
UPDATE ON OPEN OPTIMIZED VVC IMPLEMENTATIONS

VVENC AND VVDEC

JVET-AB0044

A. Wieckowski, J. Brandenburg, C. Bartnik, V. George, J. Güther,
G. Hege, C. Helmrich, A. Henkel, T. Hinz, C. Lehmann, C. Stoffers,
B. Bross, H. Schwarz, D. Marpe, T Schierl

Open optimized VVC implementations

Available on GitHub since Sep. 2020

- Fraunhofer Versatile Video Encoder (VVencC)

<https://github.com/fraunhoferhhi/vvenc>

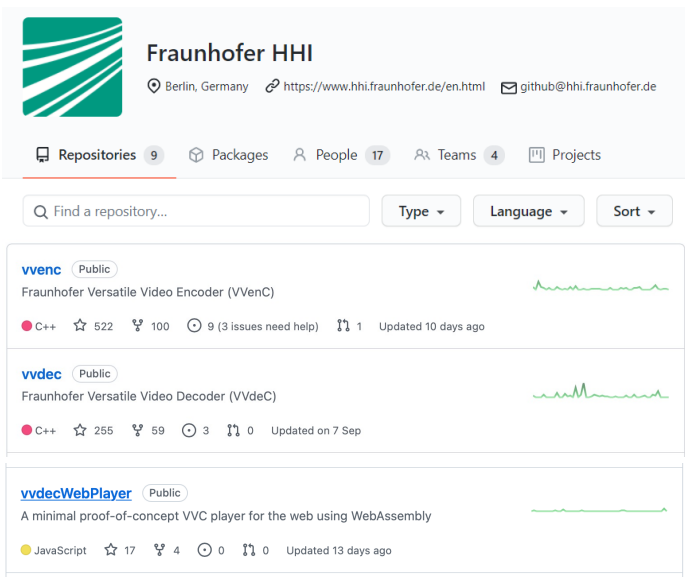
- Fraunhofer Versatile Video Decoder (VVdecC)

<https://github.com/fraunhoferhhi/vvdec>

- Fraunhofer VVdecC Web Player

<https://github.com/fraunhoferhhi/vvdecWebPlayer>

- Copyright BSD 3-clause clear license (since Mar. 2022)



VVenC v1.6.1

Main changes since July 2022 (v1.5.0):

■ Improved presets:

- Speedups: ~15% faster/fast/medium, ~5% slow (1.6.0), ~18% slower (1.6.1)
- Minor efficiency improvements (1.6.0)

■ Improved rate-control for 1pRC, high rates (1.6.0) and low rates (1.6.1)

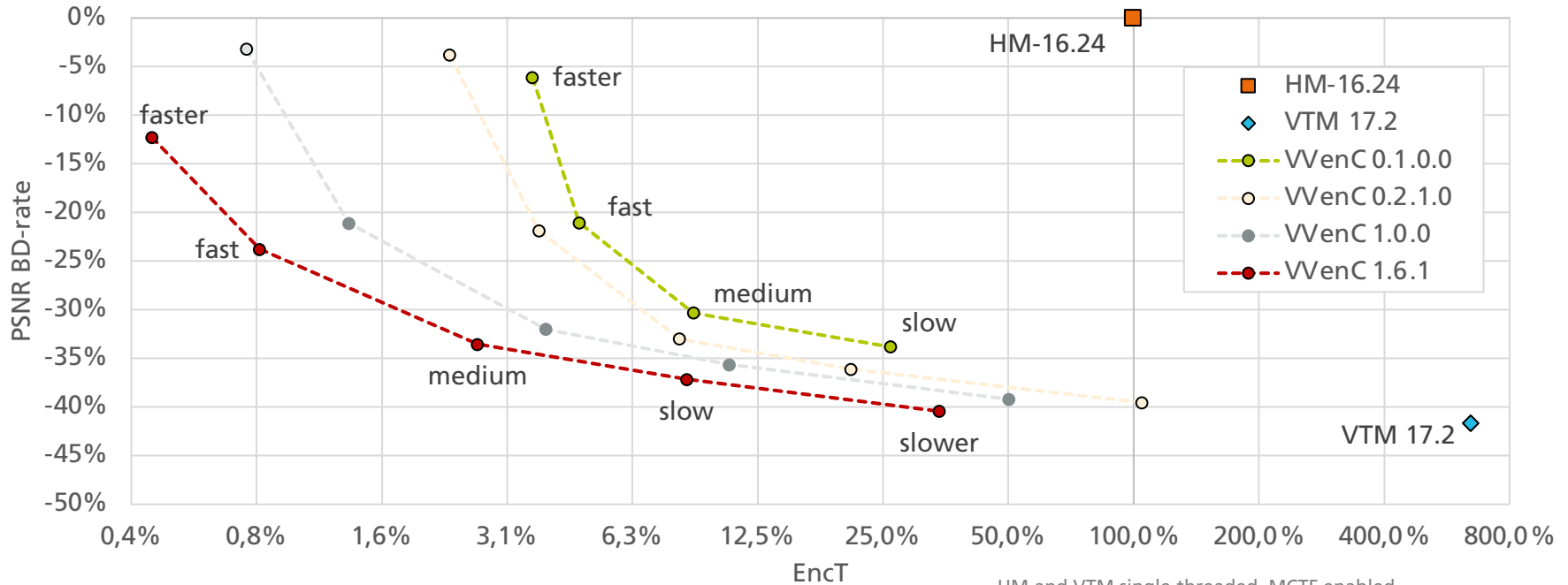
■ Fixes:

- Bugfix to multi-threaded encoding with tiles (1.6.1)
- Added auto-logic for correct ticksperssec setting (1.6.1)
- Made apputils an internal header only lib, fixing linking problems (1.6.1)

■ Various cleanups and improvements

VVenC

JVET CTC Performance – YUV PSNR for HD und UHD (PSNR optimized)

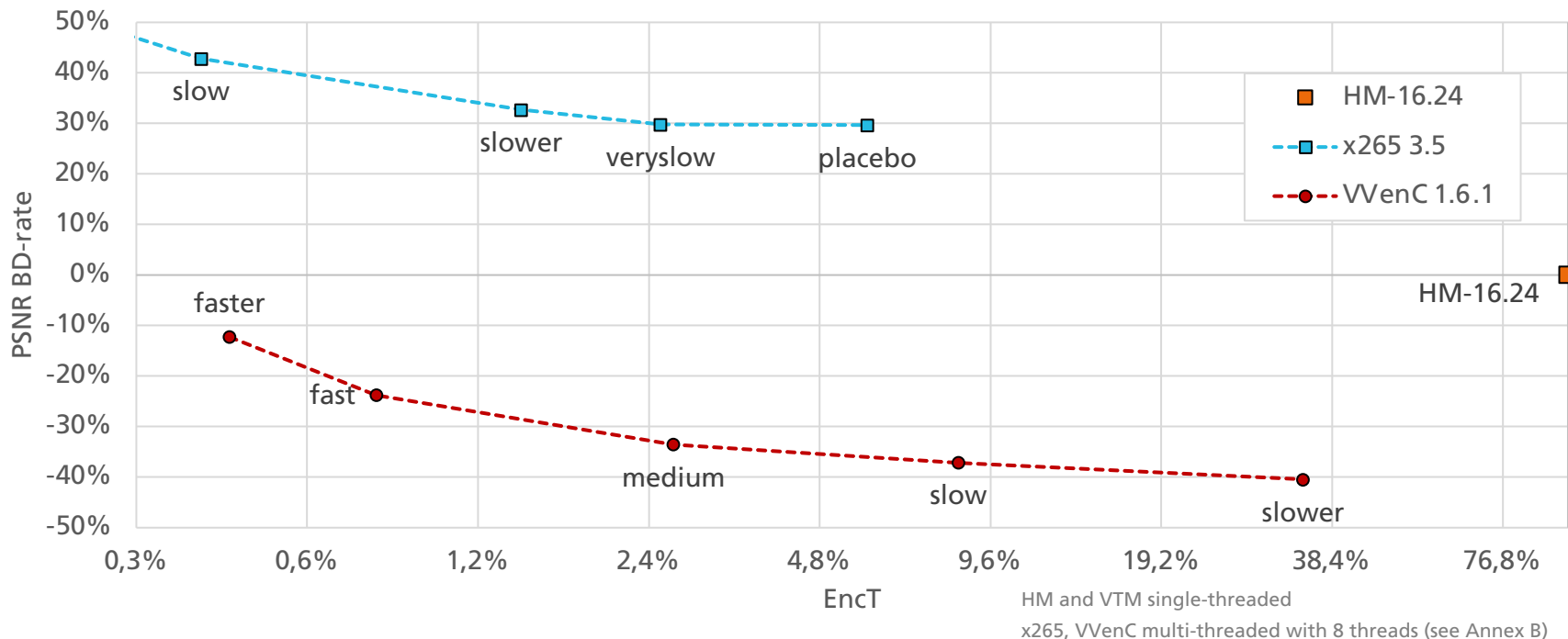


HM and VTM single-threaded, MCTF enabled

VVenC multi-threaded - 8 threads for 1.6.1 and 1.0, otherwise 6

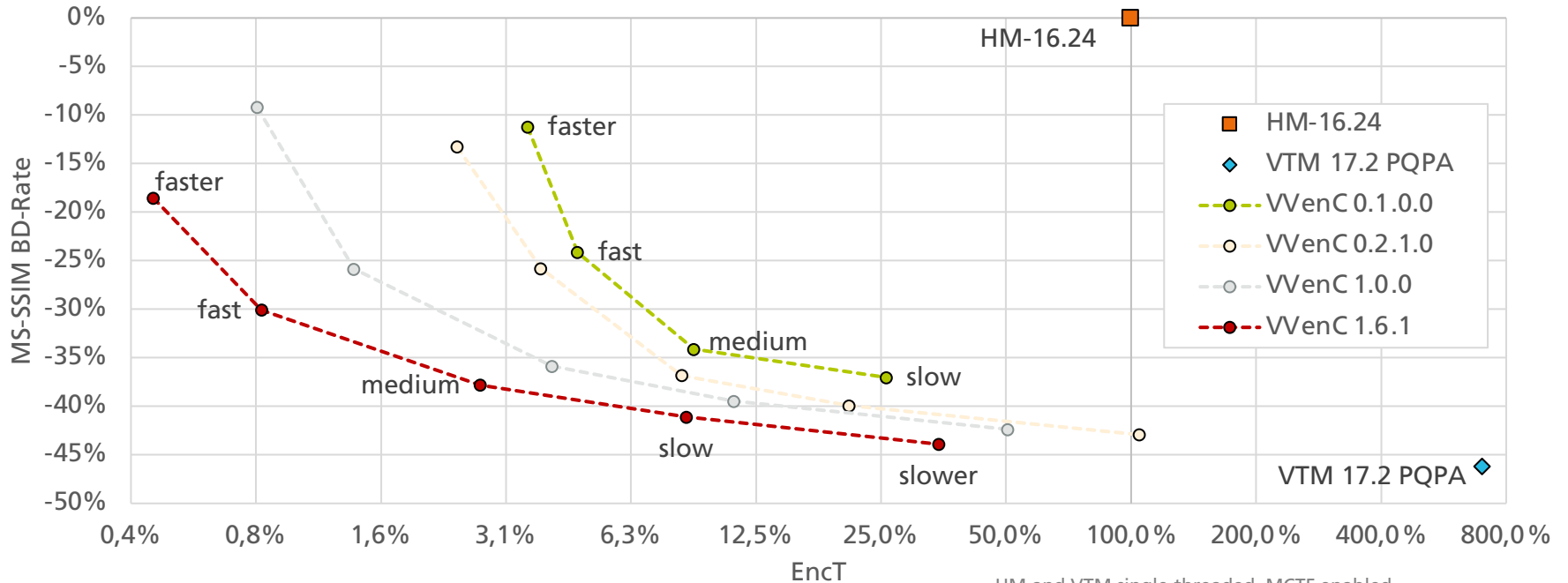
VVenC

JVET CTC Performance – YUV PSNR for HD und UHD (PSNR optimized)



VVenC

JVET CTC Performance – YUV MS-SSIM for HD und UHD (QPA)

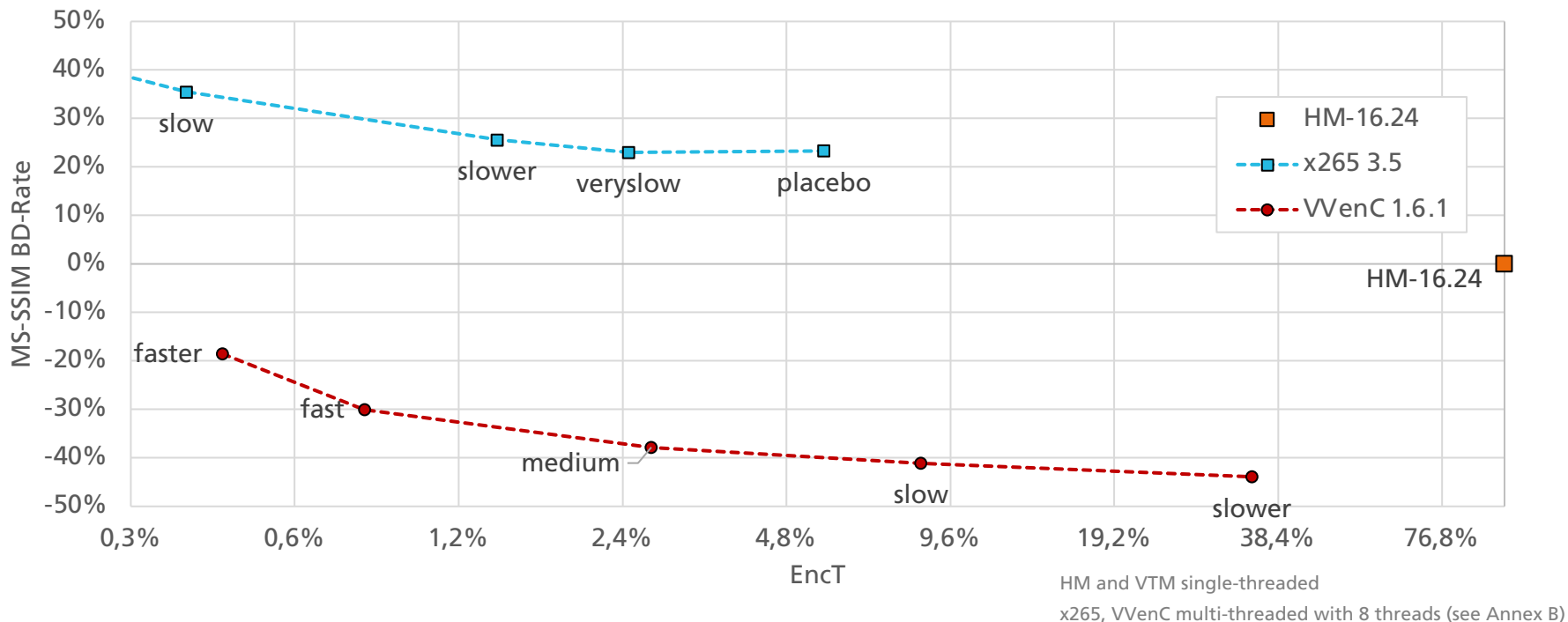


HM and VTM single-threaded, MCTF enabled

VVenC multi-threaded - 8 threads for 1.6.1 and 1.0, otherwise 6

VVenC

JVET CTC Performance – YUV MS-SSIM for HD und UHD (QPA)



VVdeC v1.6.0

Main changes since July 2022 (v1.5.0):

■ Various fixes and improvements

- Added an allocator API to reduce copying to application buffers
- Fixed conformance (VPS handling, SbTMVP)
- Improved build (reduced exported symbols, fixed Xcode build)
- Minor speedups and improvements

VVdeC

Android Playback

- aarch64 build of VVdeC on Android:
 - MPV media player
 - NEW: ExoPlayer
- 1080p60 10bit @ 3Mbit/s VVC coded video playback on Samsung
 - Galaxy 7+ (Snapdragon 865+)
 - Galaxy 8 (Snapdragon 8 Gen.1)
- 5h non-stop playback with energy optimized encoder settings as described in the ICIP 2022 paper*

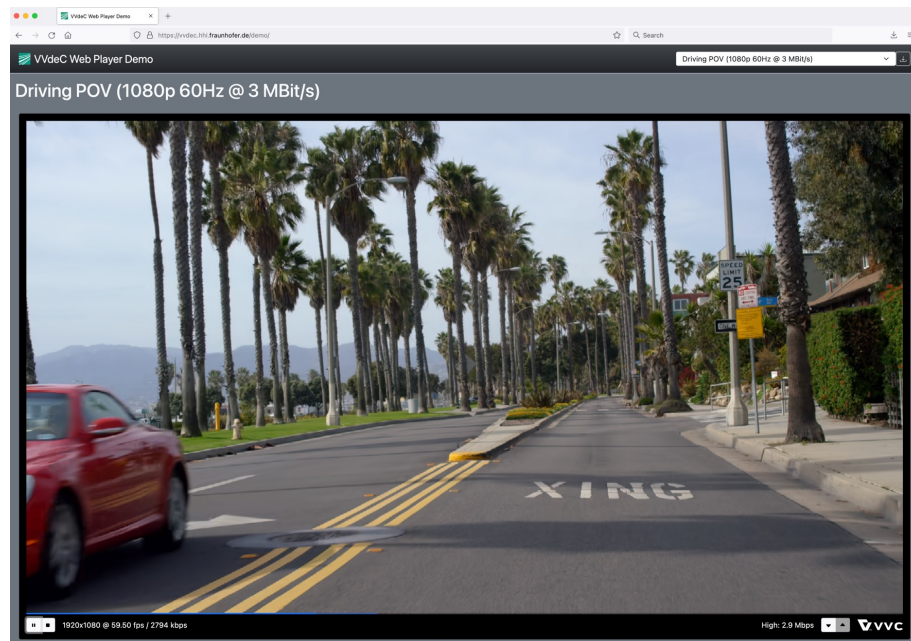
<https://arxiv.org/abs/2206.13483>



VVdeC

Browser Playback with WebAssembly

- Simple web player using
 - WebAssembly (WASM) build of VVdeC
 - WebGL for playback
 - MP4 support using mp4box.js
 - NEW: DASH support
- 1080p60 10bit @ 3Mbit/s playback in Edge, Firefox, Chrome on MacBook M1
- Sources available on [GitHub](https://github.com/fraunhoferhhi/vvdecWebPlayer):
<https://github.com/fraunhoferhhi/vvdecWebPlayer>



End-to-End Open-Source VVC Toolchain

Components used in a full VVC open-source toolchain

■ Encoding

- **VVenC** – standalone VVC encoder

■ Muxing – Transport – Demuxing

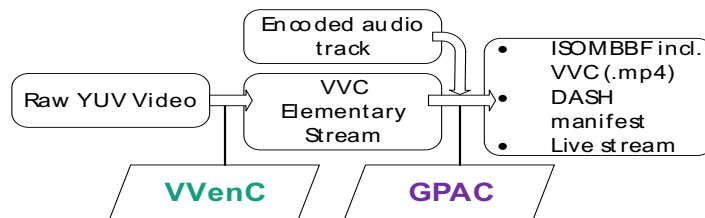
- **GPAC** – integration in upstream repo
- Supports open GOP resolution switching*
- VVC integration dependent on **FFmpeg** (**libavcodec**)

■ Decoding

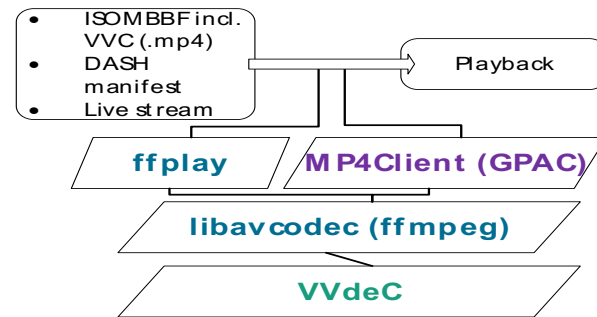
- **VVdeC** – library embedded in FFmpeg

■ Playback

- **FFmpeg** (**libavcodec**) – in a fork



a) Encoding/Muxing



b) Decoding/Demuxing/Playback

End-to-End Open Source VVC Toolchain

Where to find the and how to setup the SW

- **VVenC:** <https://github.com/fraunhoferhhi/vvenc>
 - *make install install-prefix=/usr/local*
- **VVdeC:** <https://github.com/fraunhoferhhi/vvdec>
 - *make install install-prefix=/usr/local*
- **FFmpeg:** <https://github.com/tbiat/FFmpeg>
 - *./configure --enable-libvvdec --enable-pic --enable-openssl --enable-libxml2*
 - *make -j && sudo make install*
- **GPAC:** <https://github.com/gpac/gpac>
 - *./configure --extra-ldflags='-Wl,-Bsymbolic' --extra-lldflags='-pie'*
 - *make -j && sudo make install*
- **ACM Open Access Paper:** <https://dl.acm.org/doi/10.1145/3474085.3478320>

Annex A – Additional resources

VVenC and VVdeC wiki pages

Most recent information on:

- How to build?
- How to use?
- Performance
- Publications

<https://github.com/fraunhoferhhi/vvenc/wiki>

<https://github.com/fraunhoferhhi/vvdec/wiki>

Home

Christian Lehmann edited this page 7 days ago · 46 revisions

Build	Usage	Performance
Compiling, installing and developing for VVenC	Using VVenC	Encoder Performance and comparisons

🔗 **VVenC: Fraunhofer Versatile Video Encoder**

VVenC is a fast and efficient VVC encoder implementation. After the release of the initial version in September 2020, performance and runtime are improved in each next version. The graph below shows the presets' development over time in multi-threaded operation (0.1-0.2: 6 threads, 8 threads for later versions).

Clone this wiki locally
<https://github.com/fraunhoferhhi/vvenc/wiki>

Home

Adam Wleickowski edited this page on 20 Apr · 51 revisions

Build	Usage
Compiling, installing and developing for VVdeC	Using VVdeC

🔗 **VVdeC: Fraunhofer Versatile Video Decoder**

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 has been published by ITU-T as H.266 and by ISO/IEC as MPEG-H 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 Decoder (VVdeC) development was initiated to provide a publicly available and fast VVC decoder implementation. The VVdeC software is based on VVC Test Model (VTM), with optimizations including software redesign to mitigate performance bottlenecks, extensive SIMD optimizations and extensive multi-threading support to exploit parallelization.

VVdeC can decode raw bitstreams created by any VVC standard compliant encoder, e.g. the

Clone this wiki locally
<https://github.com/fraunhoferhhi/vvdec/wiki>

Annex A – Additional resources

Publications

- J. Brandenburg et al., “**Towards Fast and Efficient VVC Encoding**”, IEEE 22nd Workshop on Multimedia Signal Processing (MMSP 2020), Tampere, Finland, 2020.
- B. Bross, C. Helmrich, A. Wieckowski “**Versatile Video Coding – Open Optimized Implementations**”, Workshop on the IEEE Picture Coding Symposium (PCS) 2021, Jul. 2021.
<https://youtu.be/IWPBuS2diVg>
- 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
<https://ieeexplore.ieee.org/document/9455944>
- C. R. Helmrich et al., “**A study of the extended perceptually weighted peak signal-to-noise ratio (XPSNR) for video compression with different resolutions and bit depths**”, in ITU Journal: ICT Discoveries, vol. 3, no. 1, May 2020.
<http://handle.itu.int/11.1002/pub/8153d78b-en>
- R. Skupin, C. Bartnik, A. Wieckowski, Y. Sanchez, B. Bross, C. Hellge, and T. Schierl, “**Open GOP Resolution Switching in HTTP Adaptive Streaming with VVC**,” 35th Picture Coding Symposium (PCS), Bristol, US, June-July 2021.
- R. Skupin, C. Bartnik, A. Wieckowski, Y. Sanchez, B. Bross, “**Constrained RASL encoding for bitstream switching**,” document JVET-W0133, Joint Video Experts Team (JVET), July 2021.

Annex B – Encoder comparison settings

Encoding with preset P for quality Q

- HD and UHD sequences from JVET common test conditions JVET-T2010:

https://jvet-experts.org/doc_end_user/documents/20_Teleconference/wg11/JVET-T2010-v2.zip

- Command line options for **x265*** encoder (no sequence specific parameters) tuned for PSNR (**T** = **psnr**) or MS-SSIM (**T** = **ssim**)

```
-D 10 --preset P --tune T --crf Q --keyint <1s> --min-keyint <1s> --profile main10 --  
output-depth 10
```

* optimal multithreading with x265 is achieved by restricting the number of CPU cores to be used to the desired number of threads (e.g. 8) since x265 determines the best combination of frame level and WPP parallel threads.