Towards Much Better SVT-AV1 Quality-Cycles Tradeoffs for VOD Applications

Ping-Hao Wu[a], Ioannis Katsavounidis[a], Zhijun Lei[a], David Ronca[a]
Hassene Tmar[b], Omran Abdelkafi[b], Colton Cheung[b], Foued Ben Amara*[b], Faouzi Kossentini[b]
[a]Facebook, Menlo Park, CA, USA
[b]Intel Corporation, Suite 700, 450 SW Marine Drive, Vancouver, BC V5X 0C3 Canada

ABSTRACT

Software video encoders that have been developed based on the AVC, HEVC, VP9, and AV1 video coding standards have provided improved compression efficiency but at the cost of large increases in encoding complexity. As a result, there is currently no software video encoder that provides competitive quality-cycles tradeoffs extending from the AV1 high-quality range to the AVC low-complexity range. This paper describes methods to further improve the SVT-AV1 overall quality-cycles tradeoffs for high-latency VOD applications.

The paper first presents the latest SVT-AV1 encoder evaluation results generated using the conventional convex-hull-based evaluation approach. To better account for all the content being considered in the evaluation, the combined convex hull approach is then introduced and corresponding results outlined. A more practical version of the latter approach, referred to as the restricted discrete approach, is then presented. In this approach, only few discrete quality levels that are of relevance to typical adaptive streaming applications are considered. This restriction makes the evaluation data more relevant to real adaptive streaming applications and reduces the storage requirements for the encodings generated at the identified optimal encoder settings. A fast version of the latter approach is then presented, where the generation of the optimal encoder parameters is performed using a fast encoder, and the final encodings needed in the evaluation are generated using the desired encoder based on the identified optimal encoder parameters. These optimizations result in a complexity reduction by up to a factor of 10 for at most 1.5% loss in BD-rate.

Keywords: AV1, SVT-AV1, Video compression, Coding efficiency, Adaptive streaming, Convex hull, Dynamic optimizer, Encoder complexity

1. INTRODUCTION

The generation, sharing and consumption of video data has experienced an explosive growth in recent years. This growth is fueled by the ubiquitous use of portable devices with video encoding and decoding capabilities, the emergence of relatively new streaming applications that allow for the viewing of video content anywhere and at any time, the widespread adoption of real-time video communication applications and the continuous growth of broadcast services. As a result, the video processing infrastructure is being increasingly strained by the large amount of data that would need to be processed before it can be distributed through communication networks. The resources available on communication networks, mainly in the form of bandwidth, are also being strained given the amount of data that is being shared among users of the networks. In response to these challenges, new video coding standards are being continuously developed to help improve the video coding efficiency and therefore help alleviate the pressure on the required network bandwidth. Examples of these standards include H.264/AVC [1], H.265/HEVC [2], VP9 [3], AV1 [4] and more recently VVC [5]. Typically, a new standard is developed every seven years and would offer coding efficiency gains of about 30% on average as compared to the preceding standard. However, such gains in compression efficiency would normally come at the expense of an increase in the complexity for both the encoder and decoder, where the increase in the encoder complexity would typically be by a factor of at least 10x, and the decoder complexity increase would be by a factor of about 2x. From a video encoder development perspective, the wide variety of the video processing applications mentioned above results in a wide range and sometimes conflicting quality-speed-latency-memory tradeoff requirements. To address this particular problem, the Scalable Video Technology (SVT) was developed recently to provide a seamless way of addressing the variety of video processing tradeoffs [6]. In particular, the SVT-AV1 encoder is an open source video encoder based on the AV1

*foued.ben.amara@intel.com

Applications of Digital Image Processing XLIV, edited by Andrew G. Tescher, Touradj Ebrahimi, Proc. of SPIE
Vol. 11842, 118420T · © 2021 SPIE · CCC code: 0277-786X/21/$21 · doi: 10.1117/12.2595598

specifications and was recently adopted by the Alliance for Open Media as the productization platform for the AV1 specifications [7][8]. The architectural features of the SVT-AV1 encoder as well as the associated algorithmic optimizations are the key enablers of the flexibility the encoder has in successfully meeting the requirements of a wide range of video processing applications.

Among the video processing applications mentioned above, HTTP adaptive streaming, or simply adaptive streaming (AS), has emerged as a key enabler behind the processing and delivery of the increasing amount of shared video data. Adaptive streaming allows for the adjustment of quality or bitrate of the delivered video bitstream in response to the network conditions and the available bandwidth. In conventional AS, the encoding of a given content is performed at different resolutions and/or bit rates, with typically five to ten versions of the encoded content made available for use in a streaming session. During a streaming session, a change in the network bandwidth would result in switching to the encoded version of the streamed content that provides the highest quality under the current bandwidth limitations. The ability to switch in the middle of a streaming session is made possible (1) by encoding the different versions of the original content using a closed Group-of-Pictures (GOP) configuration and (2) by temporally aligning the Key (or Intra) pictures in all of the encoded versions. Although AS allows for adaptation in response to network conditions, the conventional approach, which generates encoded versions using the same encoder settings from beginning to end of a long video sequence, doesn't take into account this key feature of AS and is thus suboptimal.

An improvement over the traditional adaptive streaming encoding approach was introduced by the dynamic optimizer (DO) framework presented in [9] and further discussed in [10]. The DO approach is based on two key ideas. First, the processing of the input content is performed at a finer granularity, referred to as shots, as opposed to being performed for the entire input video sequence. Second, the generation of different encoded versions of the input content is performed by concatenating shots encoded at different resolutions and rates so that each of the generated bitstreams would correspond to either a pre-specified quality level or bitrate. Shots are segments of the input video content that have relatively homogeneous properties and that are of durations that typically last from 2 to 10 sec. Consequently, shots can simply be encoded using the fixed quantization parameter (QP) or constant rate factor (CRF) approach without relying on rate control to achieve a desired bit rate. Adjacent shots in a final generated bitstream might be encoded at different resolutions and encoder settings if the content of the two adjacent shots is different. It follows in this case that the change in content type going from shot to the next helps in masking any visual effects associated with changes in the encoding parameters for adjacent shots. The generation of different encoded versions of the input content is performed using the convex hull approach. Shots are first encoded at different resolutions and bitrates, the convex hull of distortion vs. rate data associated with all such encodings for a given shot is generated, and points on the convex hull are used to identify the best rate for a given distortion - or vise-versa - for the shot. Shots that achieve a prespecified quality level or bitrate are then put together to generate the corresponding bitstream. Multiple bitstreams could be generated using this approach for different quality levels or bitrates. Another ingredient in the DO framework is the use of "equal slope" in the selection of operating points among different shots. This method is also known as "constant slope" in the context of statistical multiplexing of different streams for carriage in terrestrial, cable or satellite communication channels. The conventional DO approach was shown to provide bit rate savings of up to 30% as compared to the conventional AS approach [9,10].

Even though the DO provides more optimized encodings as compared to the conventional AS approach, the improvements come at a significant increase in computational complexity of the overall process. The increase could be roughly represented by a factor corresponding to the number of bitrates considered when generating the convex hull for each shot. To address this problem, a fast DO approach was proposed in [11] where the convex hull generation is achieved by considering a relatively fast encoder, whereas the generation of the final bitstreams is performed using the optimal encoding parameters (i.e. (resolution, bitrate) pairs) from the first step for each shot and completing the final encodings using the desired high quality but computationally costly encoder. The two encoders used in the fast DO approach could correspond to two different presets of the same encoder, or to two encoders that support different coding standards. The fast encoder could also be a fast hardware encoder implementation. It is argued in [11] that the convex hull representing the optimal encoding parameters should be relatively the same, regardless of the encoder preset used in generating the convex hull for the shot. The key finding from the investigation reported in [11] is that the fast DO approach results in a reduction in the total complexity of the process while incurring a minor loss in BD-rate gain. For example, in the case of the x264 encoder, the total complexity increase was reduced from 6x to about 2.3x while the associated BD-rate gain was reduced by only about 1% from -29.71% to -28.76%.

The purpose of this paper is to evaluate the latest version of the SVT-AV1 encoder using different variants of the DO approach. First, the performance of the SVT-AV1 encoder is evaluated using the conventional DO approach and is compared to that of other encoders. In the conventional DO approach, the encoder performance represents the average of the encoder BD-rate data over all shots considered in the evaluation. A second evaluation approach, referred to as the combined DO approach, is considered where the encoder performance is conceptually computed for one clip representing the concatenation of all clips in the test set. The combined DO approach extends the use of the constant slope approach to the single clip under consideration, allowing for a dense convex hull data for the clip and for optimized encoder settings that account for all the content being tested. A variation of the combined DO approach, referred to as the restricted discrete DO approach, is considered where the range of quality values considered in the evaluation is reflective of quality values common in AS applications, and where the encoder BD-rate performance is evaluated by considering few points on the convex hull. To reduce the complexity associated with the restricted discrete DO approach, a fast DO approach is then evaluated, where the identification of optimal encoder parameters is performed based on encodings generated using a fast encoder. The optimal encoder parameters are then used to generate final encodings using the desired encoder. Convex hull data corresponding to the final encodings is used to generate the encoder BD-rate performance data. Evaluation results indicate the fast approach results in a significant reduction in complexity, reaching a factor of 10 reduction in complexity for a loss of about 1.5% in BD-rate as compared to the restricted discrete DO approach. Concluding remarks are presented in the last section of the paper.

2. PERFORMANCE UPDATE FOR THE SVT-AV1 ENCODER

The SVT-AV1 encoder architecture and features are discussed in detail in [6]. In the following, a summary of the recent updates in the SVT-AV1 encoder is presented followed by an update on SVT-AV1 encoder performance based on the conventional DO approach.

2.1 Recent Updates in the SVT-AV1 Encoder

The SVT-AV1 encoder is designed to handle a wide range of often conflicting performance requirements involving complexity, quality and latency. The scalability features in the SVT-AV1 encoders provide the tools that enable the SVT-AV1 encoder to effectively address those requirements through the full exploitation of the available computational resources. The main enablers of scalability in the encoder are the multidimensional parallelism, the structural optimizations and the algorithmic optimizations, all of which provide effective means to shape the encoder performance.

Parallelism in the SVT-AV1 encoder could be considered at the process, picture or segment level. The flexibility offered by these three parallelism dimensions makes it possible to address various speed and latency tradeoffs. In process-based parallelism, tasks in the encoder pipeline are grouped into processes that can run in parallel. Picture level parallelism is achieved by designing the prediction structure in the encoder in such a way that groups of pictures could be processed simultaneously once their corresponding references become available. To enable segment level parallelism, input pictures are split into smaller segments that can be processed in parallel while respecting coding dependencies. Realtime performance in the encoder could therefore be achieved with a large enough number of processor cores provided the processor memory constraints are not violated. In general, picture level parallelism would result in longer latencies as compared to segment level parallelism. The resulting bitstream from the encoder is the same whether a single threaded mode or a multithreaded mode is considered, implying that the real-time performance of the encoder would be realized at a high CPU utilization without any degradation in quality.

At the structural level, the SVT-AV1 encoder features a multi-staged processing approach to the selection of partitioning and coding modes. In this approach, the encoder initially processes a large number of candidates to identify a subset of promising candidates. The initial selection of this subset is performed by considering prediction and coding tools that are relatively inaccurate but computationally less costly to run as compared to the normative tools. The same process is repeated in subsequent stages where more and more accurate but computationally more expensive tools are considered to ultimately converge to the best partitioning and coding modes. A more detailed discussion of this topic is included in [6]. In addition to the structural and algorithmic optimizations reported in [6], further improvements in the encoder performance have been realized through the integration of additional features and/or the optimization of existing features. The improvements in the encoder cover mostly temporal filtering, motion estimation, mode decision and inloop filtering features. Examples of these improvements are briefly summarized in the following.

- BD-rate-wise lossless changes: A number of lossless changes were completed in the code and yielded significant reduction in the encoder complexity.
- Added presets 7 and 8: The total number of presets in the encoder is increased from seven to nine. The two new presets cover faster speed ranges with preset 8 achieving similar complexity as that of x265 Medium preset (v3.4) with good BD-rate gain in favor of preset 8.
- Temporal filtering: The purpose of temporal filtering is to generate filtered versions of input pictures that would be encoded instead of or in addition to the original source pictures for better compression efficiency. Motion-compensated temporal filtering is considered in SVT-AV1 for pictures in temporal layers 0 and 1. The complexity of this feature was optimized along several directions. First, the number of pictures considered in temporal filtering is adjusted according to (1) the noise level in the pictures, where pictures with high noise level are not included in the filtering process, (2) the activity level in the reference pictures, where reference pictures with high activity level are not included in the filtering operation. Second, with respect to subpel refinement, the subpel search is first performed using large block sizes, e.g. 32x32 blocks, and if the resulting prediction is satisfactory, the subpel search for smaller block sizes is skipped.
- Pre-Hierarchical Motion Estimation (Pre-HME): Even though Hierarchical Motion Estimation (HME) is designed to quickly identify Motion Estimation (ME) search centers, especially for large motion content, it fails to do so for very high motion content. To address this problem, a Pre-HME stage is added just before HME to handle such cases. In Pre-HME, a very crude search is performed in extended vertical and horizontal search strips to identify good HME search centers when the motion range is very large.
- The Temporal Dependency Model (TPL) algorithm: The TPL algorithm represents an extension of the mb tree algorithm introduced in x264 encoder. The main purpose of the algorithm is to optimize the encoder settings to reduce the impact reference pictures have on the degradation in quality/rate in the pictures that reference them directly or indirectly. In the SVT-AV1 encoder, the goal of the algorithm is to reduce the impact of base layer pictures on such degradations. The algorithm involves two main steps. In the first step, encoding is performed using an elementary encoder to collect prediction information. The second step involves using the collected prediction information to optimize the encoder settings that would be used in the final encoding of the input pictures. The affected encoder settings include QP scaling, QP modulation and the lambda parameter considered in the RD cost calculations. A multithreaded implementation of the analysis component of the algorithm is considered to improve the efficiency of the algorithm. The lookahead distance, which refers to a window of future pictures considered in the analysis process, is designed to be a user defined parameter and can be adjusted to yield different complexity/quality/memory tradeoffs.
- Transform type search: The AV1 specifications provide a relatively large number of transform types to consider. Searching for the transform type that provides the best coding cost would normally involve evaluating each of the transform types and selecting the best type. The forward transform type search represents one of the costliest operations in the encoder. To reduce the cost associated with such a search, the different transform types are analyzed and are organized into nested groups. Each of the encoder presets is assigned a given group of transform types to work with based on the associated speed/quality tradeoffs. As such, the complexity associated with the transform type search is limited as a function of the encoder preset.
- Constrained Directional Enhancement Filter: The Constrained Directional Enhancement Filter (CDEF) is an important inloop filter that helps with addressing edge artifacts. The selection of the CDEF filter to work with is based on evaluating a large number of (primary filter strength, secondary filter strength) pairs to identify the pairs with the best coding cost. To reduce the cost associated with the search for the best filter pair, a subset of filter strength pairs to search is considered for each encoder preset depending on the associated tradeoffs.
- Rate control algorithms: Both one-pass and two-pass VBR rate control algorithms are included in the code and are still work in progress. The one-pass VBR features a lookahead window that allows the one-pass algorithm to approach the performance of the two-pass algorithm for a long enough lookahead distance.

2.2 Update on the SVT-AV1 Encoder Performance

This section presents the evaluation of the SVT-AV1 encoder using the conventional DO approach discussed in [9]. Building on the work done in [6] and as described in Figure 1, the rest of this section presents a comparison of the quality vs. complexity tradeoffs of different open-source production grade encoder software implementations representing the AVC, HEVC, VP9, AV1 and VVC video coding standards.

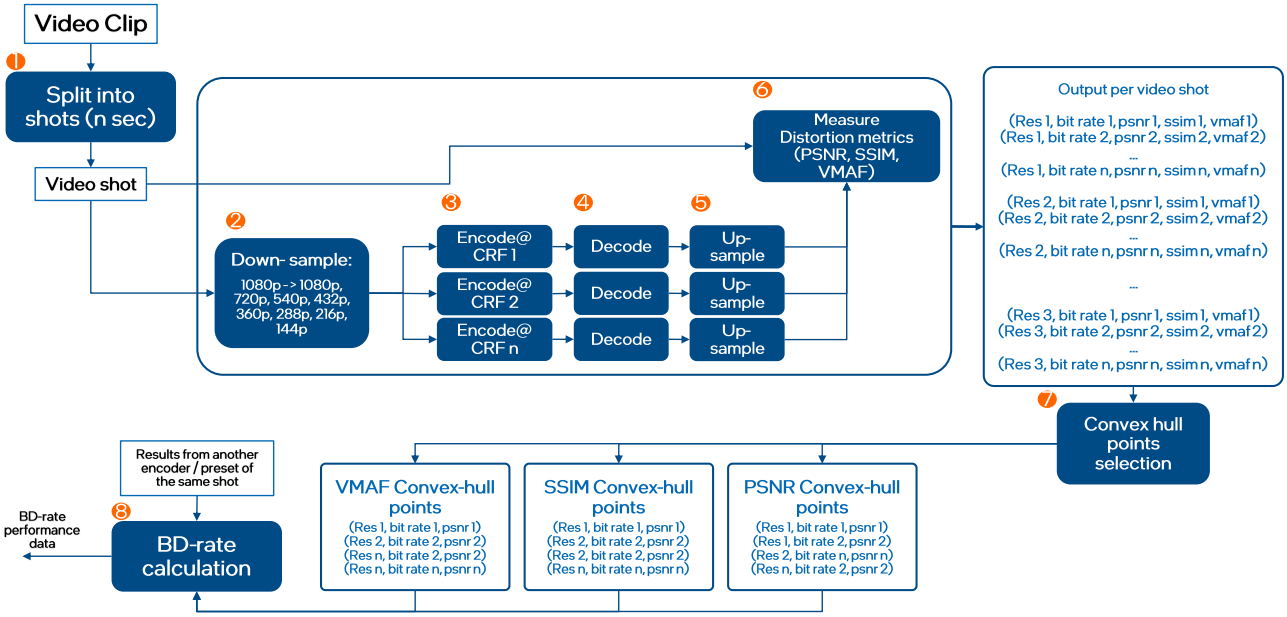

Figure 1. Shot-based encoding workflow.

2.3 Selection of the test video clips

Three public video clips test sets were selected to perform these experiments that present a mix between premium VOD content and UGC test sets all at the same spatial video resolution 1920x1080:

- The first test set (testset-1) builds on the work done in [6] by considering the same 20 clips and cropping them to a maximum length of 4 seconds, hence creating a total length of 1 min 20 sec of video content having a mix of premium and UGC VOD content.
- The second test set (testset-2) is the open source ElFuente clip [12] that contains 14296 frames with a frame rate of 29.97 corresponding to 7 min 52 sec of premium VOD content. The video has been partitioned into 140 shots with a shot length of 2-5 seconds, averaging a 3.4 sec shot length.
- The third test set (testset-3) is a selection of UGC clips from the YouTube UGC dataset [13]. The selection included 81 shots of 5 sec each at a heterogeneous set of frame rates (12, 15, 23.98, 24, 24.98, 25, 29.72, 29.9, 29.97, 30, 30.02, 30.07, 31.14, 50, 59.94 and 60) creating a total video length of 6 min 45 sec.

2.4 Spatial down-sampling of the video clips

As depicted in step 2 of Figure 1, starting at the 1080p resolution of the test sequences, seven lower resolutions were considered in these tests, resulting in a total of 8 encoding resolutions: **1920x1080, 1280x720, 960x540, 768x432, 640x360, 512x288, 384x216, 256x144.** FFMPEG (n4.4 - git revision a37109d555) was used to perform the down-sampling with the sample command line shown below, resulting in a total of **160**, **1120**, and **648** video shots for testset-1, testset-2, and testset-3 respectively:

- ffmpeg -y -i input.y4m -sws_flags lanczos+accurate_rnd+full_chroma_int -sws_dither none -param0 5 -strict -1 -s:v 1280x720 output.y4m

2.5 Encoders and encoding configurations

In order to give a wide range of complexities and coding efficiencies, six open source encoders representing production grade software implementations of AVC, HEVC, VP9, AV1 and VVC video coding standards were used in this comparison: x264 (0.161.3049) [14], x265 (3.5) [15], libvpx (1.10) [16], libaom (3.1.0) [17], SVT-AV1 (0.8.8-rc) [18], and VVenC (1.0.0) [19]. The detailed CRF values, presets, and command lines used for the experiments are presented in the following.

CRF values

In order to maximize compression efficiency, the Constant Rate Factor (CRF) mode [20] was used for all encoders. The selection of CRF values is based on first selecting the AV1 / VP9 CRF values ranging from 23 to 63 and selecting equally spaced intermediate points to end up with the following set **(23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63)** of CRF values. All the clips in the test sets are then encoded using libaom at the above mentioned 11 CRF values, and x264 at a range of CRF values from 14-51. The resulting average quality scores across all clips per CRF value indicated that the AV1 CRF values of 23 and 63 yield a quality level that matches approximately the one generated by CRF 19 and 41 of x264 respectively. As a result, 11 CRF points **(19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 41)** were chosen for the x264, x265 and VVenC encoders.

Presets

The libaom encoder in its highest quality preset (--cpu-used=0) is used as the BD-rate anchor in all the comparisons. As for the rest of the encoders, all commonly used presets were tested with the exception of x264 ultrafast as it results in a large quality loss in the context of these experiments.

Command lines

- **libaom:** ./aomenc --passes=2 --verbose --lag-in-frames=35 --auto-alt-ref=1 --end-usage=q --kf-min-dist=999 --kf-max-dist=999 --cq-level=<crf value> --cpu-used=0 -o output.bin input.y4m
- **x264:** ./x264 --preset <1-9> --threads 1 --tune psnr --stats output.stat --asm avx2 --crf <crf> --keyint 999 --min-keyint 999 --no-scenecut -o output.bin input.y4m
- **x265:** ./x265 --preset [0-9] --tune psnr --stats output.stat --crf <crf value> --keyint 999 --min-keyint 999 --pools 1 --no-scenecut --no-wpp input.y4m -o output.bin
- **libvpx:** ./vpxenc --ivf --codec=vp9 --cpu-used=[0-5] --tile-columns=0 --arnr-maxframes=7 --arnr-strength=5 --aq-mode=0 --bias-pct=100 --minsection-pct=1 --maxsection-pct=10000 --i420 --min-q=0 --frame-parallel=0 --min-gf-interval=4 --max-gf-interval=16 --verbose --passes=2 --end-usage=q --lag-in-frames=25 --auto-alt-ref=6 --threads=1 --profile=0 --bit-depth=8 --input-bit-depth=8 --fps=30000/1001 --kf-min-dist=999 --kf-max-dist=999 --cq-level=<crf value> -o output.bin input.y4m
- **SVT-AV1:** ./SvtAv1EncApp --preset [0-8] -q <crf value> --keyint -1 -lp 1 -i input.y4m -b output.bin
- **vvenc:** ./vvencapp --input in.yuv --size <wxh> --qp <crf value> --preset slow --format yuv420 --internal-bitdepth 8 --intraperiod 999 --framerate 59.94 --threads 1 --output out.bin

Please note that, for all encoding tests, only the first frame is encoded as an Intra (Key) frame and all other frames are encoded as Inter frames. Different parameters are used for different encoders, such as --intraperiod, --keyinit, etc., to achieve this encoding configuration.

Machines running the experiment

Encodings for this experiment are performed on AWS EC2 instances, specifically, c5.12xlarge instances for testset-1 and c5.24xlarge for testset-2 and testset-3. All instances ran Ubuntu Server 20.04 Linux OS Intel® Xeon® Platinum 8275CL. Hyper-threading and Turbo frequency are both enabled on these instances allowing the instance to access 48 and 96 logical cores (vCPU) for the c5.12xlarge and c5.24xlarge, respectively, with a maximum all-core turbo speed of 3.6 GHz [21]. Running the list of commands is done by invoking the ***parallel*** [22] Linux tool and passing to it the command lines generated as explained above. The ***parallel*** tool would then schedule running the command lines by executing newer commands when older ones retire, while maintaining N command lines running at a time. N is chosen to be equal to the number of available vCPUs on the instances in order to maximize the CPU utilization. Each set of encodings per preset per encoder is run independently while capturing the run time using the GNU ***time*** command as follows:

- /usr/bin/time --verbose parallel -j 96 < commands_encoder<x>_preset<n>.txt

2.6 Encoding results

Once all encodings are done, the resulting bitstreams are collected, decoded, and upsampled to 1080p resolution using the ffmpeg command line shown below:

- ffmpeg -i input1.bin -i input2.y4m -lavfi "scale2ref=flags=lanczos+accurate_rnd+full_chroma_int:sws_dither=none:param0=5 [scaled][ref]" -map "[ref]" -f null - -map "[scaled]" -strict -1 output.y4m

Three performance metrics representing a measure of the Y component distortion between the up-sampled decoded clip and the initial clip at the 1080p resolution are generated using vmaf (v2.1.1) [23] based on the following command line:

- vmaf --reference input1.y4m --distorted output.y4m --aom_ctc v1.0 --output output.xml

The performance metrics are Peak-Signal-to-Noise-Ratio (Y-PSNR), Structural Similarity Index (Y-SSIM), and Video Multimethod Assessment Fusion (VMAF). In order to pick the RD points that would result in the best-possible tradeoffs between quality, bit rate and resolution, all resulting elementary bitstream file sizes and quality metrics across all resolutions for each clip are passed to a C sample application [24] that determines the convex hull points based on all available points. These points are chosen to allow the application to switch between encodings corresponding to different resolutions (based on the available bandwidth) while also maintaining the best possible video quality at a certain bit rate. With respect to the performed simulations, the input to this process is a set of 88 encoding results (8 resolutions * 11 CRF points) per video shot.

The BD-rate [25] results for all encodings are generated by comparing the resulting convex hull points for each of the video clips to those generated using libaom preset 0 (i.e. --cpu-used=0), which represents the anchor encoder in this comparison. The BD-rate percentages are then averaged over all clips within the testset being tested, and an average BD-rate percentage is generated per encoder per preset, representing the percent BD-rate deviation of that preset as compared to libaom preset 0.

Figures 2, 3, and 4 show the results of the different encoder quality vs. complexity tradeoffs for testsets 1, 2, and 3 respectively. Every point on the graph corresponds to an encoder preset. The y-axis represents the average BD-rate of each encoder preset relative to that of libaom cpu0. The average BD-rate data is generated by averaging the Y-PSNR, Y-SSIM and VMAF BD-rate data for each preset to create one number describing the quality level for each of the presets. A positive BD-rate value indicates an encoder with worse coding efficiency, i.e. one that requires more bits to achieve the same video quality as that of the anchor, while a negative BD-rate value indicates an encoder with better coding efficiency. The x-axis represents, on a logarithmic scale, the encoding time in seconds. The latter represents the sum of the "user time" and the "system time", where each of those two components of the encoding time are obtained through the GNU utility ***time***. The encoding time represents the aggregate per preset of the encoding times of the 1760, 12320 and 7216 encodings for testset-1, testset-2, and testset-3 respectively.

The results in Figures 2, 3, and 4 show that newer video coding standards produce benefits to video applications along three dimensions:

- Maximizing video coding efficiency gains at a higher compute cost, using the highest quality presets of each of the implementations going from ***x264 placebo***, to ***x265 placebo*** and ***libvpx cpu0***, to ***SVT-AV1 M0*** to ***VVenC slower.***
- Maximizing video coding efficiency gains while maintaining the same compute cost, for example as in the case of ***SVT-AV1 M5*** and ***libvpx cpu1.***
- Minimizing the encoding complexity while maintaining similar coding efficiency levels, for example as in the case of ***SVT-AV1 M5*** and ***libvpx cpu1.***

The wider the complexity range an encoder covers, the more of the above three benefits it can represent. In fact, across all three test sets, the SVT-AV1 encoder maintains a consistent performance, i.e. quality vs complexity tradeoffs, across a wide range of complexity levels. The M8 preset presents a specifically interesting point as it is:

- More than 275x faster than the M0 preset, yielding the widest complexity range across all of the tested encoders.
- Matching similar or higher speed levels to those of the ***x264 slower*** and ***x265 medium*** presets while yielding similar quality levels to those of ***libvpx cpu1*** and ***x265 veryslow*** presets.
- An order of magnitude faster than ***x265 veryslow*** and ~7x faster than ***libvpx cpu1*** while maintaining similar quality levels.

Based on the observed trends for the SVT-AV1 quality-complexity tradeoffs, future presets M9-M12 are indicated in Figure 2, 3 and 4 and would allow SVT-AV1 to cover a complexity range that extends from the higher quality AV1 to the higher speeds AVC presets corresponding to more than 1000x change in complexity.

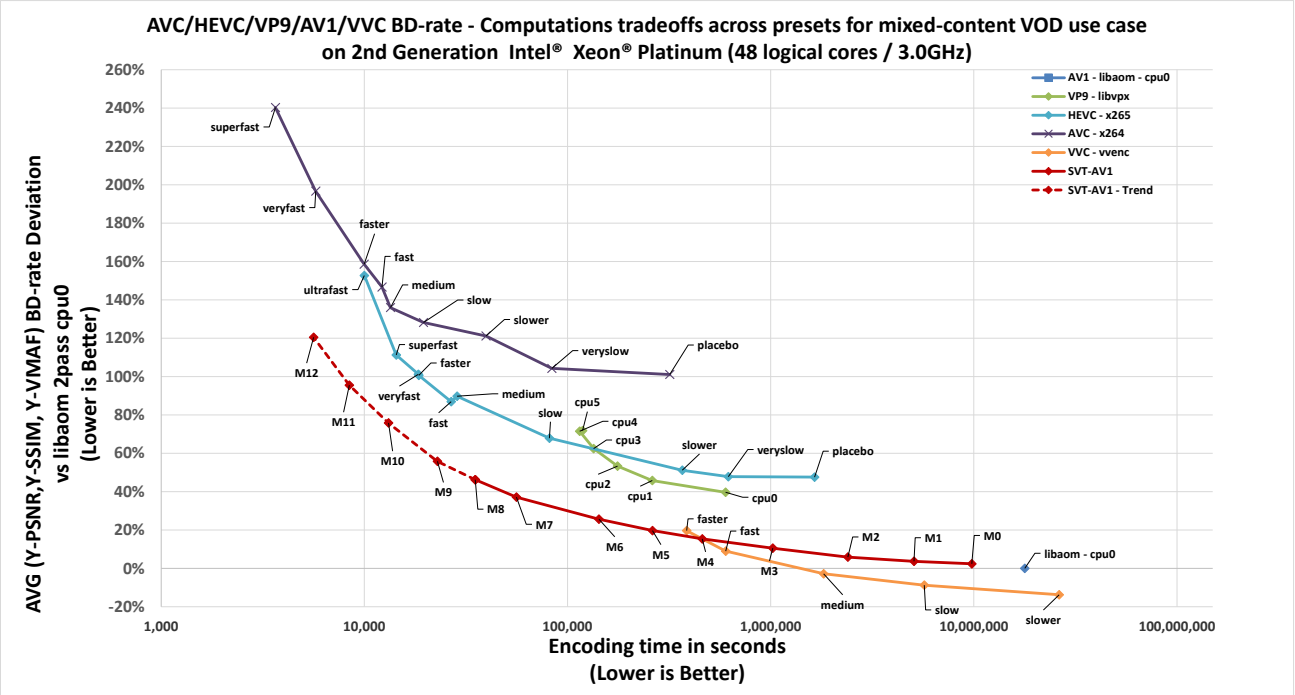

Figure 2. BD-rate vs complexity tradeoffs across preset for testset-1 using the conventional convex hull approach.

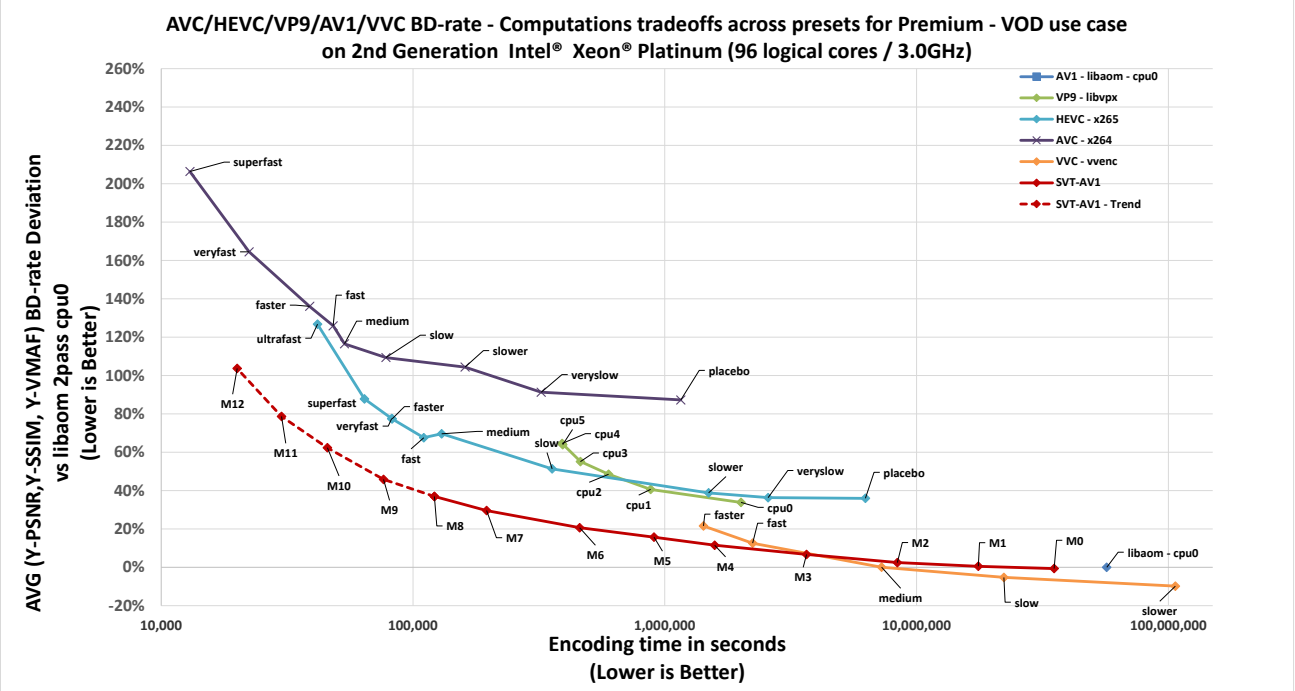

Figure 3. BD-rate vs complexity tradeoffs across preset for testset-2 using the conventional convex hull approach.

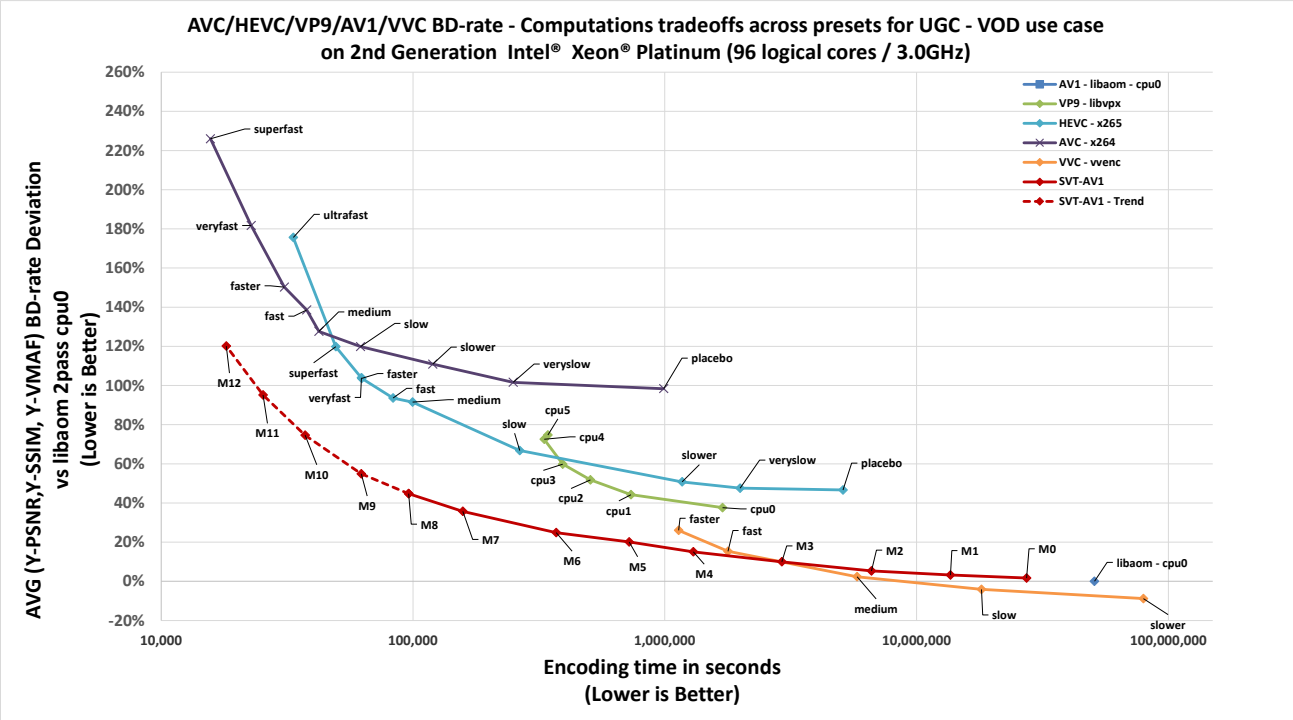

Figure 4. BD-rate vs complexity tradeoffs across preset for testset-3 using the conventional convex hull approach.

3. COMBINED CONVEX HULL APPROACH VS. CONVENTIONAL APPROACH

The issue of aggregating coding efficiency performance across different test video sequences is well known and has been debated before. It is a well-known fact that video coding is highly dependent on the characteristics of the video that is being applied to, with large variations in its performance. Our main motivation for exploring multiple avenues in this work is the desire to find an approach that will better reflect what actual users of video coding in their products or services would observe when they swap an older encoder or coding standard for a newer one. Figure 5 depicts the three different approaches considered in this section. The details of each of the approaches are discussed in the following sections.

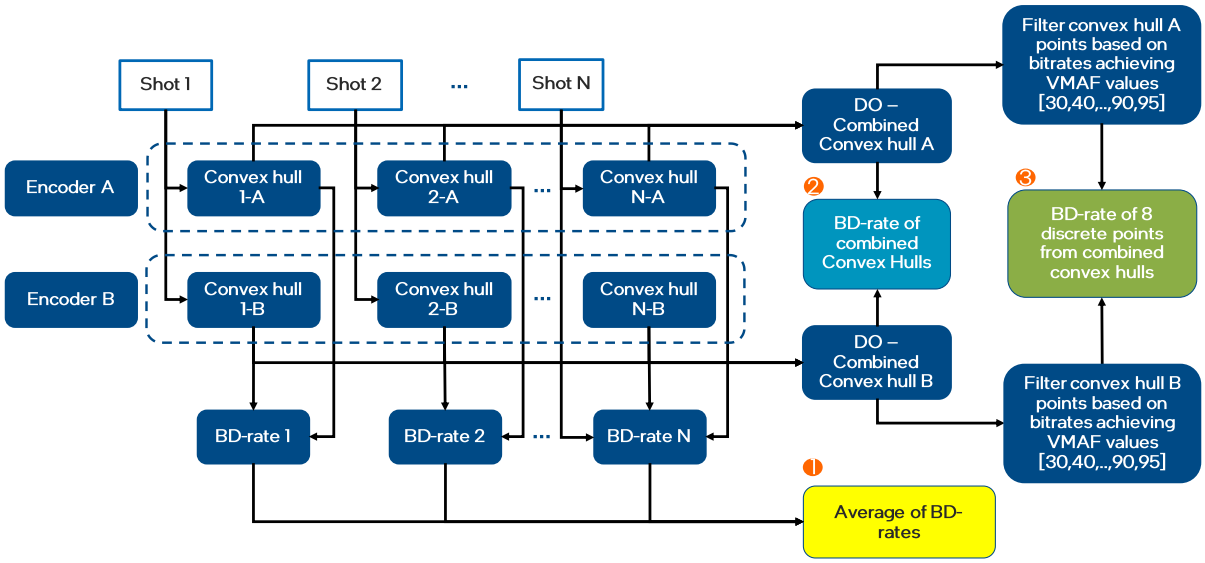

Figure 5. The three proposed approaches - "conventional" in yellow, "combined" in light blue and "restricted discrete" in green. DO stands for Dynamic Optimizer.

3.1 Conventional convex hull approach

In section 2.6, we described the most popular approach to aggregate coding efficiency results across different test sequences, which we will call "Conventional approach" or "averaging of BD-rates approach". This is the approach typically used in video standardization, and can be summarized as follows:

- Each test video sequence is treated as an independent unit, for which a certain BD-rate figure is obtained.
- Calculation of the BD-rate figure for a test sequence is done in an unrestricted manner, i.e. there is no bitrate or quality range restriction when integrating bitrate difference between an anchor and a test encoder.
- All BD-rate figures obtained for a given "class", which typically means sequences of a given spatial resolution, such as 1920x1080, or 1280x720, are then averaged using arithmetic mean.
- A further averaging across different classes of sequences is reported as a "global goodness" measure.

Although simple to explain and implement, there are multiple issues with the conventional approach. First, there is an assumption - which is in fact mostly a hope - that the range of QPs used for each encoder are "reasonable", i.e. they correspond to the range of bitrates/qualities that a certain application would use; this is in fact one of the main factors taken into account by those working in video coding standardization when choosing both test video sequences and QP values to be used in the so-called "common test conditions". Inevitably, though, the wide variety of video content type, together with the typical restriction of using the exact same QP values for all test content, results in cases where the qualities/bitrates are unreasonably high (for example, a 1080p sequence that results in 100Mbps bitrate) or unreasonably low (the same example of a 1080p sequence encoded at 80kbps).
Furthermore, using test sequences with multiple and sometimes very different frame rates results in different time duration, thus one can't reasonably argue that these sequences should have the same weight. On the other hand, the attempt to select sequences that behave as uniformly as possible to the selection of QPs ignores the fact that the vast majority of practical services have videos that can be drastically different, thus there is a certain selection bias towards rather complex sequences, which are atypical for commercial services. It would be ideal to offer the possibility to benchmark encoder performance by using:

- All types of video content; from very simple to very complex.
- Content of different frame rates and different time lengths, while still being able to draw meaningful results.
- The range of qualities/bitrates that are most likely to be used in a commercial service.
- The state of the art in video encoding, which today is represented by convex-hull encoding, i.e. the ability for a system to choose the optimal (resolution, QP) pair that maximizes perceptual quality at a certain bitrate.

Please note that our "Conventional approach" addressed the last aspect (convex-hull encoding), and that in itself should be considered a major improvement over the even more traditional fixed-QP/fixed-resolution test conditions, typically referred to as "Random Access" (RA) in video coding standardization. We would like to acknowledge and highlight that the ongoing efforts within the Alliance for Open Media (AOM) to research new coding tools that can eventually lead to a next-generation royalty-free coding standard has already adopted this testing methodology, called "adaptive streaming", for its CTC [26].

3.2 Combined (unrestricted, continuous) convex hull approach

In order to address the issue of encoding very simple and very complex video sequences, we introduce the same "constant-slope" approach that is the core behind the Dynamic Optimizer [9] framework. To do that, we apply the following steps:

- First, we group all test sequences that belong to the same class (e.g., 1080p) into subclasses that also have the same - or very similar, as is the case with 29.97 and 30fps, for example - frame rate.
- For each subclass, we treat each test sequence as a "shot", part of a longer sequence referred to as the combined sequence that is the virtual "collage" of all sequences in a subclass. For example, if we have 10 test clips of 1920x1080 resolution @29.97fps, with a total number of 1000 frames, we consider the ensemble of these 1000 frames as a single combined video sequence.
- Having different lengths for each "shot" doesn't pose any issue and can be easily dealt with by the DO framework.
- After obtaining the convex hull for each "shot", the multiple convex hulls - 10, for the previous example - are combined as described in the Dynamic Optimizer work [9] using the constant slope principle. This results in a single rate-distortion (or rate-quality, equivalently) curve describing the coding performance over the entire combined sequence.

- Having two such combined rate-distortion curves, one for the anchor and another for a test encoder, allows the evaluation to be based on a single BD-rate figure that captures the performance over the entire ensemble.

The advantages of this methodology, referred to as the "combined convex hull" approach, are that each of the individual test sequences are treated optimally, relative to the rest of the ensemble. Thus, if a very simple sequence requires very low bitrates to achieve high quality, it is only that range of its R-D curve that is actually taken into account in the combined convex hull. On the other hand, a very complex sequence that requires unrealistically high bitrates to achieve high quality is mostly contributing to the ensemble BD-rate through the lower part of its R-D curve.

To better understand the effect of combining shot convex hulls using the constant-slope principle, we show two selected consecutive shots from the "ElFuente" sequence, corresponding to frames 6535-6663 and 6664-6749. The thumbnails from these two shots are shown in Figure 6 while their convex hulls are shown in Figure 7. It is worth pointing out that the first of these two shots contain high detail, but it is mostly static, while the second one is dominated by very high motion.

Figure 6. Two selected consecutive shots from ElFuente - frames 6535-6663 on the left and frames 6664-6749 on the right.

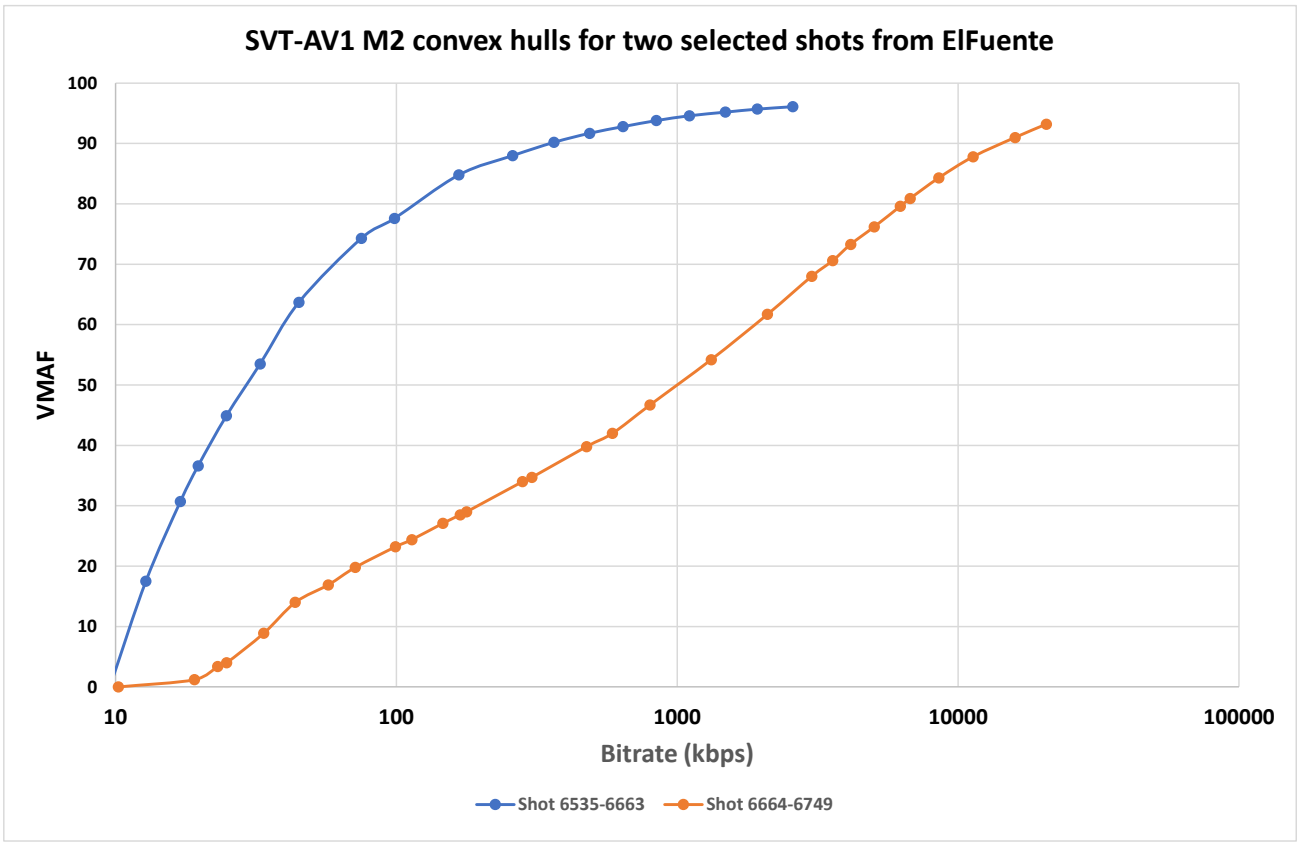

Figure 7. Convex rate-quality hulls for two selected consecutive shots from "ElFuente", using the VMAF quality metric.

When encoding the entire ElFuente sequence using SVT-AV1, preset M2, at 1920x1080 resolution and a fixed QP value of 47, an average bitrate of 1866kbps is achieved, with an average VMAF quality of 91.6. In particular, the two selected shots achieve the rates and qualities as shown on the left part of Table 1.

When using the Dynamic Optimizer for the same encoder and preset, we can obtain an average bitrate of 1545kbps at the same average VMAF quality of 91.6. This corresponds to bitrate savings of about 18% and it showcases the power of the Dynamic Optimizer. In order to achieve this optimal encode, the two selected shots use the following settings and achieve the characteristics shown on the right part of Table 1.

Table 1. Comparison between fixed resolution/QP encoding vs. Dynamic Optimizer encodings.

Shot	Fixed resolution	Fixed QP	Fixed QP bitrate (kbps)	Fixed QP VMAF	Dynamic Optimizer resolution	Dynamic Optimizer CRF	Dynamic Optimizer bitrate (kbps)	Dynamic Optimizer VMAF
6535-6663	1920x1080	47	487	91.7	1920x1080	43	641	92.8
6664-6749	1920x1080	47	2540	62.9	768x432	23	4146	73.3
All (0-14295)	1920x1080	47	1866	91.6	Variable	Variable	1545 (-18%)	91.6

To better understand how these two points have been chosen, in Figure 8, we show the rate-distortion curves for the same two shots, focusing on the bitrate range of 100-6100kbps. The distortion-quality mapping is the one corresponding to the HVMAF (harmonic VMAF) aggregation method, as described in [10], i.e. distortion is roughly proportional to the inverse of quality. One can notice the slope of the easy-to-encode shot (blue curve) around 600kbps is similar to that of the difficult-to-encode shot (red curve) around 4100kbps and equal to approximately ***-0.8*10^(-6)***. This same slope is used to choose the operating points from all other shots, as well, in order to achieve the optimal encoding listed above. Unsurprisingly, the Dynamic Optimizer chose the highest encoding resolution for the first shot (high detail, static) while it chose a much lower resolution but at higher quality (lower QP) for the second shot (very high motion), reflecting a proper perceptual quality tradeoff.

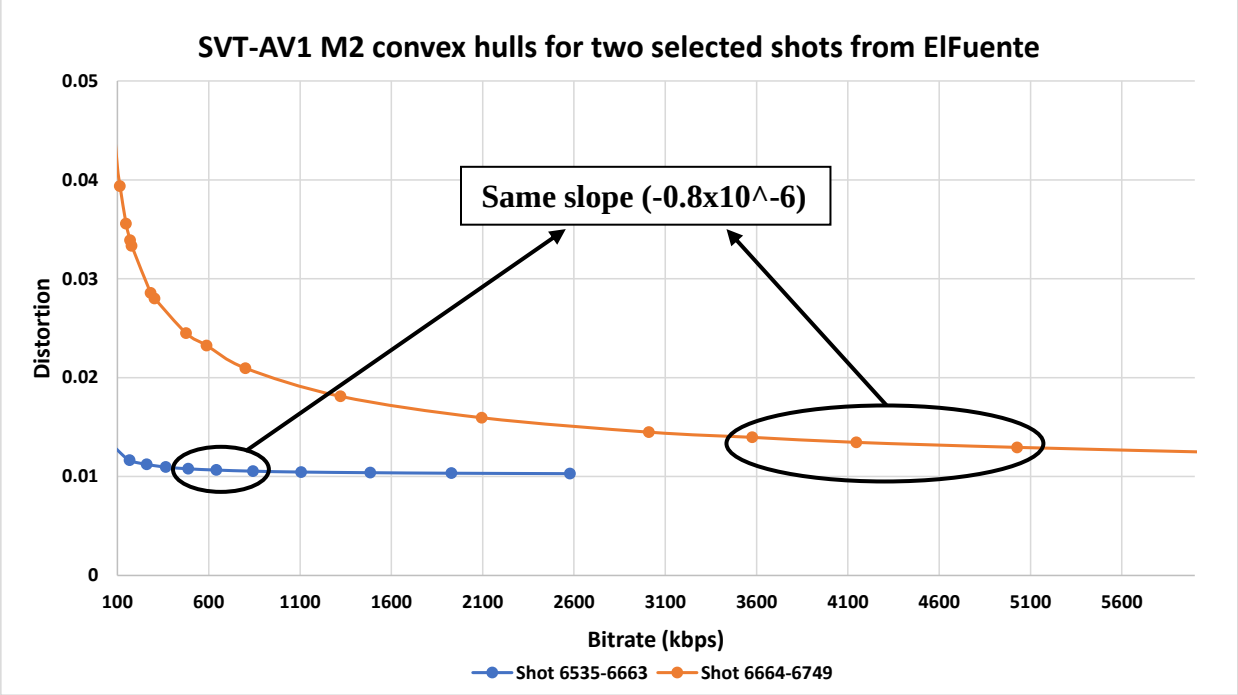

Figure 8. Convex rate-distortion hulls for two selected consecutive shots from "ElFuente", using the VMAF quality metric.

To better understand the differences between the conventional approach of averaging BD-rates obtained between convex hulls and the combined convex hull BD-rate, we use the same two shots from ElFuente in the following graph. In this case, we use the libaom cpu0 encoder as our anchor and the much faster SVT-AV1 M8 preset as the test encoder.

Figure 9. Convex hulls obtained from two shots (6535-6663 and 6664-6749) of ElFuente, using libaom cpu0 and SVT-AV1-M8 encoders; combined convex hulls representing both shots are shown in the middle for libaom cpu0 and SVT-AV1.

We notice right away that the easier shot (6535-6663) has a much smaller BD-rate difference than the more difficult shot (6664-6749). We also notice that the combined convex hull BD-rate difference is closer to that of the difficult shot than it is to that of the easy shot. Intuitively, this makes sense, since the difficult shot requires much higher bitrates to achieve the same level of qualities than the easy shot. In other words, if an encoder saves 10% in bitrate for a shot that is compressed at 100kbps, while saving 20% for another shot that is compressed at 1Mbps, the average savings are (10+200)/(100+1000) = 19%, rather than the arithmetic mean of (10%+20%)/2 = 15%.

In the following Table 2, we show the actual BD-rates measured for these 2 shots and their combined convex hull for 2 speed settings of SVT-AV1, using libaom cpu0 as the anchor and using VMAF as the quality metric.

Table 2. BD-rates for ElFuente shots 6535-6663, 6664-6749 and their combined convex hull for SVT-AV1-M2 and SVT-AV1-M8 speed settings using libaom cpu0 as anchor.

Shot	libaom - cpu0	SVT-AV1-M2	SVT-AV1-M8
6535-6663	0%	-2.15%	9.83%
6664-6749	0%	1.75%	24.68%
Conventional (average)	0%	-0.20%	17.26%
Combined Convex Hull	0%	0.33%	21.92%

We can immediately notice that the conventional average of BD-rates paints a different picture compared to the combined convex hull BD-rate. For the M2 preset of SVT-AV1 it actually reverses from -0.2% to +0.33% while for the faster M8 preset of SVT-AV1 it changes from +17.26% to +21.92%. In both cases, the combined convex hull BD-rate is closer to the corresponding BD-rate for the difficult shot (6664-6749) than to that of the easy shot (6535-6663).

Figures 10 and 11 below show the results of using the combined convex hull approach on the BD-rate vs complexity results for only test sets 2 and 3 as test set 1 did not have enough encoded shots to use in any meaningful analysis. Note that in test set 3, some of the clips that have odd frame rates or did not have as many shots in the frame rate class were taken out of consideration. Also due to time constraints and the high encoding time of VVenC, we were not able to perform the analysis for the VVenC encoder past the conventional convex-hull approach.

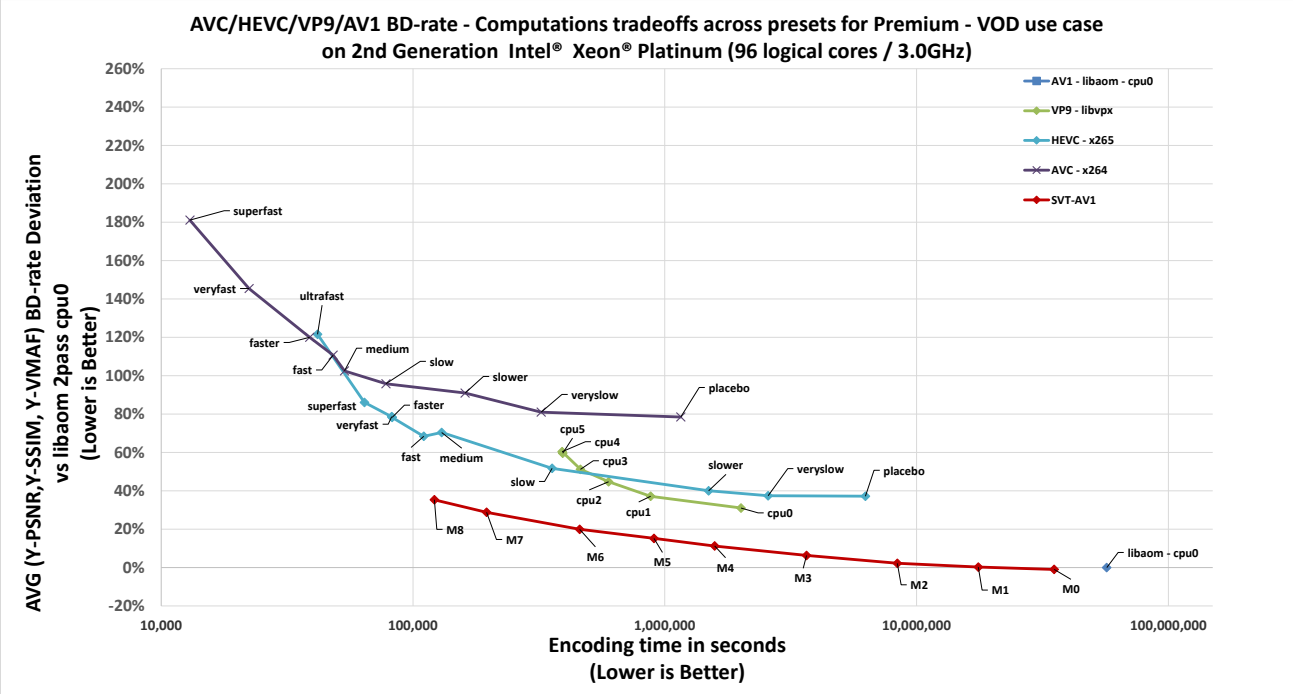

Figure 10. BD-rate vs complexity tradeoffs across presets for testset-2 using the combined convex hull approach.

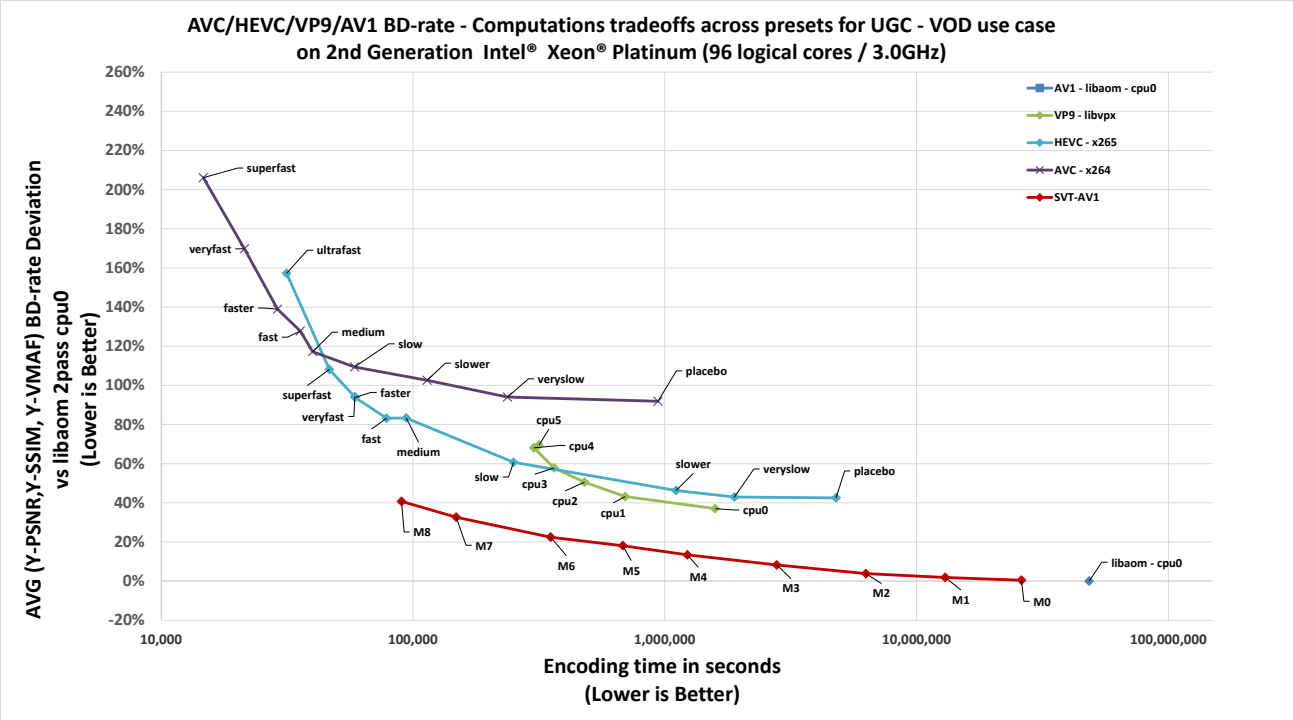

Figure 11. BD-rate vs complexity tradeoffs across preset for testset-3 using the combined convex hull approach.

3.3 Restricted discrete convex hull approach

One of the positive side-effects of the "combined convex hull" approach described above is that the combined single convex hull is very dense and thus makes BD-rate calculations easy and in many cases without the need for a polynomial interpolation. Yet, offering a very big number of operating points is rather unrealistic for a practical video service, where typically one chooses to encode videos by creating a so-called "bitrate ladder", where each step represents a given quality/bitrate for an input video. The number of steps in such a ladder are a system parameter that is optimized taking into account multiple factors, such as amount of storage required, edge cache efficiency for streaming and perceptibility of different representations of the same video content to the human eye.

Taking all the above into consideration and inspired by the previous work on the same topic [10], we propose a third approach based on the "combined convex hull" described previously, which shares the same first steps.

- After combining and obtaining a single Rate-VMAF curve for the combined sequence that represents the concatenation of the test clips with the same resolution/frame rate, 8 operating points are chosen that are as close as possible to the following VMAF quality values: 30, 40, 50, 60, 70, 80, 90, 95.
- The resulting bitrates for these 8 quality targets are then used to obtain the corresponding points on the Rate-PSNR and Rate-SSIM curves.
- These 8 (rate, quality) points - where quality is one of PSNR, VMAF, SSIM - for the anchor are then compared with the 8 points obtained by the same methodology for a test encoder to obtain the corresponding BD-rate figures.

We refer to this approach as "restricted discrete". By restricting bitrates such that we cover VMAF values of [30,95], we ensure that we cover qualities deployed in adaptive bitrate streaming. By choosing values with increments of 10 we uniformly cover that useful range with approximately 1 just-noticeable difference (JND) steps [27].

Figures 12 and 13 below show the results of using the restricted discrete convex hull approach on the BD-rate vs computations results for test sets 2 and 3 over the range of VMAF quality levels (30,40,50,60,70,80,90,95) and the corresponding bitrate ranges of SSIM and PSNR then averaging the BD-rate results from all three metrics.

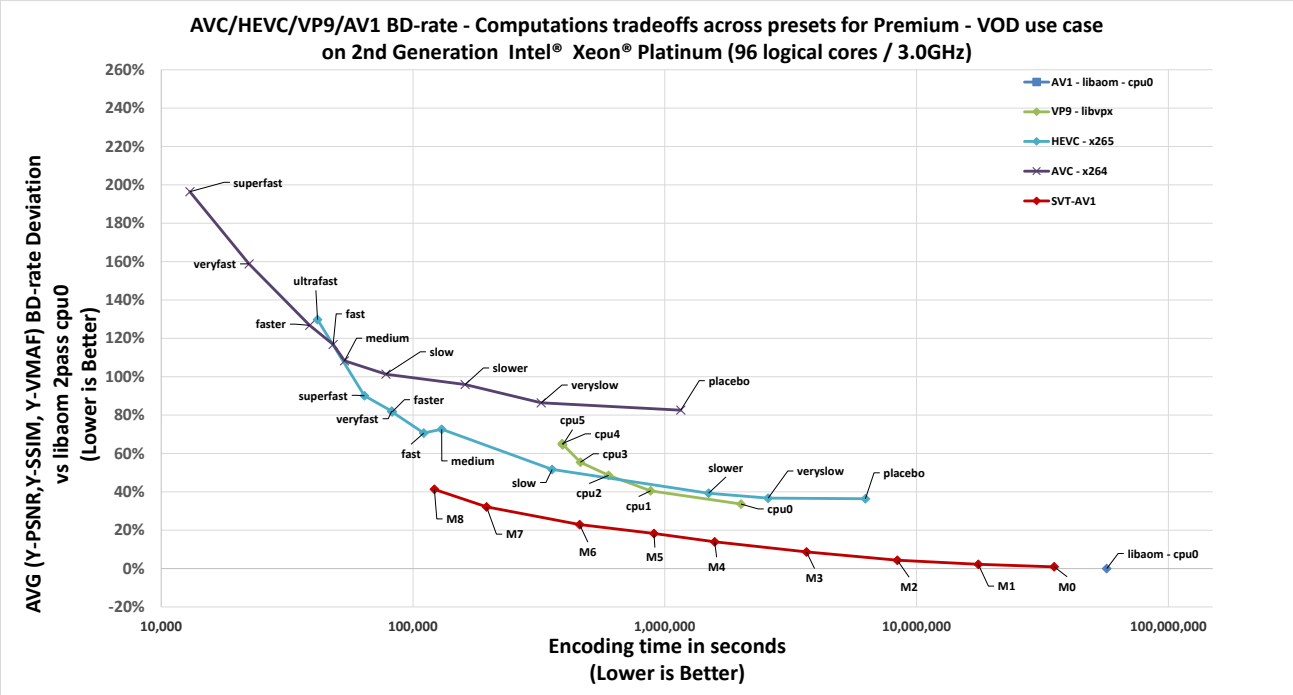

Figure 12. BD-rate vs complexity tradeoffs across presets for testset-2 using the restricted discrete convex hull approach.

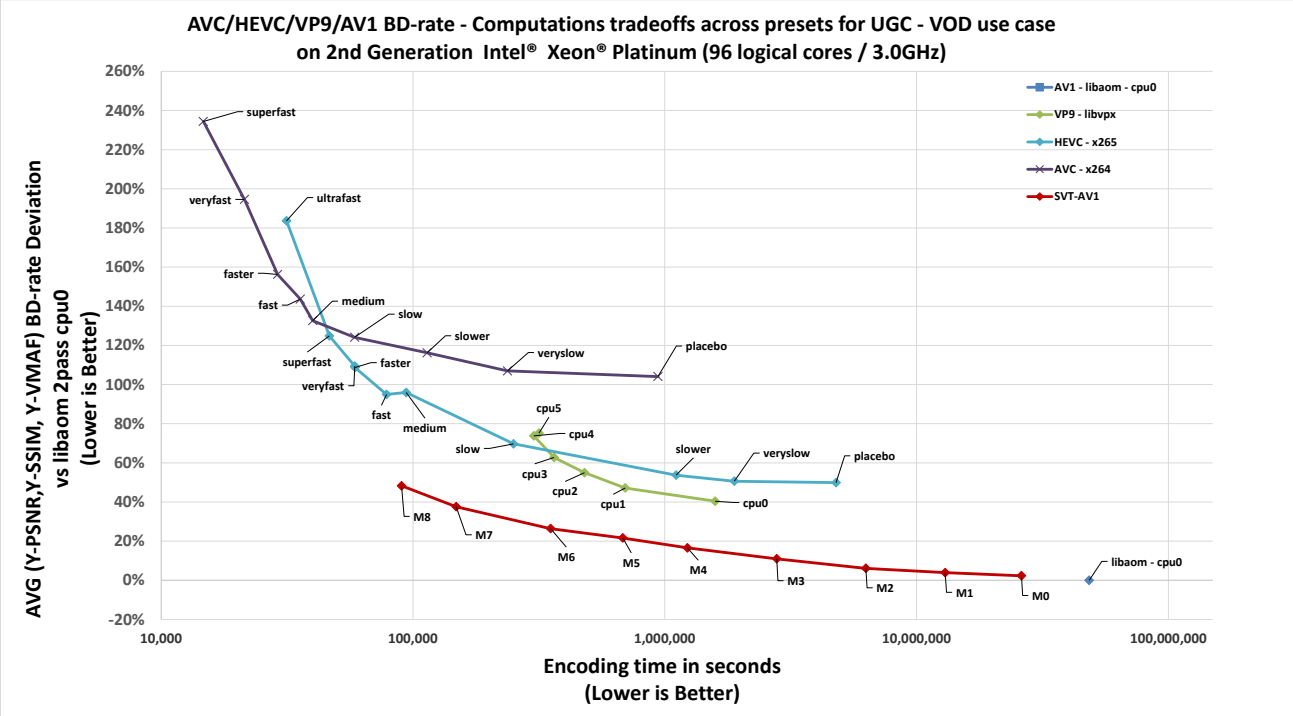

Figure 13. BD-rate vs complexity tradeoffs across presets for testset-3 using the restricted discrete convex hull approach.

3.4 Assessing compute complexity

We previously presented how we measure and aggregate the CPU time required to complete all encodings for all clips as the complexity measure of a given encoder/speed setting. On the other hand, the actual encodings that are part of the convex hull for a given sequence are only a subset of the total number of encodings performed, which means we are using

a subset of the encodings for BD-rate calculations, while we use all encodings for computational complexity calculations. Yet, we argue that this is very reasonable, given that we don't know *a priori* which of the (resolution, QP) pairs would be suboptimal, thus the additional encodes that are not part of the convex hull can be considered as a compute tax to achieve optimality. As such, we have maintained this "total complexity" figure when presenting our results in the previous sections for all three proposed approaches - "conventional", "combined" and "restricted discrete".
We need to understand, though, that there is an opportunity to greatly reduce computational complexity in the "restricted discrete" case, if we have a good way to predict - or, rather, estimate - which (resolution, QP) pairs are to be used in order to achieve the desired quality/bitrate targets. In that case, the complexity of such an optimized bitrate ladder generation would be equal to the complexity of producing only those 8 (resolution, QP) encodings for each clip that correspond to the final 8 selected aggregate discrete points, plus the cost of the predictor/estimator of these encoding parameters.

$$Total\ encoding\ \boldsymbol{cost} = encoder\ parameter\ estimation\ \boldsymbol{cost}\ (constant) + \boldsymbol{cost}\ of\ the\ selected\ 8\ encodings\ per\ shot\ (variable, depending\ on\ encoder\ preset)$$

As shown in previous work [11, 28], such an estimation method is readily available by using the same video encoder in its fastest setting in order to obtain the convex hull per shot and the combined convex hull. This introduces only a minor loss in coding efficiency but with a very significant improvement in encoding speed. We will discuss this method in more detail in section 4.

3.5 Comparing the results from the three approaches

Based on the three approaches described in sections 3.1, 3.2, and 3.3, we can notice a change in the BD-rate data across all encoders. We picked two speed presets from each of the encoders and summarized the results in Table 3 below showing the differences between the results for each of the methods.

Table 3. Average (PSNR / SSIM / VMAF) BD-rate deviation vs libaom cpu0 across evaluation methods.

	Test set 2 - ElFuente			**Test set 3 - UGC**		
Encoder - Preset	**Conventional CH**	**Combined CH**	**Restricted Discrete CH**	**Conventional CH**	**Combined CH**	**Restricted Discrete CH**
x264 - placebo	87.3%	78.5%	82.6%	98.4%	91.8%	104.1%
libvpx - cpu0	33.9%	31.1%	33.7%	37.6%	37.0%	40.4%
x265 - placebo	36.0%	37.2%	36.4%	46.7%	42.5%	49.9%
SVT-AV1 - M0	-0.6%	-0.9%	1.0%	1.7%	0.4%	2.3%
libaom - cpu0	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Encoder - Preset	**Conventional CH**	**Combined CH**	**Restricted Discrete CH**	**Conventional CH**	**Combined CH**	**Restricted Discrete CH**
x264 - veryslow	91.3%	81.0%	86.4%	101.6%	94.0%	107.0%
libvpx - cpu5	64.0%	59.6%	64.5%	74.7%	69.5%	75.1%
x265 - slow	51.3%	51.7%	51.7%	66.8%	60.7%	69.7%
SVT-AV1 - M6	20.7%	20.0%	23.0%	24.9%	22.5%	26.4%
libaom - cpu0	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

In summary, the three approaches presented above offer great insights into understanding the advantages and disadvantages of a new encoder or encoding standard over existing ones. It is also argued that the last proposed approach ("restricted discrete") can better reflect actual gains that a practical adaptive streaming video service should achieve upon deployment of a new encoder in their systems. It should be noted that the representation of each class/subclass of video content, as well as the distribution of different test sequences in each subclass, need to be carefully studied and matched against the actual usage statistics of such a video service.

4. FAST ENCODING PARAMETER SELECTION FOR THE SVT-AV1 ENCODER

As described in Sec. 3 and in [10], the optimal choice of qualities/bitrate operating points for a long video sequence can be achieved with per-shot constant quality encoding, and combining shots by maintaining a constant slope in the R-D domain. We have also presented the "combined convex hull" approach as an alternative way of aggregation across multiple videos for understanding and evaluating encoder performances.

However, computing such a combined convex hull with the Dynamic Optimizer requires encoding the same shot or video at multiple operating points (resolutions/qualities/bitrates). As a result, complexity typically increases by a factor of 6x-10x as compared to traditional fixed-QP encodings, which is less practical in large scale deployment. To address this, the fast encoding parameter selection technique has been proposed in [11, 28]. It leverages the observation that, at the same target quality level, as typically specified by the quantization parameter (QP), a faster and a slower encoder implementations differ primary in bitrate, where the bitrate difference would be proportional to the content complexity, while at the same time the distortion stays roughly the same. Thus a constant multiplicative factor exists for the slopes in the R-D domain between a faster and a slower encoder implementations, and that enables a faster encoder to be used to encode the shots at multiple operating points, and determine the optimal resolutions and target quality levels for individual shots, which would then be used encode the shot again with a slower encoder. In other words, an "analysis" step is done first with a faster encoder and the actual "encode" step is done on a slower encoder, with the encoding parameters obtained from the "analysis" step. In [28], the technique was taken further to enable cross-codec prediction between the "analysis" and the "encode" steps. It was shown that it is possible to map the optimal encoding parameters for an earlier generation codec, such as H.264, to the predicted parameters for a later generation codec, such as VP9 or AV1, by building a mapping based on perceptual quality metrics. The steps involved in the fast encoding parameter selection approach are as follows:

- Compute the "combined convex hull" as described in Sec. 3.3, but with a faster encoder A. This would give us 8 (rate, quality) operating points, with an associated list of (resolution, QP) encoding parameter pairs to achieve each operating point.
- From the list of encoding parameters for each operating point, we can predict the encoding parameters for a slower encoder B, by using the techniques presented in [11, 28].
- From the predicted encoding parameters, we can encode with the slower encoder and obtain the 8 operating points for the bitrate ladder construction.

As can be seen in the performance update in Sec. 2, the fastest available speed setting for SVT-AV1 is roughly at the level of x264 at the "slower" preset, which is more than two orders of magnitude faster than the slowest speed settings on both SVT-AV1 and libaom. Such mode would thus be an ideal candidate encoder for the aforementioned "analysis" step. In the following experiments, we would use the fastest available speed setting on SVT-AV1 currently, M8, to perform the "analysis" step to determine the optimal (resolution, QP) encoding parameter pairs, and apply them with other slower speed settings from M7 to M0 (slowest).

Note that, as described in Sec. 3.4, if the target bitrate ladder has 8 steps, the complexity of producing the optimized bitrate ladder would be the cost of producing these 8 encodings at the chosen target quality levels, with the predicted encoding parameters, plus the cost of performing the prediction of said parameters. Specifically, it would be the cost of producing the 8 particular encodings at, say, M0, plus the cost of producing all the encodings (at all resolutions and QPs) at M8. This would be how we characterize the BD-rate vs. complexity in the following results.

Figure. 14 shows the results of using the M8 preset in the analysis stage on the BD-rate vs complexity tradeoffs of the rest of the SVT-AV1 presets. The total encoding time shown on the x-axis is as discussed above.

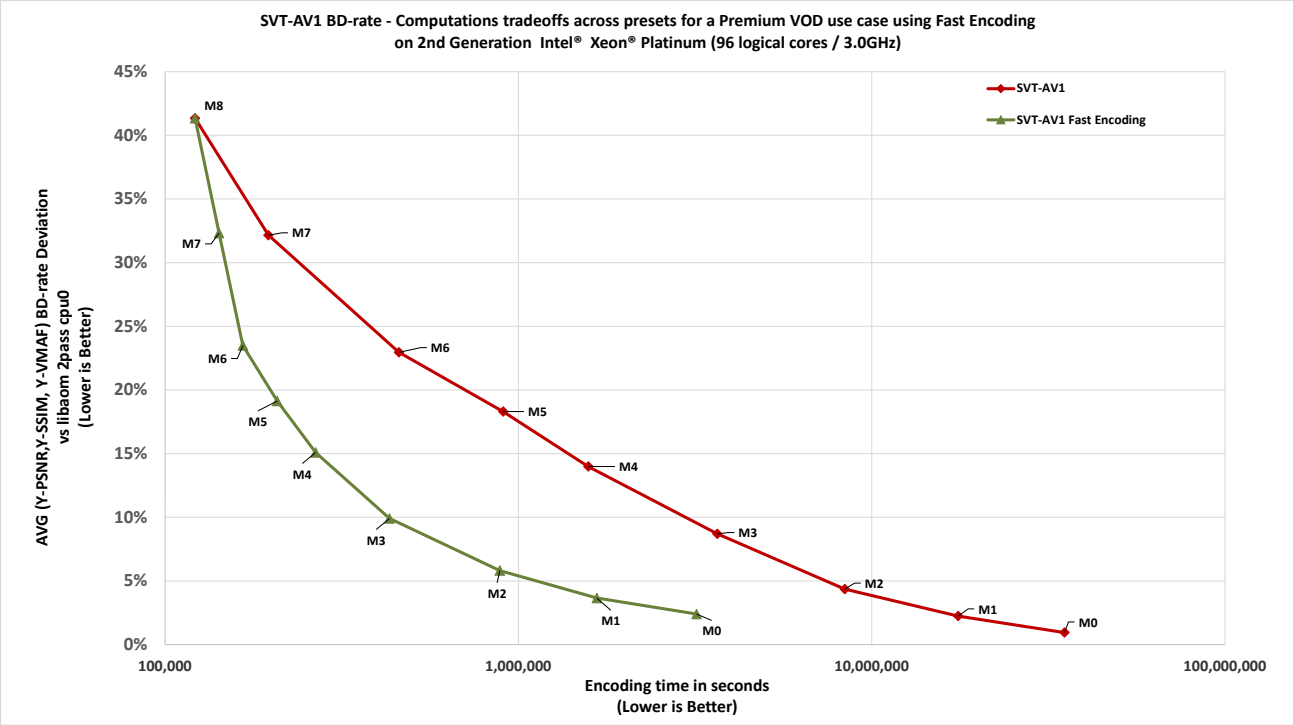

Figure 14. The impact of using fast encoding parameter selection with SVT-AV1 M8 on the SVT-AV1 BD-rate vs complexity tradeoffs across preset for testset-2.

Table 4. Average BD-rate loss and cycle usage vs discrete convex hull.

Preset	BD-rate cost of fast encoding	% of cycles used by fast encoding vs combined CH
M0	1.5%	9.1%
M1	1.4%	9.5%
M2	1.5%	10.6%
M3	1.2%	11.8%
M4	1.1%	16.9%
M5	0.8%	22.9%
M6	0.5%	36.2%
M7	0.2%	72.5%
M8	0.0%	100.0%

Compared to the combined convex hull as presented in [10], the fast encoding parameter selection significantly reduces the total encoding time needed while maintaining the BD-rate loss to less than 1.5% as shown in Table 4. When using preset M8 to select the encoding parameters and then obtaining the final encodings with preset M0, only 9.1% of the combined convex hull total encoding time is used. Both the reduction in total encoding time and the BD-rate loss decrease as faster presets are considered in generating the final encodings. Using M8 for the convex hull parameter selection is shown here as an example that yields the most benefits for higher quality presets. Furthermore, as shown in [11], a much faster encoder such as a hardware-based encoder implementation can be used to benefit even the faster presets of SVT-AV1 in the context of the Dynamic Optimizer framework. This demonstrates that, with the fast encoding parameter selection technique, an encoder that features high speed presets can significantly benefit in reducing the overall encoding time associated with the Dynamic Optimizer framework.

5. CONCLUSIONS

In this paper, the performance of the SVT-AV1 encoder was evaluated in the context of VOD applications against other open-source video encoders using the convex hull approach, at multiple computational complexity operating modes, using multiple video quality objective metrics (PSNR, SSIM, VMAF). Apart from the conventional method of averaging BD-rates across multiple video sequences, we introduced two new evaluation methodologies based on the Dynamic Optimizer framework that offer alternative ways of aggregating results from multiple sequences. The results of these evaluations show that the SVT-AV1 encoder keeps a consistent advantage over encoder implementations from preceding video coding standards in terms of quality vs. computational complexity tradeoff across various test sets and evaluation approaches. The open-source implementation of the newer VVC standard offers some limited advantage over SVT-AV1, but only at its highest compute complexity range.

The three evaluation methodologies provided broadly similar results, without observing any consistent global trends, even though the wealth of data we obtained over multiple metrics, datasets, encoders and encoder settings deserves more detailed analysis that can help us better understand whether such trends exist and how to interpret them.

The fast encoding parameter selection approach that uses a faster encoder preset to determine optimal encoder settings within the dynamic optimizer framework does show a significant reduction in computational complexity as compared to other dynamic optimizer-based approaches. Our future work includes further refining the evaluated approaches for different content, encoders and other encoding modes and parameters.

ACKNOWLEDGMENTS

The authors are very thankful to P'sao Pollard-Flamand and David Yeung for their help in running the experiments and scripting the workflow along with the core SVT team at Intel for their continuous hard work and contributions to the SVT-AV1 project.

REFERENCES

[1] Advanced video coding for generic audiovisual services, *ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC)* (2003).
[2] High efficiency video coding, *ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC)* (2013).
[3] Grange, A., de Rivaz, P. and Hunt, J., "VP9 bitstream and decoding process specification," https://www.webmproject.org/vp9/ , (2016).
[4] de Rivaz, P. and Haughton, J., "AV1 bitstream & decoding process specification,", https://aomediacodec.github.io/av1-spec/ (2019).
[5] Bross, B., Chen, J., Liu, S. and Wang, Y.-K., "Versatile Video Coding (Draft 10), "*ITU-T and ISO/IEC JVET-S2001* (2020).
[6] 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.
[7] 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.
[8] Comp, L. "Why More Companies Are Using the Open Source AV1 Video Codec". https://itpeernetwork.intel.com/open-source-svt-av1/#gs.zg5w90.
[9] 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).
[10] 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.

[11] 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.
[12] Netflix ElFutente test video, https://netflixtechblog.com/engineers-making-movies-aka-open-source-test-content-f21363ea3781.
[13] YouTube UGC test videos, https://media.withyoutube.com/.
[14] x264, code repository - open-source AVC encoder software, https://code.videolan.org/videolan/x264.
[15] x265, open source HEVC software encoder, https://www.videolan.org/developers/x265.html.
[16] libvpx, code repository - open-source VP9 encoder/decoder software, https://github.com/webmproject/libvpx.
[17] libaom, code repository - open-source AV1 encoder/decoder software, https://aomedia.googlesource.com/aom/.
[18] SVT-AV1, code repository - open-source SVT-AV1 encoder software, https://gitlab.com/AOMediaCodec/SVT-AV1.
[19] VVenC, open source VVC software encoder, https://github.com/fraunhoferhhi/vvenc.
[20] Constant Rate Factor, https://trac.ffmpeg.org/wiki/Encode/H.264#crf.
[21] Amazon EC2 C5 Instances, https://aws.amazon.com/ec2/instance-types/c5/.
[22] Tange, O., "GNU Parallel 2018," March 2018, https://doi.org/10.5281/zenodo.1146014.
[23] Libvmaf, https://github.com/Netflix/vmaf/tree/master/libvmaf.
[24] Convex hull test app, https://github.com/ikatsavounidisFB/convex_hull.
[25] Bjøntegaard, G., "Calculation of average PSNR differences between RD-Curves," ITU-T SG16/Q6, Doc. VCEG-M33, Austin, Apr. 2001, http://wftp3.itu.int/av-arch/video-site/0104_Aus/.
[26] Zhao, X., Lei, Z., Norkin, A., Daede, T. and Tourapis, A., "AV2 CTC," https://groups.aomedia.org/g/wg-codec/files/OutputDocuments/B2021/CWG-B005o_AV2_CTC/CWG-B005o_AV2_CTC_v1.pdf.
[27] Wang, H., Katsavounidis, I., Zhou, J., Park, J., Lei, S., Zhou, X., Pun, M.-O., Jin, X., Wang, R., Wang, X, Zhang, J., Kwong, S. and Kuo C.-C.J., "VideoSet: A large-scale compressed video quality dataset based on JND measurement," Journal of Visual Communication and Image Representation, 46, pp. 292-302 (2017).
[28] 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).