VVenC
Fraunhofer Versatile Video Encoder

Jens Brandenburg, Adam Wieckowski, Tobias Hinz, Benjamin Bross
Video Coding & Analytics 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]. VVC is the successor of the High Efficiency Video Coding (HEVC) standard [2] and will be 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 VVC test model (VTM) [3] serves as a common reference implementation, i.e. a test bed for evaluation and verification of proposed technologies during standardization. While VTM used to be the only publicly available encoder and decoder implementation of the VVC standard, it is aimed at correctness, completeness and readability and should not serve as a real-world example of an VVC encoder and decoder.

The Fraunhofer Versatile Video Encoder (VVenC) development was initiated to provide a publicly available, fast and efficient VVC encoder implementation. The VVenC software is based on VTM, with optimizations including software redesign to mitigate performance bottlenecks, extensive SIMD optimizations, improved encoder search algorithms and basic 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 VTM-10.0 reference software decoder. Alternatively, the fast Fraunhofer Versatile Video Decoder (VVdeC) solution can be used [4].

Features at a Glance

- Easy to use encoder implementation with four predefined quality/speed presets.
- Optimized VVC encoder providing speedups between 22x and 270x over VTM-10.0 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 3.12 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/release-static/ subdirectory.

How to build for Linux/MacOS?

In order to compile the software for Linux, gcc-7.0 or higher and CMake Version 3.12 or higher are required. For MacOS Xcode and CMake Version 3.12 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/release-static/ subdirectory.

2.3 HOW TO USE VVENC?

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

How to use the standard encoder?

The standard encoder (vvencapp) can be used in one of four 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 [5]. 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>	-	Raw yuv input file
--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
--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	Select preset for specific encoding setting (faster, fast, medium, slow)
--qp <int>	32	Quantization parameter (0..51)
--bitrate <int>	0	Bitrate for rate control (0 constant QP encoding rate control off, otherwise bits per second). Rate control requires correct framerate.
--qpa <int>	2	Perceptual QP adaption (0: off, on for 1: SDR(WPSNR), 2: SDR(XPSNR), 3: HDR(WPSNR), 4: HDR(XPSNR))
--threads <int>	size <= HD: 4 else : 6	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 -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).

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
```

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.cfg randomaccess_fast.cfg randomaccess_medium.cfg randomaccess_slow.cfg	Random access configuration for different presets. Each configuration file corresponds to one of the 4 preset modes.
qpa.cfg	Perceptually optimized QPA configuration file.
gop32.cfg	Experimental. Additional GOP size 32 configuration, replacing the default GOP size 16 configuration. Must be given at the command line after the random access configuration file.
frc.cfg	Frame level rate control configuration, overriding default fix QP setup. Note: Currently incompatible with GOP 32.

3 ENCODER PERFORMANCE

The encoder performance tests focus on the most relevant HD (1920x1080) and UHD (3840x2160) resolution use cases with random access encoding as defined in the JVET common test conditions (CTC) [6]. 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) [7]. For the multi-threading use case 6 threads have been enabled in the encoder configuration. All tests have been performed on Dell Super Micro servers with Intel Xeon processors E5-2697A v4 @2.6GHz.

3.1 STANDARD USE CASE

The PSNR BD-rate gain of VVenC over the HEVC test model reference software HM-16.22 is shown in Figure 1. PSNR BD-rate represents the approximate average bit-rate savings between two encoders for the same objective quality (PSNR). Here lower values mean larger bit-rate savings with respect to the HM-16.22 anchor. Please note the logarithmic scale of the relative encoder runtime in comparison to HM-16.22.

With the slow preset and multi-threading enabled the BD-rate gain of VVenC over HM is similar to VTM-10 common test conditions (CTC), but a speedup of more than 38x for UHD and 22x for HD sequences is achieved over VTM.

With the faster preset and multi-threading the BD-rate gain over HM is still approx. 13.7%, with a speedup of more than 270x for UHD and 160x for HD respectively over VTM. Comparing the runtime with HM gives a speedup of 36x for UHD and 20x for HD.

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. 36.1%, which is quite close to the slow preset and VTM CTC, but in comparison to VTM-10.0 the encoder runs 110x faster for UHD and 69x faster for HD sequences. Compared to HM-16.22 this is an encoder runtime speedup of 14x for UHD and 8.5x for HD. A summary of all results is shown in Table III.

Table III: PSNR BD-rate and multi-threaded encoder speedup for HD and UHD test sequences.

PRESET	HD			UHD		
	PSNR BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM	PSNR BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM
FASTER	-11.0%	20x	160x	-15.9%	36x	270x
FAST	-25.5%	16x	130x	-28.8%	26x	200x
MEDIUM	-34.4%	8.5x	69x	-37.4%	14x	110x
SLOW	-37.9%	2.7x	22x	-40.6%	5x	38x

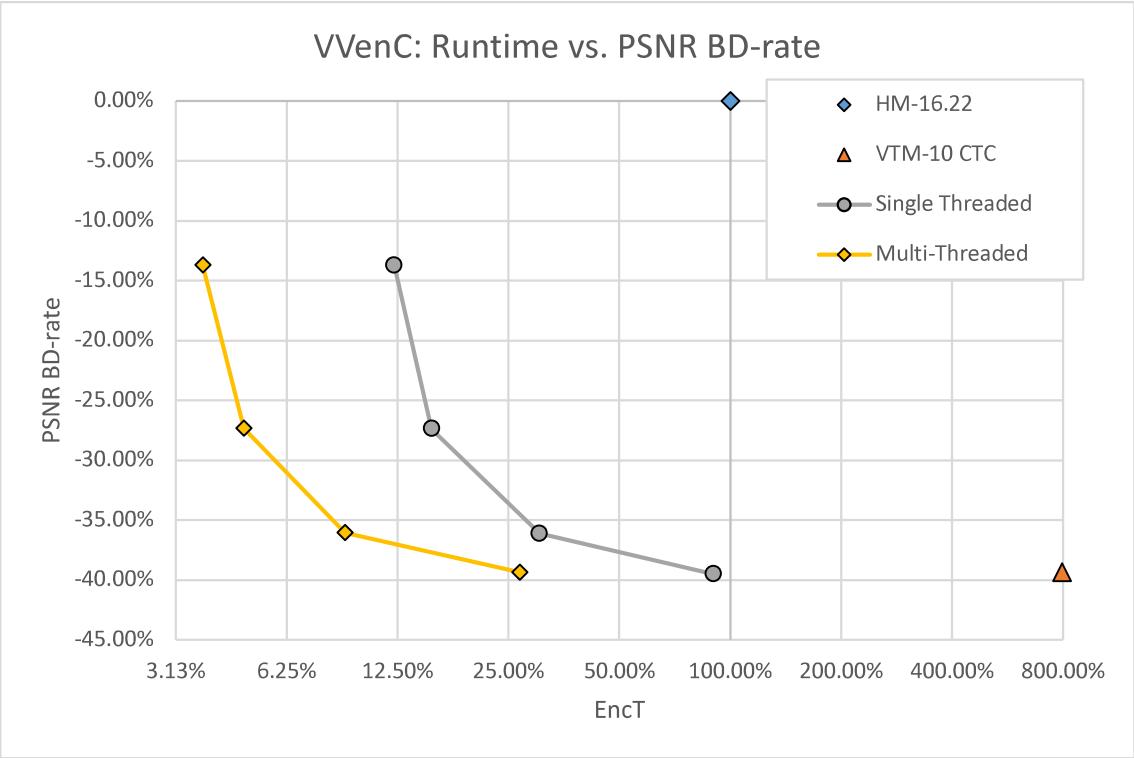

Figure 1: PSNR BD-rate gain and relative encoder runtime in comparison to HM-16.22 for VVenC. Results are given for the 4 preset options: faster, fast, medium and slow. Lower PSNR BD-rate values mean a better compression for the same objective quality in terms of PSNR.

3.2 Perceptually Optimized Quantization Parameter Adaptation

To improve the perceived (subjective) coding quality, VVenC supports a low-complexity quantization parameter adaption (QPA) algorithm based on a model of the human visual system. To evaluate the quality of the perceptually optimized quantization parameter adaption (PQPA), multiscale structural similarity measure (MS-SSIM) [8] is used.

In Figure 2 the MS-SSIM BD-rate gain of VVenC over the HEVC test model reference software HM-16.22 is shown (lower is better). With PQPA enabled the speedups achieved over HM and VTM common test conditions (CTC) 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 based BD-rates, additional bitrate reduction can be achieved. Especially for the slow preset, an MS-SSIM BD-rate gain of more than 3.7% over VTM CTC 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 results for PQPA is shown in Table IV

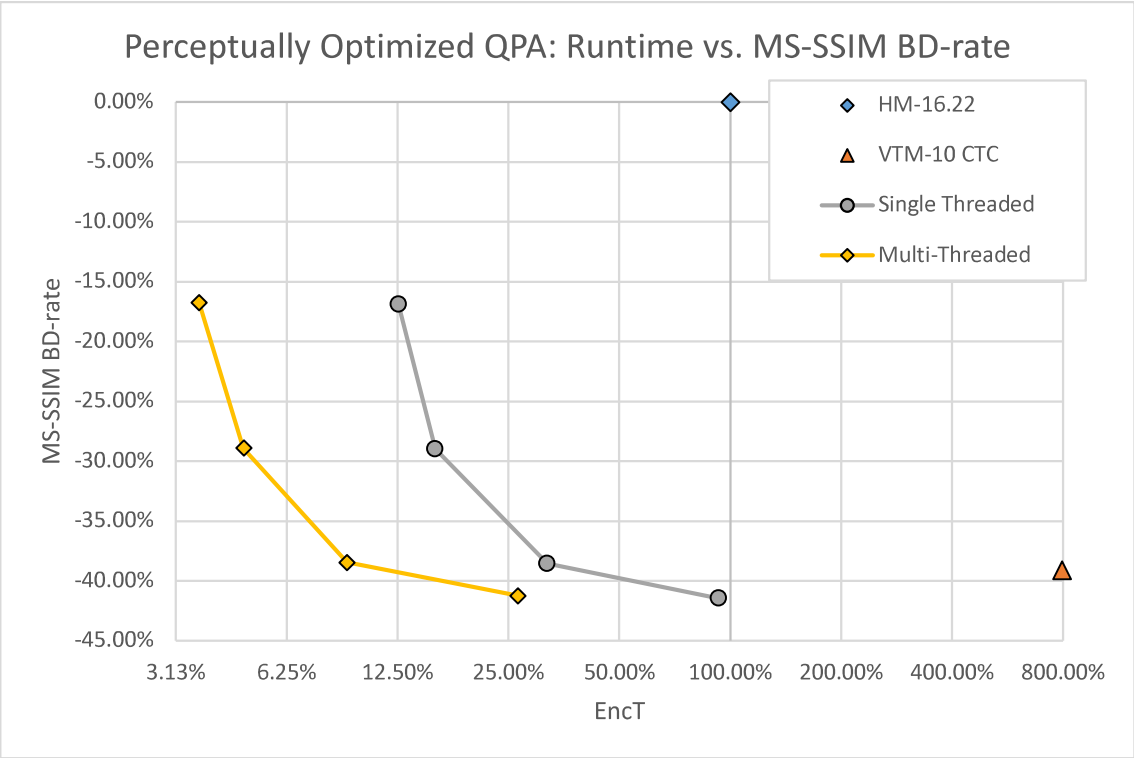

Figure 2: MS-SSIM BD-rate gain and encoder runtime in comparison to HM-16.22 for VVenC with QPA enabled. VVenC results are given for the 4 preset options: faster, fast, medium and slow. Lower MS-SSIM BD-rate values mean a better compression for the same quality in terms of MS-SSIM.

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

PRESET	HD			UHD		
	MS-SSIM BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM	MS-SSIM BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM
FASTER	-18.4%	20x	160x	-15.4%	36x	270x
FAST	-28.4%	16x	130x	-29.3%	26x	200x
MEDIUM	-37.5%	8.5x	69x	-39.2%	14x	110x
SLOW	-40.2%	2.7x	22x	-42.1%	5x	38x

4 REFERENCES

[1] B. Bross, J. Chen, S. Liu and Y.-K. Wang, "Versatile Video Coding (Draft 10)," document JVET-S2001, Joint Video Experts Team (JVET), Jul. 2020.
[2] 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.
[3] VTM software repository, version VTM-10.0. Available online: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM
[4] VVdeC software repository. Available online: https://github.com/fraunhoferhhi/vvenc
[5] J. Brandenburg et al., "Towards Fast and Efficient VVC Encoding", IEEE 22nd Workshop on Multimedia Signal Processing (MMSP 2020), Tampere, Finland, 2020.
[6] 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.
[7] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," Technical Report VCEG-M33, ITU-T SG16/Q6, Austin, Texas, USA, 2001.
[8] Z. Wang, E. Simoncelli, and A. C. Bovik, "Multi-Scale Structural Similarity for Image QualityAssessment," inProc. IEEE Asilomar Conf. Signals, Systems, and Comp., Pacific Grove, 2003.

5 LICENSE

Please see the file https://github.com/fraunhoferhhi/vvenc/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