Introduction

- Y’CbCr color conversion de-correlates luma and chroma globally, but local relationship exists
- Cross channel intra-prediction exploits local correlation
  - Pros
    - Uses information already known to decoder
    - Can predict smooth features across a block
    - Reduces signaling overhead
  - Cons
    - Increases encoder and decoder complexity
    - Needs a parameterizable model
Predicting Chroma-from-Luma: Spatial Domain

- Both encoder and decoder compute linear regression:
  \[
  \alpha = \frac{\sum_i N \cdot L_i \cdot C_i - \sum_i L_i \sum_i C_i}{\sum_i N \cdot L_i \cdot L_i - \left(\sum_i C_i\right)^2}
  \]
  \[
  \beta = \frac{\sum_i C_i - \alpha \cdot \sum_i L_i}{N}
  \]

- Use reconstructed luma coefficients to predict spatially coincident chroma coefficients:
  \[
  C(u, v) = \alpha \cdot L(u, v) + \beta
  \]

Spatial Domain CfL Properties

• Pros
  - Can predict more features than straight edge extension
  - Can be implemented without signaling $\alpha$ or $\beta$

• Cons
  - Complexity scales with block size, for $N \times N$ block
    • $4N + 2$ mul's and $8N + 3$ add's to fit model
    • $N^2$ mul's to predict coefficients
  - 4:2:0 and 4:2:2 require resampling luma coefficients to match chroma spatial extent
  - Cannot be used in codecs that use lapped transforms
Predicting Chroma-from-Luma: Frequency Domain

- Key insight: LT and DCT are both linear transforms so similar relationship exists in frequency domain
- Compute linear regression with DC and 3 AC coefficients:

\[
\begin{align*}
C_{DC} &= \alpha_{DC} \cdot L_{DC} + \beta_{DC} \\
C_{AC}(u, v) &= \alpha_{AC} \cdot L_{AC}(u, v)
\end{align*}
\]

- Use reconstructed luma to predict frequency domain chroma coefficients:
Time-Frequency Resolution Switching

- Described in Section 3.2 of `draft-terriberry-netvc-codingtools`
- Trades off spatial resolution for frequency resolution
  
  ![2x2 Lifting Transform Diagram]

- Uses 2x2 Walsh-Hadamard Transform (WHT) with only 7 add's and 1 shift

Frequency Domain CfL Properties

• Pros
  – Can predict more features than straight edge extension
  – Can be implemented without signaling $\alpha$ or $\beta$
  – Using TF avoids expensive IDCT / FDCT round trip
  – Model fitting complexity independent of block size
  – No longer required to predict chroma DC from luma DC
  – Can be used with codecs that use lapped transforms

• Cons
  – Prediction still requires 1 multiply per coefficient
Perceptual Vector Quantization

- Described in `draft-valin-netvc-pvq`
- Separate “gain” (contrast) from “shape” (spectrum)
  - Vector = Magnitude × Unit Vector (point on sphere)
- Use different quantization for each
  - “gain” is quantized using scalar quantization
  - “shape” is quantized by finding nearest VQ-codeword in an algebraically defined codebook based on the reconstructed gain
PVQ Prediction

• Given prediction vector $r$
  - “gain” predicted by magnitude $\hat{g} = \gamma_g \cdot Q + \|r\|$
  - “shape” predicted using Householder reflection

\[
v = \frac{r}{\|r\|} + s \cdot e_m
\]

\[
z = x - 2 \frac{v^T x}{v^T v} v
\]
Chroma-from-Luma with PVQ Prediction

- Consider prediction of 15 AC coefficients from a 4x4 chroma block
- The 15-dimensional predictor $\mathbf{r}$ is scalar multiple of coincident reconstructed luma coefficients $\hat{\mathbf{x}}_L$

$$C_{AC}(u, v) = \alpha_{AC} \cdot L_{AC}(u, v) \implies \mathbf{r} = \alpha_{AC} \cdot \hat{\mathbf{x}}_L$$

- Thus “shape” predictor is almost exactly $\hat{\mathbf{x}}_L$

$$\frac{\mathbf{r}}{||\mathbf{r}||} = \frac{\alpha_{AC} \cdot \hat{\mathbf{x}}_L}{||\alpha_{AC} \cdot \hat{\mathbf{x}}_L||} = \text{sgn}(\alpha_{AC}) \frac{\hat{\mathbf{x}}_L}{||\hat{\mathbf{x}}_L||}$$

- Only difference is direction of correlation!
PVQ-CfL Algorithm (Encoder)

- Code “gain” using scalar quant. (no prediction)
- Code “shape” using PVQ:

1: Let \( r = \hat{x}_L \), compute \( \theta \)
2: Code a *flip* flag, \( f = (\theta > 90^\circ) \)
3: If \( f \)
4: Let \( r = -\hat{x}_L \)
5: End
6: Code \( x_C \) with PVQ using predictor \( r \)
PVQ Chroma-from-Luma Properties

• Pros
  - Can predict more features then straight edge extension
  - No need to fit linear model to coefficients
  - Still need TF to predict 4x4 chroma from four 4x4 luma

• Cons
  - Requires using PVQ prediction
  - Must code one flip flag per block
Example – Prediction (using HV)
Example – Prediction (using CfL)
### Objective Results

#### BD-rate for Cb plane:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Rate (%)</th>
<th>DSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-4.60206</td>
<td>0.13743</td>
</tr>
<tr>
<td>PSNRHVS</td>
<td>-5.51783</td>
<td>0.24312</td>
</tr>
<tr>
<td>SSIM</td>
<td>-10.31658</td>
<td>0.16631</td>
</tr>
<tr>
<td>FASTSSIM</td>
<td>-11.50168</td>
<td>0.22043</td>
</tr>
</tbody>
</table>

#### BD-rate for Cr plane:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Rate (%)</th>
<th>DSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>-3.25591</td>
<td>0.09362</td>
</tr>
<tr>
<td>PSNRHVS</td>
<td>-4.70448</td>
<td>0.20513</td>
</tr>
<tr>
<td>SSIM</td>
<td>-7.99407</td>
<td>0.13373</td>
</tr>
<tr>
<td>FASTSSIM</td>
<td>-11.57645</td>
<td>0.22452</td>
</tr>
</tbody>
</table>
Questions?