Daala: Tools and Techniques
IETF 93 (Prague)
Daala Inter Encoder

- Reference Frames
  - OBMC
  - Prediction Frame
  - Forward Transform

- Input Frame
  - Prediction Frame
  - Forward Transform
  - Prediction Coefficients
  - DCs and PVQ data

- Input Coefficients
  - PVQ
  - Bitstream
    - Entropy Coding

- Decoded Image
  - Dequant + Inverse Transform
  - DCs and PVQ data
Daala Intra Encoder

Input Frame → Forward Transform → Input Coefficients

H+V Intra Prediction → Prediction Coefficients

Chroma from Luma → Haar DC, PVQ

Decoded Image → Dequant + Inverse Transform → DCs and PVQ data

Bitstream → Entropy Coding
Transform

- Lapped transform
  - N-point *pre-filter* removes correlation between blocks
  - N-point DCT within blocks
  - Decoder applies inverse (non-adaptive *post-filter*)
Transforms

- draft-egge-netvc-tdlt-00
- Sizes: 4x4, 8x8, 16x16, 32x32 (64x64 in progress)
- Reversible
  - $iLT(fLT(x)) = x$ for all $x$
- Biorthogonal (not orthogonal)
  - Not all basis functions have the same magnitude
  - Slight correlation between coefficients
Overlapped Block Motion Compensation

- draft-terriberry-netvc-obmc
- Goal: MC prediction with no blocking artifacts
- Variable block size (unrelated to transform size)
  - Unlike Dirac, larger blocks use larger blend window
  - Currently restricted so adjacent blocks differ by at most one size to maintain continuity
    - Possible to remove this restriction
- Subpel: separable 6-tap filters
  - Windowed sinc
  - 7-bit coefficients, no truncation/rounding between horizontal/vertical stages
OBMC → Prediction

- OBMC produces a *prediction image* for the whole frame
  - PVQ requires a prediction in the frequency domain
  - Just apply forward transforms to the prediction image
- Currently no explicit intra mode
  - PVQ can choose to discard the prediction (noref)
  - Our encoder still spends bits trying to improve it during motion search
Displaced Frame Difference

- Motion Compensation
  - Copy blocks from an already encoded frame (offset by a motion vector)
  - Subtract from the current frame
  - Code the residual
Displaced Frame Difference

- **Motion Compensation**
  - Copy blocks from an already encoded frame (offset by a motion vector)
  - Subtract from the current frame
  - Code the residual
Perceptual Vector Quantization

- Separate “gain” (contrast) from “shape” (spectrum)
  - Vector = Magnitude \times \text{Unit Vector} (point on sphere)

- Potential advantages
  - Better contrast preservation
  - Better representation of coefficients
  - Free “activity masking”
    - Can throw away more information in regions of high contrast (*relative* error is smaller)
    - The “gain” is what we need to know to do this!
PVQ with a Prediction

• Subtracting and coding a residual loses energy preservation
  – The “gain” no longer represents the contrast
• But we still want to use predictors
  – They do a really good job of reducing what we need to code
  – Hard to use prediction on the shape (on the surface of a hyper-sphere)
• Solution: transform the space to make it easier
2-D Projection Example

- Input
2-D Projection Example

- Input + Prediction
2-D Projection Example

- Input + Prediction
- Compute Householder Reflection
2-D Projection Example

- **Input + Prediction**
- **Compute Householder Reflection**
- **Apply Reflection**
2-D Projection Example

- **Input + Prediction**
- **Compute Householder Reflection**
- **Apply Reflection**
- **Compute & code angle**
2-D Projection Example

- Input + Prediction
- Compute Householder Reflection
- Apply Reflection
- Compute & code angle
- Code other dimensions
What does this accomplish?

- Creates another “intuitive” parameter, $\theta$
  - “How much like the predictor are we?”
  - $\theta = 0 \rightarrow$ use predictor exactly

- Remaining $N-1$ dimensions are coded with VQ
  - We know their magnitude is $\text{gain} \times \sin(\theta)$

- Instead of subtraction (translation), we’re scaling and reflecting
Activity Masking

- Noise is more visible in low contrast areas
  - Sensitivity $\propto g^\alpha$
Activity Masking

- Better resolution in low-contrast (gain) areas
- Compand gain with exponent $\beta$

$\beta=1$  
$\beta=1.5$
PVQ Bands

• DC coded separately with scalar quantization
  – Intra uses “Haar DC” to get better resolution over large areas
• AC coefficients grouped into bands
  – Gain, theta, etc., signaled separately for each band
  – Layout ad-hoc for now
• Scan order in each band optimized for decreasing variance
Band Structure
To Predict or Not to Predict

• $\theta > \pi/2 \rightarrow$ Prediction not helping
  – Could code large $\theta$’s, but doesn’t seem that useful
  – Need to handle zero predictors anyway

• Current approach: code a “noref” flag
  – Jointly coded with small gain and theta values
H+V Intra Prediction (luma)

• Copy first row/column of neighbor to corresponding bands
  – Only if size of corresponding blocks match
• Use noref to decide whether to use that as a predictor
• For the first band (first 15 AC coeffs), have to choose (only one noref flag)
  – Copy from neighbor with highest energy
• No explicit intra mode signaled
Chroma from Luma

• Use luma coeffs. as PVQ predictor for chroma
  - For 4:2:0 4x4 chroma blocks, TF-merge 4x4 luma blocks and take the low quadrant

• No longer building a model from neighbors
  - PVQ gains signal scaling
  - noref flag can disable prediction
  - Additional “flip” flag can reverse the whole predictor (coded on first non-noref band)

• No longer predicting chroma DC from luma
Quantization

• Per-coefficient quantizers
  – Interpolated up/down from 8x8 matrix
  – Compensation for LT basis magnitudes in separate step

• Built-in activity masking
  – Goal: better resolution in flat areas
  – Low contrast → low energy (gain)
  – Compand gain, choose resolution for $\theta$ and $K$ based on quantized gain
Entropy Coding

- draft-terriberry-netvc-codingtools
- Non-binary arithmetic coding
  - Theory: many overheads are *per-symbol*
  - Reducing the number of symbols improves throughput/cycle
- Much simpler than encoding multiple symbols in parallel
  - Decoder search in non-binary alphabet can still be parallelized (with SIMD or in hardware)
Entropy Coding

• Multiply-free partition function
  - Modified from Stuiver & Moffat 1998 design
  - \( R \rightarrow c + \min(c, R - \text{total}) \)
  - Requires \( \text{total} \leq R < 2^\times\text{total} \) (shift up if not)

\[
\begin{align*}
f_0 &= 8 \\
f_1 &= 2 \\
f_2 &= 1 \\
f_3 &= 1
\end{align*}
\]

\( \text{total} = 12 \) (scale by 2 to get \( 24 \leq 32 < 48 \))

\( R = 32 \)

over-estimated

under-estimated
Entropy Coding: Properties

- 15-bit probabilities
- Alphabet sizes up to 16
  - Want to keep as close to this as possible for maximum efficiency
- Several different adaptation strategies
  - Traditional frequency counts
  - Laplace encoder parameterized by expected value
  - “Generic Coder” combines frequency counts at several scales with exponential “tails”
Entropy Coder: Raw bits

• Appended to the end of the frame
• Coded from back to front
  – Much simpler than CABAC bypass mode
  – Same technique used in Opus
• Encoder writes to separate buffer, merged during Dirac-style carry propagation step
TODO

- PVQ needs a fixed-point implementation
- No B-frames at all (in progress)
- Need 64x64 blocks for OBMC
- Need 64x64 transforms
- Lots of potential improvements in motion search
- Lots of ways to exploit PVQ we haven’t tried
- Generic coder should be replaced by more targeted probability modeling
- Deringing filter (have prototype, too slow)
- Etc.