Novel inter and intra prediction tools under consideration for the emerging AV1 video codec

Urvang Joshi, Debargha Mukherjee, Jingning Han, Yue Chen, Sarah Parker, et al.

Urvang Joshi, Debargha Mukherjee, Jingning Han, Yue Chen, Sarah Parker, Hui Su, Angie Chiang, Yaowu Xu, Zoe Liu, Yunqing Wang, Jim Bankoski, Chen Wang, Emil Keyder, "Novel inter and intra prediction tools under consideration for the emerging AV1 video codec," Proc. SPIE 10396, Applications of Digital Image Processing XL, 103960F (19 September 2017); doi: 10.1117/12.2274022

Event: SPIE Optical Engineering + Applications, 2017, San Diego, California, United States

Novel inter and intra prediction tools under consideration for the emerging AV1 video codec

Urvang Joshi, Debargha Mukherjee, Jingning Han, Yue Chen, Sarah Parker, Hui Su, Angie Chiang, Yaowu Xu, Zoe Liu, Yunqing Wang, Jim Bankoski, Chen Wang, Emil Keyder
Email: {urvang, debargha, jingning, yuec, sarahparker, huisu, angiebird, yaowu, zoeliu, yunqingwang, jimbankoski, wangch, emilkeyder}@google.com

Google, Inc., 1600 Amphitheatre Parkway, Mountain View, CA, USA 94043.

ABSTRACT

Google started the WebM Project in 2010 to develop open source, royalty-free video codecs designed specifically for media on the Web. The second generation codec released by the WebM project, VP9, is currently served by YouTube, and enjoys billions of views per day. Realizing the need for even greater compression efficiency to cope with the growing demand for video on the web, the WebM team embarked on an ambitious project to develop a next edition codec AV1, in a consortium of major tech companies called the Alliance for Open Media, that achieves at least a generational improvement in coding efficiency over VP9. In this paper, we focus primarily on new tools in AV1 that improve the prediction of pixel blocks before transforms, quantization and entropy coding are invoked. Specifically, we describe tools and coding modes that improve intra, inter and combined inter-intra prediction. Results are presented on standard test sets.

Keywords: video coding, VP8, VP9, webm, AV1, H.264, HEVC, prediction, motion, intra, inter.

1. INTRODUCTION

Google embarked on the WebM project [1] to develop open-source, royalty unencumbered video codecs for the Web. The first codec released as part of the project was called VP8 [2] and is still used extensively in Google Hangouts. The next edition of the codec, entitled VP9 [3], was released in mid-2013 and is the current generation codec from the WebM project. It achieves a coding efficiency similar to the latest video codec from MPEG entitled HEVC [4]. VP9 has found huge success with adoption by YouTube, and has delivered big improvements to the YouTube service in terms of quality of experience metrics such as watch-time and mean-time-to-rebuffer over the primary format H.264/AVC [5]. Specifically, VP9 streams delivered by YouTube today are not only 30-40% more compact than corresponding H.264/AVC streams but are also somewhat higher in quality. Consequently, even with predominant software decoding on compatible browsers - Chrome, Firefox, Opera - on potent devices, the number of VP9 videos viewed daily by YouTube users today is in the order of billions. As VP9 hardware decoders become more readily available on mobile devices we expect the proliferation of VP9 to accelerate even more.

Even though the gains achieved with VP9 are tangible and significant, the continued growth in online video consumption has made the need for efficient video coding increasingly critical. The WebM project has been focusing on developing the next generation video codec VP10 [6] since 2014, and modest gains in coding efficiency have already been achieved. In 2015, Google joined a consortium of major tech companies called the Alliance for Open Media [7] to jointly develop a new royalty-free codec to be named AV1. The experimental tools developed in VP10 are being proposed for the AV1 codec and there are also new experimental tools being developed for AV1.

Applications of Digital Image Processing XL, edited by Andrew G. Tescher, Proc. of SPIE Vol. 10396, 103960F · © 2017 SPIE · CCC code: 0277-786X/17/$18 · doi: 10.1117/12.2274022

In this paper we primarily focus on the tools developed for AV1 that improve the prediction of pixels blocks -- before transforms, quantization and entropy coding are invoked. The rest of the paper is organized as follows: Section 2 and 3 focus on novel prediction tools for intra and inter prediction respectively. Section 4 describes other improvements to the flexibility of prediction. Finally, section 5 outlines the results.

2. NOVEL INTRA PREDICTION TOOLS

2.1. Enhanced Directional Intra Prediction

VP9 only supports 8 directional intra prediction modes: D45_PRED, D63_PRED, H_PRED, D117_PRED, D135_PRED, D153_PRED, V_PRED, D207_PRED. These modes correspond to prediction angles of 45, 63, 90, 117, 135, 153, 180, and 207 degrees, respectively. To improve intra coding efficiency, more prediction angle options are added to AV1. To achieve this, a new *angle delta* syntax is introduced, while keeping the number of prediction modes the same. The prediction angle is calculated by the prediction mode and the angle delta:

$$\textit{prediction angle} = \textit{nominal\_angle} + (\textit{angle\_delta} * \textit{angle\_step}),$$

where *norminal_angle* is determined by the prediction mode, and is roughly the same as the options in VP9; *angle_delta* is in a predefined range and *angle_step* is a predefined value. In current configuration, *angle_delta* is in the range of [-3, +3] and *angle_step* is 3. These settings are selected experimentally. The total number of supported prediction angles is therefore increased from **8** to 8 * 7 = **56**. In the bitstream, the prediction mode symbol is entropy-coded the same way as in VP9. The value of the angle_delta is signaled as literals without entropy coding.

2.2. New Non-Directional Intra Prediction Modes

VP9 has 2 non-directional intra prediction modes: DC_PRED and TM_PRED. AV1 expands on this by adding 3 new prediction modes: SMOOTH_PRED, SMOOTH_V_PRED and SMOOTH_H_PRED. Also a 4th new prediction mode PAETH_PRED [8] replaces the existing mode TM_PRED. The new modes work as follows:

1. SMOOTH_PRED: Useful for predicting blocks that have a smooth gradient. It works as follows: estimate the pixels on the rightmost column with the value of the last pixel in top row, and estimate the pixels in the last row of the current block using the last pixel in left column. Then calculate the rest of the pixels by an average of quadratic interpolation in vertical and horizontal directions, based on distance of the pixel from the predicted pixels.
2. SMOOTH_V_PRED: Similar to SMOOTH_PRED, but uses quadratic interpolation only in the *vertical* direction.
3. SMOOTH_H_PRED: Similar to SMOOTH_PRED, but uses quadratic interpolation only in the *horizontal* direction.
4. PAETH_PRED: Calculate base = left + top - top_left. Then predict this pixel as left, top or top-left pixel depending on which of them is closest to 'base'. The idea is: (i) if the estimated gradient is larger in horizontal direction, then we predict the pixel from 'top'; (ii) if it's larger in vertical direction, then we predict pixel from 'left'; otherwise (iii) if the two are same, we predict the pixel from 'top-left'.

2.3. Color Palette as a Predictor

Sometimes, given intra block can be approximated by a block with small number of unique colors. This is especially true for artificial videos like screen-capture, games etc. For such cases, AV1 introduces a new intra coding mode called PALETTE. This predictor for a block is signaled by storing (i) a color palette, with 2 to 8 colors, and (ii) color indices into the palette for all pixels in the block. The residual pixel values of the block are as usual transformed and quantized before being entropy-coded.

PALETTE mode can be used by both intra-only as well as inter frames. The number of base colors determines the trade-off between fidelity and compactness. The color indices for pixels are obtained by the nearest neighbor method. The color indices are encoded using the neighborhood-based context to be as compact as possible.

2.4. Intra Prediction Based on Recursive Filtering

Most intra prediction tools primarily focus on directional extrapolation or interpolation. We propose a family of modes where pixels in a block are predicted recursively in raster scan order, where a predicted pixel P is predicted as a weighted combination of its causal neighbors that have been already predicted [9] [10]. In this framework, a 4-tuple of weights for neighbors from 4 directions (above, left, above-left, and above-right) determines the prediction mode, which can capture different pixel correlation models. As many such modes may be supported, with the best mode being picked by rate-distortion optimization.

3. NOVEL INTER PREDICTION TOOLS

3.1 Warped Motion Compensation

Traditional modern codecs, including VP9, use block motion compensation where motion vectors are translational only. This is not sufficient for real video which often contains complex motion. For example, motion due to camera shake, panning and zoom might require transformations that support shearing, scaling, rotation and changes in aspect ratio. In AV1, we introduce warped motion compensation implemented as similarity and affine transformations to better capture the diversity of motion that exists in real video [11]. These transformations can be described as homographies with fewer than 8 degrees of freedom. For example, If α represents a weight factor, (x, y) represents coordinates of an original pixel in a current frame, and (x', y') represents its warped coordinates in a reference frame, a full homography can be described by the following:

$$\alpha \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine projections, which have only 6 degrees of freedom, allow a rectangle to parallelogram transformation. In this case the transformation matrix is restricted:

$$h_{31} = h_{32} = 0\,,\ h_{33} = 1$$

Similarity projections, which require 4 degrees of freedom, allow for a block to block transformation with rotation and zoom only. In this case:

$$h_{31} = h_{32} = 0\,,\ h_{33} = 1,$$
$$h_{11} = h_{22}\ ,\ h_{13} = -h_{21}$$

We employ warped motion compensation in two different ways:

1. Global Warped Motion Compensation: It is common for videos to contain a global camera motion which is pertinent to an entire inter frame. It is therefore beneficial to transmit a set of motion parameters at the frame level that is applicable to a large number of blocks in the frame. When a frame is encoded, a set of global motion parameters is computed and transmitted between that frame and each reference frame. These parameters may be either translational, similarity or affine motion model. Subsequently, any block in the frame can signal use of the *global motion mode* with a given reference to create a suitable predictor.
2. Local Warped Motion Compensation: Warped motion compensation is also useful to describe complex local object motion. Here, we estimate warp parameters for a single block using the translational motion vectors that are typically conveyed for all inter blocks. Specifically, we estimate an affine or similarity model using the motion vectors from the current block and its causal neighbors which share the same reference frame.

3.2 Adaptive Overlapped Block Motion Compensation

Overlapped block prediction [12] [13], proposed and implemented back in the era of h.263,was proved to largely reduce prediction errors but not adopted by recent codecs due to extra complexity in the scenario of hybrid inter/intra variable block size coding.
In AV1, a practical overlapping mechanism based on two-stage 1-D filtering is proposed for the advanced partitioning framework to implement causal overlapped block prediction [14]. This algorithm starts by initializing the block using traditional inter prediction. On top of that, prediction overlapping is performed progressively in two stages: first combine it with the adjacent blocks above, and next combine it with those to the left. In first phase, if block i in the above row is inter-coded, motion assigned to block *i* is exploited to compute a new prediction block $p_i(x, y)$ for an 'overlapping region' of p_0 and p_i: a rectangular area in block *0*. For instance, in Fig.1(a), the shaded area is the overlapping region of p_0 and p_2. Blending will only be operated in the overlapping region by applying a 1-D filter in the vertical direction to update the current predictor as the combination of itself and prediction computed using above motion vectors. Likewise at the second stage, 1-D filtering will be performed on top of the result obtained after the first phase with left neighbors. For instance, Fig.1(b) shows the overlapping region of current block and block *4*. And the 1-D filtering will be operated in the horizontal direction. Note that for overlapped block prediction, the conventional block matching is not sufficient to account for the non-uniformly weighted inaccuracy at different pixel locations. Therefore a special motion estimation is executed according to a weighted distortion metric. The proposed two-sided OBMC mode is added as an extra coding mode competing with traditional non-overlapping motion compensated prediction.

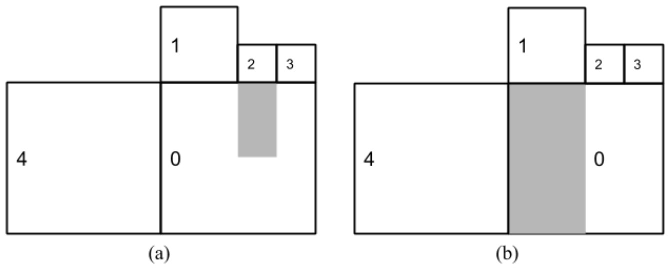

Fig. 1. Overlapping regions

3.3 Dual Interpolation Filter Kernel

The motion vector used in modern video codecs is allowed to have a fractional position for a better prediction quality. So, an interpolation filter module is needed to generate the prediction block at a fractional position in the reference frame. VP9 codec uses a separable interpolation filter to perform inter prediction with ⅛ motion vector precision. Three filter types, SHARP, REGULAR and SMOOTH, in descending order of cutoff frequencies, are provided to deal with various types of noise/distortions that can occur in reference frames/blocks. Given a filter type and a motion vector, the interpolation filter is performed by two one-dimensional filters, one for horizontal direction and one for vertical direction.
In AV1 codec, dual interpolation filter kernel [15] is introduced on top of the interpolation module inherited from VP9. Dual filter allows each block/frame to use a different interpolation filter type in horizontal and vertical direction. This idea is based on the observation that a reference frame/block's horizontal and vertical signals may have distinct frequency characteristics; therefore, using different filter types may produce a better prediction. As before, both the filter types are transmitted in the bitstream on a per block or per frame basis.

3.4 Extended Inter Compound Prediction Modes

In AV1, a collection of new compound prediction tools is implemented to address the limitations of compound prediction paradigm in VP9 and therefore to create a more versatile predictive coding framework.
Note that any compound prediction operation described in this section 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 $[0.0, 1.0]$ that are designed for different use cases and can be easily generated from predefined tables or made mode dependent.
The traditional compound modes in VP9 or AV1 baseline uses only averaging weights. In other words, m(i, j) above is 0.5 that can be readily implemented as:

$$p_f(\mathrm{i}, \mathrm{j}) = (p_1(i, j) + p_2(i, j)) / 2$$

3.4.1 Flexible MV Mode-combinations for Inter Compound Prediction

VP9 restricts motion vectors for a compound predictor to share one motion vector referencing mode, even though they may use different reference frames. To add more flexibility, on top of existing four combinations (`NEAREST_NEARESTMV`, `NEAR_NEARMV`, `NEW_NEWMV`, `ZERO_ZEROMV`) in VP9, AV1 supports four more empirically selected combinations: `NEAREST_NEWMV`, `NEW_NEARESTMV`, `NEAR_NEWMV`, and `NEW_NEARMV`. In addition to providing compression improvement, this flexibility also makes the new compound tools described in next two sections possible.

3.4.2 Compound Wedge Prediction

Boundaries of moving objects in a video often separate two regions with distinct motions. Coding these regions with separate motion vector reference combinations should be beneficial; however, finding exact object boundaries is not only difficult, but expensive to communicate in the bitstream. Our approach is to design a codebook of masks with only a few possible partitioning combinations and signaling the codebook index in the bitstream. The AV1 wedge codebook contains partition orientations that are either horizontal, vertical or oblique with slopes: 2, -2, 0.5 and -0.5. The wedge prediction mode is used for all square and rectangular blocks, using the 16-ary shape codebooks shown in Fig. 2.

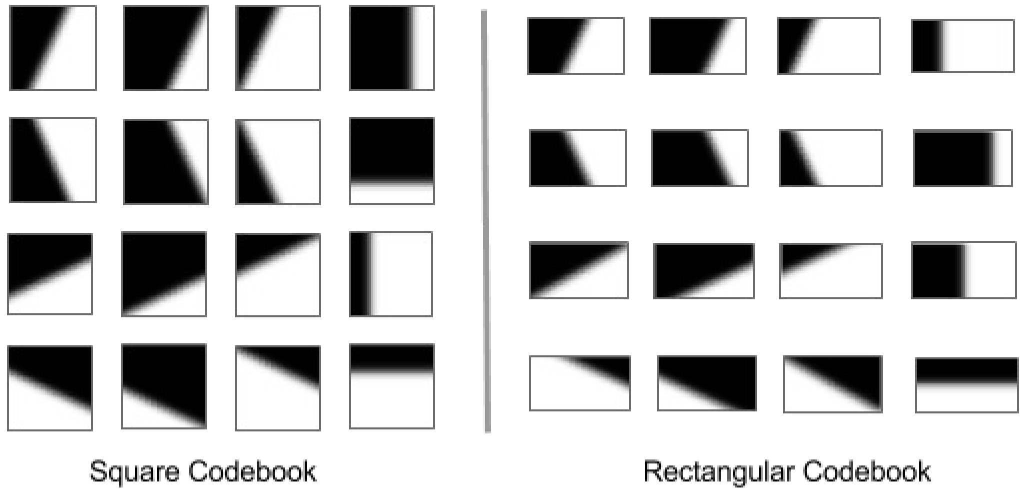

Fig. 2. Wedge shape codebooks for square and rectangular blocks

Taking pixels from two separate predictors and juxtaposing them according to one of these wedge masks invariably leads to spurious high frequency components that hurt coding efficiency. The solution is to use a softer blending to smooth the edges around the intended partition using cliff-shaped 2-D masks. The resultant method of compounding can be referred to as *overlapped wedge motion compensation* because of the obvious similarity with overlapped block motion compensation. The smooth cliff shaped 2-D mask $m(i, j)$ can in general be formulated as:

$$m(i, j) = g(d(i, j)),$$

where $d(i, j)$ is the distance of a pixel (i, j) to an underlying partition line, $g(\cdot)$ is a smoothing function of the type shown in Fig. 3 with g(0) = 0.5, and g(.) = 0 or 1 at either end.

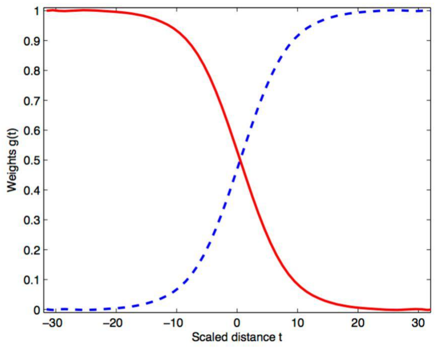

Fig. 3 . An example of 1-D weights $g(\cdot)$

3.4.3 Compound Segment Prediction

In many cases, regions in one predictor will contain useful content that is not present in the other. For example, one predictor might include information that was previously occluded by a moving object. While one of the wedges described above might be sufficient in some of these instances, a straight partition is not always suitable. In these instances, it is useful to create a non-uniform weighting based on information from the two predictors. Here, we use the pixel difference between the predictors as a simple and efficient scheme for identifying differing content. Specifically, the pixel difference between the two individual predictors is used to determine the weighting by modulating mask values on top of some base value. The mask is generated in the following way:

$$m(i, j) = b + a| p_1(i, j) - p_2(i, j)|$$

where b is a constant that controls how strongly one predictor is weighted over the other within the differing regions and a *is* a scaling factor which ensures a smoother modulation and guarantees that no mask weight exceeds 1. In practice, we try both mask m and mask $1 - m$ and select whichever mask produces the best rd-cost.

3.4.4 Compound Inter-intra Prediction

Blocks cannot always perfectly partition moving objects. For example, occlusion can occur in the middle of a block, it is better to apply different prediction techniques to different contents. Contents that are not occluded in reference frame will prefer inter prediction, while newly revealed content could benefit more from intra prediction using local reference. Therefore, a family of compound inter-intra prediction modes are proposed to combine intra prediction $p_1(i, j)$ and single-reference inter prediction $p_2(i, j)$ in different ways. To reduce the overhead, only 4 frequently-selected intra modes are supported in compound inter-intra modes, including horizontal mode, vertical mode, DC_PRED, and SMOOTH_PRED from Sec. 2.2. The mask $m(i, j)$ applied incorporates two types of smoothing functions:

(i) *wedge inter-intra mode* is an extension of the wedge inter-inter mode mentioned above using smooth cliff masks. The wedge codebooks are similar to what we use for wedge inter-inter modes, except that the wedge sign bit does not need to be communicated since we pre-define that the slice closer to the top-left corner will be occupied by intra predictor.

(ii) The other type of mask weights the intra part $p_1(i, j)$ in a smoothly decaying pattern determined by the direction of pixel extrapolation. For a mode-variant inter-intra mode, $m(i, j)$ at each pixel location can be inferred from the intra mode, block size and a primitive 64-tap 1-D decaying function $w_{ii1d}(.)$. The intra mode determines the main direction of weight variation, and the primitive function is used at proper scales according to a scaling factor

$$a = 64 / size\_of\_long\_edge(block\_size).$$

Weight computation for each mode is elaborated below:

$$m^{DC}(i, j) = 0.5,$$
$$m^{HORZ}(i, j) = w_{ii1d}(aj),$$
$$m^{VERT}(i, j) = w_{ii1d}(ai),$$
$$m^{SMOOTH}(i, j) = w_{ii1d}(a \cdot min(i, j)).$$

4. OTHER IMPROVEMENTS

4.1 Increased Available Reference Frames

AV1 extends the number of references from three to six. The new references are: `LAST2_FRAME`, `LAST3_FRAME`, and `BWDREF_FRAME`. In particular, `LAST2_FRAME` and `LAST3_FRAME` are two forward references, similar to `LAST_FRAME`, whereas `BWDREF_FRAME` is a backward reference, similar to `ALTREF_FRAME`. The use of `BWDREF_FRAME` exploits the existing "`show_existing_frame`" syntax provided by VP9, to encode a look-ahead frame without applying temporal filtering, thus more applicable as a backward reference in a relatively shorter distance.

Specifically, in the single reference mode, where Inter-coded blocks use a single prediction obtained from one reference frame, the proposed coding tool allows each frame to choose from up to six reference frames. In the compound reference mode, where Inter-coded blocks use a combination of two predictions, the new coding tool provides an abundant set of options obtained from a variety of reference frame pairs. Each video frame consequently is offered an extensively larger set of multi-reference prediction modes, thus leading to a greater potential for the RD performance improvement.

Moreover, leveraging the use of `BWDREF_FRAME` and `ALTREF_FRAME`, a symmetric framework of multi-reference prediction is established for the compound mode prediction: (1) A `BWDREF_FRAME` may be selected from a nearer future frame, as opposed to the `LAST_FRAME`; (2) A `BWDREF_FRAME` or `ALTREF_FRAME` may be selected from a farther future frame, as opposed to the `LAST2_FRAME`; and (3) An `ALTREF_FRAME` may be selected from a distant future frame, as opposed to the `GOLDEN_FRAME` in the distant past. Such framework provides an opportunity to encode a variety of videos with dynamic temporal correlation characteristics in a more adaptive and optimal way.

4.2 More Partitioning Options

The VP9 codec supports recursive coding block partition from 64x64 down to 4x4. For coding block sizes below 8x8, it is constrained that the sub8x8 blocks within an 8x8 block to either all use inter or intra modes. If they are coded in the inter mode, all the sub8x8 blocks should have identical reference frames and interpolation filter types. In AV1 such constraints are largely relaxed. AV1 supports partition down to 4x4 coding block unit, where each 4x4 luma block is allowed to independently select inter or intra mode, its reference mode, and interpolation filter type. This provides more flexibility to improve the prediction quality and hence the compression performance.

In the common YUV 420 format, the chroma block is a quarter size of the collocated luma block. A direct extension would incur the need to predict and transform chroma block in 2x2 block unit, if the luma block is coded at 4x4 block size. However, the use of 2x2 intra prediction would negatively affect the pipeline throughput in hardware design. To address this issue, the coding of sub8x8 blocks in YUV 420 format is restructured to process all the luma blocks within an 8x8 block area first. The chroma component would now have access to the entire 8x8 luma block coding information. If all the luma blocks therein are coded in the inter mode, there is no spatial dependency from intra prediction within this 8x8 block. Hence the chroma component would derive the collocated luma block's motion information and get its motion compensated prediction in 2x2 unit, and compose a 4x4 prediction block. If any of the luma blocks is coded in the intra mode, the whole 4x4 chroma component will be coded in the intra mode as a single block. In both cases, one has a residual block of size 4x4. A 4x4 transform block unit is then applied to residual coding.

On the other end of the spectrum, AV1 codec also allows the use of 128x128 coding block size, which is especially useful for high resolution videos (1080p, 4K etc). At sequence level, the codec can select its partition root as either 64x64 or 128x128. The later would make the maximum coding block size 128x128. To support the hardware design reuse the existing 64x64 block based pipeline, a number of modifications are made for the 128x128 block. (1) The transform coefficients are coded in the unit of 64x64 block. That is, it always completes them in this order: top left 64x64 block, then the top right 64x64 block, followed by the bottom left and bottom right 64x64 blocks. Within the 64x64 range, the order of blocks remains the same as a regular 64x64 coding block. (2) The post-processing filter updates its control parameters per 64x64 block regardless the maximum coding block size. (3) To apply the overlapped block motion compensation to a 128x128 block, the overlapped area is limited to the top 32 rows and leftmost 32 columns, each is processed same as a 64x64 coding block would do.

5. CODING RESULTS

To evaluate our new tools, we performed a controlled bitrate test using 3 different video sets:

- *lowres*, which includes 40 videos of CIF resolution,
- *midres*, wich includes 30 videos of 480p and 360p resolution, and
- *hdres*, which contains 38 videos at 720p and 1080p resolution.

where we code 150 frames of each video with a single keyframe. The coding results are shown in Tables 1-3 below.

For quality metrics we use average sequence PSNR and SSIM [16] - computed by the arithmetic average of the combined PSNRs and SSIMs respectively for each frame. Combined PSNR for each frame is computed from the combined MSE of the Y, Cb and Cr components. In other words:

$$MSE_{combined} = [4MSE_y + MSE_{Cb} + MSE_{Cr}]/6, \text{ assuming 4:2:0 sampling}$$

$$PSNR_{combined} = \min\left(10\log_{10}(255^2 / MSE_{combined}), 100 \right)$$

SSIM for each component in each frame is computed by averaging the SSIM scores computed without applying a windowing function over 8x8 windows for each component. Combined SSIM for the frame is computed from the SSIMs of the Y, Cb and Cr components as follows:

$$SSIM_{combined} = 0.8\ SSIM_y + 0.1\ (SSIM_{Cb} + SSIM_{Cr})$$

To compare RD curves obtained by two codecs we use a modified BDRATE [17] metric that uses piecewise cubic Hermite polynomial interpolation (*pchip*) on the the rate-distortion points before integrating the difference over a fine grid using the trapezoid method. The **OVERALL** number at the bottom is the arithmetic average of the BDRATE numbers over all the videos in the same column. The BDRATE is computed separately based on the average sequence PSNR and SSIM metrics as computed above.

For all the tables below, we use the current revision of AV1, from the master branch of aomedia repository, where the baseline and test configurations are as follows:

[Note that the current codebase does not allow disabling two of the predictions tools -- cb4x4 and chroma-sub8x8. Hence, both the baseline and test configs include these tools. We mention the BDRATE improvement from these two tools in table 4. The combined improvement using all the prediction tools is noted at the end of this section.]

Baseline: AV1 *without* the new prediction tools:

--enable-av1 --enable-experimental --disable-ext-refs --enable-cb4x4 --enable-chroma-sub8x8 --disable-ext-partition --disable-ext-intra --disable-alt-intra --disable-smooth-hv --disable-rect-intra-pred --disable-pale tte --disable-filter-intra --disable-global-motion --disable-warped-motion --disable-motion-var --disable-dual-filter --disable-ext-inter --disable-wedge --disable-compound-segment --disable-interintra

Test Config: AV1 *with* the new AV1 prediction tools:

--enable-av1 --enable-experimental --enable-ext-refs --enable-cb4x4 --enable-chroma-sub8x8 --enable-ext-partition --enable-ext-intra --enable-alt-intra --enable-smooth-hv --enable-rect-intra-pred --enable-palette --enable-filter-intra --enable-global-motion --enable-warped-motion --enable-motion-var --enable-dual-filter --enable-ext-inter --enable-wedge --enable-compound-segment --enable-interintra

The tables 1 to 3 below note the improvement in BDRATE of test config over the baseline config:

Table 1. AV1 BDRATE results on *lowres* set

Video	BDRATE (PSNR)	BDRATE (SSIM)
akiyo_cif.y4m	-10.648	-4.532
basketballpass_240p.y4m	-6.907	-4.127
blowingbubbles_240p.y4m	-9.873	-9.187
bowing_cif.y4m	-7.305	-4.704
bqsquare_240p.y4m	-22.721	-18.688
bridge_close_cif.y4m	-5.412	-6.118
bridge_far_cif.y4m	-11.556	-11.732
bus_cif.y4m	-13.565	-13.208
cheer_sif.y4m	-7.446	-5.136
city_cif.y4m	-10.479	-5.245
coastguard_cif.y4m	-10.006	-10.166
container_cif.y4m	-14.462	-12.849
crew_cif.y4m	-6.453	-5.293
deadline_cif.y4m	-5.953	-1.914
flower_cif.y4m	-16.029	-8.908
flowervase_240p.y4m	-18.619	-14.022
football_cif.y4m	-4.073	-3.274
foreman_cif.y4m	-7.925	-3.825
garden_sif.y4m	-11.276	-7.398
hallmonitor_cif.y4m	-3.355	-2.514
harbour_cif.y4m	-11.452	-13.089
highway_cif.y4m	-6.62	-6.459
husky_cif.y4m	-8.639	-7.948
ice_cif.y4m	-6.447	-3.667
keiba_240p.y4m	-4.725	-3.66
mobile_cif.y4m	-18.579	-16.241
mobisode2_240p.y4m	-16.798	-14.756

motherdaughter_cif.y4m	-9.018	-5.93
news_cif.y4m	-5.996	-0.768
pamphlet_cif.y4m	-4.335	-2.429
paris_cif.y4m	-4.719	1.477
racehorses_240p.y4m	-4.461	-1.966
signirene_cif.y4m	-8.489	-6.054
silent_cif.y4m	-6.493	-4.862
soccer_cif.y4m	-6.014	-2.077
stefan_sif.y4m	-11.764	-11.583
students_cif.y4m	-9.215	-6.403
tempete_cif.y4m	-21.955	-24.595
tennis_sif.y4m	-11.742	-14.381
waterfall_cif.y4m	-24.949	-24.689
OVERALL	**-10.162**	**-8.073**

Table 2. AV1 BDRATE results on *midres* set

Video	BDRATE (PSNR)	BDRATE (SSIM)
BQMall_832x480_60.y4m	-10.414	-8.261
BasketballDrillText_832x480_50.y4m	-6.96	-2.203
BasketballDrill_832x480_50.y4m	-7.064	-2.657
Flowervase_832x480_30.y4m	-10.627	-9.912
Keiba_832x480_30.y4m	-6.049	-5.584
Mobisode2_832x480_30.y4m	-9.988	-8.024
PartyScene_832x480_50.y4m	-12.016	-9.527
RaceHorses_832x480_30.y4m	-6.381	-4.383
aspen_480p.y4m	-10.539	-6.836
city_4cif_30fps.y4m	-12.838	-8.554
controlled_burn_480p.y4m	-4.863	-1.229
crew_4cif_30fps.y4m	-5.292	-2.228
crowd_run_480p.y4m	-12.622	-6.661
ducks_take_off_480p.y4m	-21.262	-17.409
harbour_4cif_30fps.y4m	-12.4	-11.582
ice_4cif_30fps.y4m	-5.725	-1.562
into_tree_480p.y4m	-10.445	-8.402

old_town_cross_480p.y4m	-11.897	-8.93
park_joy_480p.y4m	-10.992	-4.56
red_kayak_480p.y4m	-2.637	-3.519
rush_field_cuts_480p.y4m	-9.493	-5.453
sintel_trailer_2k_480p24.y4m	-11.193	-11.351
snow_mnt_480p.y4m	-1.48	0.471
soccer_4cif_30fps.y4m	-7.336	-3.017
speed_bag_480p.y4m	-12.99	-10.768
station2_480p25.y4m	-44.741	-46.166
tears_of_steel1_480p.y4m	-9.2	-5.691
tears_of_steel2_480p.y4m	-10.689	-4.708
touchdown_pass_480p.y4m	-7.328	-2.059
west_wind_easy_480p.y4m	-4.259	-4.927
OVERALL	**-10.324**	**-7.523**

Table 3. AV1 BDRATE results on *hdres* set

Video	BDRATE (PSNR)	BDRATE (SSIM)
basketballdrive_1080p50.y4m	-9.618	-7.998
blue_sky_1080p30.y4m	-14.029	-13.141
bqterrace_1080p60.y4m	-7.921	-7.171
cactus_1080p50.y4m	-10.716	-9.206
chinaspeed_xga.y4m	-1.968	-3.682
city_720p30.y4m	-7.06	-4.207
crew_720p30.y4m	-5.895	-4.722
crowd_run_1080p50.y4m	-7.457	-4.206
cyclists_720p30.y4m	-6.79	-4.376
dinner_1080p30.y4m	-5.173	-6.864
ducks_take_off_1080p50.y4m	-17.713	-15.678
factory_1080p30.y4m	-8.333	-7.806
fourpeople_720p60.y4m	-6.981	-6.169
in_to_tree_1080p50.y4m	-0.952	-0.049
jets_720p30.y4m	-25.001	-25.843
johnny_720p60.y4m	-10.526	-10.079
kimono1_1080p24.y4m	-6.428	-5.637

kristenandsara_720p60.y4m	-9.945	-9.634
life_1080p30.y4m	-8.057	-5.86
mobcal_720p50.y4m	-3.122	-1.901
night_720p30.y4m	-6.619	-4.769
old_town_cross_720p50.y4m	-8.639	-7.54
parkjoy_1080p50.y4m	-6.332	-3.233
parkrun_720p50.y4m	-6.788	-4.277
parkscene_1080p24.y4m	-6.286	-4.577
ped_1080p25.y4m	-6.383	-5.915
riverbed_1080p25.y4m	-4.386	-4.082
rush_hour_1080p25.y4m	-5.109	-4.733
sheriff_720p30.y4m	-8.223	-9.943
shields_720p50.y4m	-5.64	-4.007
station2_1080p25.y4m	-25.255	-26.629
stockholm_ter_720p60.y4m	-2.96	-2.518
sunflower_720p25.y4m	-8.321	-5.684
tennis_1080p24.y4m	-10.004	-9.551
tractor_1080p25.y4m	-8.838	-7.576
vidyo1_720p60.y4m	-10.046	-10.408
vidyo3_720p60.y4m	-8.966	-7.318
vidyo4_720p60.y4m	-9.007	-8.474
OVERALL	**-8.46**	**-7.512**

Table 4. Additional BDRATE improvements using CB4x4 and Chroma-sub8x8 tools

Test Set	BDRATE (PSNR)	BDRATE (SSIM)
lowres	-2.4%	-1.85%
midres	-1.2%	-0.90%
hdres	-0.9%	-0.71%

Thus, with all prediction tools combined, we observe **9.36% to 12.56%** reduction in **BDRATE for PSNR**, and **8.22% to 9.92%** reduction in **BDRATE for SSIM** for the 3 video sets, which confirms the benefit of the prediction tools.

6. CONCLUSION

In this paper we have presented a brief overview of the novel prediction tools that are being explored as part of AV1 development. Preliminary results indicate that introduction of these new tools can achieve on average at least a 9.36%

reduction in BDRATE for PSNR and 8.22% BDRATE for SSIM. Although this is an encouraging improvement, we continue to explore more avenues within the space of improving the prediction of pixels. AV1 is an open-source project, and we invite the rest of the video coding community to join the effort to create tomorrow's royalty-free codec.

REFERENCES

[1] http://www.webmproject.org/

[2] https://tools.ietf.org/html/rfc6386

[3] https://storage.googleapis.com/downloads.webmproject.org/docs/vp9/vp9-bitstream-specification-v0.6-20160331-draft.pdf

[4] G. J. Sullivan, J. R. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 22, No. 12, Dec 2012.

[5] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, 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, Jan 2011.

[6] D. Mukherjee, H. Su, J. Bankoski, A. Converse, J. Han, Z. Liu, and Y. Xu, "An overview of video coding tools under consideration for VP10: the successor to VP9," Proc. SPIE, Applications of Digital Image Processing XXXVIII, vol. 9599, Sep 2015.and K. Rose, "A Recursive Extrapolation Approach to Intra Prediction in Video Coding", Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2013.

[7] http://aomedia.org/

[8] https://www.w3.org/TR/PNG-Filters.html

[9] Y. Chen, J. Han and K. Rose, "A recursive extrapolation approach to intra prediction in video coding", Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2013.

[10] S. Li, Y. Chen, J. Han, T. Nanjundaswamy, and K. Rose, "Rate-distortion optimization and adaptation of intra prediction filter parameters", Proc. IEEE International Conference on Image Processing (ICIP), Oct. 2014

[11] S. Parker, Y. Chen, D. Barker, P. de Rivaz, and D. Mukherjee, "Global and locally adaptive warped motion compensation in video compression," in Image Processing, 2017. ICIP 2017.

[12] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: an estimation-theoretic approach," IEEE Transactions on Image Processing, vol. 3, no. 5, 1994.

[13] H. Watanabe and S. Singhal, "Windowed motion compensation," Proc. SPIE Conf. Visual Commun. Image, vol. 1605, pp. 582–589, 1991.

[14] Y. Chen, and D. Mukherjee, "Variable block-size overlapped block motion compensation in the next generation open-source video codec," in Image Processing, 2017. ICIP 2017.

[15] C. Chiang, J Han, S. Vitvitskyy, D. Mukherjee, and Y. Xu, "Adaptive interpolation filter scheme in AV1," in Image Processing, 2017. ICIP 2017.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, No. 4, pp. 600–612, April 2004.

[17] G. Bjøntegaard, "Calculation of average psnr differences between rdcurves," VCEGM33, 13th VCEG meeting, Austin, Texas, March 2001.