VVenC
Fraunhofer Versatile Video Encoder

v1.3.1

Jens Brandenburg, Adam Wieckowski, Tobias Hinz, Ivan Zupancic, Benjamin Bross
Video Communication & Applications Department,
Fraunhofer Heinrich Hertz Institute (HHI), Berlin, Germany

1 INTRODUCTION

In July 2020, the Joint Video Experts Team (JVET), a collaborative project of the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG), has finalized a new video coding standard called Versatile Video Coding (VVC) [1][2]. VVC is the successor of the High Efficiency Video Coding (HEVC) standard [3][4] and has been released by ITU-T as H.266 and by ISO/IEC as MPEG-I Part 3 (ISO/IEC 23090-3). The new standard targets a 50% bit-rate reduction over HEVC at the same visual quality. In addition, VVC proves to be truly versatile by including tools for efficient coding of video content in emerging applications, e.g. high dynamic range (HDR), adaptive streaming, computer generated content as well as immersive applications like 360 degree video and augmented reality (AR).

The Fraunhofer Versatile Video Encoder (VVenC) [5] development was initiated to provide a publicly available, fast and efficient VVC encoder implementation. The VVenC software is based on VTM [6], with optimizations including software redesign to mitigate performance bottlenecks, extensive SIMD optimizations, improved encoder search algorithms and multi-threading support to exploit parallelization. Additionally, VVenC supports real-world encoder features, including frame-level rate control and perceptually optimized encoding in order to provide a flexible, fast and easy to use video encoding solution for the VVC standard.

Bit-streams encoded with VVenC can be decoded by any VVC standard compliant decoder, e.g.the fast Fraunhofer Versatile Video Decoder (VVdeC) [7].. Alternatively, the VTM reference software decoder solution can be used. Resulting encoded raw VVC bitstreams can be encapsulated into multimedia files and used in multimedia pipelines as described in [8].

Features at a Glance

- Easy to use encoder implementation with five predefined quality/speed presets.
- Optimized VVC encoder providing speedups between 17x and 1300x over VTM-14.2 for HD and UHD test sequences depending on the chosen quality/speed preset.
- Perceptual optimization to improve subjective video quality.
- Frame-level rate control supporting VBR encoding.
- Expert mode encoder interface available, allowing fine-grained control of the encoding process.

2 GETTING STARTED

2.1 HOW TO OBTAIN VVENC?

The software is hosted at GitHub under: https://github.com/fraunhoferhhi/vvenc

To clone the project use:

```
git clone https://github.com/fraunhoferhhi/vvenc
```

2.2 HOW TO BUILD VVENC?

The software uses CMake to create platform-specific build files. A working CMake installation is required for building the software. Download CMake from http://www.cmake.org/ and install it. The following targets are supported: Windows (Visual Studio), Linux (gcc) and MacOS (clang).

How to build for Windows?

In order to compile the software for Windows, Visual Studio 15 2017 or higher and CMake Version 3.13 or higher are required. Install gnuwin32 that provides make for Windows. To build the software open a command prompt window, change into the project directory and use:

```
make install-release
```

This will create the statically linked release version of the encoder applications in the install/bin/ subdirectory.

How to build for Linux/MacOS?

In order to compile the software for Linux, gcc-5.0 or higher and CMake Version 3.13 or higher are required. For MacOS Xcode and CMake Version 3.13 or higher are required. To simplify the build process a Makefile with predefined targets is available. To build the VVenC encoder applications open a terminal, change into the project directory and use:

```
make install-release
```

This will create the statically linked release version of the encoder applications in the install/bin/ subdirectory.

2.3 HOW TO USE VVENC?

The encoder project includes two encoder executables, a simple encoder app (vvencapp) and a full featured expert encoder (vvencFFapp).

How to use the simple encoder app?

The simple encoder app (vvencapp) can be used in one of five predefined presets. Each preset represents a different tradeoff between encoder runtime and video quality. In the slowest preset, the encoder reaches the highest compression gain, whilst in the fastest preset the runtime is significantly decreased. These preset configurations have been determined based on the Pareto optimal configuration set of the encoder configuration space, which is detailed in [9]. A list of the main encoder command line parameters is shown in Table I.

Table I: List of important encoder options. For full list please use the --help option.

OPTION	DEFAULT	DESCRIPTION
--help,-h		Show basic help
--input <str>	not set	Raw yuv input file (use `-` to read from standard input)
--size <wxh>	1920x1080	Input file resolution (width x height)
--framerate <int>	60	Temporal rate of input file. Required for rate control and calculation of output bit-rate. Also recommended with perceptual QP adaptation (see `--qpa` option below).
--framescale <int>	1	The denominator of the framerate to enable fractional rate specification.
--fps <int>	60/1	Fractional framerate specification, setting `--framerate` and `--framescale` using a single parameter with a fraction syntax (denominator defaults to `1` if not present).
--format <str>	yuv420	Set input format to YUV 4:2:0 8bit (yuv420) or YUV 4:2:0 10bit (yuv420_10)
--output <str>	not set	Bit-stream output file
--preset <str>	medium	Preset for specific encoding setting (faster, fast, medium, slow, slower)
--qp <int>	32	Quantization parameter (0..63)
--bitrate <int>	0	Bit-rate for rate control (0 constant QP encoding rate control off, otherwise bits per second). Rate control requires correct framerate.
--passes <int>	2	Number of passes used for rate control
--pass <int>	not set	Set current rate control pass. If not set, encoder will run both passes one after the other in one call. If set to [1,2], encoder will execute first or second pass only. Requires `--rcstatsfile` to be set (see below).
--rcstatsfile <str>	not set	Rate control statistics file, to store or load first pass rate control statistics data.
--qpa <int>	1	Perceptual QP adaptation (QPA) to improve subjective video quality (0: off, 1: on)
--hdr <str>	off	HDR mode of the input signal: SDR (off), HDR10 with PQ transfer function and BT.709 color primaries (pq or hdr10) or BT.2020 color primaries (pq_2020 or hdr10_2020), HLG transfer function and BT.709 color primaries (hlg) or BT.2020 color primaries (hlg_2020)
--threads <int>	Size ≥ 720p: 8 else: 4	Number of threads (1..N)

Example usage: Given a YUV 4:2:0 input file with a bit-depth of 8bit and a resolution of 176x144 pixels, the following call will encode the input file with the medium speedup preset:

```
vvencapp --preset medium -i BUS_176x144_75@15.yuv -s 176x144 -r 15 -o str.266
```

How to use the full featured expert mode encoder?

The expert mode encoder (vvencFFapp) is based on the VTM configuration scheme. Most of the parameters have been kept similar to VTM, but for some parameters, additional modes are available. Furthermore, not supported options have been removed. Example configuration files for the expert mode encoder can be found in the **cfg** sub-directory (see Table II).

Table II: Expert mode encoder configuration files provided in the ./cfg sub-directory.

CONFIGURATION FILE	**DESCRIPTION**
sequence.cfg	Sequence specific configuration parameters. Must be always adapted to the input sequence.
randomaccess_[faster, fast, medium, slow, slower].cfg	Random access configuration for different presets. Each configuration file corresponds to one of the 5 preset modes.
qpa.cfg	Perceptually optimized QPA configuration file.
rc1p.cfg	Single pass rate control configuration, overriding default fix QP setup.
rc2p.cfg	Two pass rate control configuration, overriding default fix QP setup

Example usage: In order to start your first experiments with the expert mode encoder, adapt the sequence.cfg configuration file to your input YUV source file and use the following command:

```
vvencFFapp -c randomaccess_medium.cfg -c sequence.cfg
```

3 Encoder Performance

The encoder performance tests focus on the most relevant HD (1920x1080) and UHD (3840x2160) resolution use cases, in the following denoted as HD4K use case, with random access encoding as defined in the JVET common test conditions (CTC) [10]. Unless stated otherwise, all presented results are shown for the CTC test sequences, i.e. classes A1 and A2 for UHD and class B for HD sequences. Constant QP encoding is used with QP values of 22, 27, 32 and 37 according to VTM CTC. Reported rate distortion results are calculated as Bjøntegaard delta rate (BD-rate) [11][12] to evaluate the compression performance based on the following weighted average of peak signal-to-noise ratio (PSNR) and multiscale structural similarity measure (MS-SSIM) [13] values per color component:

$$PSNR_{YUV} = \frac{1}{8}\left(6 * PSNR_{Y'} + PSNR_{C_B} + PSNR_{C_R}\right),$$

$$MS\text{-}SSIM_{YUV} = \frac{1}{8}\left(6 * MS\text{-}SSIM_{Y'} + MS\text{-}SSIM_{C_B} + MS\text{-}SSIM_{C_R}\right).$$

For VVenC and x265, multi-threading with 8 threads has been enabled to generate results. For HM and VTM, multi-threading is not supported. All tests have been performed on Supermicro servers with Intel Xeon processors E5-2697A v4 @2.6GHz.

3.1 PSNR Optimized Use Case

The $PSNR_{YUV}$ BD-rate gain of VVenC over the HEVC test model reference software HM-16.24 is shown in Figure 1. $PSNR_{YUV}$ BD-rate represents the approximate average bit-rate savings between two encoders for the same objective quality ($PSNR_{YUV}$). Here lower values mean larger bit-rate savings with respect to the HM-16.24 anchor. Please note the logarithmic scale of the relative encoder runtime in comparison to HM-16.24.

With the slower preset and multi-threading enabled the BD-rate gain of VVenC over HM is similar to VTM-14.2 common test conditions (CTC), but a speedup of more than 17x for HD4K sequences is achieved over VTM.

With the faster preset and multi-threading the BD-rate gain over HM is still approx. 10.2%, with a speedup of more than 1300x for HD4K over VTM. Comparing the runtime with HM gives a speedup of around 180x.

As a good tradeoff between encoder runtime and BD-rate performance, we recommend the medium preset with multi-threading enabled. Here, the BD-rate gain over HM is approx. 33.2%, which is close to the slower preset and VTM CTC, but in comparison to VTM-14.2 the encoder runs 210x faster for HD4K sequences. Compared to HM-16.24 this is an encoder runtime speedup of 29x. A summary of all results is shown in Table III.

Table III: PSNR$_{YUV}$ BD-rate and multi-threaded (8 threads) encoder speedup for HD, UHD and both test sequences.

	HD			**UHD**			**HD4K**		
Preset	PSNR BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM	PSNR BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM	PSNR BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM
faster	-7.7%	170x	1200x	-12.4%	200x	1400x	-10.2%	180x	1300x
fast	-21.2%	94x	700x	-23.9%	110x	780x	-22.7%	100x	740x
medium	-31.5%	25x	180x	-34.5%	33x	240x	-33.2%	29x	210x
slow	-35.1%	8.7x	65x	-37.9%	12x	88x	-36.6%	10x	76x
slower	-38.6%	1.9x	14x	-41.4%	2.6x	19x	-40.2%	2.3x	17x

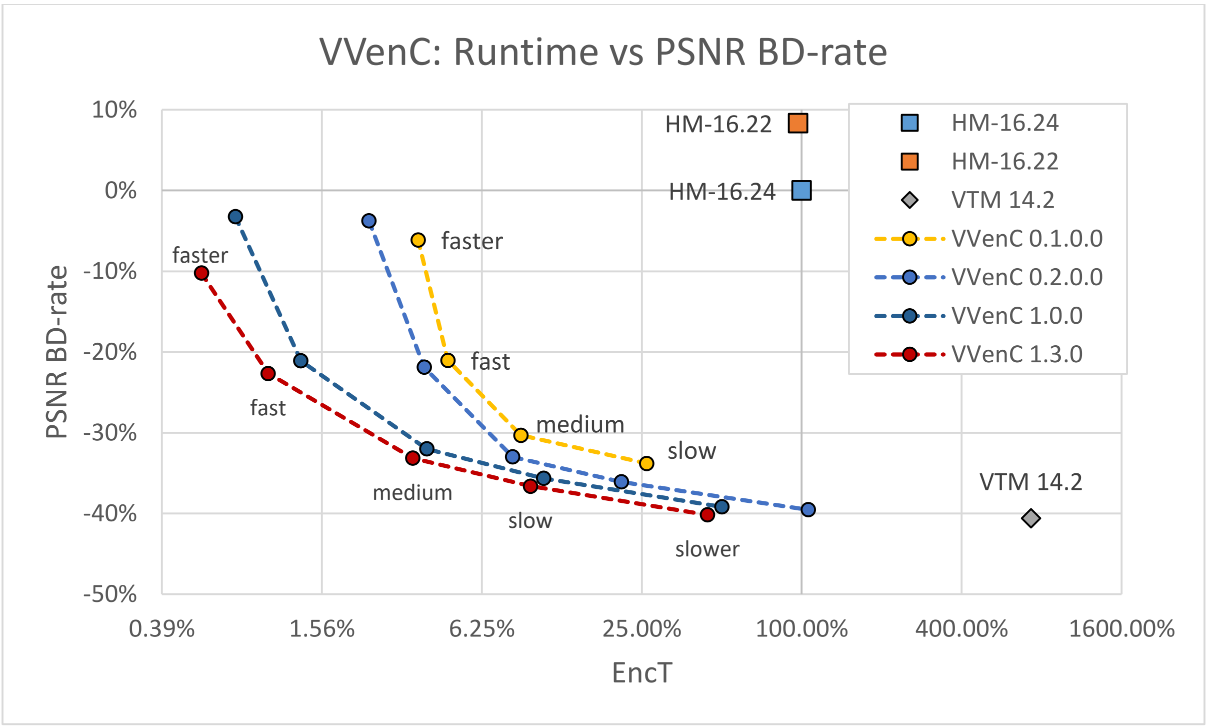

Figure 1: PSNR$_{YUV}$ BD-rate gain and relative encoder runtime for VVenC in comparison to HM-16.24 and VTM (JVET HD and UHD test sequences, MCTF enabled for HM-16.24 and VTM-14.2). Results are given for the 5 preset options: faster, fast, medium, slow and slower. VVenC is running multi-threaded using 6 threads for version <= 0.2 and 8 threads for version >=1.0. Lower PSNR$_{YUV}$ BD-rate values mean a better compression for the same objective quality in terms of PSNR$_{YUV}$.

Additionally, Figure 2 includes multi-threaded results for the HEVC open-source encoder x265 v3.4 at comparable speed presets [14]. For the comparison with VVenC, also x265 was configured to run with 8 threads. Besides sequence-specific parameters, the following parameter settings have been used:

```
--preset {0,1,2,3,…,9}  --tune psnr  --crf {17,22,27,32}  --keyint 1s  --min-keyint 1s
--profile main10  --output-depth 10
```

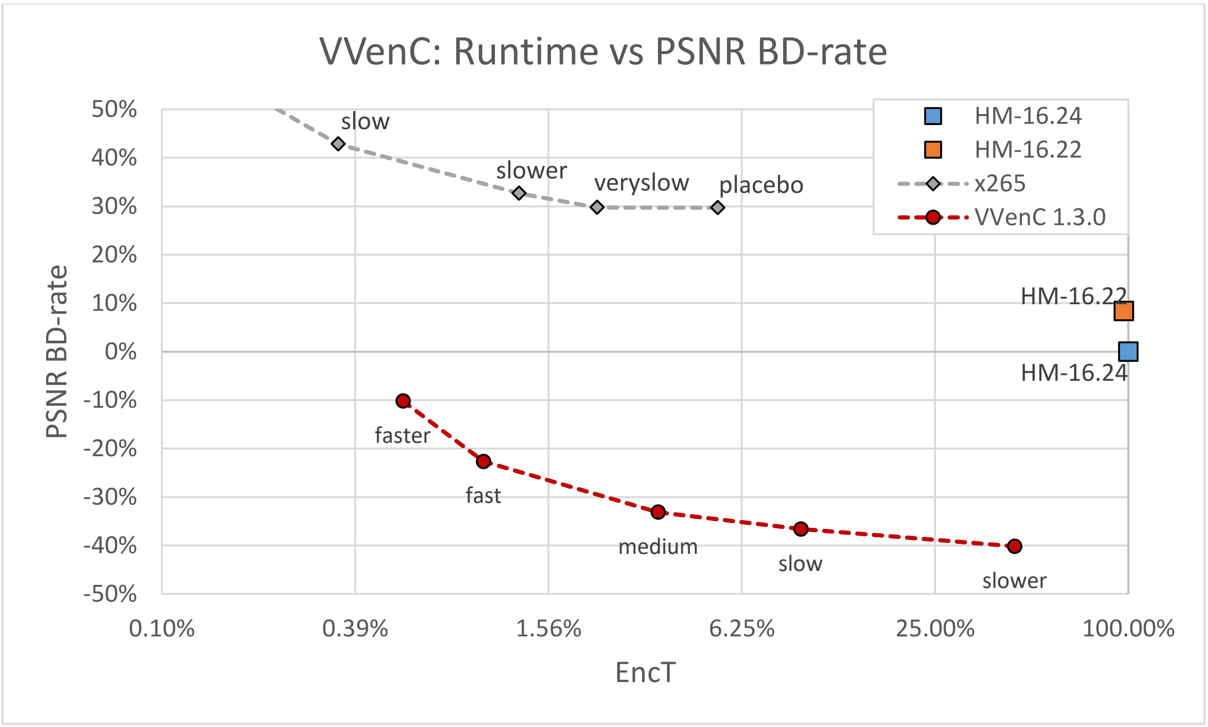

Figure 2: $PSNR_{YUV}$ BD-rate gain and relative encoder runtime in comparison to HM-16.24 for VVenC and x265 running with 8 threads. Lower $PSNR_{YUV}$ BD-rate values mean a better compression for the same objective quality in terms of $PSNR_{YUV}$.

3.2 Perceptually Optimized Quantization Parameter Adaptation

To improve the perceived (subjective) coding quality, VVenC supports a low-complexity quantization parameter adaptation (QPA) algorithm based on a model of the human visual system. To evaluate the quality of the perceptually optimized quantization parameter adaptation (PQPA), MS-$SSIM_{YUV}$ is used.

In Figure 3 the MS-$SSIM_{YUV}$ BD-rate gain of VVenC over the HEVC test model reference software HM-16.24 is shown (lower is better). For VTM simulation, common test conditions CTC with additional PQPA is used (--PerceptQPA=1). With PQPA enabled, the speedups achieved over HM and VTM are similar to the Non-PQPA results presented in the previous section. This demonstrates the low-complexity nature of the PQPA algorithm. Comparing the MS-$SSIM_{YUV}$ based BD-rates, additional bit-rate reduction can be achieved. Especially for the slower preset, an MS-SSIM BD-rate gain of more than 2% over VTM CTC without PQPA is realized. We recommend using the medium preset with multi-threading and PQPA enabled as a good tradeoff between encoder runtime and resulting perceived video quality. A summary of the MS-$SSIM_{YUV}$ results for PQPA is shown in Table IV.

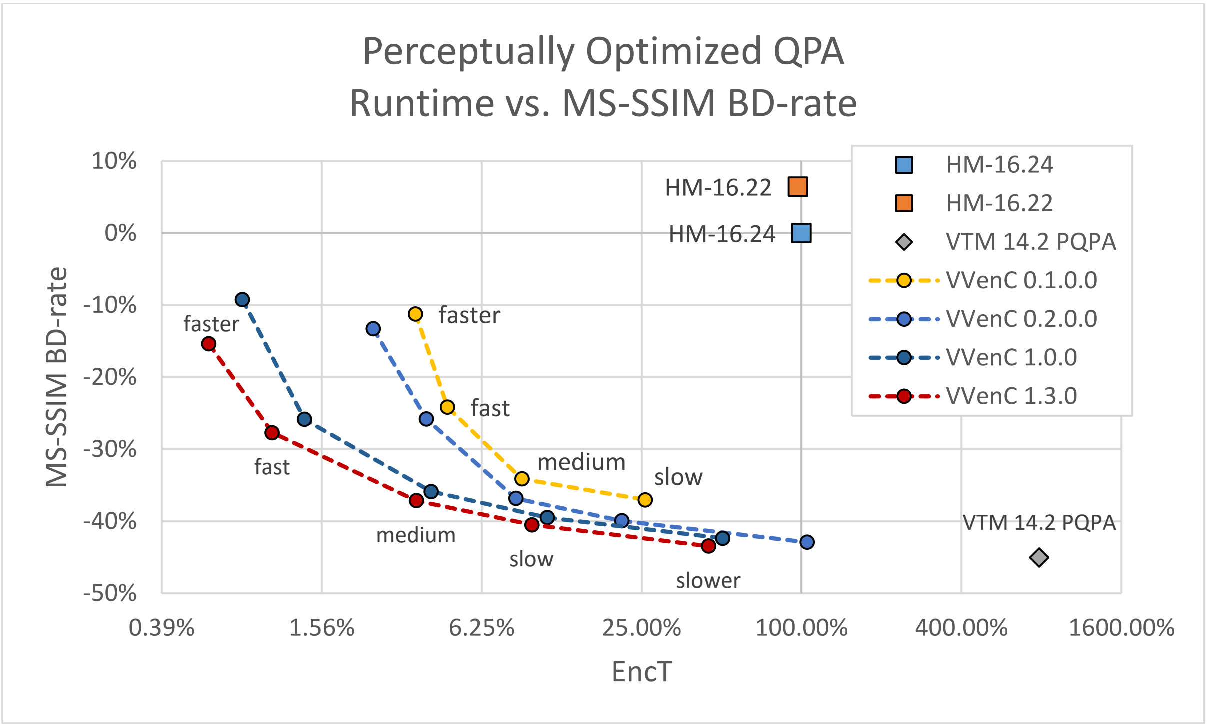

Figure 3: MS-SSIM$_{YUV}$ BD-rate gain and encoder runtime in comparison to HM-16.24 for VTM and VVenC with perceptually optimized quantization parameter adaptation enabled for HD4K sequences (MCTF enabled for HM-16.24 and VTM-14.2). VVenC results are given for the 5 preset options: faster, fast, medium, slow and slower. VVenC is running multi-threaded using 6 threads for version <= 0.2 and 8 threads for version >=1.0. Lower MS-SSIM$_{YUV}$ BD-rate values mean a better compression for the same quality in terms of MS-SSIM$_{YUV}$.

Additionally, Figure 4 includes multi-threaded results for the HEVC open-source encoder x265 v3.4 tuned for SSIM at comparable speed presets [14]. For the comparison with VVenC, also x265 was configured to run with 8 threads. Besides sequence-specific parameters, the following parameter settings have been used:

```
--preset {0,1,2,3,…,9}  --tune ssim  --crf {17,22,27,32}  --keyint 1s  --min-keyint 1s
--profile main10  --output-depth 10
```

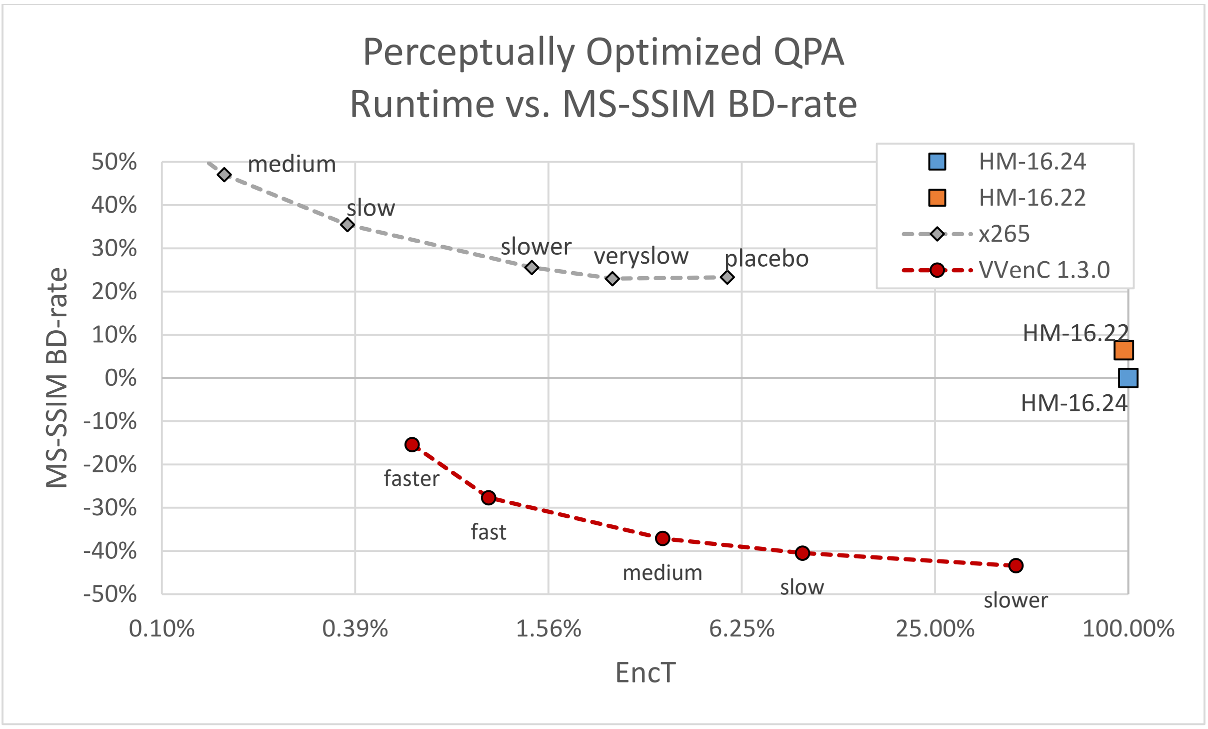

Figure 4: MS-SSIM$_{YUV}$ BD-rate gain and encoder runtime in comparison to HM-16.24 for VVenC with QPA enabled and x265 with --tune=ssim. Both VVenC and x265 are running with 8 threads. Lower YUV MS-SSIM$_{YUV}$ BD-rate values mean a better compression for the same quality in terms of MS-SSIM$_{YUV}$.

Table IV: MS-SSIM$_{YUV}$ BD-rate gain and multi-threaded encoder speedup for HD and UHD test sequences for VVenC with perceptually optimized QPA enabled.

	HD			**UHD**			**HD4K**		
Preset	SSIM BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM	SSIM BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM	SSIM BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM
faster	-16.1%	150x	1200x	-14.8%	190x	1500x	-15.4%	170x	1300x
fast	-28.3%	90x	700x	-27.2%	110x	840x	-27.7%	98x	770x
medium	-36.6%	24x	190x	-37.6%	32x	250x	-37.1%	28x	220x
slow	-40.0%	8.7x	67x	-40.9%	12x	95x	-40.5%	10x	81x
slower	-42.6%	1.9x	15x	-44.2%	2.6x	20x	-43.4%	2.2x	18x

3.3 Multi-Threading

In Figure 5 the multi-threading performance scaling is shown for the faster and fast presets for up to 16 threads and up to 32 threads for HD and UHD test sequences respectively (for version 0.3.0.0). Additionally, the scaling for the medium preset for up to 16 threads is given. The advanced threading since version 0.3 greatly improves scaling over earlier versions. It is apparent, that the threading performance is better with the fast and faster presets when utilizing more than 8 cores for HD or 12 for UHD.

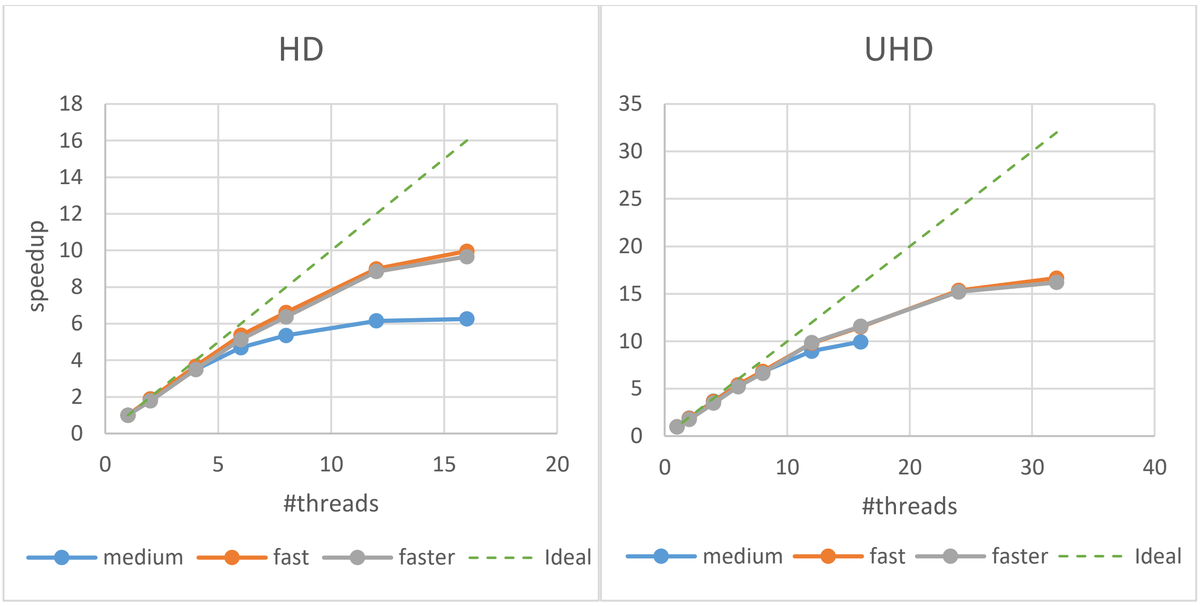

Figure 5: Multi-threading speedup dependent on the number of used threads for version 0.3.0.0. The tests were performed on high-performance computer with an AMD EPYC 7502P 32-core processor.

While the current multi-threading approach has only minimal influence on the compression performance of less than 1% bit-rate increase for the same visual quality, there are additional options allowing to significantly increase the multi-threading performance.

When operating the encoder in an area where the speedup curve saturates, the multi-threading performance can be improved by setting some additional parameters. For all presets --WaveFrontSynchro=1 can be enabled, which reduces the encoding lag of CTU lines within a picture. Additionally, for medium, slow and slower presets, --CTUSize=64 can be set, allowing it to achieve a scaling similar to the faster and fast presets. Both options are only available in the expert app and come with a gain trade-off. The latter option is enabled by default for the fast and faster preset. Since version 1.3.0, tile parallelism can be enabled by adding --Tiles=CxR to the full featured encoder command line or --tiles CxR to the smple encoder command line. *C* and *R* specify the number of tile columns and rows, respectively. Tile parallelism can be beneficial when the threading speedup saturates, especially for low CTU sizes and large resolutions, but it also comes at an efficiency trade-off.

3.4 RATE CONTROL

To support encoding with a predefined target rate instead of fixed QP in which the final bitrate is generally unknown beforehand, VVenC includes one- and two-pass rate control. The one-pass rate control uses limited information about the sequence and is intended for applications where encoder latency is a critical factor. On the other hand, two-pass rate control includes the first pass in which the statistics for the encoded sequence are collected. This information is then used in the second pass to improve the rate control performance at a cost of the encoding time increase.

PSNR_{YUV} BD-rate results of both rate control versions for HD4K content over a fixed QP VVenC encoding are shown in Table V. It should be noted that 10-second versions of CTC sequences from classes A1 and A2 are used for the rate control tests. The target rates for the rate control runs were set based on the resulting target rates from the fixed QP runs. All runs were executed using 8 threads. The BD-rate performance of the two-pass version is almost the same as that of fixed QP encoding. At the same time, the relative complexity of the two-pass approach is decreasing as presets become slower due to the constant complexity of the first pass. The one-pass approach achieves acceptable BD-rate performance while keeping the encoding runtime overhead low, especially for faster presets. The average bitrate deviation for one-pass approach is around 1.1%, while in the two-pass case it is around 0.6%.

Table V: $PSNR_{YUV}$ BD-rate and relative encoding runtime for 1- and 2-pass rate control on HD4K sequences in comparison to a VVenC fixed QP encoding for all presets. Encoders were running with 8 threads.

	HD4K					
	1-pass RC		Look-ahead RC		2-pass RC	
Preset	$PSNR_{YUV}$ BD-rate vs. Fixed QP	Encoding Time vs. Fixed QP	$PSNR_{YUV}$ BD-rate vs. Fixed QP	Encoding Time vs. Fixed QP	$PSNR_{YUV}$ BD-rate vs. Fixed QP	Encoding Time vs. Fixed QP
faster	7.43%	113%	0.50%	130%	-0.58%	141%
fast	7.61%	112%	1.20%	121%	-0.08%	128%
medium	6.76%	128%	1.55%	110%	0.37%	112%
slow	6.90%	132%	1.72%	106%	0.52%	109%
slower	6.85%	132%	1.67%	103%	0.65%	105%

For the improved perceived video quality, the rate control can be used in combination with PQPA. The $MS\text{-}SSIM_{YUV}$ BD-rate results for a combination of rate control and QPA over a fixed QP VVenC encoding with QPA are shown in Table VI. The results are similar to the results shown in Table V, apart from the higher BD-rates for one-pass rate control.

Table VI: $MS\text{-}SSIM_{YUV}$ BD-rate and relative encoding runtime for 1- and 2-pass rate control on HD4K sequences in comparison to a VVenC fixed QP encoding with QPA for all presets. Encoders were running with 8 threads.

	HD4K					
	1-pass RC		Look-ahead RC		2-pass RC	
Preset	$MS\text{-}SSIM_{YUV}$ BD-rate vs. Fixed QP with QPA	Encoding Time vs. Fixed QP with QPA	$MS\text{-}SSIM_{YUV}$ BD-rate vs. Fixed QP with QPA	Encoding Time vs. Fixed QP with QPA	$MS\text{-}SSIM_{YUV}$ BD-rate vs. Fixed QP with QPA	Encoding Time vs. Fixed QP with QPA
faster	16.93%	119%	1.21%	132%	-0.18%	145%
fast	16.73%	118%	1.26%	124%	0.50%	134%
medium	15.85%	123%	1.75%	113%	0.58%	114%
slow	15.96%	129%	1.87%	106%	1.07%	107%
slower	16.33%	132%	1.77%	103%	0.77%	103%

Experimental pre-analysis for single-pass rate control

VVenC 1.3.0 includes a new experimental algorithm for look-ahead based single-pass rate control, offering improved objective and subjective performance over VVenC's existing single-pass rate control method (see the results in Table V and Table VI). This new algorithm can be activated using --LookAhead=1 in the full feature application's command-line, e.g., as follows:

```
vvencFFapp options --TargetBitrate=1000000 --NumPasses=1 --LookAhead=1
```

The look-ahead duration used by this experimental mode is currently set to match the Intra period, causing excessive RAM consumption. Therefore, it is currently disallowed with long Intra periods.

4 CHANGELOG

4.1 v1.3.1

- Fixed tiles when QPA is enabled
- Fixed version prompt in the simple app, and aligned the prompts of simple and FF apps
- Added encoding for subpicture merging support

4.2 v1.3.0

- Combined quality improvements and speedup improvements for all presets
- Introduce string API to set encoder configuration parameters
- Perceptual QPA improvements for lower bitrates
- Reduce overall encoder memory consumption
- Change logging interface to support multiple encoder instances
- Enable setting of fractional framerate
- Write HRD timing information per default into the bit-stream
- Add CRA with constrained RASL encoding (cra_cre), previously --RPR=2
- Added tile support and tile parallelism
- Experimental: Add pre-analysis to improve single pass rate control (currently expert app only)

4.3 v1.2.0

- Major efficiency improvement for faster, minor speedups for fast and medium
- Implemented two-pass rate control encoding with separate encoder calls (storing the statistics data in a file)
- Added support for packed 10 bit YUV
- Improved single picture rate control
- Added IDR2 refresh type using IDR_W_RADL instead of IDR_N_LP in the first intra period
- Various improvements, cleanups and bugfixes

4.4 v1.1.0

- Speedups:
  - 20% for faster
  - 15% for fast
  - 7% for medium
- Added BCW
- Redefined the presets
- Improved SCC and 2-pass RC accuracy
- Various bugfixes and improvements

4.5 v1.0.0

- External interface rewritten to C
- New tool added: IBC
- Support for piped raw YUV input
- Improvements to rate control based on the XPSNR model
- Various bugfixes

4.6 v0.3.1.0

- Corrected compatibility with C++17
- Rate control cleanup (changes to external interface)
- Added vvencCfgExpert::m_log2MinCodingBlockSize interface variable
- Added Log2MinCodingBlockSize parameter to the expert app to restrict minimal allowed block size
- Fixed interaction between LMCS and ISP
- Fixed motion clipping in DMVR
- Reduced and defragmented memory usage
- Adapted and improved fast and faster presets
- Minor speed-ups to other presets

4.7 v0.3.0.0

- Minor bugfixes
- Preset refinement
- Added HDR parameter support
- Memory reduction
- Improved multi-threading
- Added a mode allowing for RPR switching with constrained drift
- Additional improvements and speed-ups
- New parameters: -aud, -vui, -hrd, -hdr (controlling the generation of appropriate parameter sets and optimized HDR settings)
- Changed parameter: --qpa (now only 0/1)
- Lib-interface changed (unified between the apps, now supporting all options available to the expert app)

4.8 v0.2.1.0

- Harmonization of QPA and 2-pass rate control
- Memory savings
- Minor bugfixes

4.9 v0.2.0.0

- Changed the license to modified 3-clause BSD
- Bugfixes to AUD and Decoded Picture Hash SEI generation

- New tools added: Intra Sub-Partitions (ISP), Transform Skip Residual Coding (TSRC), Block-level Differential Pulse Code Modulation (BDPCM)
- Added 2-pass rate control
- Added 1-pass rate control support for GOP32
- Using GOP32 as default in all presets
- Added slower preset
- New simple-app parameters: --refreshsec, --internal-bitdepth, --passes, --segment
- Using GNUInstallDirs
- Using Cmake versioning
- Added basic tests ("make test")
- Added address sanitizer support
- Many small speed-ups and improvements

4.10 v0.1.0.1

- Fix for SIMD setting using the simple-app

4.11 v0.1.0.0

- Initial release

5 REFERENCES

[1] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, "Overview of the Versatile Video Coding (VVC) Standard and its Applications," IEEE Transactions on Circuits and Systems for Video Technology, doi: 10.1109/TCSVT.2021.3101953, 2021.

[2] ITU-T and ISO/IEC JTC 1, Versatile Video Coding, Rec. ITU-T H.266 and ISO/IEC 23090-3 (VVC), Aug. 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] ITU-T and ISO/IEC JTC 1, High Efficiency Video Coding, Rec. ITU-T H.265 and ISO/IEC 23008-2 (HEVC), Apr. 2013 (and subsequent editions).

[5] A. Wieckowski et al., "VVenC: An Open And Optimized VVC Encoder Implementation," 2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 2021, pp. 1-2, doi: 10.1109/ICMEW53276.2021.9455944.

[6] VTM software repository, version VTM-14.2. Available online: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM

[7] VVdeC software repository. Available online: https://github.com/fraunhoferhhi/vvdec

[8] A. Wieckowski, C. Lehmann, B. Bross, D. Marpe, T. Biatek, M. Raulet, and J. Le Feuvre, "A Complete End-To-End Open Source Toolchain for the Versatile Video Coding (VVC) Standard," 29th ACM International Conference on Multimedia (MM'21), 2021. doi: 10.1145/3474085.3478320

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

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

[11] G. Bjøntegaard, "Improvement of BD-PSNR Model," Doc. VCEG-AI11 of ITU-T SG16/Q6, Berlin, Germany, July 2008. [Online]. Available: http://wftp3.itu.int/av-arch/video-site/0807_Ber/

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

[13] Z. Wang, E. Simoncelli, and A. C. Bovik, "Multi-Scale Structural Similarity for Image Quality Assessment," in Proc. IEEE Asilomar Conf. Signals, Systems, and Comp., Pacific Grove, 2003.

[14] x265 software repository, version 3.4. Available online: https://github.com/videolan/x265/tree/Release_3.4

6 LICENSE

Please see the file https://github.com/fraunhoferhhi/vvenc/blob/master/LICENSE.txt in the repository for the terms of use of the contents of the VVenC repository.

For more information, please contact: vvc@hhi.fraunhofer.de