Dependent on video length.	• Offline video coding for long sequences.
	3	2.51×	0.00%	0.00%			
	4	2.83×	0.00%	0.00%			
	5	3.26×	0.00%	0.00%			
Spatio-temporal	**2**	1.81×	4.16%	4.32%	• Good scalability. • Good load-balancing.	• Additional throughput latency of one block of pictures. • Minor BD-rate losses. • Mildly dependent on video resolution and/or video length.	• Cloud encoding service with on-demand resource allocation.
	3	2.54×	4.16%	4.32%			
	4	3.15×	4.16%	4.32%			
	5	3.55×	4.16%	4.32%			

6. REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 13, no. 7, Aug. 2003, pp. 560–576.

[2] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 22, no. 12, Dec. 2012, pp. 1649–1668.

[3] ITU, "New 'Versatile Video Coding' standard to enable next-generation video compression," Sept. 2020, [Online]. Available: https://www.itu.int/en/mediacentre/Pages/pr13-2020-New-Versatile-Video-coding-standard-video-compression.aspx.

[4] Bitmovin, "Bitmovin video developer report 2019," 2019, [Online]. Available: https://cdn2.hubspot.net/hubfs/3411032/Bitmovin%20Magazine/Video%20Developer%20Report%202019/bitmovin-video-developer-report-2019.pdf.

[5] J. Vanne, M. Viitanen, T. D. Hämäläinen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 22, no. 12, Dec. 2012, pp. 1885–1898.

[6] P. Piñol, H. Migallón, O. López-Granado, and M. P. Malumbres, "Parallel strategies analysis over the HEVC encoder," *J. Supercomput.,* vol. 70, no. 2, Mar. 2014, pp. 671–683.

[7] H. Migallón et al., "Synchronous and asynchronous HEVC parallel encoder versions based on a GOP approach," *Advances Eng. Soft.,* vol. 101, Nov. 2016, pp. 37–49.

[8] H. Migallón, V. Galiano, P. Piñol, O. López-Granado, and M. P. Malumbres, "Distributed memory parallel approaches for HEVC encoder," *J. Supercomput.,* vol. 73, no. 1, Feb. 2017, pp. 164–175.

[9] H. Migallón, P. Piñol, O. López-Granado, V. Galiano, and M. P. Malumbres, "Performance analysis of frame partitioning in parallel HEVC encoders," *J. Supercomput.,* vol. 73, no. 1, Nov. 2016, pp. 543–556.

[10] H. Migallón, P. Piñol, O. López-Granado, V. Galiano, and M. P. Malumbres, "Frame-based and subpicture-based parallelization approaches of the HEVC video encoder," *Appl. Sci.,* vol. 8, no. 6, May 2018, pp. 854.

[11] T. K. Heng et al., "A highly parallelized H. 265/HEVC real-time UHD software encoder," *in Proc. IEEE Int. Conf. Image Process.*, Paris, France, Oct. 2014, pp. 1213–1217.

[12] "dve," [Online]. Available: https://github.com/patademahesh/dve.

[13] S. Ahovainio, A. Mercat, M. Viitanen, and J. Vanne, "Multi-level parallelization scheme for distributed HEVC encoding on multi-computer systems," *in Proc. IEEE Int. Symp. Circuits Syst.*, Seville, Spain, Sept. 2020, pp. 1–5.

[14] S. Ahovainio, A. Mercat, and J. Vanne, "Live demonstration: multi-laptop HEVC encoding," *in Proc. IEEE Int. Symp. Circuits Syst.*, Seville, Spain, Sept. 2020, pp. 1–1.

[15] A. Lemmetti, M. Viitanen, A. Mercat, and J. Vanne, "Kvazaar 2.0: fast and efficient open-source HEVC inter encoder," *in Proc. ACM Multimedia Syst. Conf.*, Istanbul, Turkey, May 2020, pp. 237–242.

[16] Ultra Video Group, "Kvazaar open-source HEVC encoder," [Online]. Available: https://github.com/ultravideo/kvazaar.

[17] G. Clare, F. Henry, and S. Pateux, "Wavefront parallel processing for HEVC encoding and decoding," *document JCTVC-F274*, Torino, Italy, Jul. 2011.

[18] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, V. George, and T. Schierl, "Improving the parallelization efficiency of HEVC decoding," *in Proc. IEEE Int. Conf. Image Process.*, Orlando, Florida, USA, Sept. 2012, pp. 213–216.

[19] J. Sainio, A. Mercat, and J. Vanne, "uvgVenctester: open-source test automation framework for comprehensive video encoder benchmarking," *in Proc. ACM Multimedia Syst. Conf.*, Istanbul, Turkey, Jun. 2021, pp. 255–260.

[20] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, "JVET common test conditions and software reference configurations for SDR video," *document JVET-N1010*, Geneva, Switzerland, Mar. 2019.

[21] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120fps 4K sequences for video codec analysis and development," *in Proc. ACM Multimedia Syst. Conf.*, Istanbul, Turkey, May 2020, pp. 297–302.

[22] Blender Foundation, "Big Buck Bunny," [Online]. Available: https://peach.blender.org.

[23] Blender Foundation, "Tears of Steel, Mango Open Movie Project," [Online]. Available: http://tearsofsteel.org.

[24] G. Bjøntegaard, "Improvements of the BD-PSNR model," *document VCEG-AI11*, Berlin, Germany, Jul. 2008.

[25] ITU-T and ISO/IEC JTC 1, "Working practices using objective metrics for evaluation of video coding efficiency experiments," *document ITU-T HSTP-VID-WPOM and ISO/IEC DTR 23002-8*, Jul. 2020.

[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.,* vol. 13, no. 4, Apr. 2004, pp. 600–612.