Daala: One year later

Timothy B. Terriberry
Original Plan

• *Finish Daala by the end of 2015*
  – This obviously ain’t gonna happen
Original Plan

- Finish Daala by the end of 2015 ✓
  - This obviously ain’t gonna happen
Original Plan

- **Finish Daala by the end of 2015 ✓**
  - This obviously ain’t gonna happen
- Not exactly:
  - Main development moved to AV1
  - Still using Daala as a research test bed
  - May have future life as a still-image codec
  - Techniques not ready for AV1 now may mature over time and become more compelling
• Last year, we had
  – 32×32 transforms and MC
  – Multiple reference frames, but no B-frames
  – Bilinear loop filter and deringing filter
  – No intra mode in our motion search
• Last year, we had
  – 32×32 transforms and MC
  – Multiple reference frames, but no B-frames
  – Bilinear loop filter and deringing filter
  – *No intra mode in our motion search*

• This year, we have
  – 64×64 transforms and MC
  – Basic MPEG-2 style B-frames (no bi-prediction, no direct mode)
  – No bilinear loop filter
  – Hardware-friendly deringing
Other major developments

- Better handling of frame padding
- Full Precision References
- New transform coefficient coder
- Fixed-point PVQ implementation (almost done, for real!)
- Rate control
- Better chroma quality
- Encoder turning and better SIMD
64×64 Motion Compensation

- Have restrictions on MC sizes for neighbors
  - Neighboring block size must be within factor of 2
- Complicated RDO scheme uses unwieldy tables
- Found disabling 4×4 MC was an improvement
- Just scaled up all the tables
  - Now we have 64×64 (down to 8×8)

<table>
<thead>
<tr>
<th></th>
<th>RATE (%)</th>
<th>DSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-6.64171</td>
<td>0.19601</td>
</tr>
<tr>
<td>PSNRHVS</td>
<td>-6.12492</td>
<td>0.28447</td>
</tr>
<tr>
<td>SSIM</td>
<td>-6.62569</td>
<td>0.15629</td>
</tr>
<tr>
<td>FASTSSSIM</td>
<td>-4.34795</td>
<td>0.11693</td>
</tr>
</tbody>
</table>
64×64 Transforms

• We have them

<table>
<thead>
<tr>
<th>Rate (%)</th>
<th>DSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-1.10946</td>
</tr>
<tr>
<td>PSNRHVS</td>
<td>-1.52479</td>
</tr>
<tr>
<td>SSIM</td>
<td>-1.22348</td>
</tr>
<tr>
<td>FASTSSIM</td>
<td>-1.16836</td>
</tr>
</tbody>
</table>

• But now we have 64×64 padding
Better Handling of Frame Padding

- Our transform coding doesn’t understand that some regions are padding
- MC ignores prediction errors in the padding
  - We then coded all of these errors
- After MC, replace the padding in the input frame by the MC predictor

<table>
<thead>
<tr>
<th></th>
<th>RATE (%)</th>
<th>DSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-1.58367</td>
<td>0.04947</td>
</tr>
<tr>
<td>PSNRHVS</td>
<td>-1.69591</td>
<td>0.08251</td>
</tr>
<tr>
<td>SSIM</td>
<td>-1.57043</td>
<td>0.03814</td>
</tr>
<tr>
<td>FASTSSIM</td>
<td>-1.43134</td>
<td>0.04049</td>
</tr>
</tbody>
</table>
Full Precision References (Currently off by default)

- Daala always operates on transform coefficients in 12-bit precision
  - 8-bit inputs are shifted up by 4 before transforms
  - Used to shift inverse transform output back to 8 bits
    - Saves memory, but adds rounding noise
- FPR: Stop converting back to 8 bits

<table>
<thead>
<tr>
<th></th>
<th>RATE (%)</th>
<th>DSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-1.95527</td>
<td>0.06122</td>
</tr>
<tr>
<td>PSNRHVS</td>
<td>-1.64452</td>
<td>0.07952</td>
</tr>
<tr>
<td>SSIM</td>
<td>-2.69109</td>
<td>0.06513</td>
</tr>
<tr>
<td>FASTSSSIM</td>
<td>-1.97242</td>
<td>0.05554</td>
</tr>
</tbody>
</table>
Deringing Filter Updates
Deringing Filter Updates

- Fixed several issues from hardware review
  - Made block-level threshold calculation independent of other blocks
    - Removed term involving an average over the whole 64x64 superblock
  - In the 45-degree case, changed second filter to run horizontally instead of vertically
    - Reduced the number of line buffers required in hardware
  - Removed divisions in the direction search
    - Used to divide by small, fixed constants (1...8) when averaging pixels along each direction (implemented in practice by multiplies)
    - Multiply by the LCM instead: no rounding errors, still fits in 32 bits
- Also changed filter taps from \([2,2,3,2,3,2,2]/16\) to \([1,2,3,4,3,2,1]/16\)
Bilinear Loop Filter

- Not a standard deblocking filter
  - Doesn’t look outside of current block!
  - Compare decoded block to bilinear interpolation of corner pixels, blend with optimal Wiener filter gain

\[ w = \min \left( 1, \frac{\alpha Q^2}{12 \sigma^2} \right)^2 \]
Bilinear Loop Filter

- Not a standard deblocking filter
  - Doesn’t look outside of current block!
  - Compare decoded block to bilinear interpolation of corner pixels, blend with optimal Wiener filter gain

\[
w = \min \left( 1, \frac{\alpha Q^2}{12 \sigma^2} \right)
\]
New Coefficient Coder (1)

- Basic idea: sum of absolute values, $K$, known
  - True for PVQ, must be encoded for scalar

- Split coefficient vector in half

$$K = K_{\text{left}} + K_{\text{right}}$$

- Code $K_{\text{right}}$: $K+1$ possible values (0...$K$)
  - If $K$ larger than 7, code top 3 bits of $K_{\text{right}}$ with arithmetic coder, code rest with raw bits
  - Context chosen from vector dimension, top bits of $K$
New Coefficient Coder (2)

- Special case: $K = 1$ and vector dimension $\leq 16$
  - Code exact location of the 1 with one symbol
  - 12 contexts based on vector dimension
    - 4 for vectors that start out with dimension $\leq 16$
    - 8 for vectors that get split down to dimension $\leq 16$

- Sign bits coded with raw bits

<table>
<thead>
<tr>
<th>Metric</th>
<th>Rate (%)</th>
<th>DSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-0.11934</td>
<td>0.00353</td>
</tr>
<tr>
<td>PSNRHVS</td>
<td>-0.06492</td>
<td>0.00298</td>
</tr>
<tr>
<td>SSIM</td>
<td>-0.36226</td>
<td>0.00815</td>
</tr>
<tr>
<td>FASTSSIM</td>
<td>-0.73242</td>
<td>0.01960</td>
</tr>
</tbody>
</table>
Room for Improvement

- Rate by pulse position for two random blocks

![Graph showing pulse position rates for two random blocks](graph.png)
Dyadic Probability Adaptation (1)

- Probabilities that sum to a power of 2 can be coded with less overhead (~1%)
- Adaptation mostly done now with simple frequency counts
  - Total probability changes with each coded symbol
- Want: adaptation scheme with fixed total
  - Problem: need to ensure no probability goes to 0
Dyadic Probability Adaptation (2)

- Fix total, \( T \), at 32768 (or any power of 2)
- Subtract probability floor \( \{1,2,3,...,M\} \) from CDF
  - \( M \) is the alphabet size (\( \leq 16 \))
- Blend with \( \{0,0,...,0,T-M,T-M,...,T-M\} \) CDF
  - Weights \( 2^{-\text{rate}} \) and \( (1 - 2^{-\text{rate}}) \)
- Add back in probability floor \( \{1,2,3,...,M\} \)
- No matter how you round/truncate in the blending, total remains \( T \), no probability is zero
Dyadic Probability Adaptation (3)

- Simplify
  - Symbol $i < \text{coded value}$
    - $f_i \rightarrow f_i - \lfloor (f_i + 2^{rate} - i - 2)/2^{rate} \rfloor$
  - Symbol $i \geq \text{coded value}$
    - $f_i \rightarrow f_i - \lfloor (f_i + M - i - T - 1)/2^{rate} \rfloor$

- $T$ (total probability), $M$ (alphabet size), $i$ (symbol index), and $rate$ are constants
  - Two 15-bit vector adds and one shift with pre-computed tables

- Change rate for first few symbols in context to speed up adaptation
  
<table>
<thead>
<tr>
<th>RATE (%)</th>
<th>DSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-0.45209</td>
</tr>
<tr>
<td>PSNRHVS</td>
<td>-0.45243</td>
</tr>
<tr>
<td>SSIM</td>
<td>-0.32941</td>
</tr>
<tr>
<td>FASTSSIM</td>
<td>-0.47029</td>
</tr>
</tbody>
</table>
Directions for AV1

- Directional Deringing
  - Fully SIMDable, good perceptual improvements

- Non-binary Arithmetic Coding
  - Small effective parallelism in entropy coding

- Perceptual Vector Quantization
  - Already showing small gains vs. scalar on PSNR
  - Potential for large perceptual improvements
  - Enables freq. Domain Chroma-from-Luma, others

- Rate control improvements
Progress and Metrics
Daala Progress (Fast MS-SSIM): January 2014 to April 2016

H.265

HQ YouTube

LQ Video Conference

Jan

May

Jun

up and left is better

The Xiph.Org Foundation & The Mozilla Corporation
Daala Progress (PSNR-HVS): January 2014 to April 2016

The Xiph.Org Foundation & The Mozilla Corporation
Questions?