Chroma-from-Luma Intraprediction for NETVC

`draft-egge-netvc-cfl-00`

Nathan Egge

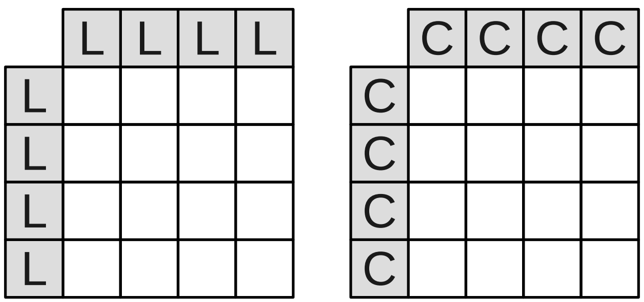
IETF 94 – Yokohama
2015 Nov 3

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

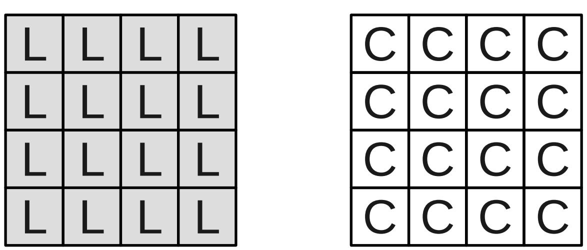
Predicting Chroma-from-Luma: Spatial Domain

- Both encoder and decoder compute linear regression:



$$\alpha = \frac{N \cdot \sum_i L_i \cdot C_i - \sum_i L_i \sum_i C_i}{N \cdot \sum_i L_i \cdot L_i - \left(\sum_i C_i\right)^2} \qquad \beta = \frac{\sum_i C_i - \alpha \cdot \sum_i L_i}{N}$$

- Use reconstructed luma coefficients to predict spatially coincident chroma coefficients:



$$C(u,v) = \alpha \cdot L(u,v) + \beta$$

[1] S.H. Lee & N.I. Cho: “Intra prediction method based on the linear relationship between the channels for YUV 4:2:0 intra coding” ICIP 2009, pp. 1033-1036

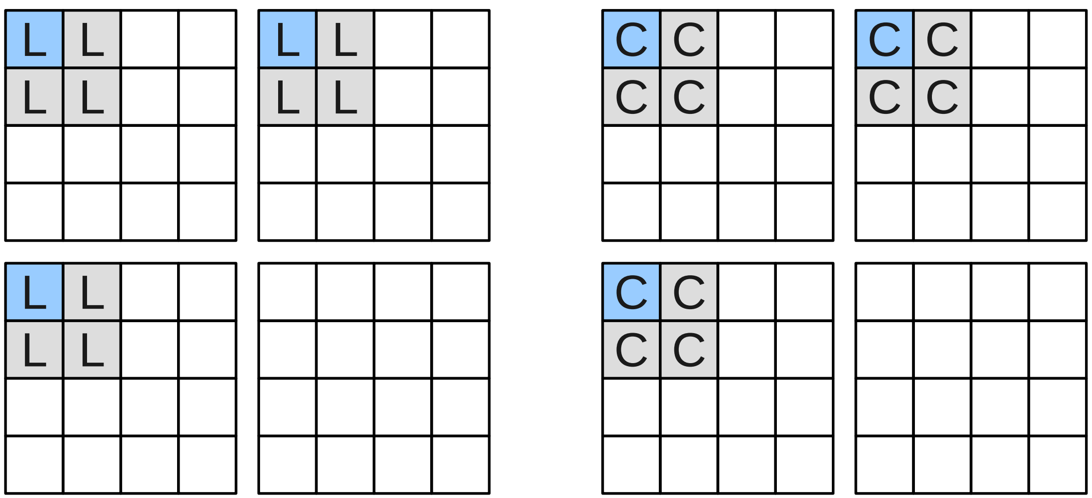
Spatial Domain CfL Properties

- Pros
  - Can predict more features then straight edge extension
  - Can be implemented without signaling α or β

- Cons
  - Complexity scales with block size, for NxN block
    - 4*N + 2 mul's and 8*N + 3 add's to fit model
    - N*N 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:



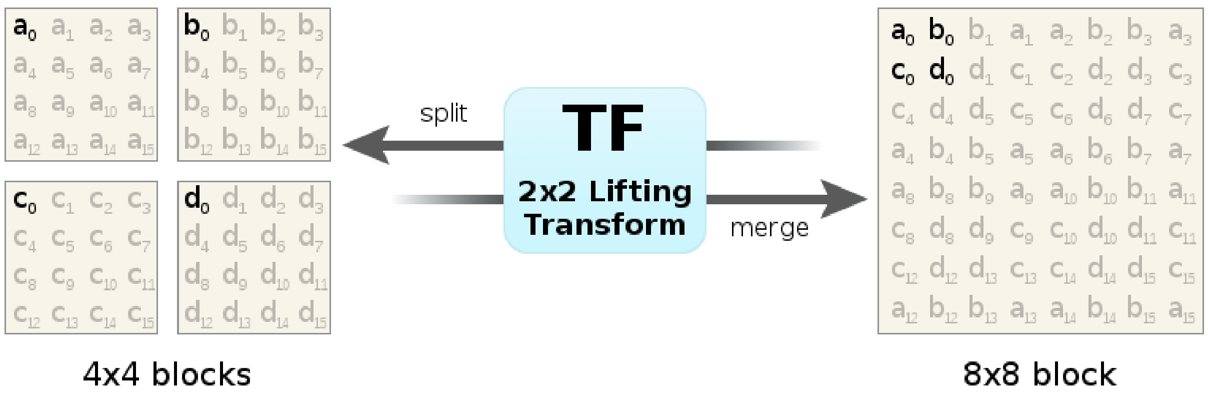
- Use reconstructed luma to predict frequency domain chroma coefficients:

$$C_{DC} = \alpha_{DC} \cdot L_{DC} + \beta_{DC}$$

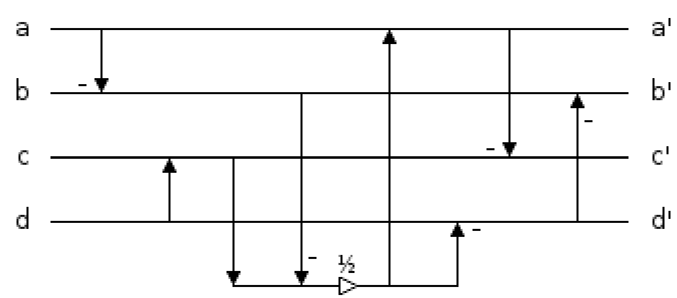
$$C_{AC}(u,v) = \alpha_{AC} \cdot L_{AC}(u,v)$$

Time-Frequency Resolution Switching

- Described in Section 3.2 of `draft-terriberry-netvc-codingtools`
- Trades off spatial resolution for frequency resolution



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



[2] https://xiph.org/~xiphmont/demo/daala/demo3.shtml

Frequency Domain CfL Properties

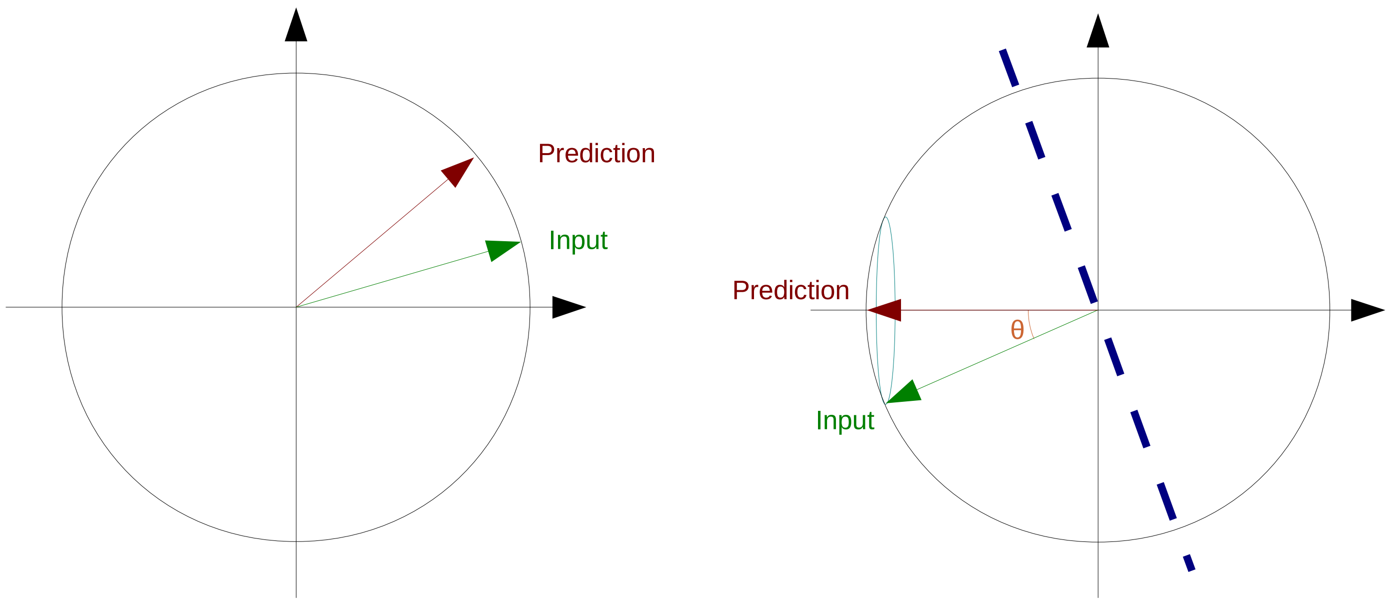
- Pros
  - Can predict more features then straight edge extension
  - Can be implemented without signaling α or β
  - Using TF avoids expensive IDCT / FDCT round trip
  - Model fitting complexity independent of block size
  - No longer required to predict chroma DC from luma DC
  - Can be used with codecs 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 $\mathbf{r}$
  - "gain" predicted by magnitude $\hat{g} = \gamma_g \cdot Q + \|\mathbf{r}\|$
  - "shape" predicted using Householder reflection



$$\mathbf{v} = \frac{\mathbf{r}}{\|\mathbf{r}\|} + s \cdot \mathbf{e}_m$$

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

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 $\hat{\mathbf{x}}_L$

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

- Only difference is *direction* of correlation!

PVQ-CfL Algorithm (Encoder)

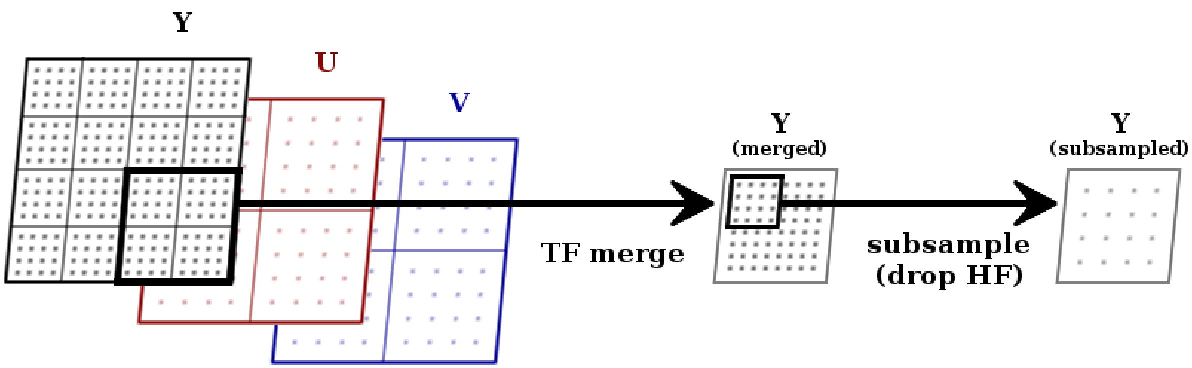
- Code “gain” using scalar quant. (no prediction)

- Code “shape” using PVQ:

  1: Let $\mathbf{r} = \hat{\mathbf{x}}_L$, compute *θ*
  2: Code a *flip* flag, *f* = (*θ* > 90°)
  3: If *f*
  4: Let $\mathbf{r}$ = -$\hat{\mathbf{x}}_L$
  5: End
  6: Code $\mathbf{x}_C$ with PVQ using predictor $\mathbf{r}$

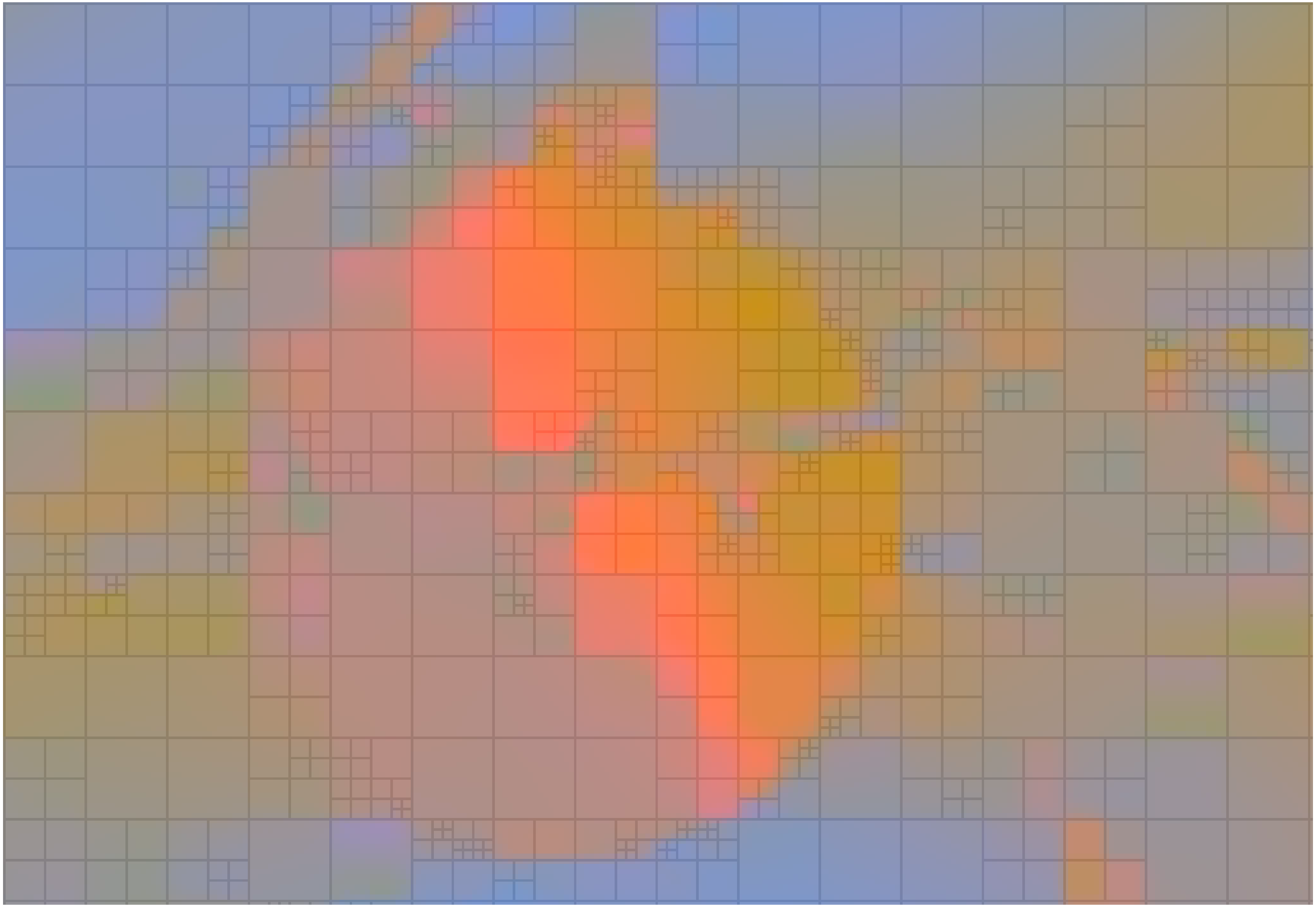
PVQ Chroma-from-Luma Properties

- 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

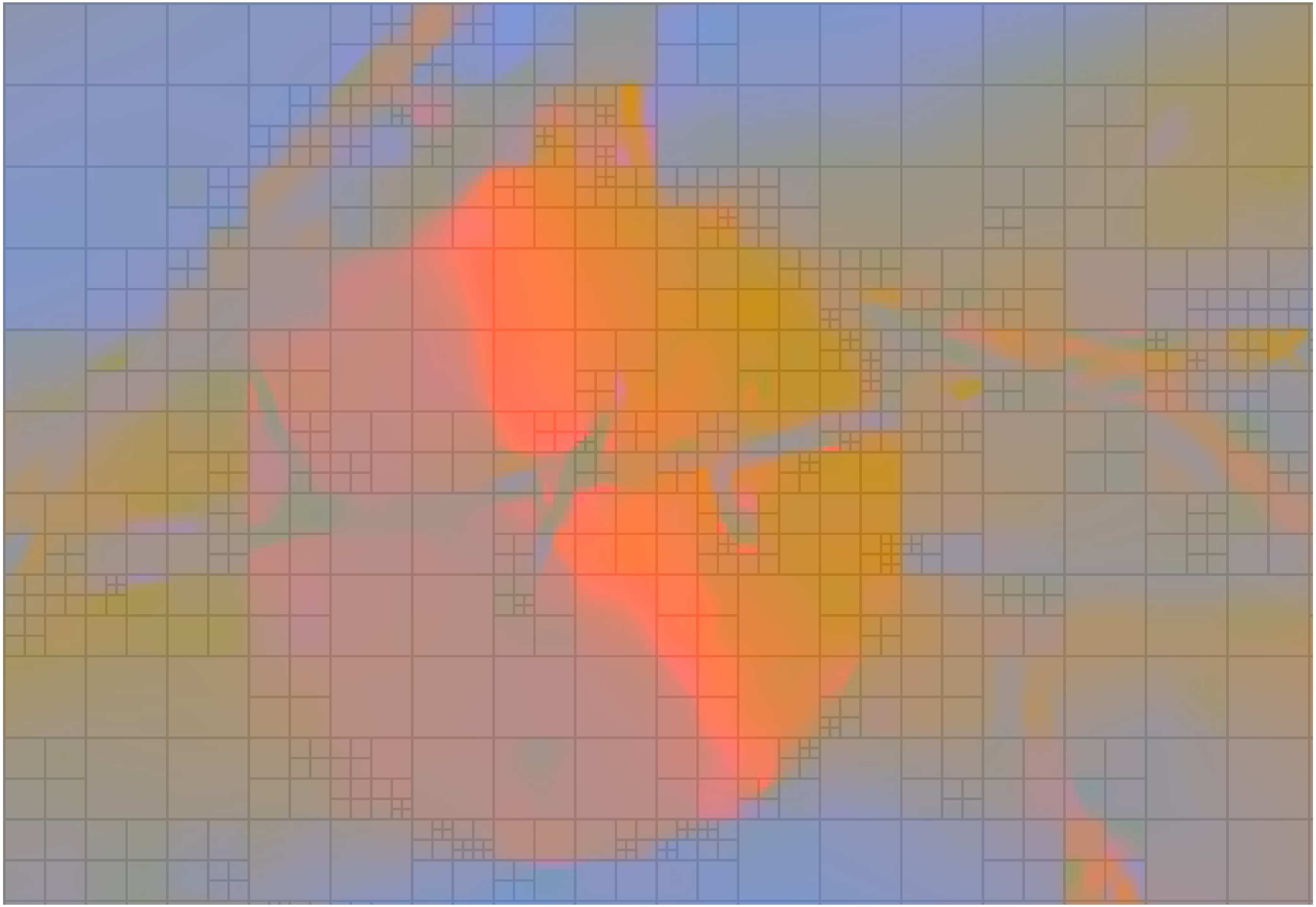


- Cons
  - Requires using PVQ prediction
  - Must code one flip flag per block

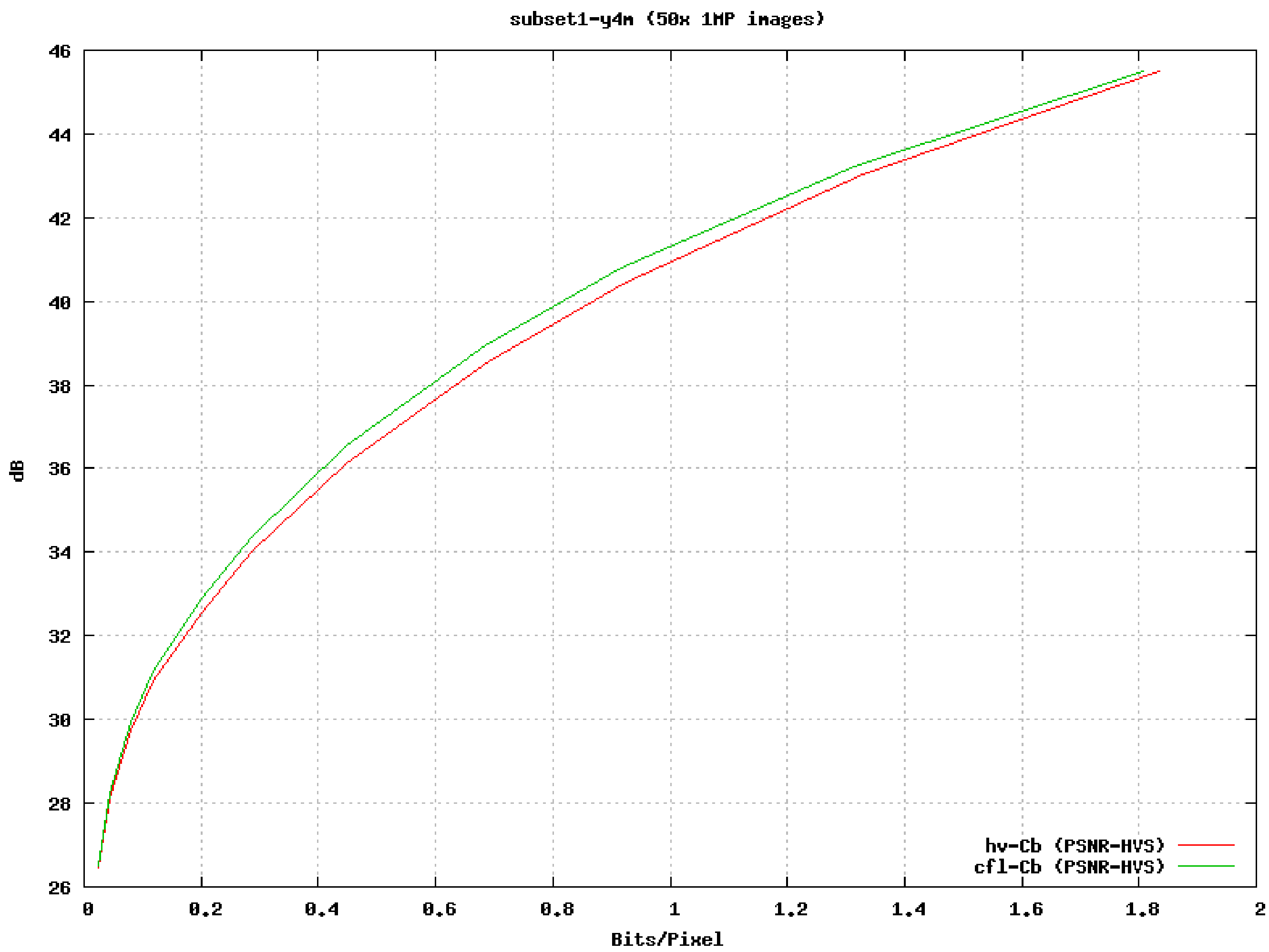
Example – Prediction (using HV)



Example – Prediction (using CfL)



Objective Results



BD-rate for Cb plane:

	RATE (%)	DSNR (dB)
PSNR	-4.60206	0.13743
PSNRHVS	-5.51783	0.24312
SSIM	-10.31658	0.16631
FASTSSIM	-11.50168	0.22043

BD-rate for Cr plane:

	RATE (%)	DSNR (dB)
PSNR	-3.25591	0.09362
PSNRHVS	-4.70448	0.20513
SSIM	-7.99407	0.13373
FASTSSIM	-11.57645	0.22452

Questions?