Daala: Building A Next-Generation Video Codec From Unconventional Technology

Jean-Marc Valin, Timothy B. Terriberry, Nathan E. Egge, Thomas Daede,
Yushin Cho, Christopher Montgomery, Michael Bebenita

Mozilla, Mountain View, USA
Xiph.Org Foundation, USA
jmvalin@jmvalin.ca

Abstract—**Daala is a new royalty-free video codec that attempts to compete with state-of-the-art royalty-bearing codecs. To do so, it must achieve good compression while avoiding all of their patented techniques. We use technology that is as different as possible from traditional approaches to achieve this. This paper describes the technology behind Daala and discusses where it fits in the newly created AV1 codec from the Alliance for Open Media. We show that Daala is approaching the performance level of more mature, state-of-the art video codecs and can contribute to improving AV1.**

I. Introduction

Daala [1] is a video codec designed to explore a set of atypical techniques, outlined in Section II, to avoid the patent thickets built around most current codecs. Some of these techniques are new to Daala, while others already existed, but are not used in popular standards. Although Daala is not yet a competitive codec on its own, some of the techniques it uses are currently being integrated in the Alliance for Open Media (AOM) [2] codec, AV1, which we discuss in Section III. Section IV presents results obtained with both Daala and AV1.

II. Daala Techniques

Most of the techniques Daala uses have been fundamental to the design since the initial stages of the project. Some of the techniques described below, like lapped transforms, Overlapped Block Motion Compensation (OBMC), and non-binary arithmetic coding, have forced the entire codec to move in a very different direction.

Vector variables are denoted in bold ($\mathbf{x}$) and their individual components are denoted with indices (x_i). Quantized variables are denoted with a hat ($\hat{x}$). Unless otherwise noted, $\|\mathbf{x}\|$ denotes the $L2$-norm of a vector $\mathbf{x}$.

A. Lapped Transforms

Rather than using a deblocking filter to attenuate blocking artifacts caused by quantizing DCT coefficients, Daala uses biorthogonal lapped transforms [3], [4]. The transform applies a *decorrelating* pre-filter to the input image before computing the DCT and applies a *deblocking* post-filter to the reconstruction after the inverse DCT. Since the pre-filter is the inverse of the post-filter, there is no need for complex adaptation of the filter strength to avoid blurring details. As shown in Fig. 1, the post-filter causes the basis functions to decay smoothly at transform block edges, avoiding blocking artifacts.
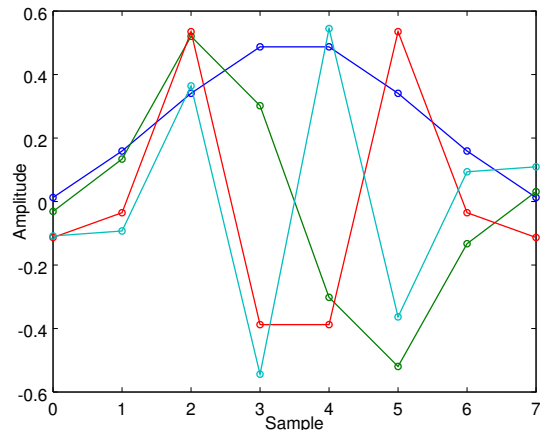

Figure 1: 1D synthesis basis functions for the lapped 4×4 DCT.

Transform block sizes in Daala range from 4×4 to 64×64, based on recursive quad-tree subdivision. A *superblock* refers to the largest area of a frame on which Daala can operate. Their size is 64×64.

Although applying lapping that spans the entire width of the transform is sometimes desirable (giving an $N \times N$ transform a support of $2N \times 2N$), it makes block size decision using rate-distortion optimization (RDO) intractable. Any choice of block size affects the coding efficiency of neighboring blocks. For this reason, Daala uses 4-point lapping for all transform sizes.

One disadvantage of lapped transforms is that it complicates intra prediction. Because of the overlap, the pixels adjacent to the block being predicted are not available for use in intra prediction. They cannot be reconstructed without quantized transform coefficients from the block being predicted. Instead of pixel-domain intra prediction, Daala uses a simple frequency-domain intra predictor. We predict AC coefficients along horizontal and vertical directions by directly copying a row or column of AC coefficients from the block above and the block to the left [5]. Some previous attempts at a more general frequency-domain intra predictor [6] were ultimately abandoned, as they failed to achieve good results with tractable complexity on large block sizes and mixed block sizes.
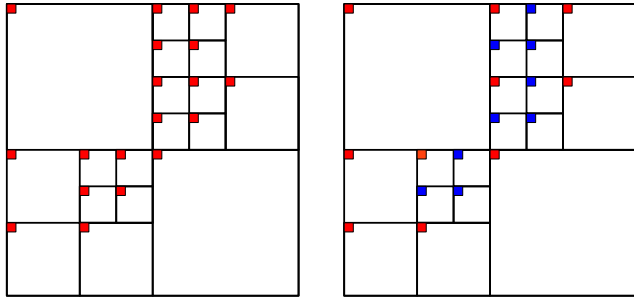
(a) Original DC coefficients before Haar DC

(b) DC coefficients from 8×8 blocks are combined
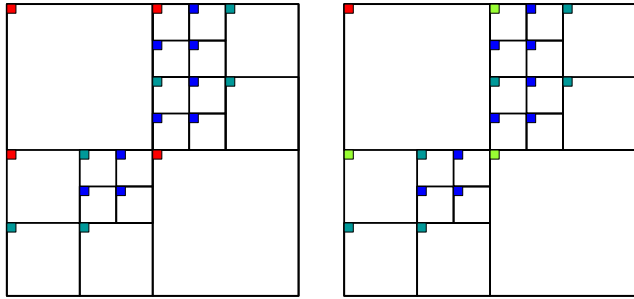
(c) DC coefficients from 16×16 sub-blocks are combined

(d) DC coefficients from 32×32 sub-blocks are combined

Figure 2: Example of applying Haar DC over three levels on a 64×64 superblock subdivided into transform blocks of 8×8, 16×16, and 32×32. At each step (a) to (d), the remaining DC coefficients are shown in red

B. Haar DC

Instead of using intra prediction for DC coefficients in keyframes, they are further transformed with a 2D Haar wavelet. Since Daala transform blocks are always split as quad-trees, the transform is applied bottom-up, recursively, up to the level of the corresponding superblock, as shown in Fig. 2. At each level, four DCs are combined into four Haar coefficients: one horizontal, one vertical, one diagonal, and one new DC corresponding to a larger block size.

The highest level (64×64) DC is predicted as a linear combination of the neighboring superblock DC coefficients: left, top-left, top, and top-right. The prediction coefficients are trained on an image database and constrained to sum to unity. The (non-DC) horizontal and vertical Haar coefficients are predicted from co-located horizontal and vertical coefficients at a larger scale. This slightly reduces bitrate in smooth areas.

C. Multi-Symbol Entropy Coder

Most recent video codecs encode information using binary arithmetic coding, meaning that each symbol can only take two values. The Daala range coder supports up to 16 values per symbol, making it possible to encode fewer symbols [7]. This is equivalent to coding up to four binary values in

parallel and reduces serial dependencies, allowing hardware implementations to use lower clock rates, and thus less power.

The range coder itself is based on a multiply-free approximation presented in [8], called piecewise integer mapping. The original approximation overestimates probabilities for symbols at the end of the alphabet, and underestimates probabilities at the beginning of the alphabet. Daala does this the other way around, since 0 is frequently the most probable symbol. Overestimating its probability leads to lower approximation overhead than underestimating it without reordering the alphabet.

Piecewise integer mapping replaces the multiplication and division of a traditional arithmetic coder with a subtraction, a minimum operation, and an addition. This allows it to work with any input probability distribution, without requiring that its total be normalized to a power of two. That lets Daala use traditional frequency counts to model probabilities, instead of having to extend the table-based schemes usually used by binary coders to larger alphabet sizes. Probabilities can typically be updated with just one or two SIMD instructions in software, and are similarly cheap to update in parallel in hardware. The overall cost is a bitrate overhead of around 1% in practice, which is comparable to CABAC [9].

We are currently exploring approaches for reducing this overhead without sacrificing throughput or modeling efficiency with hardware implementers in the AOM.

D. Overlapped Block Motion Compensation

Because hard block edges are expensive to code with lapped transforms, we want to avoid creating blocking artifacts in the motion-compensated prediction. For this we use overlapped block motion compensation (OBMC) [10]. We use an adaptive grid of Motion Compensation (MC) blocks ranging from 8×8 up to 64×64, in order to scale to high-resolution content. This grid is completely independent of the transform blocks, and the block sizes of one do not impose any constraints on the block sizes of the other. We also experimented with 4×4 MC blocks, but they were rarely used in practice, and actually caused a small quality regression at equal bitrate with the encoder used at the time.

The use of variable-sized MC blocks requires a blending scheme that maintains continuity between neighboring regions of different sizes. One approach, used by codecs like Dirac [11], is to fix the overlap size at the largest overlap allowed by the smallest motion-compensated block (8 pixels, in this case). This makes a large block equivalent to a group of smaller blocks with the same motion vector. However, this can create low-passing artifacts in a predictable grid pattern. These can be visually annoying, and require many bits to correct because of their locality. Unlike with transform blocks, there is no pre-filter in the encoder to compensate for the blending done in the decoder. Zhang *et al.* showed that splitting large blocks, but only as far as necessary to prevent their blending windows from overlapping more than one neighboring block, improved quality [12]. However, they implemented this as a

post-process to a non-overlapped block-based motion search, and did not incorporate RDO.

Instead, Daala structures its grid as a 4-8 mesh [13] to ensure the sizes of neighboring MC blocks differ by no more than a factor of two. This makes it easy to design OBMC blending windows that both span an entire block and ensure continuity across block size changes. Although R-D optimal block size decisions with this data structure are still NP-hard, there is a fast dynamic programming approximation that achieves good results in practice [14]. The details of the 4-8 grid structure, the blending windows, and the dynamic programming algorithm are outlined in [10].

E. Perceptual Vector Quantization

Most video codecs subtract a motion-compensated reference frame from the input frame to compute a residual, then transform and code it using scalar quantization. Instead, Daala uses gain-shape vector quantization. This encodes the signal as a vector by splitting it into a magnitude (gain), and direction (shape). Most importantly, in order to ensure the gain represents the amount of energy in the original signal, the motion-compensated reference is never subtracted from the input frame. Instead, Daala uses that reference to build a transformation that makes the input easier to code. This technique is called Perceptual Vector Quantization (PVQ) [15].

Perceptual Vector Quantization originates from the pyramid vector quantizer previously used for music in the Opus audio codec [16]. The pyramid vector quantizer is also a gain-shape quantizer that Opus uses to ensure that the signal energy is always conserved. Using a gain-shape quantizer in a video codec is more complicated, since it must also take into account a predictor. While we could quantize the scalar difference between the prediction and the input, conserving the energy of that difference is perceptually meaningless.

Rather than attempting to encode the difference, we derive a Householder reflection from the predictor that makes the input easier to encode. Let $\mathbf{x}$ be the input and $\mathbf{r}$ be the prediction. We construct a Householder reflection plane

$$\mathbf{v} = \frac{\mathbf{r}}{\|\mathbf{r}\|} + s\mathbf{e}_m \ , \tag{1}$$

where $\mathbf{e}_m$ is a unit vector along dimension m and $s = \pm 1$. The values of m and s are arbitrary, but to maximize numerical stability, we typically choose m to be the position of the largest absolute value in $\mathbf{r}$ and s to be the sign of that value.

We then apply the reflection to the input vector $\mathbf{x}$ to produce the reflected vector $\mathbf{z}$:

$$\mathbf{z} = \mathbf{x} - 2\frac{\mathbf{x}^T\mathbf{v}}{\mathbf{v}^T\mathbf{v}}\mathbf{v} \ . \tag{2}$$

When the input is similar to the prediction itself, the direction of the reflected vector $\mathbf{z}$ is close to the axis $-\mathbf{e}_m$. To take advantage of that fact, we express it as

$$\mathbf{z} = g\left(-s\cos\theta + \mathbf{u}\sin\theta\right) \ , \tag{3}$$

where g is the magnitude of $\mathbf{z}$ (and thus also the magnitude of $\mathbf{x}$), $\mathbf{u}$ is a unit vector with no component along the $\mathbf{e}_m$

direction, and θ is the angle between $\mathbf{r}$ and $\mathbf{x}$ (a meaningful parameter that represents the similarity between the prediction and the input). Since the Householder reflection is orthonormal, it follows that

$$\theta = \arccos\frac{\mathbf{x}^T\mathbf{r}}{\|\mathbf{x}\|\,\|\mathbf{r}\|} \ . \tag{4}$$

We code the unit vector $\mathbf{u}$ using a spherical quantizer derived from the pyramid vector quantizer [17]:

$$\mathbf{u} = \frac{\mathbf{y}}{\|\mathbf{y}\|} \ , \tag{5}$$

with

$$\mathbf{y} \in \mathbb{Z}^N : \|\mathbf{y}\|_{L1} = K \wedge y_m = 0 \ , \tag{6}$$

where the number of *pulses* K controls the size of the codebook.

The encoder quantizes g and θ and encodes them in the bitstream along with the integer vector $\mathbf{y}$ (excluding y_m which is 0). The codebook size K is determined only from g and θ and does not need to be transmitted. Since the decoder has access to the prediction vector $\mathbf{r}$, it can compute the reflection vector $\mathbf{v}$ without the need to transmit m and s. There are $N-2$ degrees of freedom to code in $\mathbf{y}$, and two more for g and θ. Thus we still code parameters with a total of N degrees of freedom. The main difference from scalar quantization is that two of the coded values have a perceptual meaning: g is the amount of contrast and θ is the amount of deviation from the prediction. By coding g as a parameter, it is easier to preserve the amount of contrast than by coding only DCT coefficients.

In practice, the vectors $\mathbf{x}$ and $\mathbf{r}$ are transform coefficients rather than pixel values. This requires an extra forward DCT in both the encoder and the decoder since the input and the prediction need to be transformed separately. Only the AC coefficients are coded using PVQ, and for transform blocks larger than 4×4, the AC coefficients are divided into multiple *bands*, where each band is coded separately. This allows us to control the contrast separately in each octave and orientation.

PVQ also allows us to take into account masking effects with no extra signaling. Since the gain is explicitly signaled, we can make the quantization resolution depend on the gain:

$$E\left\{\|\mathbf{x} - \hat{\mathbf{x}}\|^2\right\} \propto g^{2\alpha} \ , 0 \le \alpha \le 1 \ , \tag{7}$$

where $\alpha = 0$ behaves like a standard linear scalar quantizer and $\alpha = 1$ produces a constant relative error like in the Opus audio codec. Daala uses $\alpha = 1/3$. To achieve this, we quantize the companded gain

$$\gamma = g^{1-\alpha} \ , \tag{8}$$

giving finer resolution to smaller gains and coarser resolution to larger gains.

The decoder always decodes the quantized companded gain $\hat{\gamma}$ first. From there it can compute the quantization step size for θ as

$$Q_\theta = \frac{\beta}{\hat{\gamma}} \ , \tag{9}$$

where $\beta = \frac{1}{1-\alpha}$.

We determined the size of the codebook K through curve fitting [15] to be

$$K = \frac{\hat{\gamma} \sin \hat{\theta}}{\beta} \sqrt{\frac{N+2}{2}} \ . \tag{10}$$

The formulation in (10) is not robust to packet loss when the gain is predicted. If there are errors in the prediction, the decoder may obtain the wrong K and decode the wrong number of symbols. Fortunately, by making the $\sin \hat{\theta} \simeq \hat{\theta}$ approximation and substituting $\hat{\theta} = Q_\theta \hat{\tau}$, where $\hat{\tau}$ is the quantization index of the angle, we obtain

$$K = \hat{\tau} \sqrt{\frac{N+2}{2}} \ . \tag{11}$$

F. Chroma from Luma (CfL) Prediction

Although the use of $Y'C_B C_R$ reduces the correlation across planes compared to RGB, the chroma planes C_B and C_R and the luma plane Y' are still often locally correlated. Edges in chroma tend to align very well with edges in luma, with only the amount of contrast (gain) differing. PVQ's separation of signals into a gain and a shape makes it especially easy to predict chroma planes from the luma plane. Daala's chroma from luma (CfL) [18] prediction uses the luma transform coefficients as the prediction vector **r** directly. The only complication is that we need to code a sign for the prediction, since the luma plane and chroma plane coefficients may be negatively correlated. We also do not predict the gain of chroma from the gain of luma.

G. Directional Deringing Filter

Like other transform codecs, Daala can cause ringing artifacts around edges. We use a directional deringing filter to attempt to eliminate the ringing without blurring the image. Unlike HEVC's Sample-Adaptive Offset (SAO) [19], the Daala deringing filter is not based on classifying pixels and applying per-class offsets. Instead, it is an outlier-robust directional filter that smooths the neighborhood of pixels while preserving edges.

Let $x(n)$ denote a 1-dimensional signal and w_k denote filter tap weights. We define a linear finite impulse response (FIR) filter with unit DC response as

$$y(n) = \frac{1}{\sum_k w_k} \sum_k w_k x(n+k) \ , \tag{12}$$

which can alternatively be written as

$$y(n) = x(n) + \frac{1}{\sum_k w_k} \sum_{k,k \neq 0} w_k [x(n+k) - x(n)] \ . \tag{13}$$

The main advantage of expressing a filter in the form of (13) is that the normalization term $\frac{1}{\sum_k w_k}$ can be approximated relatively coarsely without affecting the unit gain for DC. This makes it easy to use small integers for the weights w_k.

The disadvantage of linear filters for removing ringing artifacts is that they tend to also cause blurring. To reduce the amount of blurring, Daala uses a "conditional replacement filter" to exclude the signal taps $x(n+k)$ that would cause

blurring and replace them with $x(n)$ instead. It determines this by whether $x(n+k)$ differs from $x(n)$ by more than a threshold T. This makes the FIR filter in (13) into a conditional replacement filter:

$$y(n) = x(n) + \frac{1}{\sum_k w_k} \sum_{k,k \neq 0} w_k R(x(n+k) - x(n), T) \ , \tag{14}$$

where

$$R(x, T) = \begin{cases} x & , \ |x| < T \\ 0 & , \ \text{otherwise} \end{cases} \ . \tag{15}$$

To further reduce the risk of blurring the decoded image, we apply the conditional replacement filter along the main direction of the edges in each 8×8 block of the reconstructed image. We determine the direction from the decoded image (no side information is transmitted) by analyzing each 8×8 block as described in [20]. For each 8×8 block, the decoder determines which of eight different directions best represents the content of the block. The search can be efficiently implemented in SIMD. We apply a 7-tap conditional replacement filter along the detected direction to each pixel in the 8×8 block.

To further reduce ringing in very smooth regions of the image, we apply a second filter to combine multiple output values of the first filter. The second filter is applied either vertically or horizontally – in the direction most orthogonal to the one used in the first filter. For a 45-degree direction, we apply the second filter horizontally to reduce hardware line buffer requirements. The combined effect of the two filters is a separable deringing filter that covers a total of 35 pixel taps.

The deringing filter only requires a minimal amount of signaling. We send a global threshold for the entire a frame, and signal one of six adjustment factors (including an *off* setting) for each superblock. With entropy coding, the cost of the signaling generally averages 2 bits per superblock, or 128 bytes for a 1080p keyframe. For predicted frames, we don't apply the deringing filter to 8×8 blocks where no coefficients are coded, and no adjustment factor is coded for superblocks where no 8×8 block is filtered, further reducing the amount of signaling. Fig. 3 shows the effect of the deringing filter at low bitrates.

III. Alliance for Open Media AV1 codec

The recently-formed Alliance for Open Media (AOM) is currently specifying AV1, a royalty-free video codec. The initial development is based on technology from three existing codecs: Google's VP9 [21] codec, Cisco's Thor [22] codec, and the Daala codec presented here. For this reason, some of the techniques used in Daala are currently being considered for inclusion in AV1.

The deringing filter described in Section II-G is already fully integrated in AV1 and has been shown to reduce bitrate by around 2% at equal quality. AV1 also includes Thor's constrained lowpass filter (CLPF) [23] that attenuates ringing, but with a more limited effect and a lower complexity than Daala's deringing filter.
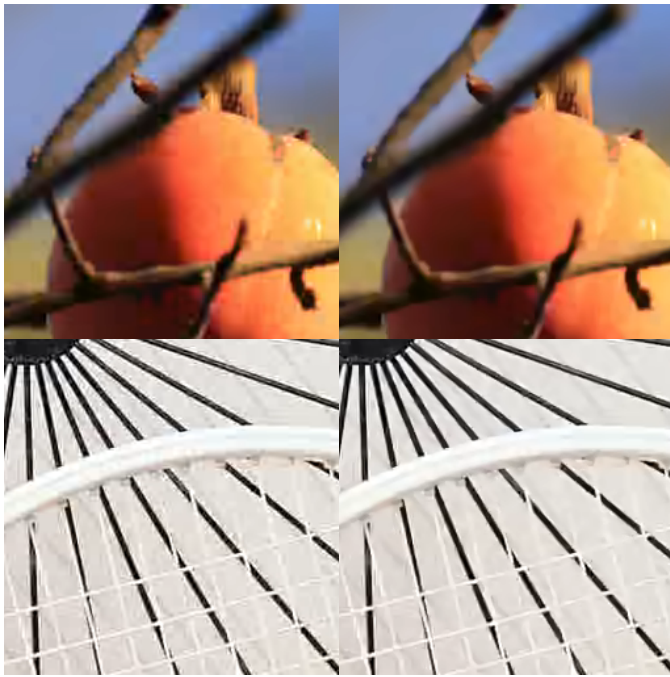
(a) Without deringing filter   (b) With deringing filter

Figure 3: Effect of the deringing filter on two images with sharp edges.

We are also evaluating the multi-symbol entropy coder (Section II-C) for use in AV1. Probability distributions in AV1 are currently fixed or explicitly signaled. In this case, we can avoid the piecewise integer mapping overhead by using probabilities whose denominator is a power of two. We expect that using adaptive distributions will produce larger gains than the overhead introduced by the mapping approximation, but using this effectively requires revisiting how every symbol is coded.

PVQ (Section II-E) is in the early stages of experimentation within AV1 and is by far the most invasive of the Daala techniques under consideration. Integration requires many changes to the bitstream, as well as the addition of a forward transform on the prediction itself. No results are available yet, but should PVQ be included in AV1, we would also be able to add CfL (Section II-F).

Lapped transforms (Section II-A) and OBMC (Section II-D) are *not* being considered for inclusion in AV1. Both these techniques have far-reaching interactions with the other coding techniques and would essentially require a complete redesign of the codec. Considering that Haar DC (Section II-B) is mostly needed to compensate for the lack of pixel-domain prediction, it also is not currently being considered for AV1.

IV. Results and Discussion

We have tested Daala on the `ntt-short-1` test set using the Are We Compressed Yet? [24] testing infrastructure. We use four different objective metrics for the comparison: PSNR, PSHR-HVS-M [25], SSIM, and FastSSIM [26] (a low-

complexity version of multiscale SSIM). We compare Daala[1] with the AV1 encoder[2], the x264 H.264 encoder[3], and the x265 HEVC encoder[4].

The results in Fig. 4 show that Daala is generally better than H.264, and slightly worse than HEVC and AV1. Based on this authors' informal evaluation, the subjective performance of Daala is close to what the PSNR-HVS-M results show in Fig. 4b. Qualitatively, the Daala artifacts tend to differ greatly from most other video codecs. Daala tends to perform more poorly on sharp details and edges, while retaining more texture in low contrast regions. This is in part due to PVQ activity masking, but remains true even without activity masking.

Considering that most of the technology used in Daala is either new or unproven in the context of video codecs, we consider these results to be encouraging. For example, Daala's results presented here do not use B-frames, while all of the other codecs do use B-frames or Alt-Refs to significantly improve coding efficiency. Moreover, these results have been steadily improving over the past two years, suggesting that Daala may be a viable approach to royalty-free codecs in the longer term. Some of the techniques presented in this paper will make their way into the AV1 codec, while others will require more time to mature before being used in a video coding standard.

References

[1] "Daala website," https://xiph.org/daala/.

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

[3] H. S. Malvar and D. H. Staelin, "The LOT: Transform coding without blocking effects." *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 37, no. 4, pp. 553–559, 1989.

[4] T. Tran, J. Liang, and C. Tu, "Lapped transform via time-domain pre- and post-filtering," *Signal Processing, IEEE Transactions on*, vol. 51, no. 6, pp. 1557–1571, June 2003.

[5] N. E. Egge, J.-M. Valin, T. B. Terriberry, T. Daede, and C. Montgomery, "Using Daala intra frames for still picture coding," in *Proceedings of Picture Coding Symposium*, 2015.

[6] C. Montgomery, "Introducing daala part 2: Frequency domain intra prediction," https://people.xiph.org/~xiphmont/demo/daala/demo2.shtml, 2013.

[7] T. B. Terriberry, "Coding tools for a next generation video codec," https://tools.ietf.org/html/draft-terriberry-codingtools-02, 2015.

[8] L. Stuiver and A. Moffat, "Piecewise integer mapping for arithmetic coding," in *Data Compression Conference, 1998. DCC'98. Proceedings*. IEEE, 1998, pp. 3–12.

[9] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, Jul. 2003.

[10] T. B. Terriberry, "Adaptive motion compensation without blocking artifacts," in *Proceedings of Visual Information Processing and Communication VI*, A. Said, O. G. Guleryuz, and R. L. Stevenson, Eds., vol. 9410, San Francisco, CA, Mar. 2015.

[1]Git version d55fff01 from April 25th, 2016 with command-line options: -k 1000 -v

[2]Git version 337b23a5 from April 7th, 2016 with deringing enabled and command-line options: –ivf –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 –cq-level=

[3]Command-line options: –preset placebo –min-keyint 1000 –keyint 1000 –no-scenecut –crf=

[4]Version 1.6 with command-line options: –preset slow –frame-threads 1 –min-keyint 1000 –keyint 1000 –no-scenecut –crf=
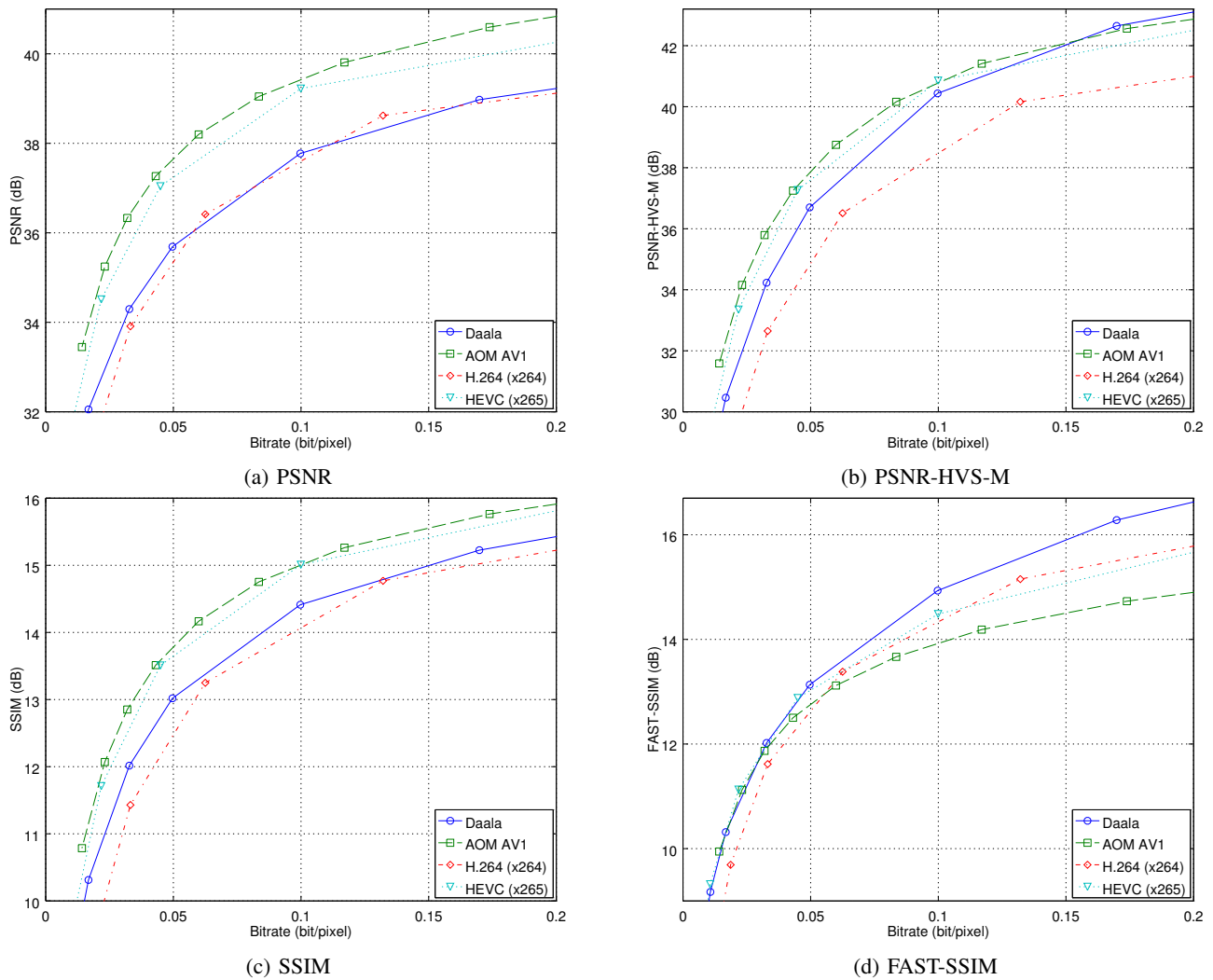
(a) PSNR

(b) PSNR-HVS-M

(c) SSIM

(d) FAST-SSIM

Figure 4: Comparison between Daala, HEVC and AV1 based on objective metrics.

[11] "Dirac specification," https://web.archive.org/web/20081010112442/http://diracvideo.org/download/specification/dirac-spec-latest.pdf, 2008.

[12] J. Zhang, M. O. Ahmad, and M. N. S. Swamy, "New windowing techniques for variable-size block motion compensation," *IEE Proceedings—Vision, Image, and Signal Processing*, vol. 145, no. 6, pp. 399–407, Dec. 1998.

[13] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein, "ROAMing terrain: Real-time optimally adapting meshes," in *Proceedings of the 8th IEEE Conference on Visualization (VIS'97)*, 1997, pp. 81–88.

[14] L. Balmelli, "Rate-distortion optimal mesh simplification for communications," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Switzerland, 2001.

[15] J.-M. Valin and T. B. Terriberry, "Perceptual vector quantization for video coding," in *Proceedings of SPIE Visual Information Processing and Communication*, vol. 9410, 2015, pp. 941 009–941 009–11, arXiv:1602.05209 [cs.MM] http://arxiv.org/abs/1602.05209.

[16] J.-M. Valin, G. Maxwell, T. B. Terriberry, and K. Vos, "High-quality, low-delay music coding in the opus codec," in *Proceedings of the 135th AES Convention*, 2013, arXiv:1602.04845 [cs.MM] http://arxiv.org/abs/1602.04845.

[17] T. R. Fischer, "A pyramid vector quantizer," *IEEE Trans. on Information Theory*, vol. 32, pp. 568–583, July 1986.

[18] N. E. Egge. and J.-M. Valin, "Predicting chroma from luma with frequency domain intra prediction," in *Proceedings of SPIE Visual Information Processing and Communication*, vol. 9410, 2015, pp. 941 008–941 008–10, arXiv:1603.03482 [cs.MM] http://arxiv.org/abs/1603.03482.

[19] C. M. Fu, E. Alshina, A. Alshin, Y. W. Huang, C. Y. Chen, C. Y. Tsai, C. W. Hsu, S. M. Lei, J. H. Park, and W. J. Han, "Sample adaptive offset in the hevc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1755–1764, Dec 2012.

[20] J.-M. Valin, "The Daala directional deringing filter," arXiv:1602.05975 [cs.MM] http://arxiv.org/abs/1602.05975, 2016.

[21] A. Grange, P. de Rivaz, and J. Hunt, "VP9 bitstream & decoding process specification," http://www.webmproject.org/vp9/, 2016.

[22] A. Fuldseth, G. Bjøntegaard, S. Midtskogen, T. Davies, and M. Zanaty, "Thor video codec," https://tools.ietf.org/html/draft-fuldseth-netvc-thor-02, 2016.

[23] S. Midtskogen, A. Fuldseth, and M. Zanaty, "Constrained low pass filter," https://tools.ietf.org/html/draft-midtskogen-netvc-clpf-01, 2016.

[24] "Are we compressed yet?" https://arewecompressedyet.com/.

[25] N. Ponomarenko, F. Silvestri, K.Egiazarian, M. Carli, and V. Lukin, "On between-coefficient contrast masking of dct basis functions," in *Proceedings of Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics VPQM-07*, 2007.

[26] M.-J. Chen and A. C. Bovik, "Fast structural similarity index algorithm," *Journal of Real-Time Image Processing*, vol. 6, no. 4, pp. 281–287, Dec. 2011.