An Overview of Core Coding Tools in the AV1 Video Codec

Yue Chen*, Debargha Murherjee*, Jingning Han*, Adrian Grange*, Yaowu Xu*, Zoe Liu*, Sarah Parker*, Cheng Chen*, Hui Su*, Urvang Joshi*, Ching-Han Chiang*, Yunqing Wang*, Paul Wilkins*, Jim Bankoski*, Luc Trudeau†, Nathan Egge†, Jean-Marc Valin†, Thomas Davies‡, Steinar Midtskogen‡, Andrey Norkin§ and Peter de Rivaz¶
*Google, USA
†Mozilla, USA
‡Cisco, UK and Norway
§Netflix, USA
¶Argon Design, UK

***Abstract*—AV1 is an emerging open-source and royalty-free video compression format, which is jointly developed and finalized in early 2018 by the Alliance for Open Media (AOMedia) industry consortium. The main goal of AV1 development is to achieve substantial compression gain over state-of-the-art codecs while maintaining practical decoding complexity and hardware feasibility. This paper provides a brief technical overview of key coding techniques in AV1 along with preliminary compression performance comparison against VP9 and HEVC.**

***Index Terms*—Video Compression, AV1, Alliance for Open Media, Open-source Video Coding**

I. Introduction

Video applications have become ubiquitous on the internet over the last decade, with modern devices driving rapid growth in the consumption of high resolution, high quality content. Services such as video-on-demand and conversational video are predominant bandwidth consumers, that impose severe challenges on delivery infrastructure and hence create an even stronger need for high efficiency video compression technology. On the other hand, a key factor in the success of the web is that the core technologies, for example, HTML, web browsers (Firefox, Chrome, etc.), and operating systems like Android, are open and freely implemetable. Therefore, in an effort to create an open video format at par with the leading commercial choices, in mid 2013, Google launched and deployed the VP9 video codec[1]. VP9 is competitive in coding efficiency with the state-of-the-art royalty-bearing H.265/HEVC[2] codec, while considerably outperforming the most commonly used format H.264/AVC[3] as well as its own predecessor VP8[4].

However, as the demand for high efficiency video applications rose and diversified, it soon became imperative to continue the advances in compression performance. To that end, in late 2015, Google co-founded the Alliance for Open Media (AOMedia)[5], a consortium of more than 30 leading hi-tech companies, to work jointly towards a next-generation open video coding format called AV1.

The focus of AV1 development includes, but is not limited to achieving: consistent high-quality real-time video delivery, scalability to modern devices at various bandwidths, tractable computational footprint, optimization for hardware, and flexibility for both commercial and non-commercial content. The codec was first initialized with VP9 tools and enhancements, and then new coding tools were proposed, tested, discussed and iterated in AOMedia's codec, hardware, and testing workgroups. As of today, the AV1 codebase has reached the final bug-fix phase, and already incorporates a variety of new compression tools, along with high-level syntax and parallelization features designed for specific use cases. This paper will present the key coding tools in AV1 that provide the majority of the almost 30% reduction in average bitrate compared with the most performant libvpx VP9 encoder at the same quality.

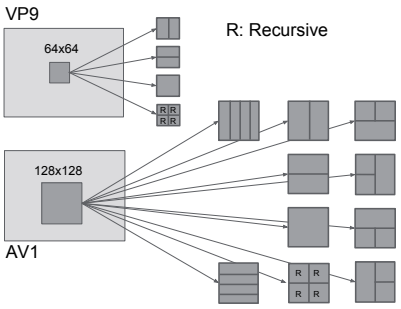

Fig. 1. Partition tree in VP9 and AV1

II. AV1 Coding Techniques

A. Coding Block Partition

VP9 uses a 4-way partition tree starting from the 64×64 level down to 4×4 level, with some additional restrictions for blocks 8×8 and below as shown in the top half of Fig.1. Note that partitions designated as R refer to as recursive in that the same partition tree is repeated at a lower scale until we reach the lowest 4×4 level.

AV1 not only expands the partition-tree to a 10-way structure as shown in the same figure, but also increases the largest size (referred to as superblock in VP9/AV1 parlance) to start from 128×128. Note that this includes 4:1/1:4 rectangular partitions that did not exist in VP9. None of the rectangular partitions can be further subdivided. In addition, AV1 adds more flexibility to the use of partitions below 8×8 level, in the sense that 2×2 chroma inter prediction now becomes possible on certain cases.

B. Intra Prediction

VP9 supports 10 intra prediction modes, including 8 directional modes corresponding to angles from 45 to 207 degrees, and 2 non-directional predictor: DC and true motion (TM) mode. In AV1, the potential of an intra coder is further explored in various ways: the granularity of directional extrapolation are upgraded, non-directional predictors are enriched by taking into account gradients and evolving correlations, coherence of luma and chroma signals is exploited, and tools are developed particularly for artificial content.

1) Enhanced Directional Intra Prediction: To exploit more varieties of spatial redundancy in directional textures, in AV1, directional intra modes are extended to an angle set with finer granularity. The original 8 angles are made nominal angles, based on which fine angle variations in a step size of 3 degrees are introduced, i.e., the prediction angle is presented by a nominal intra angle plus an angle delta, which is -3 $\sim$ 3 multiples of the step size. To implement directional prediction modes in AV1 via a generic way, the 48 extension modes are realized by a unified directional predictor that links each pixel to a

reference sub-pixel location in the edge and interpolates the reference pixel by a 2-tap bilinear filter. In total, there are 56 directional intra modes enabled in AV1.

2) Non-directional Smooth Intra Predictors: AV1 expands on non-directional intra modes by adding 3 new smooth predictors SMOOTH_V, SMOOTH_H, and SMOOTH, which predict the block using quadratic interpolation in vertical or horizontal directions, or the average thereof, after approximating the right and bottom edges as the rightmost pixel in the top edge and the bottom pixel in the left edge. In addition, the TM mode is replaced by the PAETH predictor: for each pixel, we copy one from the top, left and top-left edge references, which has the value closest to (top + left - topleft), meant to adopt the reference from the direction with the lower gradient.

3) Recursive-filtering-based Intra Predictor: To capture decaying spatial correlation with references on the edges, FILTER_INTRA modes are designed for luma blocks by viewing them as 2-D non-separable Markov processes. Five filter intra modes are pre-designed for AV1, each represented by a set of eight 7-tap filters reflecting correlation between pixels in a 4×2 patch and 7 neighbors adjacent to it. An intra block can pick one filter intra mode, and be predicted in batches of 4×2 patches. Each patch is predicted via the selected set of 7-tap filters weighting the neighbors differently at the 8 pixel locations. For those patches not fully attached to references on block boundary, predicted values of immediate neighbors are used as the reference, meaning prediction is computed recursively among the patches so as to combine more edge pixels at remote locations.

4) Chroma Predicted from Luma: Chroma from Luma (CfL) is a chroma-only intra predictor that models chroma pixels as a linear function of coincident reconstructed luma pixels. Reconstructed luma pixels are subsampled into the chroma resolution, and then the DC component is removed to form the AC contribution. To approximate chroma AC component from the AC contribution, instead of requring the decoder to imply scaling parameters as in some prior art, AV1-CfL determines the parameters based on the original chroma pixels and signals them in the bitstream. This reduces decoder complexity and yields more precise predictions. As for the DC prediction, it is computed using intra DC mode, which is sufficient for most chroma content and has mature fast implementations. More details of AV1-CfL tool can be found in [6].

5) Color Palette as a Predictor: Sometimes, especially for artificial videos like screen capture and games, blocks can be approximated by a small number of unique colors. Therefore, AV1 introduces palette modes to the intra coder as a general extra coding tool. The palette predictor for each plane of a block is specified by (i) a color palette, with 2 to 8 colors, and (ii) color indices for all pixels in the block. The number of base colors determines the trade-off between fidelity and compactness. The color indices are entropy coded using the neighborhood-based context.

6) Intra Block Copy: AV1 allows its intra coder to refer back to previously reconstructed blocks in the same frame, in a manner similar to how inter coder refers to blocks from previous frames. It can be very beneficial for screen content videos which typically contain repeated textures, patterns and characters in the same frame. Specifically, a new prediction mode named IntraBC is introduced, and will copy a reconstructed block in the current frame as prediction. The location of the reference block is specified by a displacement vector in a way similar to motion vector compression in motion compensation. Displacement vectors are in whole pixels for the luma plane, and may refer to half-pel positions on corresponding chrominance planes, where bilinear filtering is applied for sub-pel interpolation.

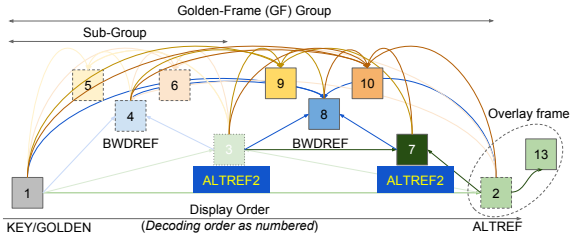

Fig. 2. Example multi-layer structure of a golden-frame group

C. Inter Prediction

Motion compensation is an essential module in video coding. In VP9, up to 2 references, amongst up to 3 candidate reference frames, are allowed, then the predictor either operates a block-based translational motion compensation, or averages two of such predictions if two references are signalled. AV1 has a more powerful inter coder, which largely extends the pool of reference frames and motion vectors, breaks the limitation of block-based translational prediction, also enhances compound prediction by using highly adaptable weighting algorithms as well as sources.

1) Extended Reference Frames: AV1 extends the number of references for each frame from 3 to 7. In addition to VP9's LAST(nearest past) frame, GOLDEN(distant past) frame and ALTREF(temporal filtered future) frame, we add two near past frames (LAST2 and LAST3) and two future frames (BWDREF and ALTREF2)[7]. Fig.2 demonstrates the multi-layer structure of a golden-frame group, in which an adaptive number of frames share the same GOLDEN and ALTREF frames. BWDREF is a look-ahead frame directly coded without applying temporal filtering, thus more applicable as a backward reference in a relatively shorter distance. ALTREF2 serves as an intermediate filtered future reference between GOLDEN and ALTREF. All the new references can be picked by a single prediction mode or be combined into a pair to form a compound mode. AV1 provides an abundant set of reference frame pairs, providing both bi-directional compound prediction and uni-directional compound prediction, thus can encode a variety of videos with dynamic temporal correlation characteristics in a more adaptive and optimal way.

2) Dynamic Spatial and Temporal Motion Vector Referencing: Efficient motion vector (MV) coding is crucial to a video codec because it takes a large portion of the rate cost for inter frames. To that end, AV1 incorporates a sophisticated MV reference selection scheme to obtain good MV references for a given block by searching both spatial and temporal candidates. AV1 not only searches a deeper spatial neighborhood than VP9 to construct a spatial candidate pool, but also utilizes a temporal motion field estimation mechanism to generate temporal candidates. The motion field estimation process works in three stages: motion vector buffering, motion trajectory creation, and motion vector projection. First, for coded frames, we store the reference frame indices and the associated motion vectors. Before decoding a current frame, we examine motion trajectories, like MV_{Ref2} in Fig.3 pointing a block in frame $Ref2$ to somewhere in frame $Ref0_{Ref2}$, that possibly pass each 64×64 processing unit, by checking the collocated 192×128 buffered motion fields in up to 3 references. By doing so, for any 8×8 block, all the trajectories it belongs to are recorded. Next, at the coding block level, once the reference frame(s) have been determined, motion vector candidates are derived by linearly project passing motion trajectories onto the desired reference frames, e.g., converting MV_{Ref2} in Fig.3 to MV_0 or MV_1. Once all spatial and temporal candidates have been aggregated in the pool, they are sorted, merged and ranked to obtain up to 4 final candidates[8]. The scoring scheme relies on calculating a

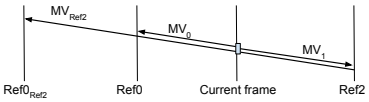

Fig. 3. Motion field estimation

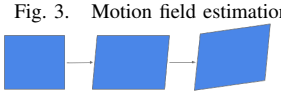
Fig. 4. Affine warping in two shears

likelihood of the current block having a particular MV as a candidate. To code a MV, AV1 signals the index of a selected reference MV from the list, followed by encoding a delta if needed. In practice, the combination of the reference MV and the delta is signaled through modes, as in VP9.

3) Overlapped Block Motion Compensation (OBMC): OBMC can largely decrease prediction errors near block edges by smoothly combining predictions created from adjacent motion vectors. In AV1, a 2-sided causal overlapping algorithm is designed to make OBMC easily fit in the advanced partitioning framework [9]. It progressively combines the block-based prediction with secondary inter predictors in the above edge and then in the left, by applying predefined 1-D filters in vertical and horizontal directions. The secondary predictors only operate in restricted overlapping regions in top/left halves of the current block, so that they do not tangle with each other on the same side. AV1 OBMC is only enabled for blocks using a single reference frame, and only works with the first predictor of any neighbor with two reference frames, therefore the worst-case memory bandwidth is same as what is demanded by a traditional compound predictor.

4) Warped Motion Compensation: Warped motion models are explored in AV1 by enabling two affine prediction modes, global and local warped motion compensation [10]. The global motion tool is meant for handling camera motions, and allows conveying motion models explicitly at the frame level for the motion between a current frame and any of its references. The local warped motion tool aims to describe varying local motion implicitly using minimal overhead, by deriving the model parameters at the block level from 2-D displacements signalled by motion vectors assigned to the causal neighborhood. Both coding tools compete with translational modes at block level, and are selected only if there is an advantage in RD cost. More importantly, affine warpings in AV1 is limited to small degrees so that they can be implemented efficiently in SIMD and hardware by a horizontal shear followed by a vertical shear (Fig.4), with 8-tap interpolation filters being used for each shear at 1/64 pixel precision.

5) Advanced Compound Prediction: A collection of new compound prediction tools is developed for AV1 to make its inter coder more versatile. In this section, any compound prediction operation can be generalized for a pixel (i, j) as: $p_f(i, j) = m(i, j)p_1(i, j) + (1 - m(i, j))p_2(i, j)$, where p_1 and p_2 are two predictors, and p_f is the final joint prediction, with the weighting coefficients $m(i, j)$ in $[0, 1]$ that are designed for different use cases and can be easily generated from predefined tables. [11]

- *Compound wedge prediction*: Boundaries of moving objects are often difficult to be approximated by on-grid block partitions. The solution in AV1 is to predefine a codebook of 16 possible wedge partitions and to signal the wedge index in the bitstream when a coding unit chooses to be further partitioned in such a way. 16-ary shape codebooks containing partition orientations that are either horizontal, vertical or oblique with slopes ± 2 or ± 0.5, are designed for both square and rectangular blocks as shown in Fig.5. To mitigate spurious high frequency components, which often are produced by directly juxtaposing two predictors, soft-cliff-shaped 2-D wedge masks are employed to smooth the edges around the intended partition, i.e. $m(i, j)$ is close to 0.5 around the edges, and gradually transforms into binary weights at either end.

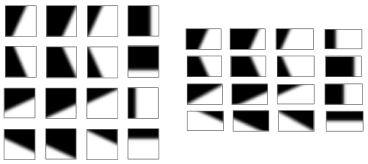
Fig. 5. Wedge codebooks for square and rectangular blocks

- *Difference-modulated masked prediction*: Straight partitions like wedges are not always effective to separate objects. Therefore AV1 compound predictor can also create non-uniform weighting by differing content from values of the two predictors. Specifically, the pixel difference between p_1 and p_2 is used to modulate weights on top of a base value. The mask is generated by $m(i, j) = b + a|p_1(i, j) - p_2(i, j)|$ where b controls how strongly one predictor is weighted over the other within the differing regions and a scaling factor a ensures a smooth modulation.
- *Frame distance based compound prediction*: Besides non-uniform weights, AV1 also utilizes a modified uniform weighting scheme by accounting for frame distances. Frame distance is defined as the absolute difference between timestamps of two frames. It naturally indicates the reliability of a motion compensated block copied from different references. When a frame distance based compound mode is selected, let d_1 and d_2 ($d_1 \geq d_2$) represent distances from current frame to reference frames, from which p_1 and p_2 are computed, the whole block will share a constant weight m. Instead of using direct linear weighting, AV1 defines quantized weights modulated by d_1/d_2, which balances the tradeoff between temporal correlation and quantization noises in the reconstructed references.
- *Compound inter-intra prediction*: Compound inter-intra prediction modes, which combine intra prediction p_1 and single-reference inter prediction p_2, are developed to handle areas with emerging new content and old objects mixed. For the intra part, 4 frequently-used intra modes are supported. The mask $m(i, j)$ incorporates two types of smoothing functions: (i) smooth wedge masks similar to what is designed for wedge inter-inter modes, (ii) mode-dependent masks that weight p_1 in a decaying pattern oriented by the primary direction of the intra mode.

D. Transform Coding

1) Transform Block Partition: Instead of enforcing fixed transform unit sizes as in VP9, AV1 allows luma inter coding blocks to be partitioned into transform units of multiple sizes that can be represented by a recursive partition going down by up to 2 levels. To incorporate AV1's extended coding block partitions, we support square, 2:1/1:2, and 4:1/1:4 transform sizes from 4×4 to 64×64. For chroma blocks, only the largest possible transform units are allowed.

2) Extended Transform Kernels: A richer set of transform kernels is defined for both intra and inter blocks in AV1. The full 2-D kernel set consists of 16 horizontal/vertical combinations of DCT, ADST, flipADST and IDTX[12]. Besides DCT and ADST that has been used in VP9, flipADST applies ADST in reverse order, and identity transform(IDTX) means skipping transform coding in a certain direction so is particularly beneficial for coding sharp edges. As block sizes get larger, some of the kernels begin to act similarly, thus the kernel sets are gradually reduced as transform sizes increase.

E. Entropy Coding

1) Multi-symbol Entropy Coding: VP9 used a tree-based boolean non-adaptive binary arithmetic encoder to encode all syntax elements.

AV1 moves to using a symbol-to-symbol adaptive multi-symbol arithmetic coder. Each syntax element in AV1 is a member of a specific alphabet of N elements, and a context consists of a set of N probabilities together with a count to facilitate fast early adaptation. The probabilities are stored as 15 bit cumulative distribution functions (CDFs). The higher precision than a binary arithmetic encoder, enables tracking probabilities of less common elements of an alphabet accurately. Probabilities are adapted by simple recursive scaling, with an update factor based on the alphabet size. Since the symbol bitrate is dominated by encoding coefficients, motion vectors and prediction modes, all of which use alphabets larger than 2, this design in effect achieves more than a factor of 2 reduction in throughput for typical coding scenarios over pure binary arithmetic coding.

In hardware, the complexity is dominated by throughput and size of the core multiplier that rescales the arithmetic coding state interval. The higher precision required for tracking probabilities is not actually required for coding. This allows reducing the multiplier size substantially by rounding from 16×15 bits to an 8×9 bit multiplier. This rounding is facilitated by enforcing a minimum interval size, which in turn allows a simplified probability update in which values may become zero. In software, the operation count is more important than complexity, and reducing throughput and simplifying updates correspondingly reduces fixed overheads of each coding/decoding operation.

2) Level Map Coefficient Coding: In VP9, the coding engine processes each quantized transform coefficient sequentially following the scan order. The probability model used for each coefficient is contexted on the previously coded coefficient levels, its frequency band, transform block sizes, etc. To properly capture the coefficient distribution in the vast cardinality space, AV1 alters to a level map design for sizeable transform coefficient modelling and compression[13]. It builds on the observation that the lower coefficient levels typically account for the major rate cost.

For each transform unit, AV1 coefficient coder starts with coding a skip sign, which will be followed by the transform kernel type and the ending position of all non-zero coefficients if the transform coding is not skipped. Then for the coefficient values, instead of uniformly assigning context models for all coefficient levels, it breaks the levels into different planes. The lower level plane corresponds to coefficient levels between 0 and 2, whereas the higher level plane takes care of the levels above 2. Such separation allows one to assign a rich context model to the lower level plane, which fully accounts the transform dimension, block size, and neighboring coefficient information for improved compression efficiency, at modest context model size. The higher level plane uses a reduced context model for levels between 3 to 15, and directly codes the residuals above level 15 using Exp-Golomb code.

F. In-Loop Filtering Tools and Post-processing

AV1 allows several in-loop filtering tools to be applied successively to a decoded frame. The first stage is the deblocking filter which is roughly the same as the one used in VP9 with minor changes. The longest filter is reduced to a 13-tap one from 15-taps in VP9. Further there is now more flexibility in signaling separate filtering levels horizontally and vertically for luma and for each chroma plane, as well as the ability to change levels from superblock to superblock. Other filtering tools in AV1 are described below.

1) Constrained Directional Enhancement Filter (CDEF): CDEF is a detail-preserving deringing filter designed to be applied after deblocking that works by estimating edge directions followed by applying a non-separable non-linear low-pass directional filter of size 5×5 with 12 non-zero weights [14]. To avoid extra signaling, the decoder computes the direction per 8×8 block using a normative fast search algorithm that minimizes the quadratic error from a perfect directional pattern.

The filter is only applied to blocks with a coded prediction residue. The filter can be expressed as:

$$y(i,j) = R(x(i,j) + g(\sum_{m,n \in N} w_{m,n} f(x\,(m,n) - x(i,j)\,, S, D)))$$

where N contains the pixels in the neighbourhood of $x(i,j)$ with the non-zero weights $w_{m,n}$, $f()$ and $g()$ are nonlinear functions described below, and $R(x)$ rounds x to the nearest integer towards zero. The $f()$ function modifies the difference between the pixel to be filtered and a neighbor, and is determined by two parameters, a strength S and a damping value D, that are specified at the 64×64 block level and frame level respectively. The strength S clamps the maximum difference allowed minus a ramp-down controlled by D. The $g()$ function clips the modification of the pixel x to be filtered to the greatest difference between x and any $x(m,n)$ in the support area to preserve the low-pass nature of the filter.

2) Loop Restoration Filters: AV1 adds a set of tools for application in-loop after CDEF, that are selected in a mutually exclusive manner in units of what is called the loop-restoration unit (LRU) of selectable size 64×64, 128×128 or 256×256. Specifically, for each LRU, AV1 allows selecting between one of two filters [15] as follows.

- *Separable symmetric normalized Wiener filter*: Pixels are filtered with a 7×7 separable Wiener filter, the coefficients of which are signaled in the bit-stream. Because of the normalization and symmetry constraints, only three parameters need to be sent for each horizontal/vertical filter. The encoder makes a smart optimization to decide the right filter taps to use, but the decoder simply applies the filter taps as received from the bit-stream.
- *Dual self-guided filter*: For each LRU, the decoder first applies two cheap integerized self-guided filters of support size 3×3 and 5×5 respectively with noise parameters signaled in the bitstream. (Note self-guided means the guide image is the same as the image to be filtered). Next, the outputs from the two filters, r_1 and r_2, are combined with weights (α, β) also signaled in the bit-stream to obtain the final restored LRU as $x + \alpha(r_1 - x) + \beta(r_2 - x)$, where x is the original degraded LRU. Even though r_1 and r_2 may not necessarily be good by themselves, an appropriate choice of weights on the encoder side can make the final combined version much closer to the undegraded source.

3) Frame Super-resolution: AV1 adds a new frame super-resolution coding mode that allows the frame to be coded at lower spatial resolution and then super-resolved normatively in-loop to full resolution before updating the reference buffers. While such methods are known to offer perceptual advantages at very low bit-rates, most super-resolution methods in the image processing literature are far too complex for in-loop operation in a video codec. In AV1, to keep operations computationally tractable, the super-resolving process is decomposed into linear upscaling followed by applying the loop restoration tool at higher spatial resolution. Specifically, the Wiener filter is particualrly good at super-resolving and recovering lost high frequencies. The only additional normative operation is then a linear upscaling prior to use of loop restoration. Further, in order to enable a cost-effective hardware implementation with no overheads in line-buffers, the upscaling/downscaling is constrained to operate only horizontally. Fig. 6 depicts the overall architecture of the in-loop filtering pipeline when using frame super-resolution, where CDEF operates on the coded (lower) resolution, but loop restoration operates after the linear upscaler has expanded the image horizontally to resolve part of the higher frequencies lost.

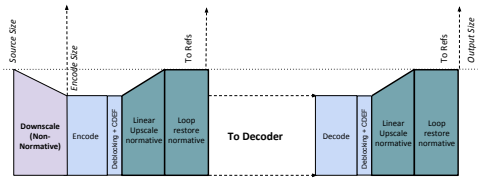

Fig. 6. In-loop Filtering pipeline with optional super-resolution

4) Film Grain Synthesis: Film grain synthesis in AV1 is normative post-processing applied outside of the encoding/decoding loop. Film grain, abundant in TV and movie content, is often part of the creative intent and needs to be preserved while encoding. Its random nature makes it difficult to compress with traditional coding tools. Instead, the grain is removed from the content before compression, its parameters are estimated and sent in the AV1 bitstream. The decoder synthesizes the grain based on the parameters and adds it to the reconstructed video.

The grain is modeled as an autoregressive (AR) process with up to 24 AR-coefficients for luma and 25 for each chroma component. These coefficients are used to generate 64×64 luma and 32×32 chroma grain templates. 32×32 luma grain patches are then taken from random positions in the template and applied to the video. Discontinuities between the patches can be mitigated by an optional overlap. Since film grain strength varies with signal intensity, each grain sample is scaled accordingly[16]. The scaling is modeled with a piece-wise linear function for each color component.

For grainy content, film grain synthesis significantly reduces the bitrate necessary to reconstruct the grain. This tool is not used in the comparison in Sec.III since it does not generally improve objective quality metrics because of mismatch in single grain positions.

III. Performance Evaluation

We compare the coding performance obtained with AV1 (Apr 3, 2018 version) on AOMedia's open test bench AWCY[17], against those obtained by the best libvpx VP9 encoder (Feb 8, 2018 version) and x265 (v2.7). The three codecs are operated on AWCY *objective-1-fast* test set[18], which includes 4:2:0 8-bit videos of various resolution and types: 12 normal 1080p clips, 4 1080p screen content clips, 7 720p clips, and 7 360p clips, all having 60 frames.

In our tests, AV1 and VP9 compress in 2-pass mode using constant quality(CQ) rate control, by which the codec is run with a single target quality parameter without specifying any bitrate constraint. The AV1 and VP9 codecs are run with the following parameters:

- *--frame-parallel=0 --tile-columns=0 --auto-alt-ref=2 --cpu-used=0 --passes=2 --threads=1 --kf-min-dist=1000 --kf-max-dist=1000 --lag-in-frames=25 --end-usage=q*

with *--cq-level=*{20, 32, 43, 55, 63}, unlimited key frame interval, and *--tune-content=screen* for AV1 to enable screen content tools. Need to mention that the first pass of AV1/VP9 2-pass mode simply conducts stats collections rather than actual encodings.

x265, a library for encoding video into the HEVC format, is also tested in its best quality mode(placebo) using constant rate factor(CRF) rate control. The x265 encoder is run with the parameters:

- *--preset placebo --no-wpp --tune psnr --frame-threads 1 --min-keyint 1000 --keyint 1000*

with *--crf=*{15, 20, 25, 30, 35}, unlimited key frame interval, and preset 60-frame lookahead buffer. Note that using the above cq levels and crf values makes RD curves produced by the three codecs close with each other in meaningful ranges for BDRate calculation.

The difference of coding performance is shown in Table I and Table II, represented by BDRate. A negative BDRate means using less bits to achieve the same quality. PSNR-Y, PSNR-Cb and PSNR-Cr are the objective metrics used to compute BDRate. At the time of writing this paper, unfortunately on AWCY test bench there is no PSNR metric implemented for averaging the PSNR across Y, Cb, Cr planes, we will update the results in later literature. Table I compares AV1 against VP9, indicating AV1 substantially outperforms VP9 by around 30% in all planes. Also in comparison with x265, Table II demonstrates a consistent 23.97% coding gain when the main quality factor PSNR-Y is considered, and even more outstanding coding capacity in Cb and Cr planes shown by about -40% BDRate in PSNR Cb/Cr metric.

TABLE I
BDRATE(%) OF AV1 IN COMPARISON WITH LIBVPX VP9 ENCODER

Metric \ Set	1080p	1080p-screen	720p	360p	Average
PSNR-Y	-26.89	-37.51	-27.21	-27.22	-28.46
PSNR-Cb	-31.27	-48.21	-26.16	-24.63	-30.78
PSNR-Cr	-31.71	-44.44	-28.64	-33.00	-32.99

TABLE II
BDRATE(%) OF AV1 IN COMPARISON WITH X265 HEVC ENCODER

Metric \ Set	1080p	1080p-screen	720p	360p	Average
PSNR-Y	-22.23	-28.95	-26.60	-21.46	-23.97
PSNR-Cb	-41.32	-50.49	-38.25	-36.08	-40.60
PSNR-Cr	-41.18	-45.14	-39.58	-43.10	-41.78

ACKNOWLEDGEMENT

Special thanks to all AOMedia members and individual contributors for their effort and dedication. Due to space limitations, we only list authors who participated in drafting this paper.

References

[1] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R.S. Bultje, "The latest open-source video codec VP9 - an overview and preliminary results," *Picture Coding Symposium (PCS)*, December 2013.

[2] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, 2012.

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, 2003.

[4] J. Bankoski, P. Wilkins, and Y. Xu, "Technical overview of VP8, an open source video codec for the web," *IEEE Int. Conference on Multimedia and Expo*, December 2011.

[5] "Alliance for Open Media," http://aomedia.org.

[6] L. N. Trudeau, N. E. Egge, and D. Barr, "Predicting chroma from luma in AV1," *Data Compression Conference*, Mar. 2018.

[7] W. Lin, Z. Liu, D. Mukherjee, J. Han, P. Wilkins, Y. Xu, and K. Rose, "Efficient AV1 video coding using a multi-layer framework," *Data Compression Conference*, Mar. 2018.

[8] J. Han, Y. Xu, and J. Bankoski, "A dynamic motion vector referencing scheme for video coding," *IEEE Int. Confernce on Image Processing*, Sep. 2016.

[9] Y. Chen and D. Mukherjee, "Variable block-size overlapped block motion compensation in the next generation open-source video codec," *IEEE Int. Confernce on Image Processing*, Sep. 2017.

[10] S. Parker, Y. Chen, and D. Mukherjee, "Global and locally adaptive warped motion comprensationin video compression," *IEEE Int. Conference on Image Processing*, Sep. 2017.

[11] U. Joshi, D. Mukherjee, J. Han, Y. Chen, S. Parker, H. Su, A. Chiang, Y. Xu, Z. Liu, Y. Wang, J. Bankoski, C. Wang, and E. Keyder, "Novel inter and intra prediction tools under consideration for the emerging AV1 video codec," *Proc. SPIE, Applications of Digital Image Processing XL*, 2017.

[12] S. Parker, Y. Chen, J. Han, Z. Liu, D. Mukherjee, H. Su, Y. Wang, J. Bankoski, and S. Li, "On transform coding tools under development for VP10," *Proc. SPIE, Applications of Digital Image Processing XXXIX*, 2016.

[13] J. Han, C.-H. Chiang, and Y. Xu, "A level map approach to transform coefficient coding," *IEEE Int. Confernce on Image Processing*, 2017.

[14] S. Midtskogen and J.-M. Valin, "The AV1 constrained directional enhancement filter (CDEF)," *IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, Apr. 2018.

[15] D. Mukherjee, S. Li, Y. Chen, A. Anis, S. Parker, and J. Bankoski, "A switchable loop-restoration with side-information framework for the emerging AV1 video codec," *IEEE Int. Confernce on Image Processing*, Sep. 2017.

[16] A. Norkin and N. Birkbeck, "Film grain synthesis for AV1 video codec," *Data Compression Conference*, Mar. 2018.

[17] "AWCY," arewecompressedyet.com.

[18] "objective-1-fast test set," https://people.xiph.org/%7etdaede/sets/objective-1-fast/.