Open-Source Toolkit for Live End-to-End 4K VVC Intra Coding

Marko Viitanen, Joose Sainio, Alexandre Mercat, Guillaume Gautier, and Jarno Vanne
Ultra Video Group, Hervanta Campus, Tampere University, Finland
{marko.viitanen, joose.sainio, alexandre.mercat, guillaume.gautier, jarno.vanne}@tuni.fi

Ibrahim Farhat, Pierre-Loup Cabarat, Wassim Hamidouche, and Daniel Menard
University of Rennes, INSA Rennes, CNRS, IETR–UMR 6164, Rennes, France
{ibrahim.farhat, pierre-loup.cabarat, wassim.hamidouche, daniel.menard}@insa-rennes.fr

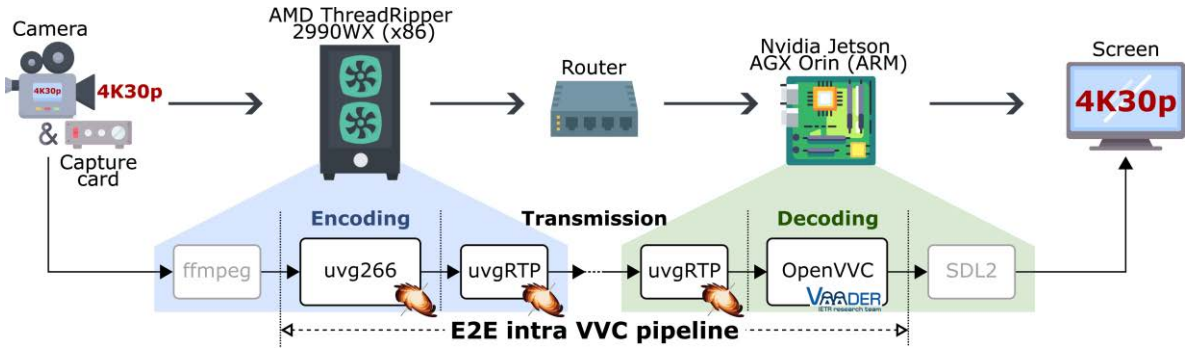

Figure 1: Overview of the proposed end-to-end (E2E) pipeline for live VVC intra coding and streaming.

ABSTRACT

Versatile Video Coding (VVC/H.266) takes video coding to the next level by doubling the coding efficiency over its predecessors for the same subjective quality, but at the cost of immense coding complexity. Therefore, VVC calls for aggressively optimized codecs to make it feasible for live streaming media applications. This paper introduces the first public end-to-end (E2E) pipeline for live 4K30p VVC intra coding and streaming. The pipeline is made up of three open-source components: 1) uvg266 for VVC encoding; 2) uvgRTP for VVC streaming; and 3) OpenVVC for VVC decoding. The proposed setup is demonstrated with a proof-of-concept prototype that implements the encoder end on AMD ThreadRipper 2990WX and the decoder end on Nvidia Jetson AGX Orin. Our prototype is almost 34 000 times as fast as the corresponding E2E pipeline built around the VTM codec. Respectively, it achieves 3.3 times speedup without any significant coding overhead over the pipeline that utilizes the fastest possible configuration of the well-known VVenC/VVdeC codec. These results indicate that our prototype is currently the only viable open-source solution for live 4K VVC intra coding and streaming.

CCS CONCEPTS

• Software and its engineering • Computing methodologies ~ Computer graphics ~ Image compression

KEYWORDS

Versatile Video Coding (VVC), VVC intra coding, end-to-end (E2E) streaming, real-time coding speed, open-source implementation

ACM Reference format:
Marko Viitanen, Joose Sainio, Alexandre Mercat, Guillaume Gautier, Jarno Vanne, Ibrahim Farhat, Pierre-Loup Cabarat, Wassim Hamidouche, and Daniel Menard. 2023. Open-Source Toolkit for Live End-to-End 4K VVC Intra Coding. In *14th ACM Multimedia Systems Conference (MMSys '23), June 7–10, 2023, Vancouver, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3587819.3593938

MMSys '23, June 7–10, 2023, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0148-1/23/06.
https://doi.org/10.1145/3587819.3593938

1 INTRODUCTION

The proliferation of new media services and products has resulted in an exponential growth of digital video. Currently, video is estimated to account for 82% of all global IP traffic [1], and the existing growth rate shows no signs of slowing down. This trend has created an urgent need for efficient and high-speed video coding and streaming technologies that are able to

encode, transmit, store, and decode video streams in the given time, quality, bandwidth, and cost constraints.

To that end, ISO/IEC MPEG and ITU-T VCEG have released a series of international video coding standards over the last three decades. The latest standard, *Versatile Video Coding* (*VVC/H.266*), was ratified in 2020 [2] as the successor to the famous *High Efficiency Video Coding* (*HEVC/H.265*) [3]. VVC aims for 50% higher coding efficiency for the same subjective visual quality, but the coding gain does not come for free as VVC is found to be from 7.4× up to 34.0× as complex as HEVC [4]. By now, HEVC has been adopted by more than 2 billion devices and around 50% of broadcast and video streaming professionals are already using it [5]. Hence, it is likely that VVC will also gain a firm foothold this decade, though efficient codec implementations will be key to its wider adoption.

Currently, there are three well-known open-source VVC encoders: 1) *VVC test model* (*VTM*) [6]; 2) *Fraunhofer Versatile Video Encoder* (*VVenC*) [7]; and 3) uvg266 [8] developed by our *Ultra Video Group* (*UVG*) [9] at Tampere University. VTM is the VVC reference software that virtually implements all normative VVC coding tools, but it is not designed for practical encoding. VVenC is an optimized implementation of VTM, whereas uvg266 is developed from the comprehensively optimized Kvazaar HEVC encoder [8]. Both of them are being developed for real-time coding.

On the decoder side, there are three noteworthy open-source solutions at the moment: 1) VTM [6], 2) *Fraunhofer Versatile Video Decoder* (*VVdeC*) [10], and 3) OpenVVC [11] developed by *Institut d'Électronique et des Technologies du numéRique* (*IETR*). VVdeC is a practical VVC decoder developed from VTM. Whereas the practical OpenVVC decoder pursues a low memory footprint on a wide range of systems. Both VVdeC and OpenVVC can achieve real-time decoding speed, and their performance comparison can be found in [12].

The first complete open-source *end-to-end* (*E2E*) VVC pipeline was introduced by Wieckowski et al. [13]. Their toolchain was made up of VVenC, VVdeC, and GPAC [14] tools. Even though they did not report any absolute coding speeds, it is evident that their solution did not reach real-time performance. Furthermore, several real-time E2E VVC pipelines have also been unveiled [15]–[19], but they all were using a proprietary ATEME TitanLive encoder.

Figure 1 illustrates our proof-of-concept prototype implementation for *All-Intra* (*AI*) 4K30p VVC video encoding, streaming, and decoding. To the best of our knowledge, our solution is the first open-source E2E pipeline for real-time VVC intra coding and streaming. It is composed of an open-source uvg266 [8], uvgRTP [20], and OpenVVC [11] tools, which are respectively presented in Section II. Section III reports our performance results, compares them with prior art, and describes our live demonstration setup. Finally, Section IV concludes the paper.

Table 1: Implemented VVC intra coding tools in uvg266.

Tool name	Description
Cross component linear model (CCLM)	Attempts to predict chroma from luma with linear model
Matrix weighted Intra Prediction (MIP)	Generates the prediction using reference line averaging, matrix multiplication, and interpolation
Multiple Reference Line prediction (MRL)	Selection between three reference pixels lines
Angular intra modes	Double the HEVC intra angles with 65 angular modes
Joint coding of chroma residual (JCCR)	When the chroma channels are similar, can reduce the redundancy in signals
Low frequency non-separable transform (LFNST)	Additional transform for the lowest frequencies done after everything
Multiple Transform Selection (MTS)	Different DCT or DST can be selected for blocks
Transform skip	Codes residual data without transform done, extended for larger sizes in VVC, up to 32×32 pixels
Adaptive Loop Filter (ALF)	Wiener-based diamond-shape filter for minimizing the mean squared error
Dual tree	Luma and Chroma channels split into separate coding passes

2 PROPOSED E2E VVC CODING PIPELINE

2.1 uvg266 Encoder

uvg266 [8] is an open-source VVC encoder distributed under the permissive 3-clause BSD license. It is developed from the Kvazaar HEVC encoder [21], [22], which is optimized for real-time encoding with various techniques, including *Advanced Vector Extensions 2* (*AVX2*) optimizations for x86 architecture. Correspondingly, uvg266 is being designed for real-time encoding by utilizing parallelization strategies, tool optimization techniques, and strategic tool selection from Kvazaar.

In general, uvg266 inherits all the HEVC main profile tools from Kvazaar, with the necessary modifications [8] to comply with the VVC standard. This includes the AVX2 optimized functions for intra prediction, transforms, quantization, and fast coefficient cost estimation, as well as the whole search procedure, which is optimized for real-time encoding.

uvg266 also implements the VVC coding tools tabulated in Table 1, but most of them are computationally too complex for the real-time encoding scenarios pursued in this work. The main issue with CCLM, MIP, and MRL is that their prediction generation is more complex compared to regular angular intra modes, thus they cannot be used for low complexity encoding. JCCR, LFNST, MTS, and *transform skip* require complete reconstruction of the block, which is too complex for real-time encoding. Finally, ALF is computationally a magnitude too complex for real-time encoding. Among these tools, only *dual tree* is enabled in the real-time configuration. It reduces

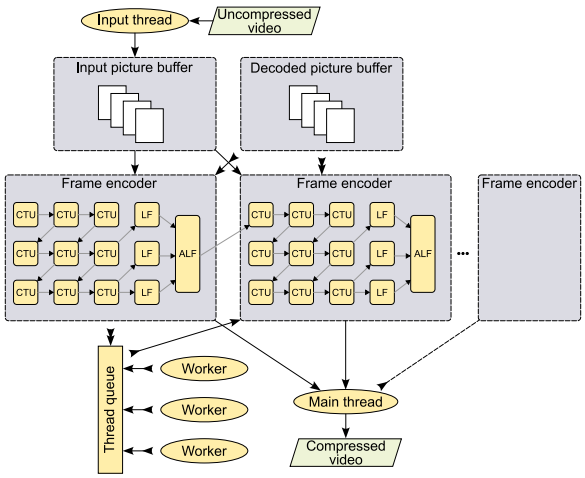

Figure 2: Diagram of the uvg266 parallelization architecture.

computational complexity by separating the luma and chroma searches. This approach allows for early termination if the encoder determines that further search will not improve encoding efficiency, instead of continuing the search for both components even if only one can improve. Moreover, the most complex part during the search is to reconstruct each tested mode. Therefore, this configuration also uses only *Sum of Absolute Transformed Differences* (*SATD*) based rough cost of the prediction, instead of the full reconstruction, for the mode selection. Finally, the coefficient cost calculation is done using the vectorized estimator, instead of calculating the accurate cost with CABAC.

uvg266 implements three different types of parallelism: 1) tiles; 2) *wavefront parallel processing* (*WPP*) [23]; and 3) *overlapping wavefront* (*OWF*) [24]. The frame can be split into separate independent tiles that can be encoded in parallel. WPP allows processing multiple *Coding Tree Unit* (*CTU*) rows in parallel without breaking coding dependencies between them. The non-normative OWF technique extends the execution of WPP across consecutive frames. Additionally, since in AI coding the frames have no dependencies between them, the frames can be encoded in parallel. However, it is important to note that only tiles or WPP can be used at a time.

Overall, the threading system consists of three components: an input, a main thread, and a worker thread pool, as shown in Figure 2. The input thread takes care of input reading. The main thread is responsible for passing the input to the encoder, receiving the encoded frames, providing the UI, and writing the bitstream to a file. The worker thread pool handles all the encoding tasks, i.e., search and bitstream generation. They are separated because the next WPP row requires only the search task to be completed.

2.2 uvgRTP Streamer

Sending bitstreams over a network is possible, but *Real-time Transport Protocol* (*RTP*) is more commonly used as it provides a means for detecting jitter and packet loss as well as handling out-of-order delivery. Among the various RTP libraries available, uvgRTP [20] is the only one that is specifically designed for VVC bitstreams.

uvgRTP is a BSD 2-Clause licensed RTP streaming solution for low-latency, real-time, and high-bandwidth video transmission. It features optimized functions, e.g., for VVC start code lookup and packet dispatching with a round-trip latency of under 5 ms and streaming speeds of up to 4.6 Gbps over a 10Gbps network. It also supports Crypto++ library for encrypted streaming.

2.3 OpenVVC Decoder

OpenVVC [11] is an open-source VVC decoder written in the C programming language. It is compiled as a cross-platform library that is compatible with most-used operating systems. Moreover, it is optimized using *Single Instruction Multiple Data* (*SIMD*): *Streaming SIMD Extensions* (*SSE*)/*AVX* on x86 and NEON architecture extension on ARM. The latest version of OpenVVC is compliant with VVC Main profile. In addition, it is integrated with VLC [25], GPAC [14], and FFplay [26] video players.

OpenVVC provides high-decoding speed by using as low memory as possible [12]. In general, the decoding process starts by parsing the parameters of the sequence. Therefore, reconstruction tasks are performed at the *Coding Unit* (*CU*) level. These tasks include transform, *Luma Mapping Chroma Sample* (*LMCS*), inter prediction, and intra prediction. *Deblocking Filter* (*DF*) is performed immediately after the reconstruction process is completed at CTU level. This approach reduces memory usage by avoiding massive storage of *Quantization Parameter* (*QP*) map and CU blocks that are necessary for DF. Finally, ALF is applied after *Sample Adaptive Offset* (*SAO*) at CTU-line level, and the decoded frame is delivered as output.

OpenVVC supports 1) frame; and 2) tile level parallelization strategies. Using frame parallelism, the main thread parses *Picture Parameter Set* (*PPS*), *Sequence Parameter Set* (*SPS*), and picture/slice header. Then, it schedules the decoding threads that are in charge of decoding the frame. The main thread provides the encoded data to the decoding threads and updates their internal structure. Once their process is completed, the decoding threads return to the thread pool and signal their availability to the main thread. Using tile parallelism, frames are split into separate independent tiles that can be decoded independently. Indeed, prediction dependencies across tile boundaries are broken and entropy decoding state is reinitialized for each tile. However, tiles can have different run time complexities resulting in unbalanced workload between decoded threads of the same frame.

To efficiently use computing resources, OpenVVC applies thread scheduling that overlooks frames and focuses only on tiles. Figure 3 exemplifies the scheduling in the case of a 2×2 grid tile partitioning. Tiles are delimited by the thicker black

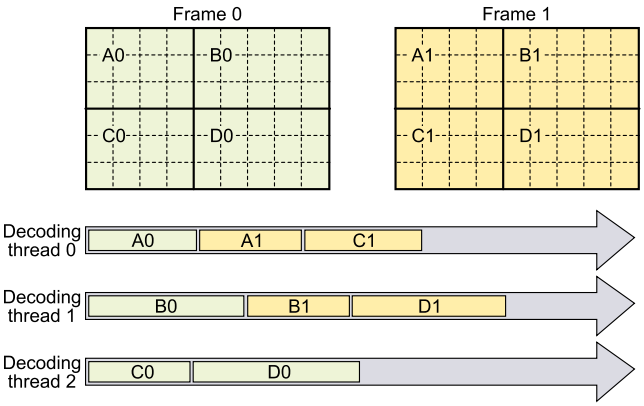

Figure 3: Example of decoding thread scheduling in OpenVVC.

lines and labelled A0, B0, C0 and D0 in frame 0 and A1, B1, C1 and D1 in frame 1. Prediction dependencies across tile boundaries are broken and entropy decoding state is reinitialized for each tile. These restrictions ensure that tiles can be independently decoded, allowing several threads to work simultaneously on different parts of the same picture. This example illustrates that regardless of the tile position, as soon as a decoding thread is available, it starts decoding the next available tile.

3 PERFORMANCE EVALUATION

3.1 Experimental Setup

The proposed uvg266/OpenVVC pipeline was benchmarked against the corresponding pipelines built around the VVenC/VVdeC and VTM codecs. Table 2 tabulates the encoder and decoder versions used in our experiments. The configuration details of each encoder are defined in Table 3, whereas default parameters are used with the decoders.

The experiments were conducted under the AI VVC *common test conditions* (*CTC*) [27] with *QP* values of 22, 27, 32, and 37. 8-bit encoding is used as it less complex and more suited for real-time applications. Moreover, the encoding configurations are tuned for the comparison:

- VTM configuration matches the CTC defined for AI to get the baseline maximum performance,
- VVenC configuration[1] disables all optional tools to achieve fastest possible setup, only 32×32 intra blocks are enabled,
- uvg266 configuration also disables almost all tools except dual-tree, 32×32, and 16×16 blocks since this configuration reaches the real time target of 4k30p.

Table 4 summarizes the applied 4K test video set that includes class A1 and A2 sequences from the VVC CTC [27] and sixteen 4K sequences from the UVG dataset [28]. Two criteria were used to compare the VVC pipelines: coding efficiency and speed. The coding efficiency is reported in *BD-rates* [29], [30] that were measured with three different objective quality metrics: *peak signal-to-noise ratio* (*PSNR*), *Structural SIMilarity* (*SSIM*) [31], and *Video Multimethod Assessment Fusion* (*VMAF*) [32].

The coding speed is reported as the minimum value of processed *frames per second (fps)* of the encoder and decoder averaged across QPs. The evaluation of each VVC pipeline was carried out on two different platforms described in Table 5. Each experiment was run five times and the results were averaged to ensure accuracy. All our experiments were automated using uvgVenctester testing framework [33].

Table 2: Encoder and decoder versions.

Encoder		Decoder	
Name	**Version**	**Name**	**Version**
uvg266	c6764df	OpenVVC	9c46721
VVenC	1.7	VVdeC	1.6.1
VTM	19.0	VTM	19.0

Table 3: Encoding parameters.

Encoder	Options
uvg266	--preset ultrafast -p 1 --hash=none --tiles=16x1 --pu-depth-intra=1-2 --no-deblock --intra-rough-granularity 4 --dual-tree --no-psnr --no-wpp
VVenC	-c vvenc_intra_all_disabled.cfg[1]
VTM	-c encoder_intra_vtm.cfg

Table 4: Test sequences.

Class	Resolution	Name(s)	Frames
A1	3840×2160	Campfire	300
		FoodMarket4	720
	4096×2160	Tango	294
A2	3840×2160	CatRobot, DaylightRoad2	300
		ParkRunning3	300
UVG	3840×2160	CityAlley, FlowerFocus, Lips, FlowerKids, FlowerPan, RaceNight, RiverBank, Twilight, Beauty, Bosphorus, HoneyBee, Jockey, ReadySetGo, YachtRide	600
		ShakeNDry, SunBath	300

Table 5: Profiling platforms.

	Encoding Platform	Decoding Platform
Processor	AMD ThreadRipper 2990WX (32 × 4.2 GHz)	Nvidia Jetson AGX Orin (12 × 2.2 GHz)
Architecture	x86 /AVX2	ARM / Neon
Memory	64 GB	64 GB
Compiler	GCC 9.4.0	GCC 11.2.0
Operating System	Ubuntu 20.04	Ubuntu 22.04

3.2 Experimental Results

Table 6 tabulates the coding efficiency and speed of the uvg266/OpenVVC pipeline over the VVenC/ VVdeC pipeline. The results are given per sequence and averaged across all sequences. VVenC/VVdeC is used as an anchor, i.e., negative BD-rate indicates that the uvg266/OpenVVC pipeline has

[1] Available at https://ultravideo.fi/vvc_e2e_2023/vvenc_intra_all_disabled.cfg

Table 6: Coding efficiency and speed of uvg266/OpenVVC pipeline over VVenC/ VVdeC pipeline.

Class	Sequence Name	Coding efficiency in BD-rate PSNR	SSIM	VMAF	Coding speed in fps Uvg266/OpenVVC	VVenC/VVdeC	Speedup
CTC-A1	Campfire	-2.7%	-6.4%	-5.2%	38	10	3.8×
	FoodMarket4	2.3%	0.7%	0.0%	33	11	3.0×
	Tango	4.3%	1.8%	-0.4%	32	10	3.2×
CTC-A2	CatRobot	-3.7%	-6.6%	-3.8%	33	10	3.3×
	DaylightRoad2	2.4%	-1.2%	0.8%	33	10	3.4×
	ParkRunning3	1.5%	-1.9%	7.0%	30	9	3.4×
UVG	Beauty	10.6%	2.0%	15.6%	34	10	3.5×
	Bosphorus	8.6%	5.8%	5.2%	37	11	3.2×
	CityAlley	-5.3%	-9.4%	-4.2%	39	11	3.4×
	Flower Focus	17.4%	12.5%	3.1%	38	11	3.3×
	FlowerKids	-2.1%	-4.0%	-3.4%	38	12	3.3×
	FlowerPan	-1.9%	-2.4%	-7.4%	31	10	3.0×
	HoneyBee	14.8%	12.6%	0.8%	31	11	2.9×
	Jockey	18.0%	12.4%	4.1%	35	11	3.2×
	Lips	9.4%	-1.3%	17.8%	37	10	3.8×
	RaceNight	17.7%	3.4%	5.8%	34	10	3.4×
	ReadySetGo	1.8%	0.8%	-1.4%	35	11	3.2×
	RiverBank	5.9%	2.6%	12.7%	36	10	3.5×
	ShakeNDry	12.3%	10.2%	7.3%	31	10	3.1×
	SunBath	-2.5%	-5.7%	-8.9%	39	11	3.6×
	Twilight	-7.9%	-12.4%	-4.8%	40	11	3.5×
	YachtRide	3.2%	1.6%	1.9%	34	11	3.0×
Average	**VVC CTC dataset**	**0.7%**	**-2.3%**	**-0.3%**	**33**	**10**	**3.4×**
	UVG dataset	**6.2%**	**1.8%**	**2.8%**	**36**	**11**	**3.3×**
	Total	**4.7%**	**0.7%**	**1.9%**	**35**	**11**	**3.3×**

better coding efficiency. It is noteworthy that the coding efficiency is only dependent on the encoder, which also dictates the overall performance of the pipeline in most, if not all cases.

The coding efficiency results show that our uvg266 increases BD-rate over that of VVenC by 4.7%, 0.7%, and 1.9% with PSNR, SSIM, and VMAF, respectively. Nevertheless, uvg266 outperforms VVenC with sequences having both darker and lighter areas, such as Campfire, CityAlley, SunBath, and Twilight. On the other hand, VVenC/VVdeC encodes sequences with shallow depth-of-field better, such as Bosphorus, FlowerFocus, HoneyBee, Lips, and ShakeNDry. This can be explained with different algorithmic optimization strategies of the encoders, which, e.g., stop the encoding search when certain threshold is reached. The main difference between the uvg266 and VVenC configurations is that the uvg266 bases its intra search decision entirely on the prediction, whereas VVenC performs reconstruction for the three best candidates after the rough search. This allows for more accurate compression of the noise caused by the shallow depth of field, but at the cost of much greater computational complexity.

On average, uvg266 achieves better coding efficiency with the VVC CTC dataset than with the UVG set. Moreover, uvg266 performs much better than VVenC when measured with SSIM. uvg266 inherited this property from Kvazaar. With VMAF, the results are less varied, except for Beauty and Lips, which have a high amount of noise.

For reference, the VTM/VTM pipeline supports all default coding tools and is able to achieve 43% better average BD-rate with PSNR than our uvg266/OpenVVC pipeline, but it barely processes 4K sequences at 0.001 fps. In comparison, the fastest configuration of the VVenC/VVdeC pipeline processes sequences at 11 fps on average. Hence, the proposed uvg266/OpenVVC pipeline is the only one that achieves real-time coding speed at 4K resolution, 35 fps on average. It has a slightly higher coding speed variance across sequences than that of the uvg266/OpenVVC pipeline, since VVenC only tests a single block size instead of the two as uvg266.

3.3 Live Demonstration Setup

Figure 1 illustrates the proposed prototype for the real-time VVC intra coding demonstration. An Epiphan AV.io HDMI capture card is used to capture raw 4K RGB video (3840×2160) at 30 fps from a Sony FDR X1000V 4K action camera. The captured RGB video is converted by FFmpeg to YUV 4:2:0 format, and then encoded to VVC bitstream by uvg266 on the AMD ThreadRipper 2990WX.

The VVC bitstream is transported with uvgRTP over Ethernet to an ARM-based Nvidia Jetson AGX Orin, where OpenVVC is used to decode the bitstream. Finally, the decoded bitstream is displayed with an SDL2 based viewer.

During the demonstration, visitors can view the decoded 4K video in real-time and monitor the CPU usage of both the encoding and decoding platforms.

4 CONCLUSION

In this paper, we introduced a complete E2E pipeline for VVC intra coding and streaming. Our prototype was made up of

open-source uvg266, uvgRTP, and OpenVVC tools. To the best of our knowledge, this proposal is the first open-source solution to demonstrate 4K VVC intra coding and streaming at 30 fps. It is able to achieve 34 000 and 3.3 times speedups over the corresponding setups built around VVenC/VVdeC and VTM codecs, respectively.

For future work, the coding overhead between uvg266 and VTM leaves room for further algorithmic optimizations to improve coding efficiency while maintaining real-time coding speed. There are a variety of other intra VVC coding tools that can be utilized and optimized in this effort. Additionally, incorporating inter coding tools into this prototype has the potential to greatly reduce the bandwidth requirements, but doing so without compromising real-time performance is a significant challenge that will require further research and development. Anyway, these activities will pave the way for future research on VVC codec design and implementation.

ACKNOWLEDGMENTS

This work was supported in part by the AI for situational Awareness (AISA) project led by Nokia and funded by Business Finland, and the Academy of Finland (decision no. 349216).

REFERENCES

[1] Cisco, Cisco Visual Networking Index: Forecast and Trends, 2017-2022, Dec. 2018.

[2] 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.

[3] 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, pp. 1649–1668, Dec. 2012.

[4] A. Mercat *et al.*, "Comparative rate-distortion-complexity analysis of VVC and HEVC video codecs," *IEEE Access,* vol. 9, pp. 67813–67828, 2021.

[5] G. Sullivan, "Deployment status of the HEVC standard," *document JVET-V0020*, Apr. 2021.

[6] "VVC Reference Software Version 18.0," [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-18.0.

[7] A. Wieckowski *et al.*, "Vvenc: An Open And Optimized Vvc Encoder Implementation," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, Shenzhen, China, Jul. 2021, pp. 1–2.

[8] M. Viitanen, J. Sainio, A. Mercat, A. Lemmetti, and J. Vanne, "From HEVC to VVC: the first development steps of a practical intra video encoder," *IEEE Trans. Consum. Electron.,* vol. 68, no. 2, pp. 139–148, May 2022.

[9] "Ultra Video Group," [Online]. Available: https://ultravideo.fi/.

[10] A. Wieckowski *et al.*, "Towards a live software decoder implementation for the upcoming versatile video coding (VVC) codec," in *Proc. IEEE Int. Conf. Image Proc.*, Virtual Conference, Oct. 2020, pp. 3124–3128.

[11] IETR Vaader, "OpenVVC open-source VVC decoder," [Online]. Available: https://github.com/OpenVVC/OpenVVC.

[12] T. Amestoy, P.-L. Cabarat, G. Gautier, W. Hamidouche, and D. Menard, "OpenVVC: a lightweight software decoder for the versatile video coding standard," arXiv, May 2022.

[13] A. Wieckowski *et al.*, "A complete end to end open source toolchain for the versatile video coding (VVC) standard," in *Proc. ACM Int. Conf. Multimedia*, Virtual Event, China, Oct. 2021, pp. 3795–3798.

[14] J. Le Feuvre, C. Concolato, and J.-C. Moissinac, "GPAC: open source multimedia framework," in *Proc. ACM Int. Conf. Multimedia*, Augsburg, Germany, Sept. 2007, pp. 1009–1012.

[15] T. Biatek, E. François, C. Thienot, and W. Hamidouche, "Sustainable OTT video distribution powered by 5G-multicast/unicast delivery and versatile video coding," in *Proc. Mile-High Video Conf.*, Denver, Colorado, USA, Mar. 2022, pp. 81–82.

[16] T. Biatek *et al.*, "Delivering universal TV services in a multi-network and multi-device world with DVB-I," in *Proc. Mile-High Video Conf.*, Denver, Colorado, USA, Mar. 2022, pp. 83–84.

[17] T. Biatek *et al.*, "Live OTT services delivery with ad-insertion using VVC, CMAF-LL and ROUTE: an end-to-end chain," in *Proc. Mile-High Video Conf.*, Denver, Colorado, USA, Mar. 2022, pp. 79–80.

[18] W. Hamidouche *et al.*, "Versatile video coding standard: a review from coding tools to consumers deployment," *IEEE Consum. Electron. Mag.,* vol. 11, no. 5, pp. 10–24, 2022.

[19] T. Biatek *et al.*, "Versatile video coding for 3.0 next generation digital TV in Brazil," *SET Int. J. Broadcast Eng.*, pp. 9–17, Dec. 2021.

[20] J. Räsänen, A. Altonen, A. Mercat, and J. Vanne, "Open-source RTP library for end-to-end encrypted real-time video streaming applications," in *Proc. IEEE Int. Symp. Multimedia*, Naples, Italy, Nov.-Dec. 2021, pp. 92–96.

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

[22] 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, Jun. 2020, pp. 237–242.

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

[24] 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.*, 2012, pp. 213–216.

[25] VideoLan, "VLC media player," 2006, [Online]. Available: https://www.videolan.org/vlc/index.html.

[26] FFmpeg, "FFmpeg github repository," [Online]. Available: https://github.com/FFmpeg/FFmpeg.

[27] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, "VTM common test conditions and software reference configurations for SDR video," *document JVET-T2010*, Teleconference, Oct. 2020.

[28] 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.

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

[30] 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*, 2020.

[31] 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, pp. 600–612, Apr. 2004.

[32] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," Jun. 2016, [Online]. Available: http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html.

[33] 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.