INSIGHTS INTO OPEN AND OPTIMIZED VVC IMPLEMENTATIONS

SVCP 2021

Presenter:

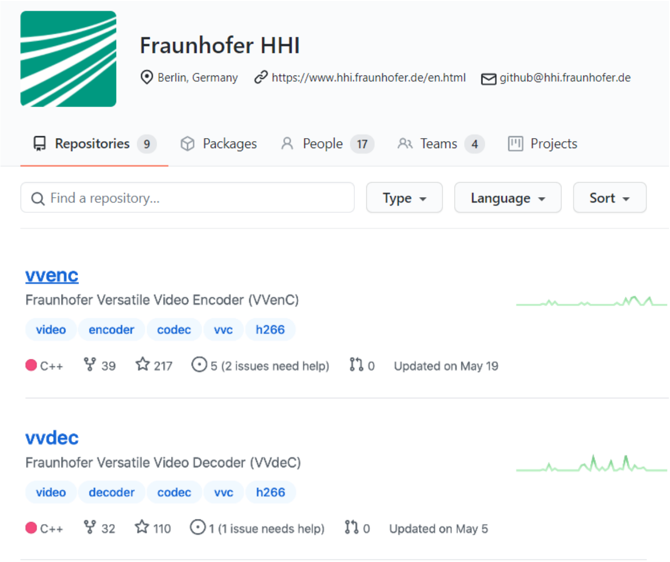
Adam Wieckowski
Fraunhofer HHI, Berlin, Germany

Fraunhofer HHI

VVC – Open, Optimized Implementations

Fraunhofer HHI developed optimized VVC software

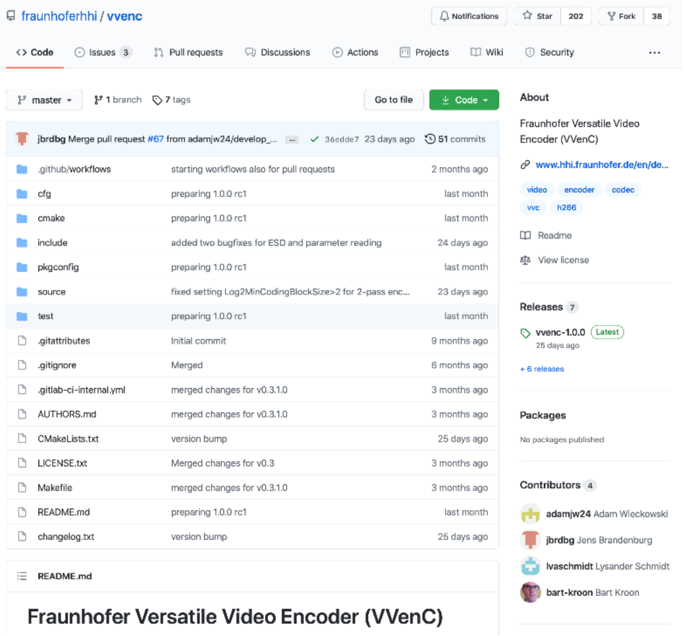
- Versatile Video Encoder (VVenC)
  - Goal: fast "real world" implementation while maintaining high coding efficiency (~VTM)
- Versatile Video Decoder (VVdeC)
  - Goal: enable 2160p60 10bit live decoding on a powerful multi-core CPU
- Source code on GitHub since Sep. 2020
- Copyright 3-clause BSD license since Dec. 2020



VVenC – At a Glance

Current version: v1.0.0 released in May 2021

- 5 predefined quality/speed presets:
  - faster, fast, medium, slow, slower
  - 16x to 1040x speedup over VTM (8 threads, UHD)
- “real world” features:
  - 1-pass and 2-pass VBR rate control
  - Subjective quality optimization
  - Multithreading
  - Simple easy to use C interface
  - Expert mode, VTM-style interface
- Happy to see first non-HHI contributions

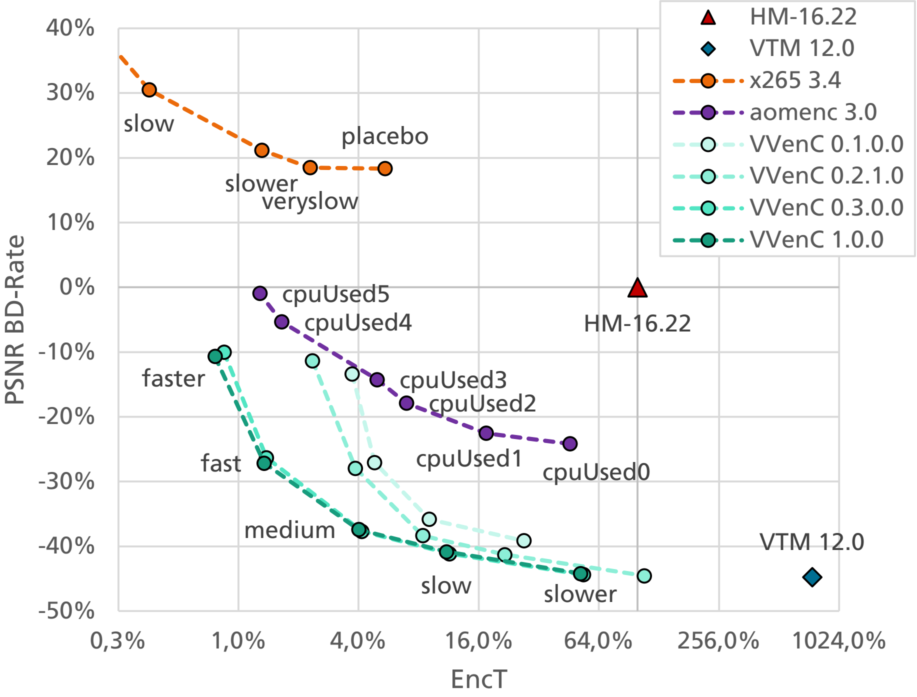


Fraunhofer HHI

VVenC – Multi-Threaded Results

Comparison to other state-of-the-art encoders

- VVenC 1.0.0 compared to:
  - HM-16.22: Over 40% BD-rate gains at 75% runtime up to 10% gains at 5% runtime
  - VTM-12: VVenC is much faster, keeps best coding efficiency plus pareto-optimal runtime-scaling
  - AV1 aomenc 3.0: VVenC has higher BD-rate at comparable runtime
  - x265 3.4: VVenc has significantly better BD-rate, catching up with runtime



See Annex A for detailed settings

Fraunhofer HHI

VVenC – Development Approach

Main objectives in VVenC development

- Implementation of usability features, e.g. rate control, subj. opt., and multithreading
- Improved implementation of the algorithms, including vectorization with SIMD
  - Mostly ported from VVdeC, but also encoder specific incl. fwd. Tr, MCTF and more
- Improved design of the search algorithms at various levels
  - Various fast strategies for most tools and tool combinations
  - Configuration space exploration for present derivation

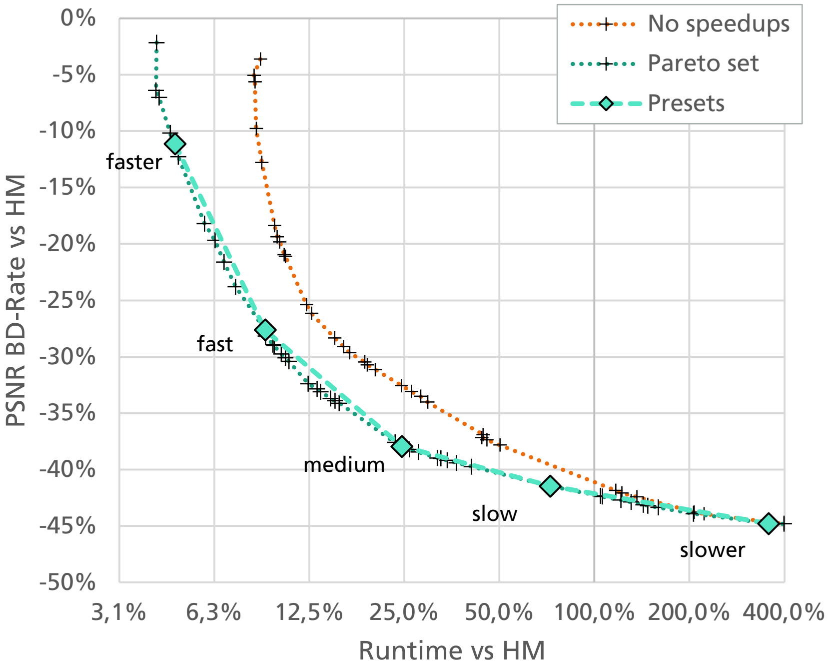
- Roadplan for future version
  - More usability features (making the encode more versatile)
  - Further speedups (both better impl. and algs.)
  - Improved compression performance, e.g. using encoding preanalysis

Fraunhofer HHI

VVenC – Preset derivation

VVenC configuration space exploration vs HM-16.22

- Iterative Pareto-Set approximation
  - Start at “HEVC”-like config
  - Next step based on “Tool-On” test
  - Both coding tools and speedups
- Pareto Set with and without speedups
  - 2x speedup up to around medium
  - Many tools with very good gain
- Less speedups towards slow and slower
  - Expensive last bit of efficiency
- ➢ In v1.1: make the starting point even faster!



Fraunhofer HHI

VVenC – Presets

"faster"

CTU64, QT44, BTT00

SAO, CCLM, TS (for SCC), TMVP

Deblocking, SH, implicit MTS, DMVR, BDOF

fast RDOQ, fast ME/partitioning, gradient based partitioning

"fast"

BTT10

Linear ALF, CC-ALF, Affine, AMVR, LFNST,

MCTF

"medium"

CTU128, *BTT21*

LMCS, DQ (~~SH~~), JCCR, MRL, MIP, SMVD, MMVD, SBTMVP, GPM

DBLF search opt, fast intra combinations

"slow"

BTT32

SBT, CIIP

AMVR, *GPM*

~~gradient based partitioning~~, ~~fast intra combi.~~

"slower"

BTT33

non-lin. ALF, expl. MTS (~~impl. MTS~~)

Affine, *ISP*, *MIP*, *SMVD*, *MMVD*

~~fast ME~~, ~~some fast partitioning~~

QT[X,Y], BTT[X,Y] max. tree depth [Intra,Inter]

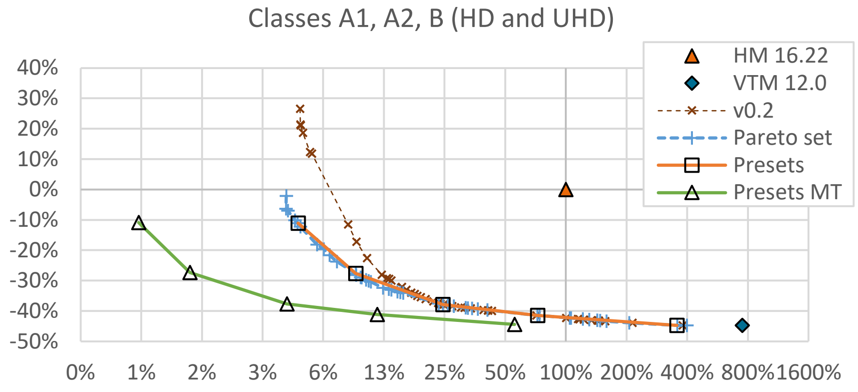
Coding tool

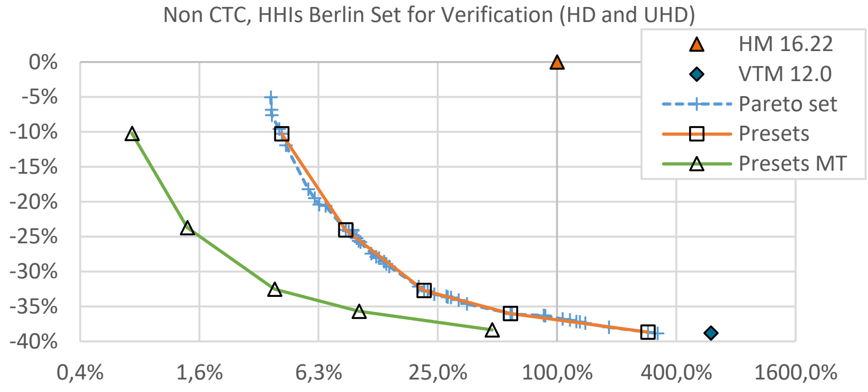
Implicit tool

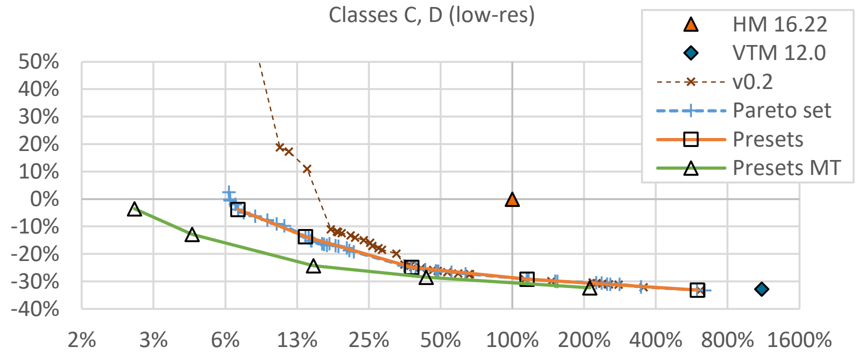
Encoder optimization

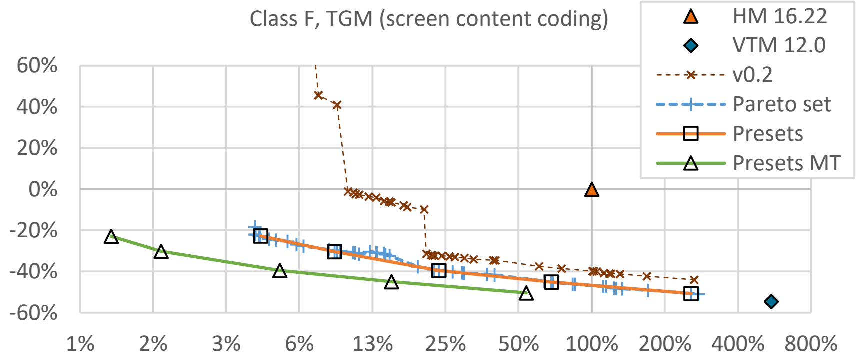
Fraunhofer HHI

VVenC – Preset performance for various use-cases





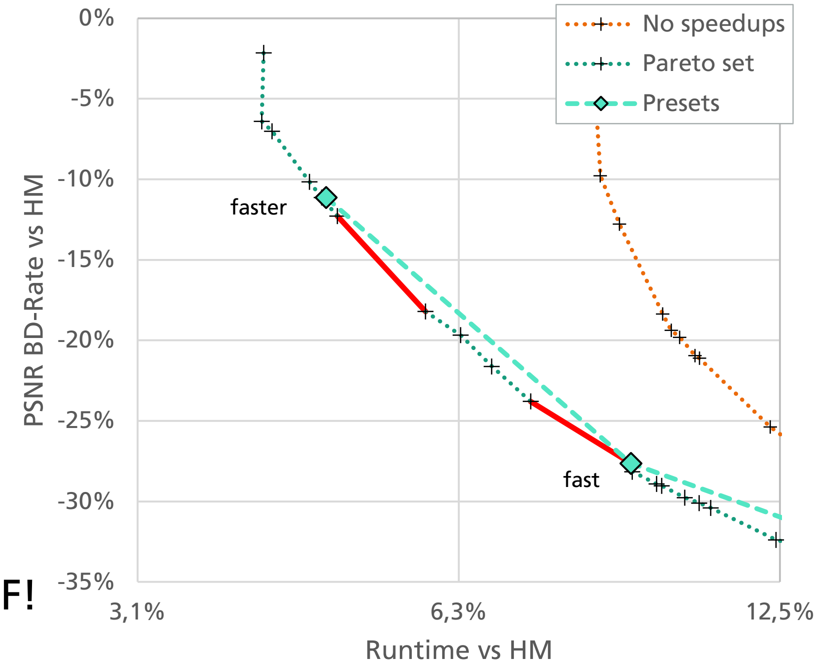




VVenC – Preset derivation towards v1.1

VVenC Pareto Set observations

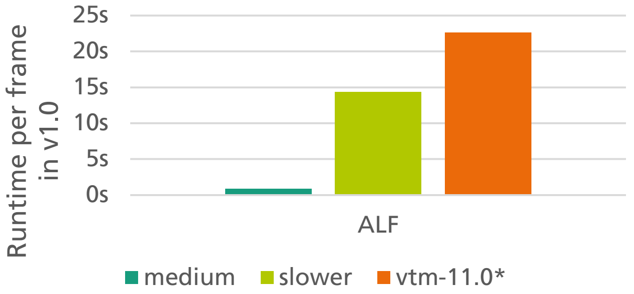
- The curve looks good
  - Overall convex characteristics
- Only 6 points between faster and fast
  - Two tools take 2/3 of the gain
  - Two tools take 2/3 of the runtime
  - ALF (w/o clipping) and MCTF (from VTM)
- Idea – split the tools up
  - Try get most of the gain
  - Minimize the runtime
- Side note – really big impact of ALF and MCTF!
  - In VTM almost no runtime increase

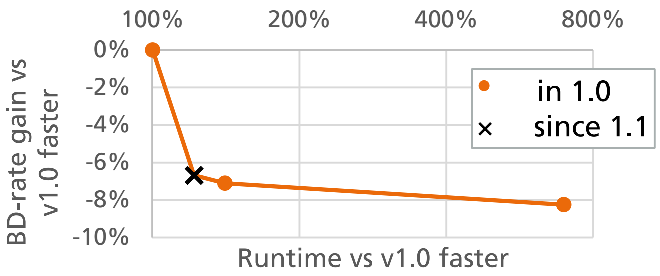


VVenC – Low hanging fruits 1

Tool deconstruction for Adaptive Loop Filter

- Typical optimization process
  - Define building blocks making out a tool
  - Test the parts independently
  - Isolate and optimize the indepedent code parts
  - Select optimal configurations for presets
- E.g. ALF working points on top of v1.0 faster
  - Full configuration: 8.3% BD-rate gain
  - No clipping (1/16 tests): 7.1% BD-rate gain
- ALF in v1.1 fast
  - Only ref frames (1/2 tests): 6.7% BD-rate gain

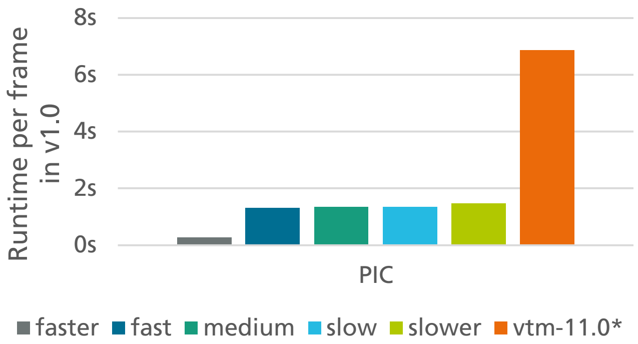


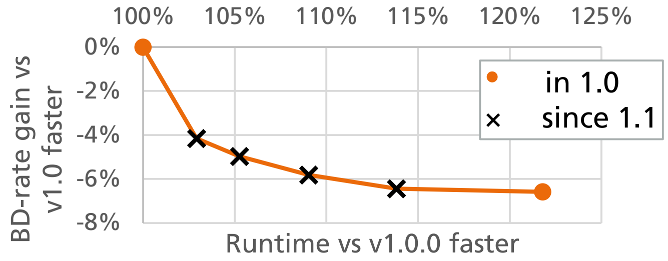


VVenC – Low hanging fruits 2

Tool deconstruction for Motion Compensated Temporal Filtering

- Motion compesated temporal filtering
  - Based on simplified motion search
  - Applied to frames with many references
  - Search up to 4 neighboring frames
- Deconstruction
  - Limit the number of frames applied
  - Limit the number of reference frames
- Results
  - 2/3 of the gain for 10% of the runtime

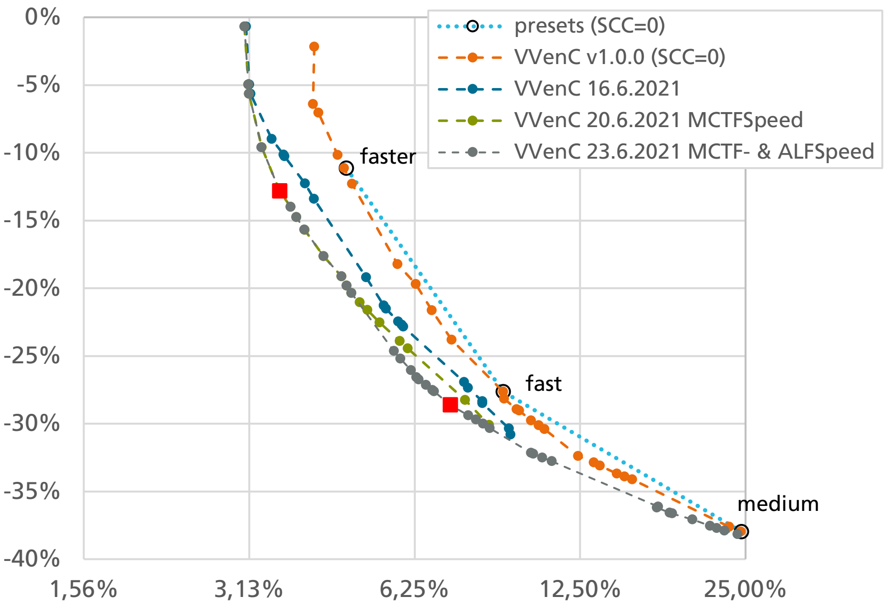


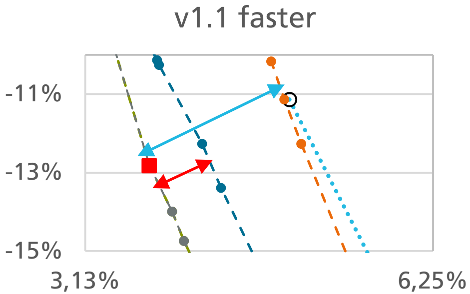


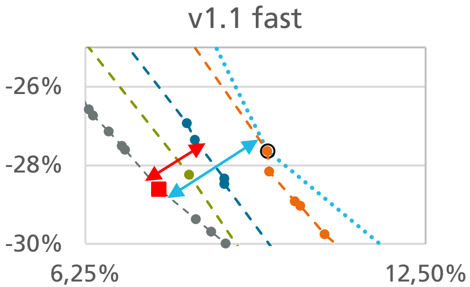
VVenC – Outlook for v1.1

New starting point and tools split-up

- Preliminary Pareto Set for v1.1
  - Improved starting point (blue line)
- Multiple ALF and MCTF working points
  - MCTF: faster and fast, ALF: fast
- Curves converge later, before medium
  - Still 7% faster than in v1.0
  - Speedup due to other factors
- Filter deconstruction impact
  - Versus improved starting point ↔
  - Versus old Pareto Set ↔
- ➢ Flip side: more options to optimize



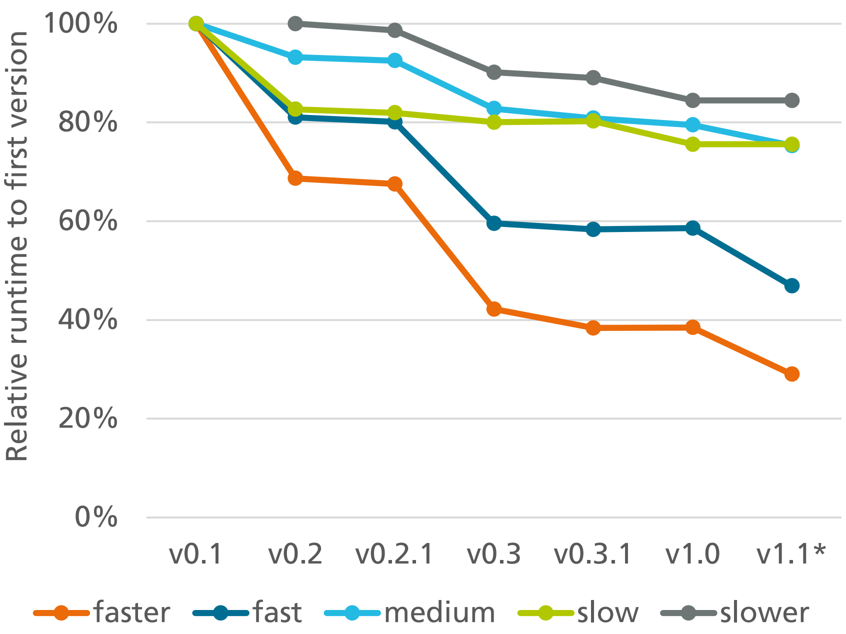




VVenC – Development history

Single threaded preset runtime development

- v1.1 to be released soon
- slower only added in v0.2
  - Only sped up by 15%
- Biggest improvement in faster and fast
  - Better starting points since v0.1
  - faster sped up by 70%
  - fast sped up by 53%
- v0.2: mostly about gains over v0.1
- v0.3: new starting point, CTU64, no MTT
- medium most optimized pre v0.1
  - Main focus in early development



VVenC – Conclusion

Insights into Open and Optimized VVC Implementations

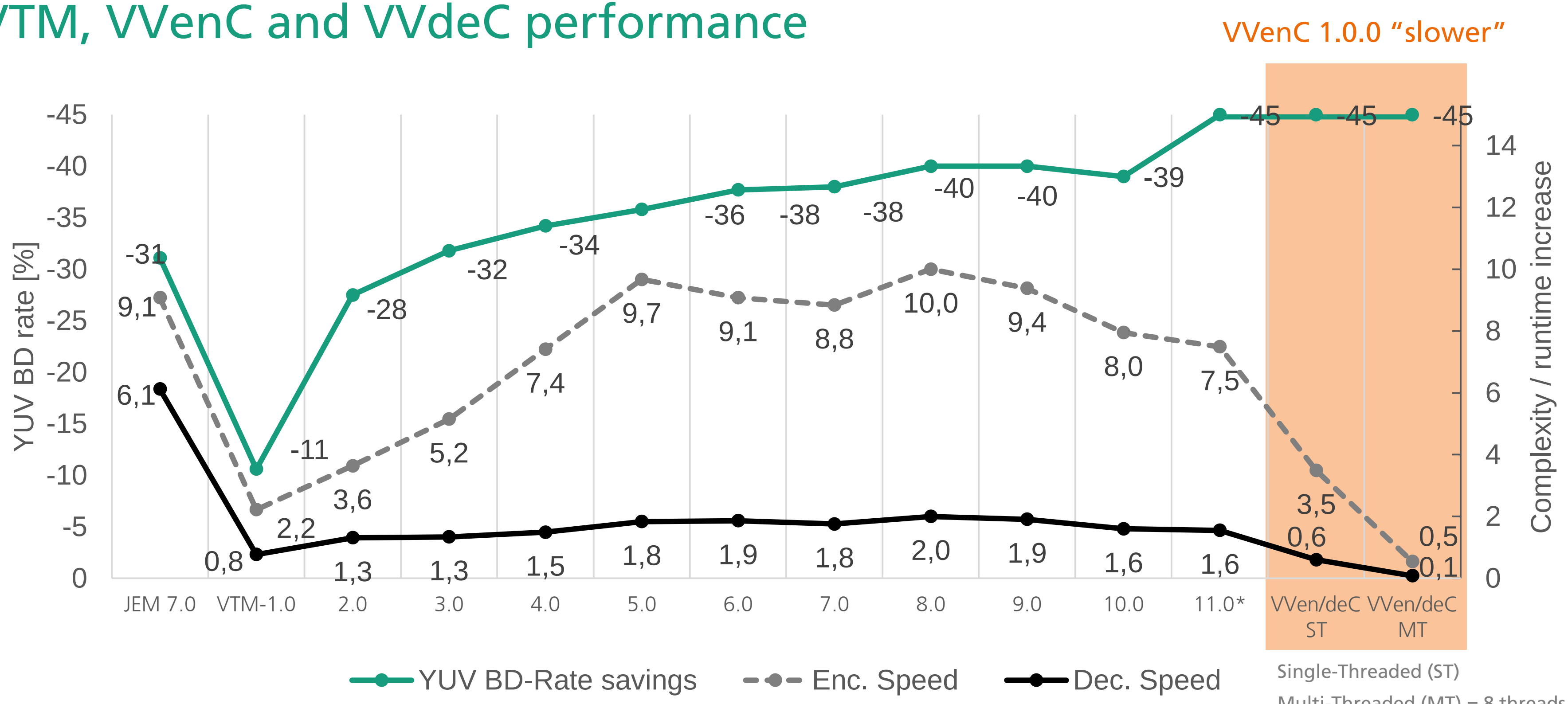
- VVenC (https://github.com/fraunhoferhhi/vvenc)
  - Open optimized VVC implementation available on GitHub
  - Single-slice, single-tile encoding
  - Optimized implementation and search algorithms
  - Subjective optimizations, rate-control & multithreading

➢ Stay tuned for v1.1 with further improvements to faster and fast

➢ Have a look at the x265 vs VVenC comparison and live decoding demo in 3IT!

VVenC – Conclusion

VTM, VVenC and VVdeC performance



Single-Threaded (ST)

Multi-Threaded (MT) = 8 threads

Fraunhofer HHI

Thank you for your attention!

adam.wieckowski@hhi.fraunhofer.de

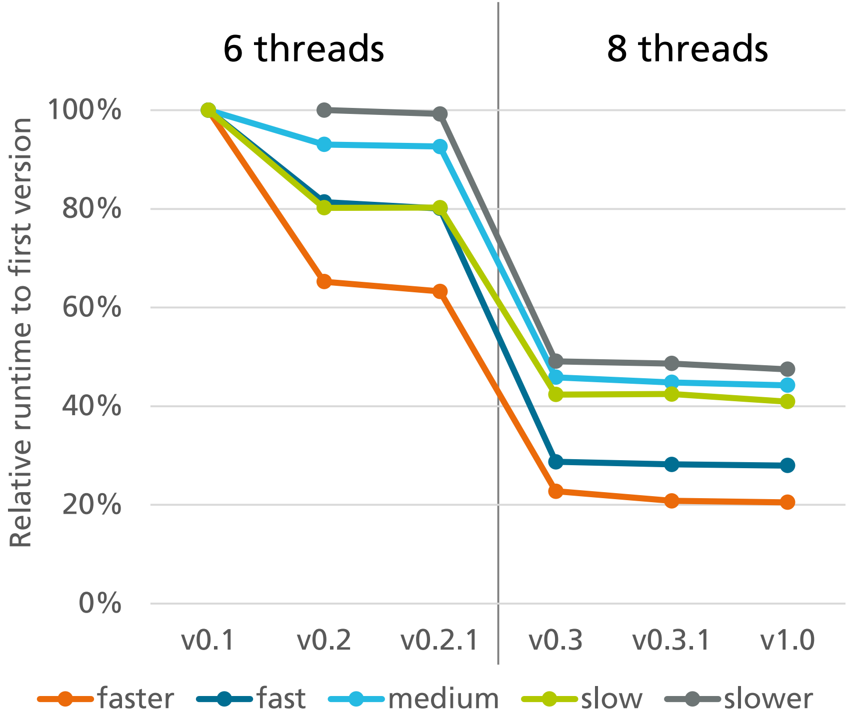
hhi.fraunhofer.de/vvc

Backup - Multithreading

VVenC – Development history

Multi-threaded preset runtime development

- v0.3 2-3x times faster than v0.2.1
  - With 33% more threads
  - Added frame parallelism
- Overall, in v1.0
  - At least 2x speedup since v0.1
  - faster, overall 4/5 runtime reduction
  - faster and fast with smaller CTU since v0.3 (more CTU lines)



VVenC – Multi-threading approach

- A combination of CTU-line and independent frame parallelism
- Task-based implementation with a single thread-pool
- 1 task per CTU, with following stages
  - CU search loop
  - LMCS and vertical deblocking
  - Horizontal deblocking
  - SAO filtering
  - 3 ALF stages: stats, filter derivation, application

- Dependencies
  - Checked by treadpool
  - Checked by tasks themselves
- Task can execute partially
- ➢ Automatic load balancing!
- ➢ Very good scaling!

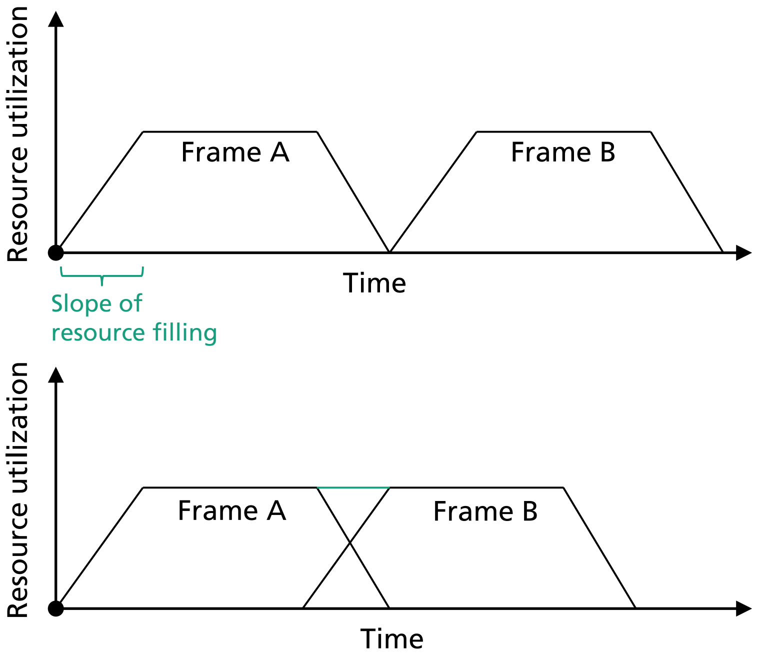
- Stats collection the most time-consuming step
- Final filter derivation requires stats for the whole picture, increasing latency if not parallelised

Fraunhofer HHI

VVenC – Multi-threading visualisation

Benefit of independent frame parallelization

- Less CTU tasks at the beginning and end of a frame
- "Slope" dependent on WaveFront usage
- Possibly not enough to fill 8+ cores
- ➢ "Overlap" indepdent frames
- In practice, just schedule all tasks at once
  - Automatic load balancing
  - 100% utilization until last frame finishes
- ➢ Downside: cannot be used without independent frames (low-delay)

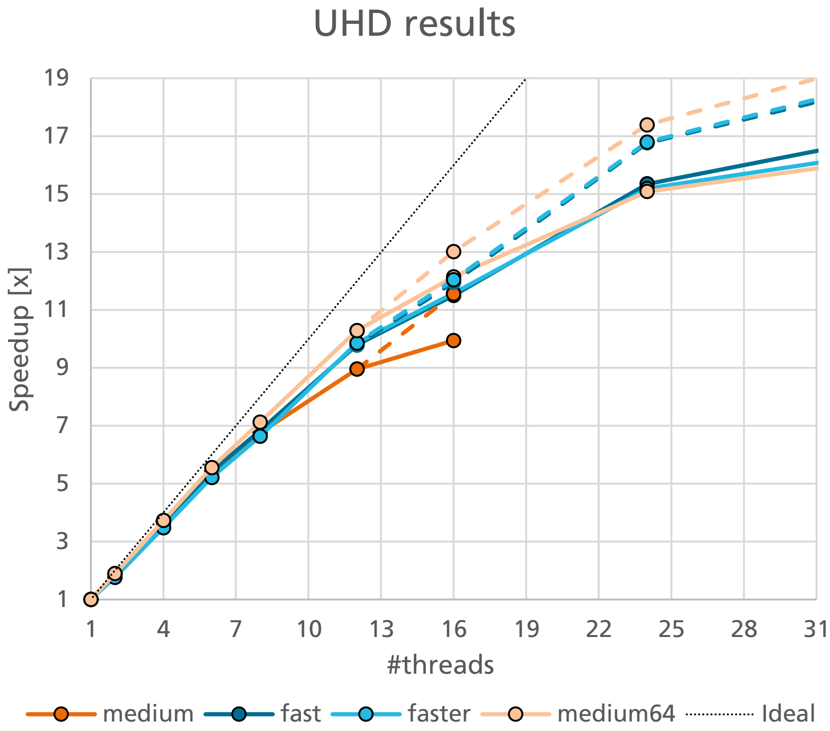


Fraunhofer HHI

VVenC – Multi-threading performance

Scaling depedent on preset and additional options

- Efficient multi-threading and scaling
- *fast* and *faster* have more CTU lines
  - Can efficiently utilize 20+ cores
- *medium*, *slow* and *slower* uses CTU128
  - Good utilization of up to 16 cores
- Relies on indepdent frames
  - ➢ Minimal efficiency impact!
- Can be improved with CTU64
- Can be improved with normative WPP (- - -)
  - Remove above-right CTU dep.



Annex A – 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 different encoders (no sequence specific parameters)
  - aomenc

    ```
    -cpu-used=P -passes=2 -cq-level=Q -kf-min-dist=<1s> -kf-max-dist=<1s> -end-usage=q -
    auto-alt-ref=1 -lag-in-frames=19 -threads=0 -bit-depth=10 -static-thresh=0 -drop-
    frame=0 -tune=psnr -q-hist=0 -rate-hist=0 -enbale-fwd-kf=1 -codec=av1 -deltaq-mode=0
    ```

  - x265

    ```
    -D 10 --preset P --tune psnr --crf Q --keyint <1s> --min-keyint <1s> --profile main10
    --output-depth 10 --frame-threads 1 --pools 0 --no-wpp
    ```