A Technical Overview of AV1 Video Codec

Jim Bankoski, Google
An Alliance of Global Media Innovators

Coming Soon to a Screen Near You
Video coding at a glance

Partition  Predict  Transform  Quantize  Reconstruct  Encode
Video coding at a glance

Partition  Predict  Transform  Quantize  Reconstruct  Encode
Coding Block Partition

R: Recursive

128x128

64x64
Video coding at a glance

Partition  Predict  Transform  Quantize  Reconstruct  Encode
Extended Directional Intra Modes

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>G</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>H</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>
Paeth Mode:
\[ P_{\text{Paeth}} = \arg\min |x - T + L - TL|, \text{ over } x \in \{L, T, TL\} \]

SMOOTH_H: \[ P_{\text{SMOOTH}_H} = w(x) L + (1-w(x)) TR \]

SMOOTH_V: \[ P_{\text{SMOOTH}_V} = w(y) T + (1-w(y)) BL \]

SMOOTH: \[ P_{\text{SMOOTH}} = \frac{1}{2} (P_{\text{SMOOTH}_H} + P_{\text{SMOOTH}_V}) \]
Chroma from Luma Prediction

1. Luma average computed over the luma transform block
2. Chroma DC_PRED computed over prediction block
Palette Mode

Encoding process proceeds in wavefront order

Palette

Code 0

Code 1

Code 2

Pixels

Wavefront Order

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>3</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>11</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

Code 0 using left value as context

Code 2 using left value as context

Code 1 using above value as context

Code 0 using left and above as context

Code 2 using left value as context

Code 0 using left and above as context

Code 0 using left and above as context
Dynamic Motion Vector Referencing

- **Current frame**
- **Prior Coded Frame**
- **Current block**
- **NEARESTMV**
  - Ref1: \{MV1\}
  - Ref2: \{MV2\}
  - Ref3: \{MV3\}

- **NEARMV**
  - \{MV1\}
  - \{MV2\}
  - \{MV3\}

- **NEWMV**
  - (Delta sent for MV)

- **GLOBALMV**

- **Header**
Overlapped Block Motion Compensation
Masked Compound Prediction

Integerized mask $m(i, j) \in [0, 64]$
Advanced Compound Predictors

Distance Weighted Predictor

Difference Weighted Predictor

distance in time determines weight for predictor

blend where similar

pick 1 where different

Wedge

pick mask
Warped Motion Compensation

Horz Shear

Vert Shear
Pyramid style encoding

Golden-Frame (GF) Group

Sub-Group

BWDREF

ALTREF2

Overlay frame

Display Order (Decoding order as numbered)
Video coding at a glance

Partition  Predict  Transform  Quantize  Reconstruct  Encode
Transform Block Partitioning

- 16 separable 2-D kernels: \{ DCT, ADST, fADST, IDTX \}^2
Video coding at a glance

Partition   Predict   Transform   Quantize   Reconstruct   Encode
**TX Coefficient Coding**

Encode EOB position
In reverse scan order starting at EOB
  • encode magnitude of coefficient (up to 15) using context of up to 5 neighbors in same block that have already been coded
In scan order
  • If coeff is not 0
    • if DC code the sign with context of above and left DC signs
    • else code sign
    • if coeff >= 15 golomb code coeff - 15
Example TX Coefficient Coding

- zig-zag scan
  - TX coeffs
    - 0 1 5 6
    - 2 4 7 12
    - 3 8 11 13
    - 9 10 14 15

- Encoding process
  - code EOB = 11
  - code 1 using context from values in yellow
  - code 0 using context from values in yellow
  - code 1 using context from values in yellow
  - code 15+ using context from values in yellow

- golomb code 2
  - (17-15) & code (-) using context left and above dc signs
  - skip because its a 0
  - code (+)
  - skip because its a 0
  - code (-)
Video coding at a glance

Partition  Predict  Transform  Quantize  **Reconstruct**  Encode
Constrained Dire. Enhancement Filtering

- Applied after deblocking
- Edge directions are estimated at 8x8 block level
- 5x5 pre-designed detail-preserving deringing filters are applied

where $a = 2$ and $b = 4$ for even strengths and $a = 3$ and $b = 3$ for odd strengths.
### In-loop restoration Filters

<table>
<thead>
<tr>
<th>RU</th>
<th>RU</th>
<th>RU</th>
<th>RU</th>
<th>RU</th>
<th>RU</th>
</tr>
</thead>
<tbody>
<tr>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
</tr>
<tr>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
</tr>
<tr>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
<td>RU</td>
</tr>
</tbody>
</table>

**Frame**

- No filtering
- Wiener Filter + Parms
- Edge Preserve Filter + Parms
In-loop restoration Filters

Type A: Wiener filter
Separable (horz + vert filter)
7-tap, symmetric, normalized

Type B: Self-guided projected filters

$X_1$ and $X_2$ are cheap restored versions,
Subspace projection can yield a much better final restoration $X_r$.

$$X_r = X + \alpha (X_1 - X) + \beta (X_2 - X)$$

[Final output]

[Clean source]

[degraded source]
Film Grain Synthesis

- Film grain is present in much of the commercial content
- It is difficult to compress but needs to be preserved as part of creative intent
- AV1 supports film grain synthesis via a normative post-processing applied outside of the encoding/decoding loop
Film Grain Synthesis
Video coding at a glance
AV1 Symbol Coding

- Most syntax elements have non-binary long alphabets
- AV1 multi-symbol arithmetic coder facilitates high throughput symbol coding and straightforward probability model adaptation
  - AV1 arithmetic coding is based on 15-bit CDF tables
  - CDFs are tracked and updated symbol-to-symbol
## Compression Efficiency

- Test condition: AWCY\(^1\) objective1-fast\(^2\), 30 x 1080p~360p clips, 60 frames
- AV1 CQ mode, libvpx-VP9 CQ mode, x265 CRF mode
- BDRate (%)

<table>
<thead>
<tr>
<th>Codecs   \ Metric</th>
<th>PSNR-Y</th>
<th>PSNR-Cb</th>
<th>PSNR-Cr</th>
<th>CIEDE-2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV1 speed 0 vs. libvpx speed 0</td>
<td>-29.06</td>
<td>-32.41</td>
<td>-34.29</td>
<td>-31.12</td>
</tr>
<tr>
<td>AV1 speed 1 vs. libvpx speed 0</td>
<td>-27.15</td>
<td>-31.70</td>
<td>-33.35</td>
<td>-29.76</td>
</tr>
<tr>
<td>AV1 speed 0 vs. x265 placebo</td>
<td>-24.82</td>
<td>-41.69</td>
<td>-42.69</td>
<td>-35.60</td>
</tr>
<tr>
<td>AV1 speed 1 vs. x265 placebo</td>
<td>-22.81</td>
<td>-41.16</td>
<td>-42.07</td>
<td>-34.34</td>
</tr>
</tbody>
</table>

\(^1\) arewecompressedyet.com  
\(^2\) https://people.xiph.org/~tdaede/sets/objective-1-fast/
Compression Efficiency

- Results from Facebook Tests[1]

## Coding Complexity

**AV1 VBR mode at speed 0~3**, compared against **libvpx-vp9 speed 0**

<table>
<thead>
<tr>
<th>Resolution, encoder speed mode</th>
<th>ENC s/frame</th>
<th>ENC time vs libvpx</th>
<th>DEC frame/s</th>
<th>DEC time vs libvpx</th>
</tr>
</thead>
<tbody>
<tr>
<td>720p-8 bit, speed 0</td>
<td>394</td>
<td>175x</td>
<td>68</td>
<td>4.0x</td>
</tr>
<tr>
<td>720p-8 bit, speed 1</td>
<td>99</td>
<td>44x</td>
<td>78</td>
<td>3.5x</td>
</tr>
<tr>
<td>720p-8 bit, speed 2</td>
<td>57</td>
<td>25x</td>
<td>66</td>
<td>3.8x</td>
</tr>
<tr>
<td>720p-8 bit, speed 3</td>
<td>34</td>
<td>15x</td>
<td>73</td>
<td>3.7x</td>
</tr>
<tr>
<td>1080p-10 bit, speed 0</td>
<td>2284</td>
<td>141x</td>
<td>18</td>
<td>3.1x</td>
</tr>
<tr>
<td>1080p-10 bit, speed 1</td>
<td>440</td>
<td>27x</td>
<td>19</td>
<td>2.9x</td>
</tr>
<tr>
<td>1080p-10 bit, speed 2</td>
<td>265</td>
<td>16x</td>
<td>18</td>
<td>3.2x</td>
</tr>
<tr>
<td>1080p-10 bit, speed 3</td>
<td>156</td>
<td>10x</td>
<td>19</td>
<td>2.9x</td>
</tr>
</tbody>
</table>

[1] fcd7166eb, 06-06-2018
[2] 3ba9a2c8b, 11-01-2017
[3] Test machine CPU: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
Outline

AOMedia and AV1
Coding Techniques
Coding Performance
What’s Next
Q & A
Prediction Type Choices

● 56 Single Reference Choices
  ○ 7 frames * 4 Modes * 2 for OBMC

● 12768 Compound Reference Choices
  ○ 7 frames * 4 modes * 6 frames * 4 modes * (16 wedges + 1 weighted + 1 difference)

● 71 Intra Modes
  ○ 8 directions * 7 deltas + 12 DC modes + PAETH + INTRABLOCK_COPY + PALETTE

● 36708 Inter Intra Choices
  ○ (7 frames * 4 modes) * (8 directions * 7 deltas + 12 DC modes + PAETH) * (3 gradual + 16 wedges)

● 49603 Total Prediction Choices
Any single 8x8 block can be in any of the following partitionings

- 128x128, 32x128, 128x32, 64x128, 128x64,
- 64x64, 16x64, 64x16, 32x64, 64x32,
- 32x32, 8x32, 32x8, 16x32, 32x16,
- 16x16, 8x16, 16x8, 8x8

That’s 19 different prediction block sizes
**Transform Choices**

- **16** separable 2-D kernels:
  
  \[ (1 \text{ DCT} + 1 \text{ ADST} + 1 \text{ fADST} + 1 \text{ IDTX} ) \times (1 \text{ DCT} + 1 \text{ ADST} + 1 \text{ fADST} + 1 \text{ IDTX} ) \]
Transform Sizes

• 3 choices for every coding blocksize
  ○ Full resolution
  ○ \(\frac{1}{2}\) width and \(\frac{1}{2}\) height
  ○ \(\frac{1}{4}\) width and \(\frac{1}{2}\) height
Huge number of choices

Think 45,237,936 (ish) choices

Try everything encoder takes

9000 times as long as VP9
Figure out simple features to prune our search tree
- split or no split partitioning
- continue looking or quit
- which modes to try
- machine learned upscaling
- Size to make frames
What’s next?

- **Speed up** the codec
  - More SIMD coverage, ML based fast mode determination, ...
  - Set up and tune lower complexity speed modes (speed 2 - 8)

- **Continue improving** compression performance
  - Rate control, adaptive quantization, frame super resolution, ...
  - Different eng usage modes will be explored, e.g. perceptual quality mode
On the table for Next Time

- Optical flow tests provided up to 50% gains (avg 15-20%)
- Render 3d to 2d + Video
- Learned Transforms
- Machine learned image / texture generation
- Hopefully some of YOUR GREAT INVENTIONS!
Outline

AOMedia and AV1

Coding Techniques

Coding Performance

What’s Next

Q & A
Q + A