SVT-AV1 Enables Highly Efficient Large-Scale Video-On-Demand (VOD) Services

Last Updated: Jul 18, 2022

By Faouzi Kossentini – Senior Director of Video Processing Architectures & Solutions, Intel * Foued Ben Amara – Software Engineering Manager, Intel * Chekib Nouira – Principal Engineer, Intel * David Ronca – Director of Video Encoding, Meta * Ryan Lei – Software Engineer - Video Codec Specialist, Meta * Ioannis Katsavounidis – Research Scientist, Meta * Hassene Tmar – Technical Program Manager, Meta

INTRODUCTION

Modern Video-On-Demand (VOD) streaming services employ multiple encoded versions of the same visual content, at varying levels of quality/bitrates. This allows streaming clients to select the version which best fits the available connection bandwidth, resulting in smooth playback. Moreover, by switching among these different representations, stalling that could take place when there are changes in bandwidth is prevented. This technique is called Adaptive Bit Rate (ABR) streaming, which is at the core of all modern video streaming services. The collection of all such encoded versions is referred to as an “ABR ladder”, with each step of the ladder representing a higher quality/bitrate representation of the same video content. The requirement to support a large variety of streaming clients having different decoding capabilities results in the generation of multiple ABR ladders, with each ladder utilizing a different video coding standard. Older clients typically support only the H.264/AVC coding standard, which was first standardized in 2003, while newer standards, such as H.265/HEVC, VP9 and AV1, are supported mostly by newer clients. Moreover, some of these clients – such as large screen TVs – are using HW video

decoders, limiting any upgrade of their capabilities, while others – such as mobile handsets – have both HW and SW video decoding capabilities, which are offered through their HW-assisted multi-core CPUs and GPUs. The choice of which video coding standard to use for this purpose gets even more complicated due to the high energy required to perform video transcoding, which grows substantially in compute/energy consumption with each newer video coding standard. This, coupled with the tremendous growth in video usage, both in terms of people sharing videos and the number of videos they produce and consume, poses one of the biggest challenges to the video streaming industry today: How do we transcode so many videos in the most bandwidth-efficient way while also maintaining reasonable compute/power consumption levels?

To address the above challenge, we present a video transcoding solution that is based on Scalable Video Technology for AV1 (SVT-AV1), an AV1 encoder that was first open-sourced in February of 2019, and that was adopted in 2020 by the Alliance for Open Media as the productization encoder reference for the AV1 specification [1][2]. We first illustrate this solution through an end-to-end AVC premium video to AV1 transcoder that is used to generate a set of 8 bitstreams that can be used effectively by a VOD streaming service. In this solution, the Dynamic Optimizer (DO)-based two-pass SVT-AV1 video transcoder (described later) is used to transcode the ***ElFuente*** open-source Netflix 7min52sec video clip, which includes various levels of video content complexity, as depicted in Figure 1.

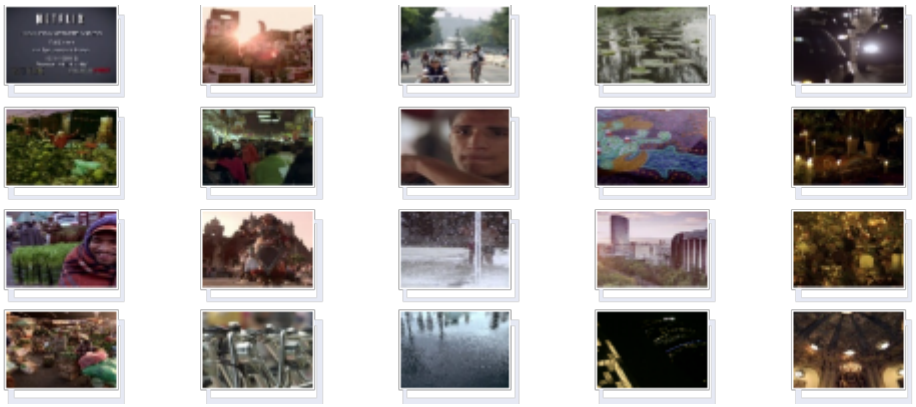


Figure 1. Snapshots of some parts of the 8-minute ElFuente video sequence.

To perform the scaling, encoding, decoding, and quality-evaluation computations, and to generate the final bitstreams, we used ***one*** FFmpeg static binary that is built with libsvtav1 v1.0 [3] and libdav1d v1.0 [4] that can be downloaded here [5]. Figure 2 shows the flow-diagram of the end-to-end transcoding process. The input video sequence is decoded from its initial format, a ~245Mbps CRF 0 encoded AVC 1080p 29.97fps bitstream, and the decoded video is then split into 140 shots that are 2-5 seconds in duration using the sample FFmpeg command line listed in Appendix I. Each video shot is then down-sampled to 8 different resolutions and encoded with SVT-AV1 preset ***M12*** to generate the convex-hull parameters, referred to as L1 in Figure 2, that would be used to encode the same video shots using the target SVT-AV1 preset ***M7*** encoder, referred to as L2 in the same figure. For each resolution, 11 QP/CRFs are used for encoding to produce bitstreams with different bitrates. The resolutions and CRF values used in L1 are shown in Figure 2, with a sample command line also provided in Appendix I. All 140 x 8 x 11 = 12320 bitstreams generated in L1 are decoded, upscaled, and the reconstructed shots are used to generate the (PSNR, bitrate) data. A sample command line is shown in Appendix I. Then, eight target (resolution, CRF) pairs are generated for every shot, yielding 1120 (resolution, CRF) pairs in total (140 shots * 8 quality-bitrate levels). The original 1080p shots are then re-encoded using SVT-AV1 at preset M7 for the identified (resolution, CRF) pairs. A sample command line of this

step can also be found in Appendix I. For each quality-bitrate level, the resulting 140 bitstreams are then combined into one bitstream using the sample FFmpeg command line shown in Appendix I. The full end-to-end transcoder takes 12 minutes and 54 seconds on a c6i.16xlarge AWS instance using 128vCPUs with 256GB of RAM of a dual Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz. Note that the encode part of the L1/L2 stages takes only 8 minutes and 14 seconds, that is, ~64% of the total end-to-end transcoding time. The resulting combined bitstreams have the bitrates, VMAF and VMAF NEG scores as shown in Table 1 below.

Combined stream quality target	bitrate (kbps)	VMAF	VMAF - NEG
Q1 - Target 100kbps	106	38.41665	36.918181
Q2 - Target 150kbps	160	47.77678	46.016472
Q3 - Target 250kbps	270	59.44893	57.419292
Q4 - Target 350kbps	376	66.21848	64.115841
Q5 - Target 500kbps	527	72.60436	70.479238
Q6 - Target 1000kbps	1038	83.70843	81.604857
Q7 - Target 1500kbps	1661	89.49798	87.558074
Q8 - Target 2500kbps	2582	93.36149	91.658397

Table 1. Resulting bitrates and VMAF scores for the eight combined bitstreams.

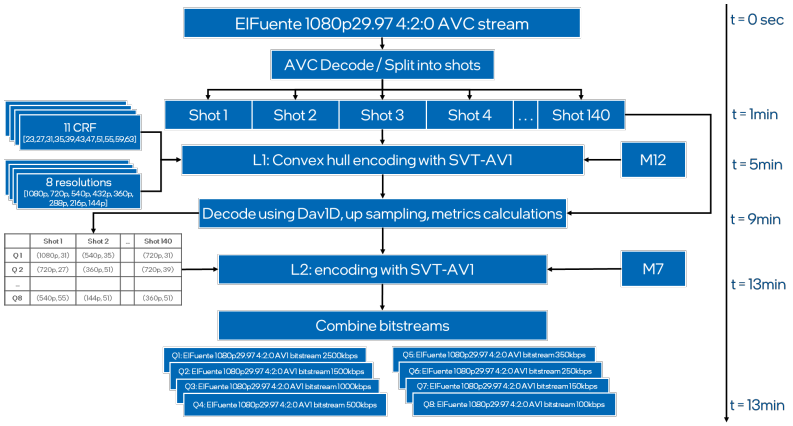


Figure 2. How to encode the 8-minute ElFuente using DO-based two-pass SVT-AV1 for VOD streaming.

We next present the components of the just-illustrated DO-based two-pass SVT-AV1 video

transcoding solution, which is shown to enable highly efficient large-scale VOD streaming services, as follows: an input video sequence is divided into multiple video shots, with each shot encoded using a fast preset in the SVT-AV1 encoder to generate the convex-hull (resolution, CRF) pairs for the shot, which is then re-encoded for each such pair using the target–preset SVT-AV1 encoder, in a way that achieves the best-possible overall quality-cycles tradeoffs.

SCALABLE VIDEO TECHNOLOGY AV1 (SVT-AV1) ENCODER

To address the fast-growing bandwidth requirements, continuous research and standardization efforts have been invested over the past two decades towards the development of newer video coding standards with higher compression efficiency. These standards include H.264/AVC [6], H.265/HEVC [7], VP9 [8], AV1 [9], and VVC [10]. In general, each new video coding standard yields ~30% savings in bandwidth as compared to the preceding standard. However, such savings would potentially come at the expense of a ~10x increase in encoder complexity, which has motivated the development of SW-efficient encoders such as x264 [11], x265 [12], libvpx [13], libaom [14], and VVENC [15], and various HW encoders. Through its excellent quality-complexity tradeoffs, and its support of 14 presets (M0-M13) that sweep the largest-ever complexity range (from libaom/VVC to x264-veryfast complexity levels), SVT-AV1 [16] not only breaks the AV1 complexity barrier, but also enables an efficient solution for what would-otherwise be a costly VOD streaming service.

The SVT-AV1 encoder architecture and features are discussed in detail in two SPIE papers [16, 17]. SVT-AV1 is highly scalable, with respect to multi-core CPU utilization, video input resolution, and encoding complexity. The enablers of SVT-AV1's elevated levels of scalability are its process- and segments-level parallelism, multi-path/multi-stage structures, and highly efficient algorithms. More specifically, SVT-AV1's **lossless** parallelism can be achieved through a process-based encoder pipeline and intra-picture segments: (i) independent segments such as in motion estimation, source-pixel-based statistics generation, temporal filtering, Constrained Directional Enhancement Filter (CDEF) parameter search and loop restoration filter parameter search, and (ii) dependent segments such as in the Temporal Dependency Model (TPL), mode decision, normative encoding, CDEF and loop restoration filtering. To maximize memory efficiency and reduce latency, SVT-AV1 creates and executes segment-based threads for each compute-intensive encoder module as soon as possible. This is performed in all independent pictures within the current and any future mini-GOPs. Moreover, SVT-AV1 features multi-path/multi-stage structures for block-based selection of the partitioning and coding modes. Through the multi-path structure, different partition/blocks are encoded at different levels of accuracy, depending on their visual importance levels. Through the multi-stage structure, the number of mode-decision candidates is pruned and reduced through consecutive stages. Moreover, many algorithmic optimizations in temporal filtering, motion estimation, TPL, transform search, interpolation filter search, and CDEF filtering, have been developed that allow SVT-AV1 to achieve excellent quality-complexity tradeoffs.

DYNAMIC-OPTIMIZER-BASED TWO-PASS SVT-AV1

In addition to the bandwidth gains from its use of new video codecs, an adaptive streaming application can also benefit bandwidth-wise from the video-shot-based DO framework, presented in a tech blog [18] and two SPIE papers [19-20]. In fact, such framework was shown to

provide an additional ~30% savings in average bitrate for the same quality as compared to traditional adaptive streaming [18-20]. Such gains would also come with a potentially substantial increase in computational complexity, requiring efficient solutions such as the two-pass DO algorithm presented in [21].

As shown in Figure 3, the DO framework is based on splitting the video content into content-uniform video shots that are usually between 2 and 10 seconds in duration, with each shot subsequently encoded at different resolutions and different QP/CRFs. The convex hull parameter values of the encodings of each shot are then generated. To construct the bitstream for the full video content, the equal-slope approach is used to aggregate the encoded shots, resulting in different full bit streams with distinct levels of quality.

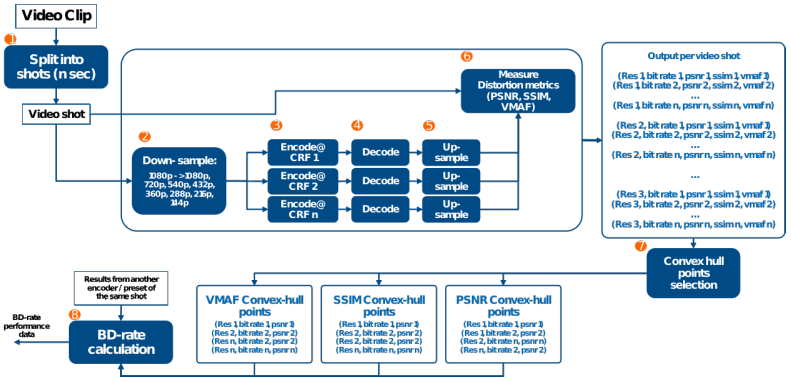
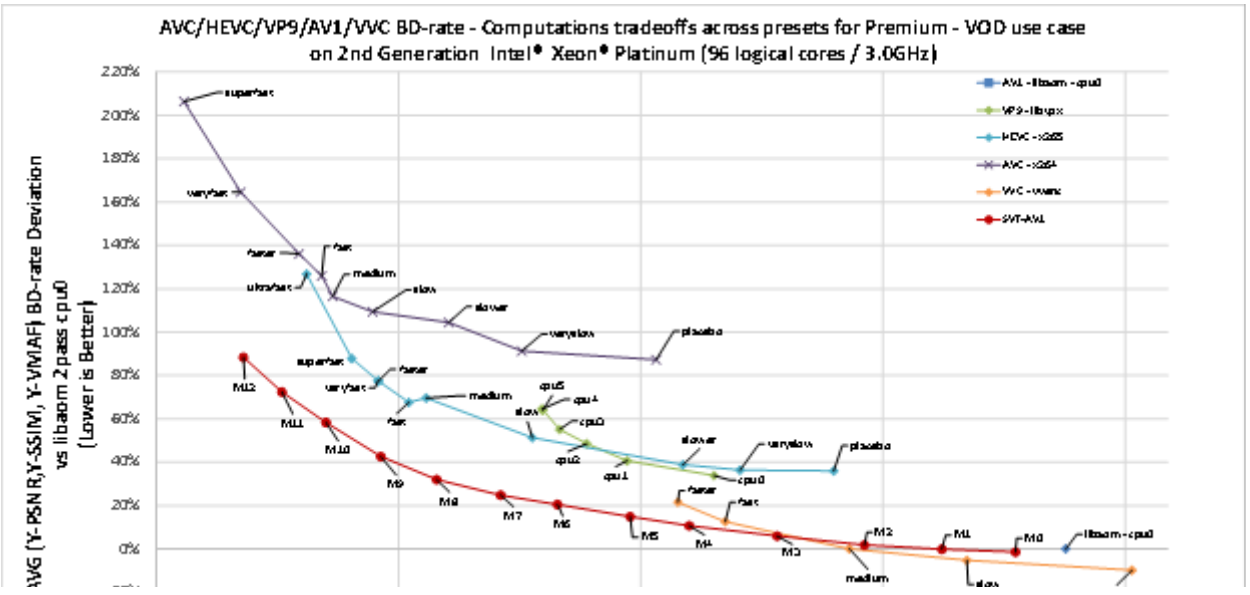


Figure 3. Shot-based encoding workflow.

Figure 4 presents the BD-rate versus time curves for DO-based encoding using premium video content, for several open-source encoders, namely SVT-AV1, AVC-x264, VP9-libvpx, HEVC-x265 and VVC-VVENC. For each preset and for each video shot, the convex hull parameter values are first generated. The YPSNR, YSSIM and VMAF BD-rate data are then generated for all presets using libaom-cpu0 as reference. Moreover, the total encode (system + user) time for each preset is recorded. As shown in the figure, SVT-AV1 M12 yields x264-veryslow BD-rate levels with a ~14x reduction in encoding time, and it achieves x264-very-fast speed levels with BD-rate savings of ~25%. Moreover, M8 achieves x265-medium speed levels while yielding ~25% gain in BD-rate.



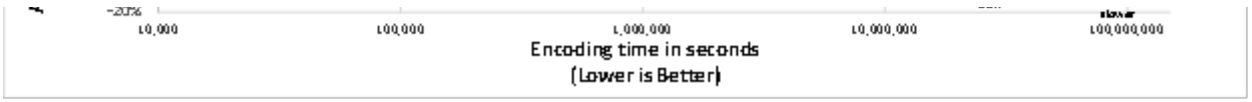


Figure 4. BD-rate vs. encode-time for DO-based SVT-AV1 and other open-source encoders.

Despite its bandwidth efficiency, DO-based streaming is potentially much more demanding in computations than conventional adaptive streaming, since each shot is encoded multiple times for many resolutions/QP pairs (8 x 11 = 88 encodings for each video shot for the results shown in Figure 4). To address the large computational demands, a two-pass SVT-AV1 solution is employed, where the convex-hull encoding parameters for each video shot are estimated using a fast preset of the SVT-AV1 encoder. Subsequently, the bit streams are generated for the desired-preset SVT-AV1 encoder using the first-pass-generated parameter values. Figure 2 illustrates the two-pass DO-based SVT-AV1 solution, with the BD-rate versus encode time tradeoffs shown in Figure 5. In this figure, the red curve shows the BD-rate versus encode time tradeoffs for conventional DO-based SVT-AV1, whereas the green curve shows the same tradeoffs for DO-based two-pass SVT-AV1. For the two-pass SVT-AV1 encoder, the presets are denoted with the pair (Mx, My), with Mx being the preset of the first-pass encoder, and My being the preset of the second-pass encoder. The Mx-preset encoder is determined for each target My-preset encoder, based on an offline analysis of a large set of different video shots, in a way that would optimize the overall quality-cycles tradeoffs. Then, the final encodings for each shot are performed for the My-preset encoder using the Mx-generated (resolution, CRF) parameters. To improve the tradeoffs of the M11/M12 encodings, we also developed a very-fast M13-preset encoder, to be used as a first-pass encoder.

As illustrated in Figure 5, the DO-based two-pass SVT-AV1 encoder not only covers the entire range of the computational complexity range that spans 3 orders of magnitudes, while providing high granularity, with a higher number of (Mx, My) presets, but it also yields clear improvements in quality-cycles tradeoffs. For example, the (M5, M0)-encoder is 8.7x faster than the M0-encoder, while maintaining the same BD-rate levels. Another example is the (M12, M7)-preset encoder, which is both ~2.8x faster and ~2.5% more bandwidth efficient than the M8-preset encoder.

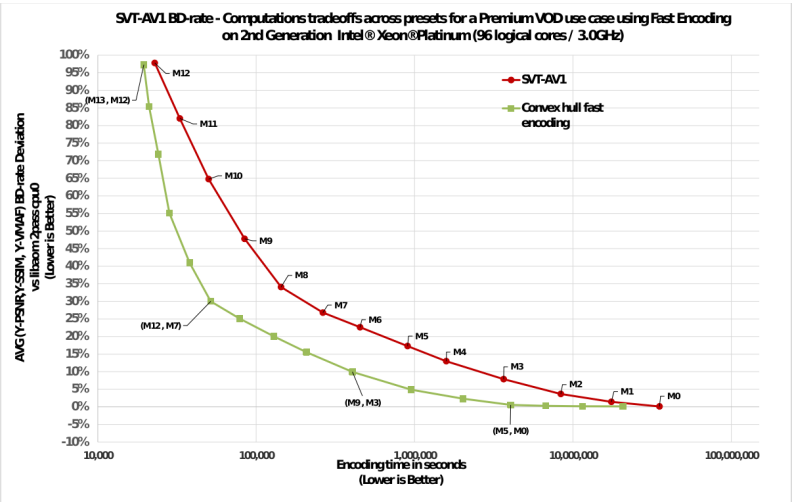


Figure 5. BD-rate – encode-time tradeoffs for the DO-based two-pass SVT-AV1 solution.

CONCLUSIONS

In 2021, ~60% of the global internet traffic was video traffic. This explosive growth will continue, putting pressure on already strained communications networks. To make video workloads bandwidth-efficient in this increasingly congested environment, new generations of video encoders, coupled with convex-hull-based dynamic-optimization algorithms, enable high-quality video encodes at significantly reduced bitrates. However, such an approach would potentially require explosively-more computing resources, which would strain data center capacity and power grids. This creates a tension between the dual goals of reducing the bandwidth requirements of video and slowing the growth of global data center energy/power consumption. Meta manages this tension by making cost/benefit decisions on a per-video basis, making sure that the videos predicted to get the most watch also get the most compute. Typically, this is achieved by using an older codec such as x264 for the less popular videos, and newer codecs such as libvpx or SVT-AV1 for the most popular videos. The presented dynamic-optimizer-based two-pass SVT-AV1 video transcoder may become a single solution that enables highly efficient large-scale VOD services, covering all content and use-cases. With the rapid adoption of highly efficient SW and HW AV1 decoders in mobile devices, this paves the way to transition the entire streaming industry from its current fragmented environment, where many generations of codecs co-exist and compete with one another, to a single, royalty-free and energy-efficient solution.

ACKNOWLEDGMENT

The co-authors of this blog would like to thank Colton Cheung who scripted and executed the pipeline yielding the L1/L2 convex hull encoding discussed in the introduction. The co-authors would like to also thank the SVT engineering team and SIWG members for their continuous contributions towards the SVT-AV1 project.

APPENDIX I - SAMPLE COMMAND LINES

Step 1: Pre-processing - Decoding and splitting the input video into shots

- ./ffmpeg -y -loglevel error -threads 128 -r 30000/1001 -i in.264 -ss 0.0 -t 3.003 Scene_000.y4m

- ./ffmpeg -y -loglevel error -threads 128 -r 30000/1001 -i in.264 -ss 3.003 -t 3.003 Scene_001.y4m

Step 2: L1 Convex-hull encoding with libsvtav1 M12:

- ./ffmpeg -y -threads 1 -i input_1080p.y4m -sws_flags lanczos+accurate_rnd:threads=1 -strict -1 -s:v 1280x720 -c:v libsvtav1 -crf 23 -g 999 -svtav1-params "preset=12:lp=1:maxbitrate=20000" -f ivf output_720p.bin

Step 3: Analysis 2 - Decoding, up sampling using libdav1d and FFmpeg:

- ./ffmpeg -y -r 25 -i output_720p.bin -r 25 -i input_1080p.y4m -threads 1 -lavfi 'scale2ref=flags=lanczos+accurate_rnd:threads=1[scl][ref]; [scl][1:v]psnr=stats_file=output_720p.psnr -threads 1 -map "[ref]" -f null -

Step 4: L2 Encoding - encoding with libsvtav1 M7:

- ./ffmpeg -y -i input_1080p.y4m -sws_flags lanczos+accurate_rnd:threads=1 -strict -1 -s:v 384x216 -c:v libsvtav1 -crf 59 -g 999 -svtav1-params "preset=7:lp=2:tune=1" -f ivf output_384x216.bin

Step 5: Preparing – Combining the final bitstreams:

- ./ffmpeg -y -safe 0 -f concat -i Netflix_Elfuente_Bitrate1000.txt -c copy Netflix_Elfuente_Bitrate1000.ivf

REFERENCES

1. AOMedia Software Implementation Working Group to Bring AV1 to More Video Platforms (https://www.businesswire.com/news/home/20200820005599/en/AOMedia-Software-Implementation-Working-Group-Bring-AV1)

2. Comp, L. "Why More Companies Are Using the Open Source AV1 Video Codec" (https://itpeernetwork.intel.com/open-source-svt-av1/#gs.zg5w90)

3. Dav1D AV1 Decoder project (https://code.videolan.org/videolan/dav1d/-/tags/1.0.0)

4. SVT-AV1 encoder and decoder project (https://gitlab.com/AOMediaCodec/SVT-AV1/-/tags/v1.0.0)

5. FFmpeg binary built with libsvtav1 and libdav1d (https://gitlab.com/AOMediaCodec/aom-testing/-/tree/tech-blog)

6. Advanced video coding for generic audiovisual services, ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) (2003).

7. High efficiency video coding, ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC) (2013).

8. Grange, A., de Rivaz, P. and Hunt, J., "VP9 bitstream and decoding process specification, (https://www.webmproject.org/vp9/)" (2016).

9. de Rivaz, P. and Haughton, J., "AV1 bitstream & decoding process specification, (https://aomediacodec.github.io/av1-spec/)", (2019).

10. Bross, B., Chen, J., Liu, S. and Wang, Y.-K., "Versatile Video Coding (Draft 10), "ITU-T and ISO/IEC JVET-S2001 (2020).

11. x264, code repository - open-source AVC encoder software (https://code.videolan.org/videolan/x264).

12. x265, open source HEVC software encoder (https://www.videolan.org/developers/x265.html).

13. libvpx, code repository - open-source VP9 encoder/decoder software (https://github.com/webmproject/libvpx).

14. libaom, code repository - open-source AV1 encoder/decoder software (https://aomedia.googlesource.com/aom/)

15. VVenC, open source VVC software encoder (https://github.com/fraunhoferhhi/vvenc).

16. Kossentini, F., Guermazi, H., Mahdi, N., Nouira, C., Naghdinezhad, A., Tmar, H., Khlif, O., Worth, P. and Ben Amara, F., "The SVT-AV1 encoder: overview, features and speed-quality tradeoffs," Proc. SPIE 11510, Applications of Digital Image Processing XLIII, 1151021 (21 August 2020); doi: 10.1117/12.2569270.

17. Wu, P-H., Katsavounidis, I., Lei, Z., Ronca, D., Tmar, H., Abdelkafi, O., Cheung, C., Ben Amara, F., and Kossentini, F., "Towards much better SVT-AV1 quality-cycles tradeoffs for VOD applications", Proc. SPIE 11842, Applications of Digital Image Processing XLIV, 118420T (1 August 2021) (https://doi.org/10.1117/12.2595598)

18. Katsavounidis, I., "The NETFLIX tech blog: Dynamic optimizer - A perceptual video encoding optimization framework (https://netflixtechblog.com/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f)," (Mar. 2018).

19. Katsavounidis, I. and Guo, L., "Video codec comparison using the dynamic optimizer framework," Proc. SPIE 10752, Applications of Digital Image Processing XLI, 107520Q (17 September 2018); doi: 10.1117/12.2322118.

20. Wu, P.-H., Kondratenko, V. and Katsavounidis, I., "Fast encoding parameter selection for convex hull video encoding," Proc. SPIE 11510, Applications of Digital Image Processing XLIII, 115100Z (21 August 2020); doi: 10.1117/12.2567502.

21. Wu, P.-H., Kondratenko, V., Chaudhari, G. and Katsavounidis, I., "Encoding Parameters Prediction for Convex Hull Video Encoding," Picture Coding Symposium (PCS), Bristol, UK, (2021).