VVenC: an open optimized VVC encoder in versatile application scenarios

Wieckowski, Adam, Stoffers, Christian, Bross, Benjamin, Marpe, Detlev

Adam Wieckowski, Christian Stoffers, Benjamin Bross, Detlev Marpe, "VVenC: an open optimized VVC encoder in versatile application scenarios," Proc. SPIE 11842, Applications of Digital Image Processing XLIV, 118420H (1 August 2021); doi: 10.1117/12.2595360

SPIE.

Event: SPIE Optical Engineering + Applications, 2021, San Diego, California, United States

VVenC, an Open Optimized VVC Encoder, in Versatile Application Scenarios

Adam Wieckowski, Christian Stoffers, Benjamin Bross, Detlev Marpe
Fraunhofer HHI, Einsteinufer 37, Berlin, Berlin Germany 030-31002-0

ABSTRACT

Versatile Video Coding (H.266/VVC) was standardized in July 2020, around seven years after its predecessor, High Efficiency Video Coding (H.265/HEVC). Typical for a successor standard, VVC aims to offer 50% bitrate savings at similar visual quality, which was confirmed in official verification tests. While HEVC provided large compression efficiency improvements over Advanced Video Coding (H.264/AVC), fast development of video technology ecosystem required more in terms of functionality. This resulted in various amendments being specified for HEVC including screen content, scalability and 3D-video extensions, which fragmented the HEVC market, rendering only the base specification being widely supported across a wide range of devices. To mitigate this, the VVC standard was from the start designed with versatile use cases in mind, and provides wide-spread support already in the first version. Shortly after the finalization of VVC, an open optimized encoder implementation VVenC was published, aiming to provide the potential of VVC at shorter runtime than the VVC reference software VTM. VVenC also supports additional features like multi-threading, rate control and subjective quality optimizations. While the software is optimized for random-access high-resolution video encoding, it can be configured to be used in alternative use cases. This paper discusses the performance of VVenC beyond its main use case, using different configurations and content types. Application specific performance is also discussed. It is shown that VVenC can mostly match VTM performance with less computation, and provides attractive additional faster working points with bitrate reduction tradeoffs.

Keywords: VVC, video coding, implementation, encoder, compression, versatility.

1. INTRODUCTION

The Versatile Video Coding standard[1] (H.266/VVC) was finalized in July 2020, standardized by the Joint Video Experts Group (JVET), a joint body of ITU-T VCEG and ISO/IEC MPEG. Since then, the envisioned bitrate savings over High Efficiency Video Coding[2] (H.265/HEVC) has been confirmed for SDR UHD[3], SDR HD[4] and other content types. Beyond the compression efficiency improvement, the other aim during VVC development was versatile applicability, with support for a wide range of content types and use cases. Different than HEVC, already in the first version VVC supports screen content coding, advanced high-level picture partitioning, high dynamic range (HDR) video with up to 10bit bitdepth, alternative and professional chroma subsampling formats including 4:4:4, spacial scalability and reference picture resampling[5]. Currently work is underway to release a second version of the VVC standard with support for bitdepths beyond 10bit, high-bitrate coding.

During the standard development, the VVC test model VTM served a common software evaluate the proposed algorithms and later provide a reference implementation. Because of its purpose, the software is not optimized for real-world application, with very verbose search algorithms optimized to show maximal gain, and inhomogeneous level of optimization and vectorization across components. Additionally, the software lacks real-world features like multi-threading, effective rate control and has only basic support for subjective quality optimization. Overall, the VTM software encoder requires around 8× the runtime of HEVC reference model HM[6] to encode a video using the random-access configuration of the common test conditions[7]. The runtime increase is caused by increased computational complexity of the VVC itself, but also by increased search space, caused by modular design of VVC with many tools being enabled per coding unit and increasing the number of encoding decisions. The new flexible partitioning scheme of VVC further extends the search space by allowing very flexible block partitioning into coding units.

Shortly after the finalization of VVC, an open optimized encoder implementation project was presented, VVenC[8] with the goal of providing a real-world VVC encoder implementation, providing all of VTM's performance at shorter runtimes, as well as additional even faster working points as defined by the pre-configured presets (*fast*, *faster*, *medium*, *slow* and

Applications of Digital Image Processing XLIV, edited by Andrew G. Tescher, Touradj Ebrahimi, Proc. of SPIE
Vol. 11842, 118420H · © 2021 SPIE · CCC code: 0277-786X/21/$21 · doi: 10.1117/12.2595360

slower)[8,9,10]. Additionally, VVenC supports real-world encoder features including efficient multi-threading for further speedup, single- and two-pass rate control as well as subjective optimizations. The development of VVenC has been concentrated to provide optimal working points for high-resolution video encoding using random access configuration as defined by JVET. With the project maturing towards the version v1.0.0 in May 2021, it is starting to embrace additional use cases. This paper provides the overview of the performance of VVenC in different configurations, including alternative temporal configurations defined by JVET[7]. Additionally, application to different content types as well as non-CTC high-resolution video is tested.

The rest of the paper is structured as follows. Section 2. introduces the VVenC encoder, including description of applied optimizations and preset derivation process. The experimental setup is described in Section 3. Section 4. presents encoding results of VVenC in different use-cases, including intra-only and low-delay scenarios, as well as application to screen content and HDR video. A conclusion and a short outlook are given in Section 5.

2. VVENC

After the initial release of the version v0.1 in September 2020, a first stable release v1.0 was published in May 2021. If not explicitly stated differently, all of the following results are based on the v1.0 version of VVenC. The project is written in C++ and provides a simple to use C-library as well as two standalone encoder executables – a simple application optimized for ease-of-use providing only basic options, and an expert mode encoder with VTM-style interface allowing fine-grained configuration.

2.1 Supported VVC features

The VVenC encoder supports single-slice/tile encoding using the Main10 profile of VVC. While alternative chroma sub-sampling formats like 4:0:0 or 4:4:4 are generally supported, the encoder does not support the Main10-4:4:4 profile allowing their application.

Most of the core coding tools available in the Main10 profile of the VVC standard are already supported, with the exception of adaptive color transform (ACT) and bi-prediction with CU weights (BCW). Support for the latter is planned in version v1.1 or v1.2. Non YUV-content can be transformed into YUV during preprocessing, making ACT low-priority during current development.

A big versatility feature of VVenC is only implicitly supported. While the encoder cannot perform reference picture resampling and produce a resolution switch, it can create bitstreams which can be segmented and mixed at different resolutions with minimal drift, e.g. for adaptive streaming with resolution change[11].

2.2 Additional optimizations and configuration space

The VVenC was developed based on a stripped down version of the VTM reference software containing only basic coding tools inherited from HEVC as well as the improved partitioning, thus borrowing a lot of internal structures and the application interface. On top of this baseline most of core VVC coding tools were ported over from VTM, taking care to fix performance bottlenecks as well as to equalize the level of optimization and vectorization. For this reason, the implementation of many of the coding tools is similar to that of VTM, but more optimal.

As stated before, VTM uses an extensive search algorithm for most of the coding tools, with only conservative early exit strategies being utilized. To provide additional working points, improvements to the search algorithms were introduced[10]. While the simple application allows only preset selection, the expert app allows to select exactly which coding tools to use in what search variant. By convention, a coding tool is disabled if the respective switch is set to ‘0’, and enabled for values larger than ‘0’. For tools allowing multiple speed settings, an increasing value would select a faster operating point. Additionally, some tool independent speedups are exposed as separate options.

Compared to VTM, additional speedups are implemented for partitioning search, motion and merge search including new motion models (affine, adaptive motion vector resolution, symmetric merge vector difference, merge with motion vector difference and geometric partitioning mode), and new intra tools including intra sub-partitions and matrix-based intra prediction. Also, an optimized integer based RDOQ is included and enabled in faster and fast presets.

Table 1. VVenC configuration settings for the partitioning, tools and additional speedups as configured for the five presets. For options with different naming conventions than [AHG13] the correct acronym is marked *cursive*.

Option	faster	fast	medium	slow	slower	Option	faster	fast	medium	slow	slower
Partitioning						MCTF	0	2	2	2	2
CTUSize	64	64	128	128	128	MIP	0	0	1	1	1
MinQTISlice[1]	4	4	8	8	8	MMVD	0	0	3	3	1
MinQtNonISlice	4	4	8	8	8	MRL	0	0	1	1	1
MaxMTTHierarchyDepth	0	0	1	2	3	MTS[4]	0	0	0	0	1
MaxMTTHierarchyDepthI[2]	0	1	2	3	3	MTSImplicit[4]	1	1	1	1	0
Tools						RDOQ[3]	2	2	1	1	1
Affine (incl. PROF)	0	2	2	2	1	SBT	0	0	0	1	1
ALF/CCALF	0	1	1	1	1	SbTMVP	0	0	1	1	1
BIO (*BDOF*)	1	1	1	1	1	SDH[3]	1	1	0	0	0
DepQuant[3]	0	0	1	1	1	SMVD	0	0	3	1	1
DMVR	1	1	1	1	1	TS/IBC/BDPCM (SCC)	2	2	2	2	2
EncDbOpt	0	0	2	2	2	Fast algorithms					
GEO (*GPM*)	0	0	3	1	1	QtbttExtraFast	2	2	2	2	1
IMV (*AMVR*)	0	5	5	1	1	ContentBasedFastQtbt	1	1	1	0	0
ISP	0	0	3	3	1	FastMRG	2	2	2	2	1
JointCbCr	0	0	1	1	1	FastMip	-	-	4	4	1
LFNST	0	1	1	1	1	FastSubPel	1	1	1	1	0
LMChroma (*CCLM*)	1	1	1	1	1	FastIntraTool	0	0	1	0	0
LMCSEnable	0	0	1	1	1	IBCFastMethod	6	4	3	1	1

[1]Option *MinQTChromaISliceInChromaSamples* is configured to half the specified size.
[2]Used for both *MaxMTTHierarchyDepthISliceL* and *MaxMTTHierarchyDepthISliceC*.
[3]DepQuant and integer RDOQ (=2) with sign data hiding (SDH) are mutually exclusive.
[4]*MTS* and *MTSImplicit* are mutually exclusive.

2.3 Preset derivation

All of the encoding options concerning tools and their search variants create a configuration vector which can be set to different values to define alternative coding-efficiency / runtime tradeoffs. The set of such options used to define alternative tradeoffs in VVenC is shown in Table 1.

The process of preset derivation in VVenC starts with determination of an approximation of the Pareto Set[10,12] of the configuration space comprising all possible configuration vector values. The Pareto Set is approximatively determined by defining a fast starting point with most of the tools disabled and all of the speedups enabled. In an iterative process, a set of candidate subsequent working points is tested. The configuration vector with the best tradeoff (biggest bitrate-reduction proportional to the time overhead) is than selected as a new point in the Pareto Set approximation. The process is than repeated relative to this point. The set of determined points is only approximate Pareto optimal, since the iterative process does not account for configuration option interactions, and does not search through the whole configuration space.

Along the approximate Pareto optimal configurations, five points are selected as presets, ranging from faster to slower, with the latter providing all of VTMs compression efficiency. The faster point should require a fraction of the time of HM to encode a video and provide a substantial gain of at least 10% BD-rate gain[13] (compared to HM-16.22). Other presets are spread evenly in-between, both with regard to runtime and compression efficiency. The exact configuration of each preset is shown in Table I.

The Pareto Set approximation is empirically determined on a set of test sequences comprising the high-resolution sequences from the JVET CTC, i.e. classes A1, A2 and B. Thus, the preset definition is also optimized for those sequences. In this paper, we will discuss the results of VVenC application to alternative configurations and sequences.

2.4 Low delay and all-intra support

The presets described in last section are derived for optimality when used in the random access configuration, with random access interval of around 1 second as specified in JVET CTC. The tools and search algorithms themselves, while possibly

being more or less effective in specific scenarios, are independent of actual bitstream requirements with regards to prediction structure or decoding latency. Additionally to the random access scenario, JVET CTC also specifies an all-intra and two low-delay temporal configurations (allowing either only P- or only B-frames). To adhere to those conditions, the encoder configuration has to be adapted. For the purpose of this paper, we do it a naïve manner. For all-intra, the temporal configuration is specified to only produce IDR and CRA picture, while all inter coding tools are disabled. We decided to only test the bi-predicted low-delay configuration. To adhere to its restrictions, the low-delay-B GOP8 temporal configuration from VTM is used (the expert mode app understand VTM parameter semantics), and all bi-prediction based tools requiring prediction from temporally equidistant sources (e.g. DMVR, BDOF) are disabled.

It is assumed that the working points of the presets could be further improved by performing a Pareto Set approximation for the specific use cases.

2.5 Screen content support

VVenC supports all of screen content coding (SCC) specific tools available in the Main10 profile of VVC. In the JVET CTC, the enabling of those tools is defined depending on the tested testset. In VVenC, which is a practical encoder, we decided not to require the end-user to know how to set optimal encoding parameters. The encoder utilizes an in-build screen content detection and allows enabling of the specific tools depending the classification result. To enable this behavior, the semantics of screen content specific tools break the previously described semantics regarding different search algorithm. For transform skip (*TS* and *ChromaTS*), intra block copy (*IBC*) and block differential pulse code modulation (*BDPCM*), a value of '0' means the tool is disabled, '1' indicates its always enabled and '2' indicates it should be enabled or disabled depending on the frame classification. For this reason IBC requires a separate parameter to select the speed variant. Additionally, the motion compensated temporal filter (MCTF) is also disabled for frames classified as screen content. The default configuration is shown in Table 1.

2.6 HDR support

With HDR and 10bit content being natively supported in VVC, and one of its main future application areas, HDR is also supported in VVenC. The somehow complicated configuration of HDR parameters comprising setting the exact color primaries, QP offsets and other options has been simplified to setting of a single parameter *Hdr* to one of the following values: *off*, *pq*, *pq_2020*, *hlg*, *hlg_2020*, defining the input signal to be either SDR, PQ or HLG, and allowing discrimination between BT.709 and BT.2020 color space for the two HDR variants. In the expert application, the fine grained configuration is still available, but for the purpose of this paper configuration *--Hdr=pq* is used for JVET CTC H1 class and *--Hdr=hlg* is used for the JVET CTC H2 class. Differently than in VTM, setting the *Hdr* parameter instructs the encoder to generate HDR SEI metadata.

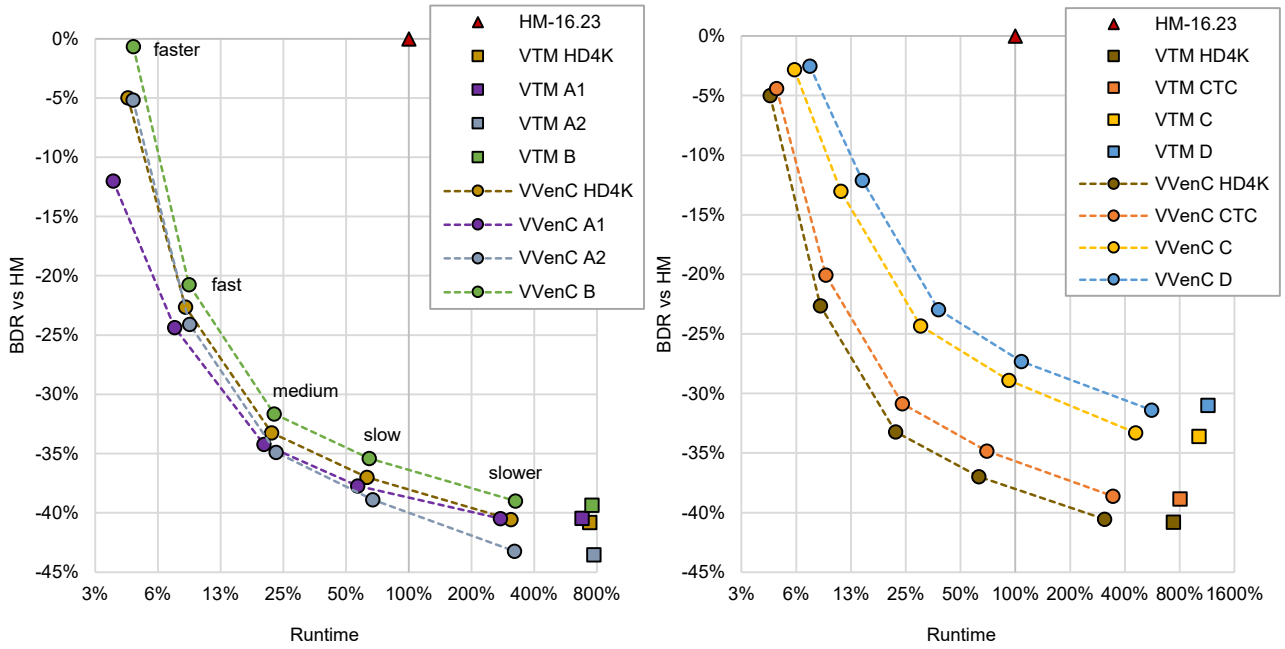
3. EXPERIMENTAL SETUP

The following experimental data was acquired using VVenC 1.0.0, HM-16.23 and VTM-12.0 in single threaded operation. Times were measured on 32 core server blades with Intel Xeon E5-2697A-v4 @2.6GHz CPU, with hyper-threading disabled. For each encoder, fixed QP encoding without subjective optimizations with four QPs (22, 27, 32, and 37) were performed. From the resulting bitstream sizes and PSNR distortion of the reconstructed video, a weighted PSNR (using 6-1-1 Y-U-V PNSR weighting) BD-rate difference[13] as recommended by JVET[14] is calculated and reported averaged across the tested sequences in the Figures 1., 2., and 3. as well as Table 2. The relative runtime vs the HM anchor is calculated as a ratio of the geometric means of encoding times of all sequences in a given set.

4. RESULTS

4.1 Performance at different resolutions

In this comparison the JVET CTC sequences were encoded using the random-access temporal configuration. For VTM and VVenC, GOP32 configurations were used, HM uses a GOP16 configuration but with intra-period aligned to VTM. All encoders but VVenC in faster preset utilize the motion compensated temporal filter. The results are shown per-class as well as combined in for the whole CTC as well as the high-resolution sequences used to derive the Pareto Set and presets (HD4K).



a) High-resolution contet.

b) Low-resolution content.

Figure 1. Random-access per class VVenC and VTM results compared to an HM-16.23 anchor based on JVET SDR CTC sequences. HD4K is a combination of classes A1, A2 and B, thus representing the testset used for preset derivation. CTC represents the average over all JVET CTC classes. Runtime is displayed on a logarithmic scale.

The results are shown in Figure 1. and Table 2. Compared to previous comparisons of VVenC compression efficiency, the gap to HM is somehow smaller. This is caused by the inclusion of the motion compensated temporal filter into the HM test conditions since the version HM-16.23 (previously reported numbers usually used HM-16.22 as anchor).

It can be observed in Figure 1. that the slower preset of VVenC always roughly meets the coding performance of VTM, at around half or less of the runtime. From there, other presets are evenly spaced towards faster and much faster tradeoffs, while reducing the compression efficiency gain over HM.

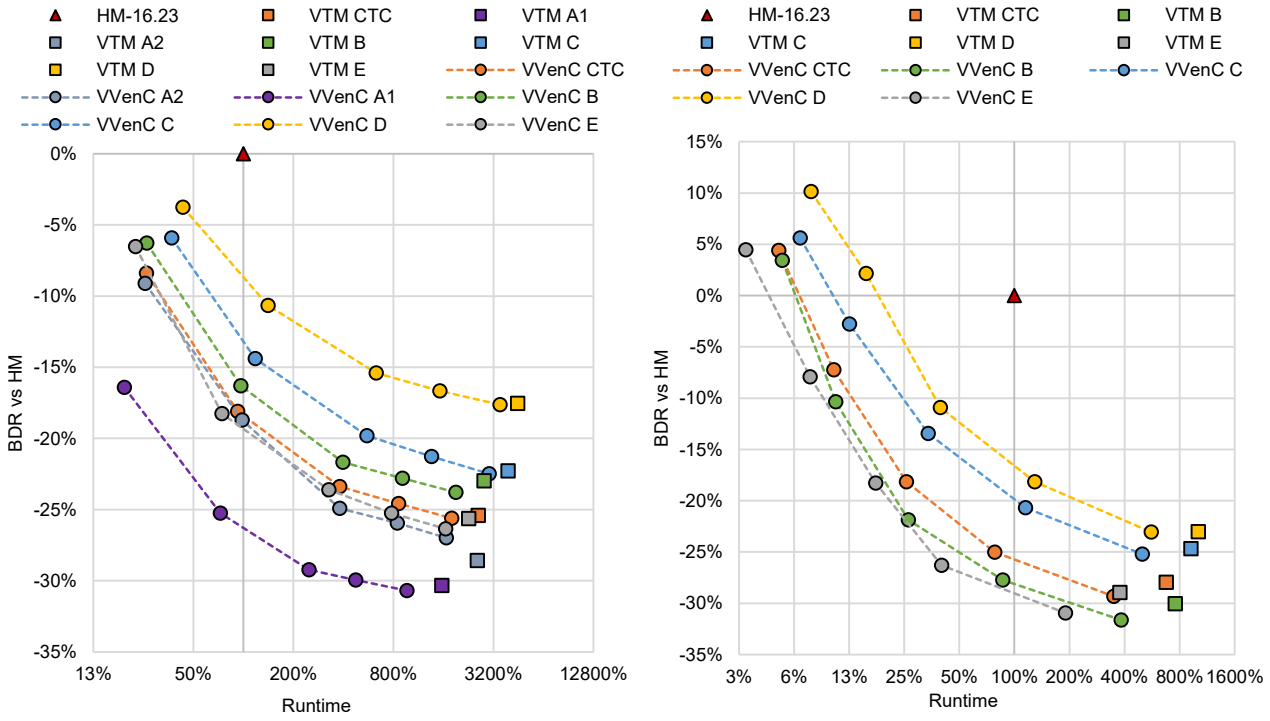
Comparing the gains between the classes, similar pattern can be observed for both VTM and VVenC – the compression efficiency increases with increasing resolution. The behavior is caused by VVC being optimized for high-resolution encoding. Interestingly, while the preset behavior in VVenC is mostly consistent between classes, the curves for classes A1 and A2 cross around the medium preset, with faster providing better tradeoff for A1 and slower, similar to VTM, showing increased performance for A2. This indicates that the special characteristics of the class A1 respond better to the reduced set of tools available in faster than class A2. Class A2 usually responds well to affine motion, which is enabled in the fast preset, at which the compression efficiency between the two presets is matched.

4.2 Performance for different temporal configurations

Figure 2. and Table 2. show the comparison of the five VVenC presets to HM and VTM in two alternative temporal configurations, all-intra in Figure 2.a) and low-delay B in Figure 2.b). The respective configurations were derived naively by adapting the temporal configuration according to VTM and disabling all inter tools for all-intra and some bi-predictive tools for low-delay B.

With the exception of the conversational video class E in low-delay B the results very much resemble the pattern visible for random access configuration in Figure 1. VVC, both VTM and VVenC, works better for higher resolutions. The conversational content in class E shows by far the best compression performance in low-delay B configuration. With the content being very static, the video contains less information in theoretical sense, similar to common high resolution video, and VVC is just very good at using those redundancies.

With regard to encoding time, VVenC slower can hold the 2× speedup versus VTM in the low-delay configuration, while maintaining or even exceeding its compression performance. The additional gain might be caused by the per-default



a) All-intra configuration.

b) Low-delay B configuration.

Figure 2. Per-class results for alternative temporal configuration of VVenC, VTM and HM. Random access results for comparison are available in Figure 1. Runtime is displayed on a logarithmic scale.

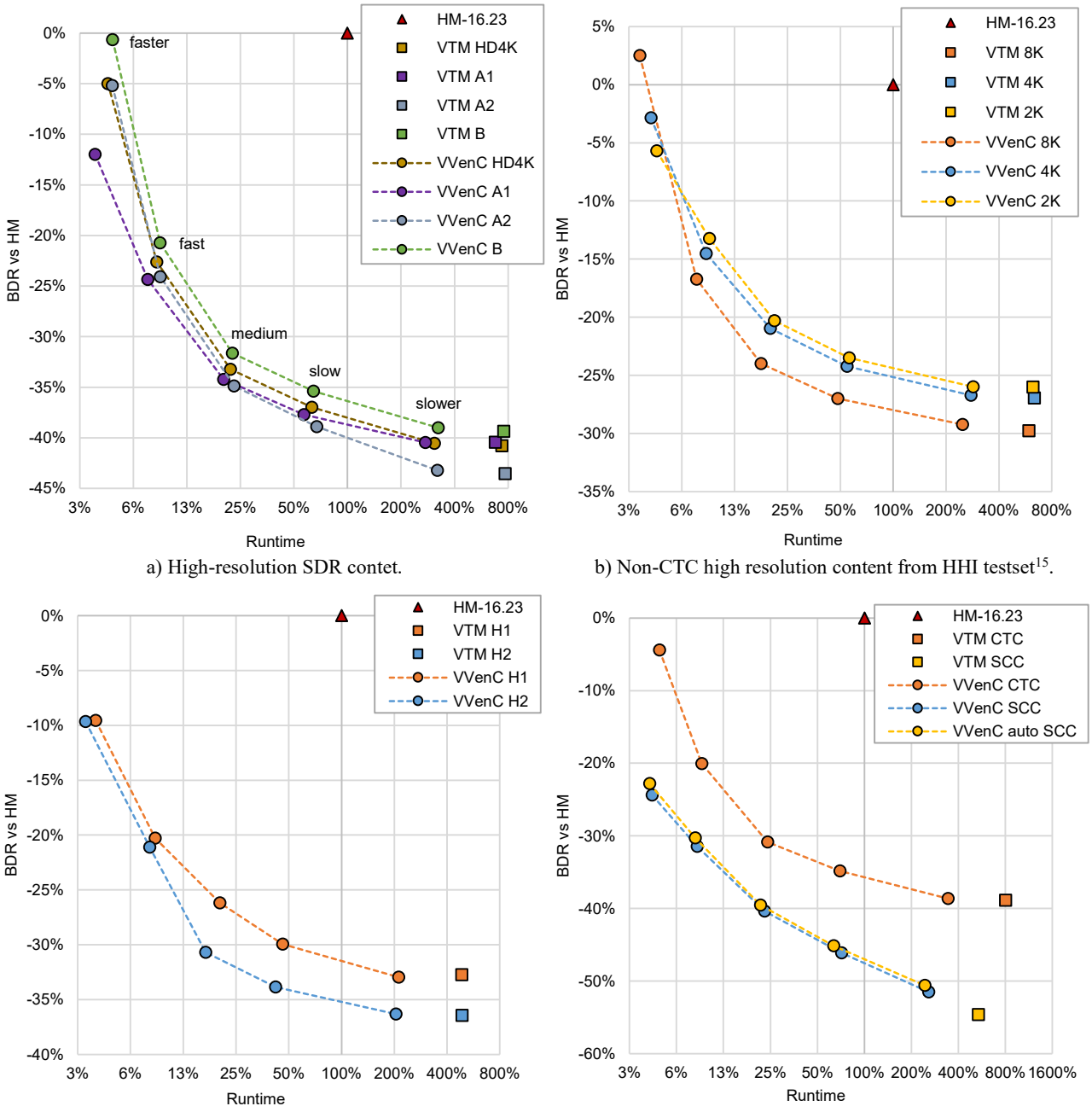
enabled SCC tools. While class E is not SCC, the static content might be classified as such and the additional tools can provide some additional gain. Interestingly, while overall the runtime behavior of VVenC for low-delay B is similar to that of random access, the produced gains are much lower with faster always showing slight loss versus HM and for class D even showing loss using the fast preset.

For the all-intra configuration, VVenC slower is only marginally faster than VTM with 1.5× speedup, while for most classes exceeding the compression performance. This is an interesting behavior, since VVenC obviously does not contain any additional coding tools optimized for this use case. The slight gain might be attributed to alternative implementation of some intra oriented coding tools like ISP and LFNST, which are in VVenC implemented inside the transform coding loop search rather than the coding unit loop search as in VTM. Interestingly, for class A2 a loss of around 2% to VTM is observed at the slower preset.

The spacing of the presets in the low-delay B configuration is very similar to that of random access and shows good distribution in both runtime and compression performance. For all-intra on the other hand, the presets faster-fast-medium are spaced much further apart than medium-slow-slower, both in terms of runtime and gain. This is caused by random-access based Pareto Set derivation. Most of the coding tools have drastically different tradeoffs between the two configurations, which makes it somehow noteworthy that the preset spacing, while uneven, looks more or less good even using the naive derivation method.

4.3 Performance for alternative content and content types

The VVenC presets are optimized for JVET high-resolution SDR sequences (classes A1, A2, and B) in JVET random access configuration[7]. While the JVET testset contains very different types of content with different kinds of motion, it is important to ensure the encoder settings are not overfitted for that specific content. In Figure 3. the results of training testset in 3.a) are compared to an alternative set of high-resolution SDR sequences from the HHI-Berlin testset[15] in Figure 3.b). The set consists of 7 8K sequences acquired around Berlin. The sequences have been downsampled to different resolutions, including 4K and 2K as shown in Figure 3.b). The overall performance compared to HM is significantly lower measured



a) High-resolution SDR contet.

b) Non-CTC high resolution content from HHI testset[15].

c) High-resolution HDR content.

d) Screent content coding (F, TGM).

Figure 3. Random-access per class VVenC and VTM results compared to an HM-16.23 anchor based on alternative content. For comparison results for high resolution JVET CTC sequences are shown in a). In b), encoding of the HHI sequences in SDR at different resolutions is shown. Classes H1 and H2 in c) are from JVET HDR testset. SCC in class d) shows combined JVET class F and TGM results. For VVenC, results with automatic SCC detection and with SCC tool always on are presented.. Runtime is displayed on a logarithmic scale.

on the HHI sequences, for both VVenC and VTM. As in the training set, VVenC in the slower presets matches the compression performance of VTM, also at around 50% of its runtime. The other presets are evenly spaced in both runtime and compression performance up to faster with around 4% HMs runtime at slightly improved compression performance.

Table 2. Recap of the most important results from Figures 1., 2., and 3. BD-rate savings and runtime of different VVenC presents and VTM over HM for different content in the random access configuration as well as all-intra (AI) and low-delay B (LDB) configurations.

BD_{YUV} [%] / EncT [%]	Random-Access							AI	LDB
Source	**JVET**				**HHI**			**JVET**	
Testset	**HD4K**	**CTC**	**SCC**	**HDR**	**2K**	**4K**	**8K**	**CTC**	**CTC**
VVenC faster	-5.0	-4.4	-22.8	-9.6	-5.7	-2.86	2.5	-8.2	4.4
	4.5	4.9	4.2	3.8	4.5	4.2	3.6	26.1	5.2
VVenC fast	-22.6	-20.1	-30.3	-20.6	-13.2	-14.5	-16.7	-15.7	-7.2
	8.5	9.1	8.2	8.4	9.0	8.6	7.6	92.3	10.3
VVenC medium	-33.2	-30.9	-39.5	-27.8	-20.3	-21.0	-24.0	-21.3	-18.2
	22.0	23.9	21.5	19.9	21.0	20.0	17.7	379.9	25.7
VVenC slow	-37.0	-34.8	-45.1	-31.4	-23.5	-24.2	-27.0	-22.4	-25.0
	63.0	69.7	63.5	44.6	56.2	54.7	48.4	857.1	78.1
VVenC slower	-40.6	-38.6	-50.6	-34.2	-26.0	-26.7	-29.2	-23.5	-29.3
	309.0	343.4	244.1	209.2	285.4	277.2	248.0	1794.7	349.1
VTM	-40.8	-38.9	-54.6	-34.1	-26.0	-27.0	-29.8	-25.0	-28.0
	738.1	804.5	537.4	486.0	623.9	635.4	592.0	2601.6	678.1

Comparing the different resolutions tested, the best compression performance can be achieved for 8K sequences, with 4K slightly outperforming 2K, for all presets but faster.

Additionally to a validation set for high-resolution SDR content, Figure 3. shows VVenC performance for two different content types – HDR in 3.c) and Screen Content in 3.d). The HDR content in 3.c) showcases the two classes from JVET HDR common test conditions[16] – class H1 containing 7 BT.709 PQ sequences and class H2 containing four BT.709 HLG sequences. The encoding process for HDR content does not differ much from that of SDR signal but some tools including LMCS and ALF have slightly different operation modes, which needs to be validated. Different than VTM, to ensure correct HDR reproduction on the end-user device VVenC generates appropriate SEI messages. Compared to VTM, the VVenC encoder can reach the compression performance at less than half the runtime. Slightly larger gains for the H2 class can be observed for VTM as well as VVenC. The preset spacing looks similar to the training set in Figure 3.a).

In Figure 3.d) VVenC performance for screen content coding (SCC) is shown, comprising the combined results for classes F and TGM from the JVET CTC. For VVenC, two curves are shown: one utilizing the default operation modi with automatic screen content detection (VVenC auto SCC) and a second with SCC specific tools explicitly enabled and always active (VVenC SCC). Both curves are very close to each with only marginal efficiency loss when using the automatic mode, resulting from misclassification of some frames. The HM anchor in this experiment is baseline HM, without the screen content coding extension, explaining larger gains of both VVenC and VTM compared to other experiments. VVenC in the slower configuration does not quite reach the performance of VTM. This behavior will be mitigated in future version by disabling additional tools during SCC and adapting e.g. the motion search strategy if SCC is detected. Different than in other use-cases, already the faster preset shows large gains against HM, caused by the inclusion of SCC-specific tools the first version of VVC and all VVenC presets. Using faster search algorithms in some presets ensures that speedups are kept and the faster preset runs around 25× faster than HM, similar to the training testset in Figure 3.a). The preset spacing between faster and slower is very regular with regard to both runtime and added gain.

5. CONCLUSION

In this paper the performance of the open optimized VVC encoder, VVenC, was tested and discussed for use-cases alternative to the main use-case of random-access high-resolution content coding. It has been shown that VVenC can provide all of VTMs compression efficiency for all but SCC content, at faster runtimes. For motion compensated encoding, a speedup of at least 2× is provided by the slower preset, with more speedups available using other presets, trading off runtime for compression performance. In the low-delay configuration, VVenC outperforms VTM in the slower preset, while keeping the speedup. Runtime difference is smaller using the all-intra configuration, which is not surprising since the software has not been optimized for that use case. Additional alternative use cases have been tested in literature including open GOP encoding for HTTP adaptive streaming, application with learning based preprocessing and others, proving VVenC to be a mature and versatile encoder for VVC.

Overall, it has been show that VVenC provides good encoding performance across a wide variety of content types and applications. The predefined presets show consistent behavior for random-access encoding as well as different temporal configurations. It has been shown that automatic content detection works very efficiently. The simplified HDR parameters allows easy of use while keeping the compression performance.

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] ITU-T and ISO/IEC JTC 1, “High Efficiency Video Coding,” Rec. ITU-T H.265 and ISO/IEC 23008-2 (HEVC), April 2013.

[3] V. Baroncini and M. Wien, “VVC Verification Test Report for UHD SDR Video Content,” doc. JVET-T2020 of ITU-T/ISO/IEC Joint Video Experts Team (JVET), 20th meeting, October 2020.

[4] V. Baroncini and M. Wien, “VVC verification test report for HD SDR and 360° video content,” doc. JVET-V2020 T2020 of ITU-T/ISO/IEC Joint Video Experts Team (JVET), 22th meeting, April 2021.

[5] B. Bross, et al. " Overview of the Versatile Video Coding (VVC) Standard and its Applications," to appear in IEEE Transactions on Circuits and Systems for Video Technology (2021).

[6] F. Bossen, X. Li, K. Sühring, K. Sharman, and V. Seregin, “JVET AHG report: Test model software development (AHG3),” doc. JVET-W0003 of ITU-T/ISO/IEC Joint Video Experts Team (JVET), 23rd meeting, July 2021.

[7] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, “VTM common test conditions and software reference configurations for SDR video,” doc. JVET-T2010 of ITU-T/ISO/IEC Joint Video Experts Team (JVET), 20th meeting, October 2020.

[8] A. Wieckowski et al., "VVenC: An Open and Optimized VVC Encoder Implementation," 2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 1-2 (2021).

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

[10] J. Brandenburg et al., "Pareto-optimized coding configurations for VVenC, a fast and efficient VVC encoder," submitted to 2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP).

[11] R. Skupin et al., “Open GOP Resolution Switching in HTTP Adaptive Streaming with VVC,” 2021 Picture Coding Symposium (PCS), 1-5 (2021).

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

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

[14] ITU-T HSTP-VID-WPOM and ISO/IEC TR 23002-8, “Working practices using objective metrics for evaluation of video coding efficiency experiments,” 2021.

[15] B.Bross, H. Kirchhoffer, C.Bartnik, M. Palkow, and D. Marpe, “AHG4 Multiformat Berlin Test Sequences,” doc. JVET-Q0791, January 2020.

[16] A. Segall, E. François, W. Husak, S. Iwamura, D. Rusanovskyy, “JVET common test conditions and evaluation procedures for HDR/WCG video,” doc. JVET-V2021 of ITU-T/ISO/IEC Joint Video Experts Team (JVET), 22nd meeting, Apr. 2021.