The Daala Project

An Initiative for Royalty-Free Video

Timothy B. Terriberry

MPEG 113
Future Video Coding
Workshop

The Xiph.Org Foundation & The Mozilla Corporation

Why Mozilla Cares About RF

- Any per-unit royalty is incompatible with the web's most successful business models
  - Software given away for free
  - All revenue indirect
  - Just *counting* how many copies are distributed is prohibitive
- The Web was built on innovation without asking permission
  - Royalty-bearing regimes are the opposite of that
  - Failure for video has negative long-term effects

The Daala Project

- Just RF is not good enough
  - Tried this twice: Theora and VP8
  - Need quality to be as good or better
- Need a better strategy than, “Trust us, we read a lot of patents”
  - People don't believe you
  - One mistake can ruin years of development
    - See: H.264 Baseline

Daala's Strategy

- Look for some elements common to broad classes of patents
  - Only need to avoid one element in a patent claim to be able to say “we don't do that”
- Replace them with fundamentally different techniques
  - Higher risk/higher reward than incremental changes
  - Can avoid vast swaths of IPR
  - Creates new challenges others haven't solved
- Still have to read lots of patents

Basic Design

- Big Pieces
  - Variable Block Size OBMC
  - Lapped Transforms
  - Perceptual Vector Quantization
  - Non-binary Arithmetic Coding
- Other tools
  - Haar DC
  - Frequency Domain Prediction
  - Loop filters

Variable Block Size OBMC

- Usual approach is to overlap with smallest block size (e.g., T13-SG16-C0806)
  - Not really “variable size”
  - Just better entropy coding for “fixed size”

Adaptive Repartitioning

- Zhang et al. (1998): Split large blocks until overlap compatible with smaller neighbors

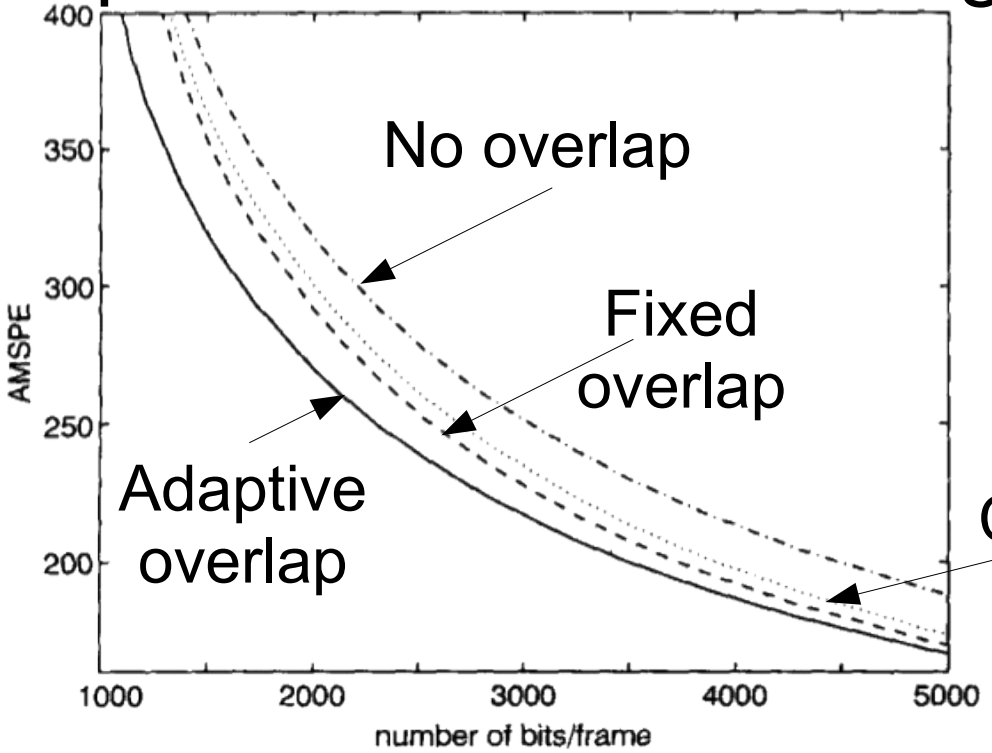


Fig.8 *Rate-distortion performance for football sequence (VSBMC)*

- VSBMC
- VSBMC + GOBMC
- VSBMC + VRP
- VSBMC + AVRP

Limitations of Adaptive Repartitioning

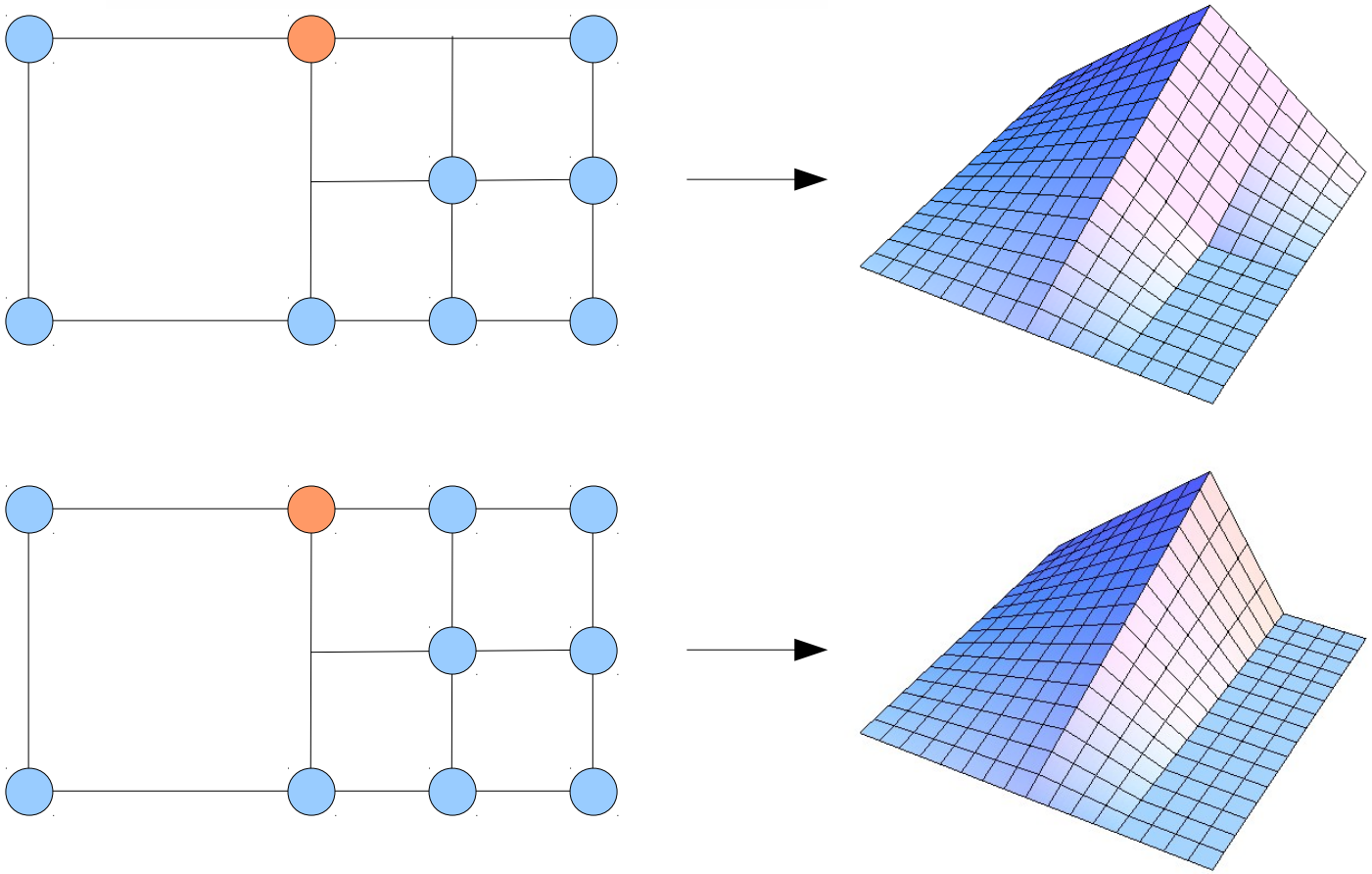
- Might still use small windows with large blocks
- How do you search?
  - 2-D dependency on neighbors' sizes → NP hard

Daala's Approach

- Restrict adjacent block sizes to differ by no more than a factor of two
- Use a “4-8 mesh” to impose the restriction
  - Data structure for surface simplification in compter graphics
  - Has a good search algorithm (Balmelli 2001)
- Define interpolation rules based on this restriction
  - Maintains continuity when one side of an edge is split and the other isn't

Blending Windows for Mismatched Block Sizes

- Simple sums of bilinear weights

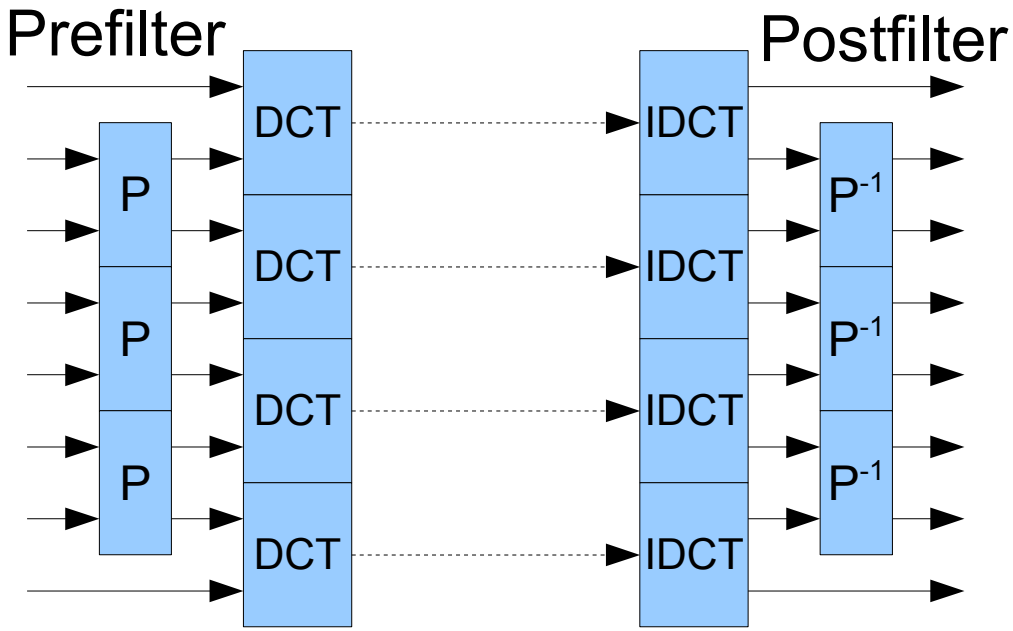


Comparison to Block Matching from Thor

- Swapped out Daala's whole MC system with the one from Cisco's Thor
- Results from IETF 93:
  - Daala's MC uses 7%...9% less rate (disabling Thor's intra modes, capping block size at 32x32)
  - Intra modes get Thor 1.5%...2%
- Results with 64x64 blocks and Thor's intra
  - 64x64 helps Thor more than Daala (entropy coding)
  - PSNR about equal, Daala 2%...4% ahead on other metrics

Lapped Transforms

- *Non*-adaptive, *invertible* deblocking post-filter
- Encoder applies the inverse (a blocking filter)
- Biorthogonal (non-unit scaling, some redundancy)



Intra Prediction with Lapped Transforms

- We can’t copy pixels until we undo the lapping
  - We can’t undo the lapping until we’ve predicted those pixels
- Tried training general linear frequency-domain predictors with multiple directional modes, etc.
- **Doesn't work**
  - Well, for 4x4 it's 0.5 dB better than DCT+VP8 intra
  - Sparse solutions lose all the gains on larger sizes
  - Issues with multiple block sizes

Intra Prediction with Lapped Transforms

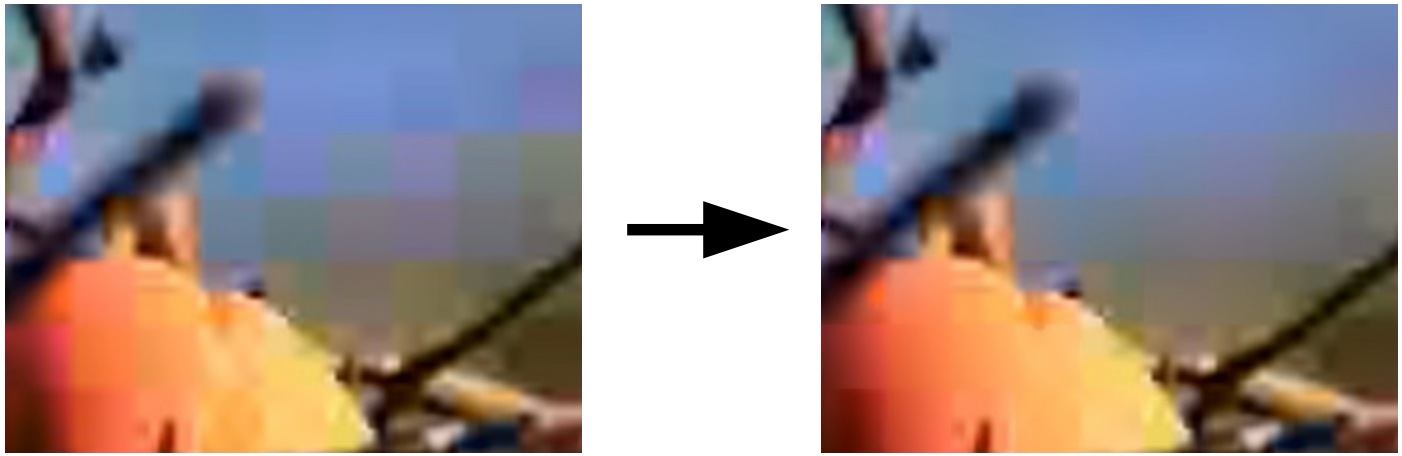
- Just copy transform coefficients
  - Currently just horizontal and vertical directions
    - Only when block sizes match
  - Chroma (color) predicted from luma (brightness)
- Not very good (1%...2% gains), but really fast
  - Mostly prevents us from looking really bad on synthetic test cases (up to 50% gains)

Lapping Sizes

- Originally tried something like Zhang's Adaptive Partitioning for OBMC
  - Maximum lapping size compatible with neighbors
- Same problem: how do you search?
  - 2-D dependency on neighbors' sizes → NP hard
- “Fixed Lapping”: Always use 4-point lapping
  - Much less ringing, more blocking (esp. gradients), less detail (coding gain)
  - 14% rate reduction for PSNR/SSIM, 5% for FastSSIM/PSNR-HVS

Bilinear Loop Filter

- Compensate for added blocking
  - Doesn’t look outside of current block!
  - Blend decoded block with bilinear interpolation of corner pixels



Perceptual Vector Quantization

- Separate “gain” (energy) from “shape” (spectrum)
  - Vector = Magnitude × Unit Vector (point on sphere)
- Potential advantages
  - Can give each piece different rate allocations
    - Preserve energy (contrast) instead of low-passing
  - 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!
  - Better representation of coefficients

PVQ with Prediction

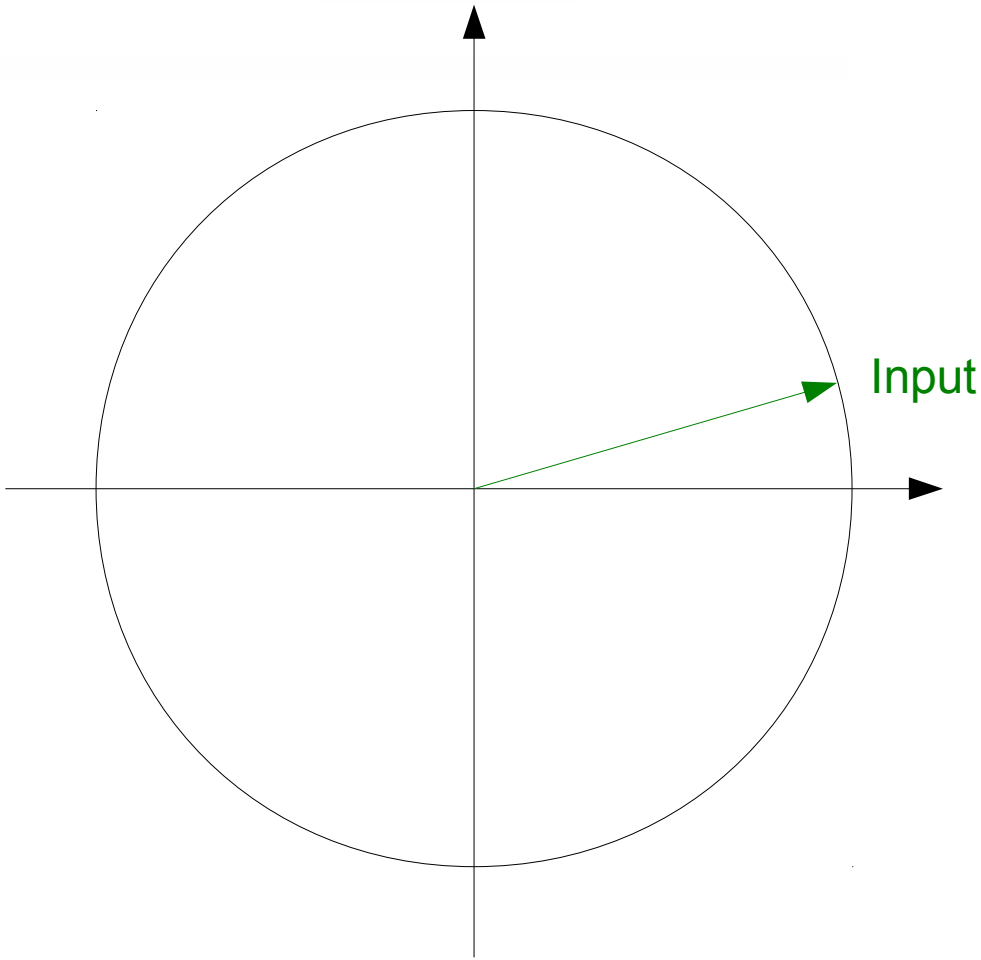
- Subtracting and coding a residual loses energy preservation
  - The “gain” no longer represents the energy of the original signal

What Does Prediction Really Do?

- Prediction changes the *probability* of points near the predictor
  - Highly probable things are cheap to code
- Predicting gains is easy
  - Subtract gain of predictor
- Enumerating points on a sphere near an arbitrary point (to model probabilities) is hard
  - Solution: *Transform* the space so we can single out points near the predictor

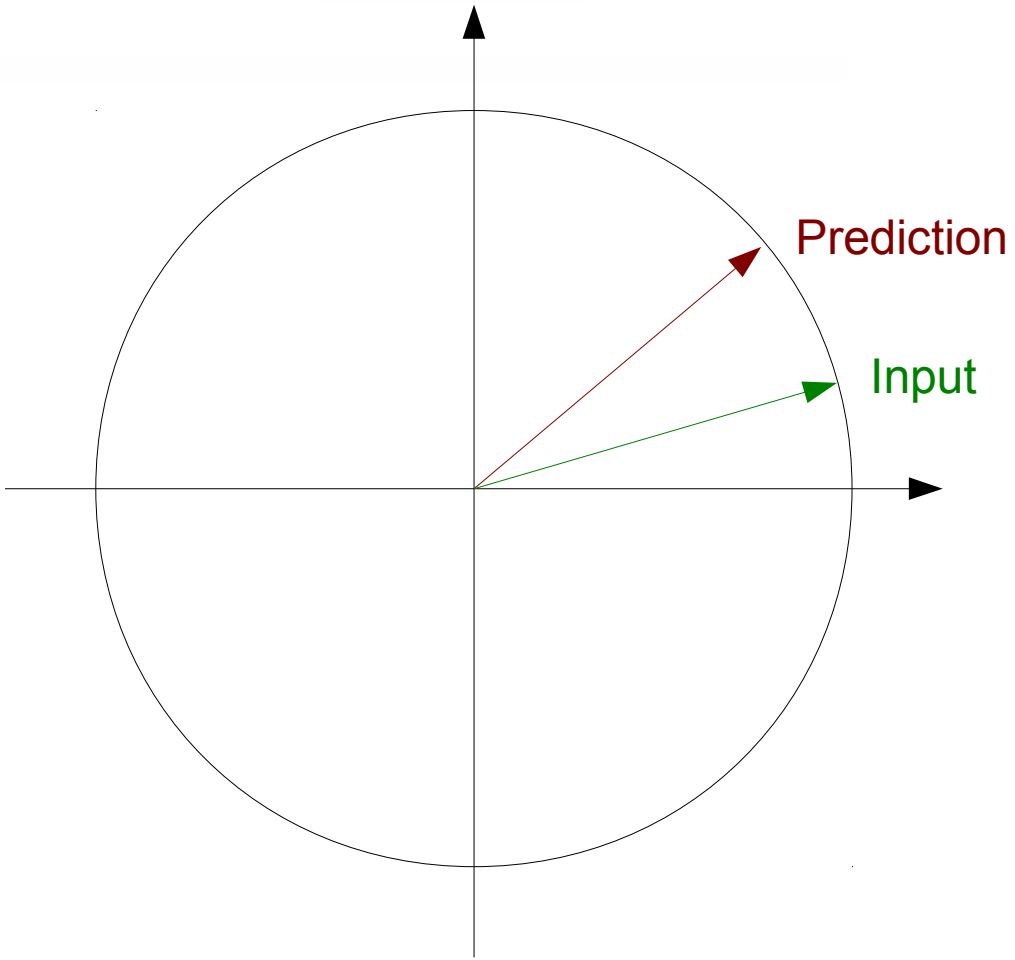
2-D Projection Example

- Input



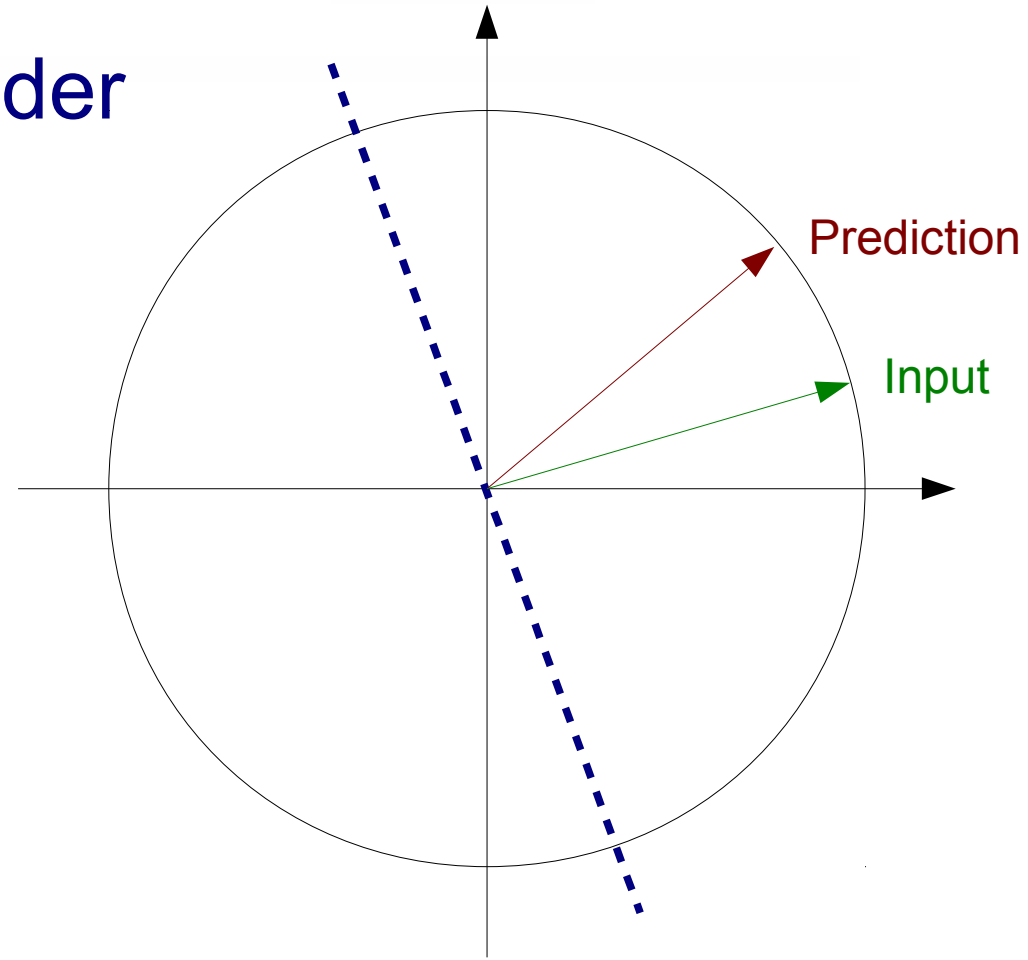
2-D Projection Example

- Input + Prediction



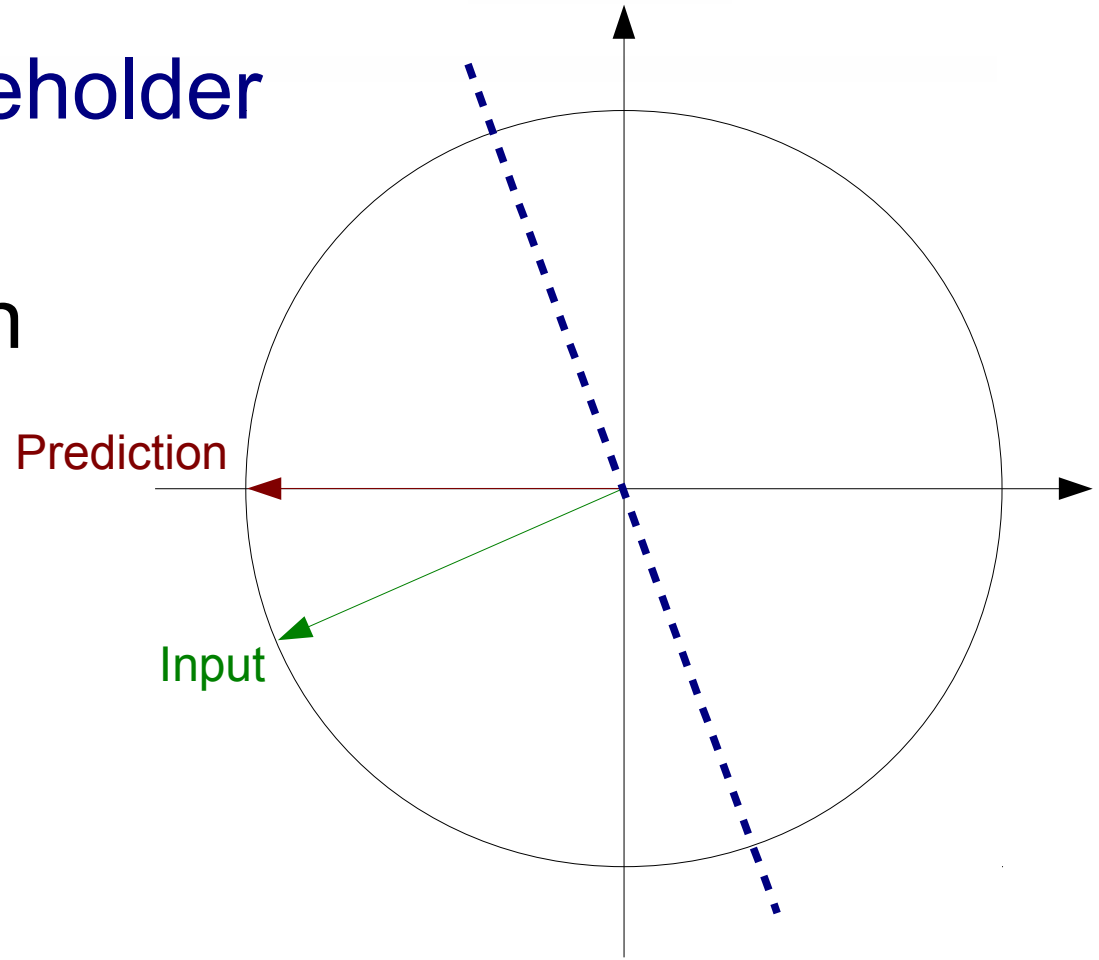
2-D Projection Example

- Input + Prediction
- Compute Householder Reflection



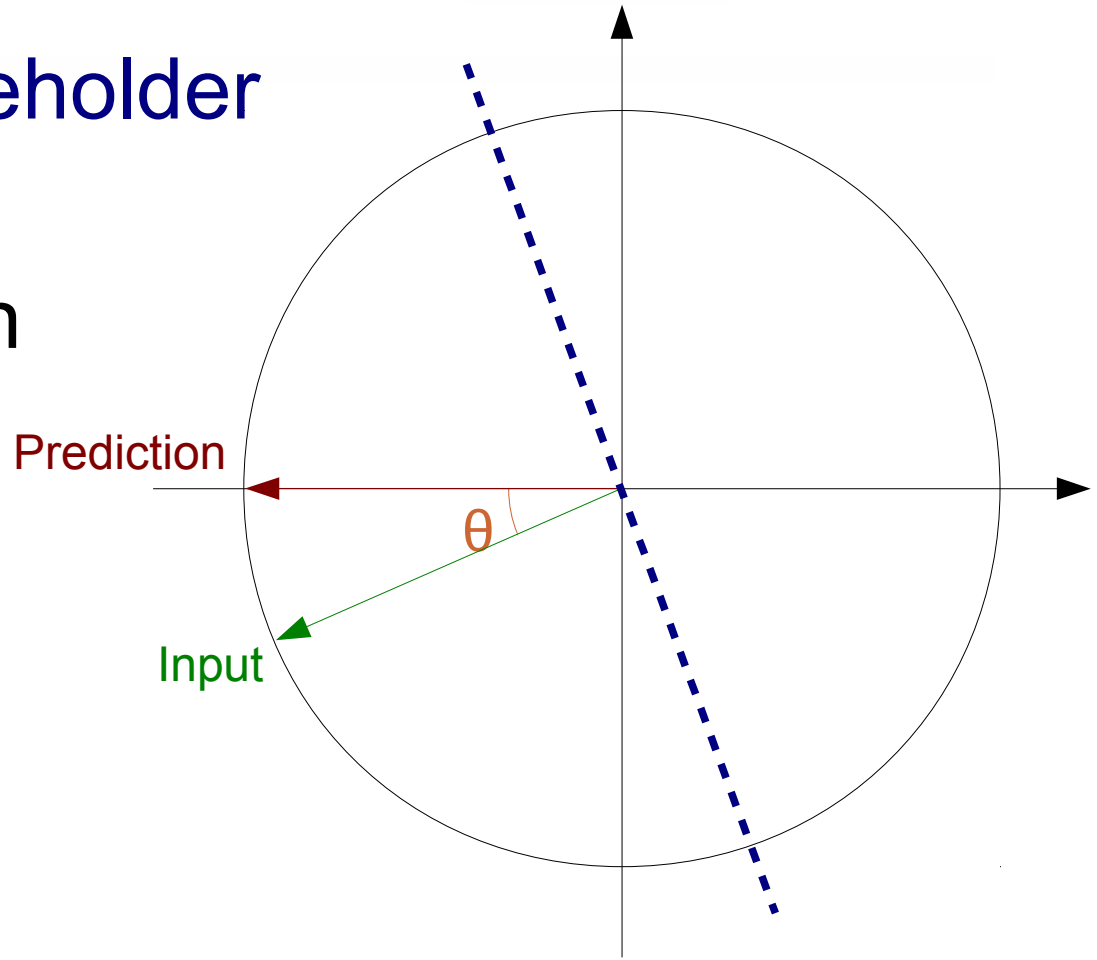
2-D Projection Example

- Input + Prediction
- Compute Householder Reflection
- Apply Reflection



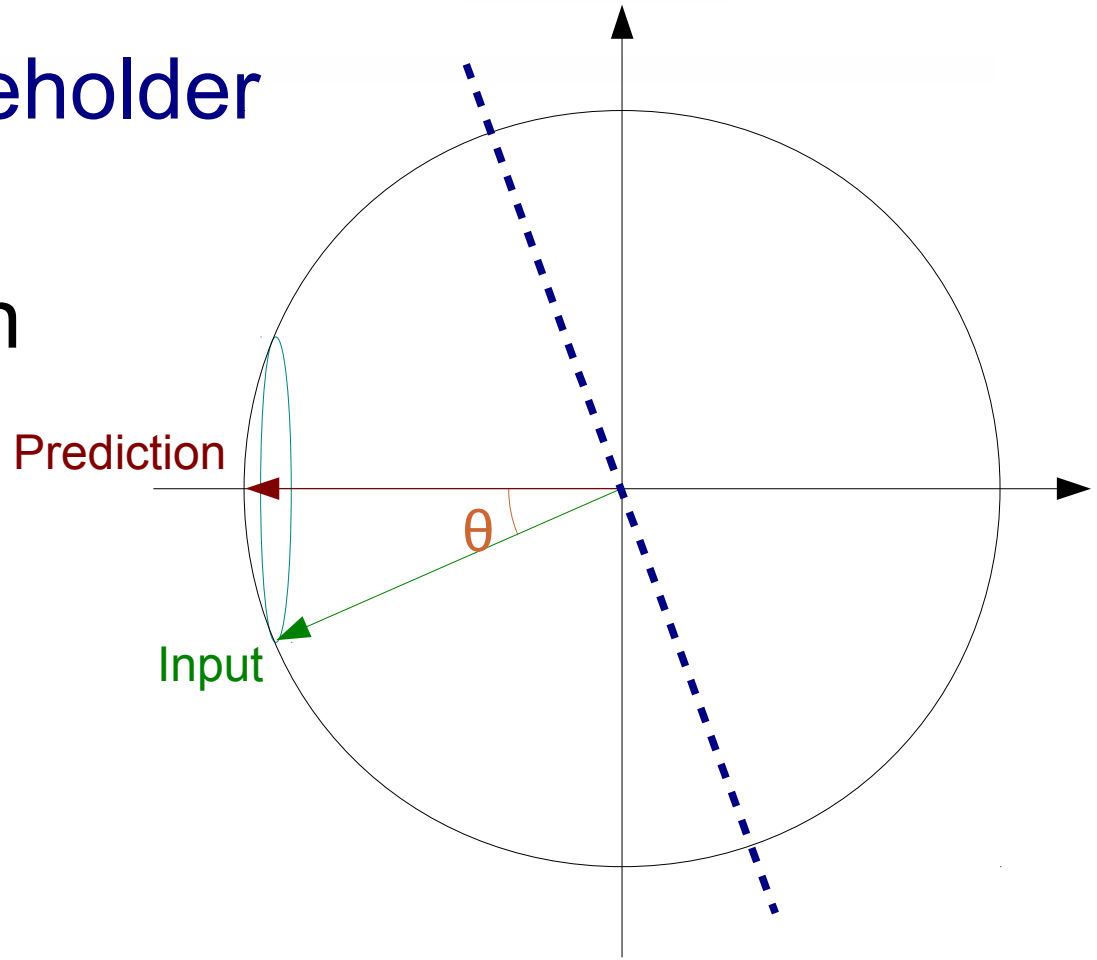
2-D Projection Example

- 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

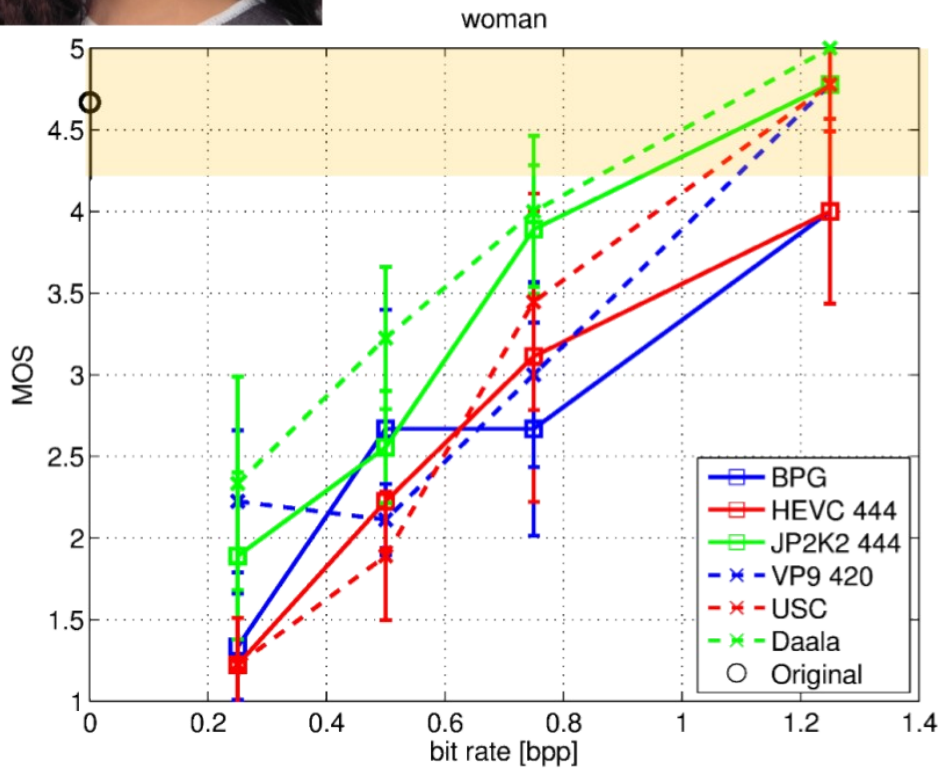


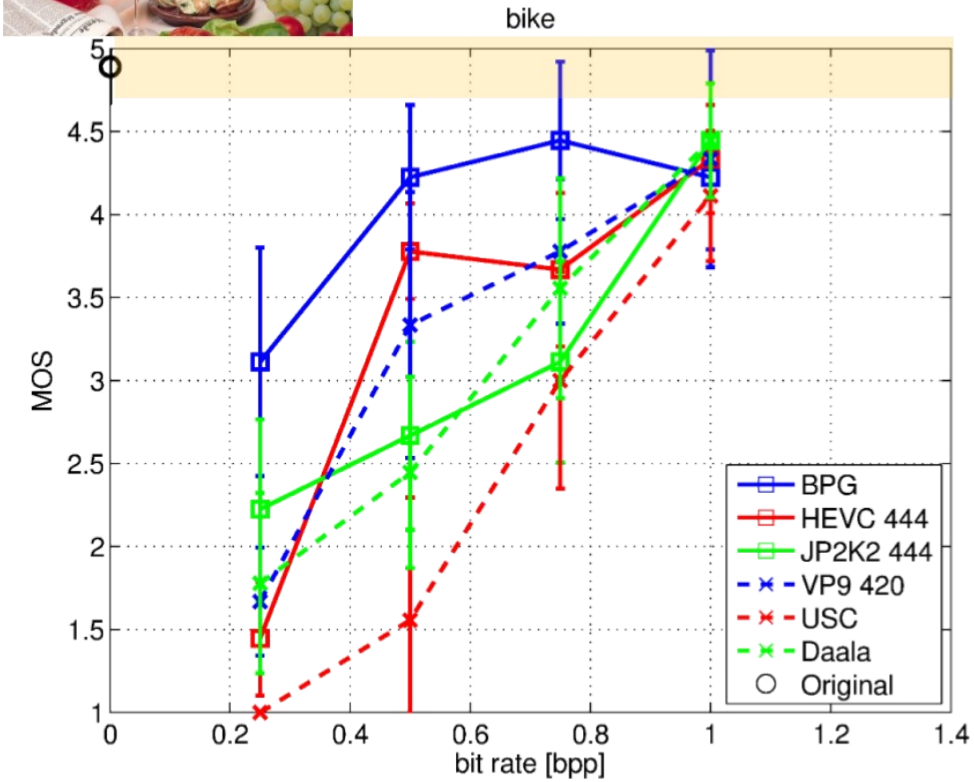
What does this accomplish?

- Creates another “intuitive” parameter, θ
  - “How much like the predictor are we?”
  - $\theta = 0 \rightarrow$ use predictor exactly
- Remaining N - 1 dimensions are coded with VQ
  - N - 2 degrees of freedom
    - We know their magnitude is gain*sin(θ)
  - VQ here just to remove redundant degrees of freedom

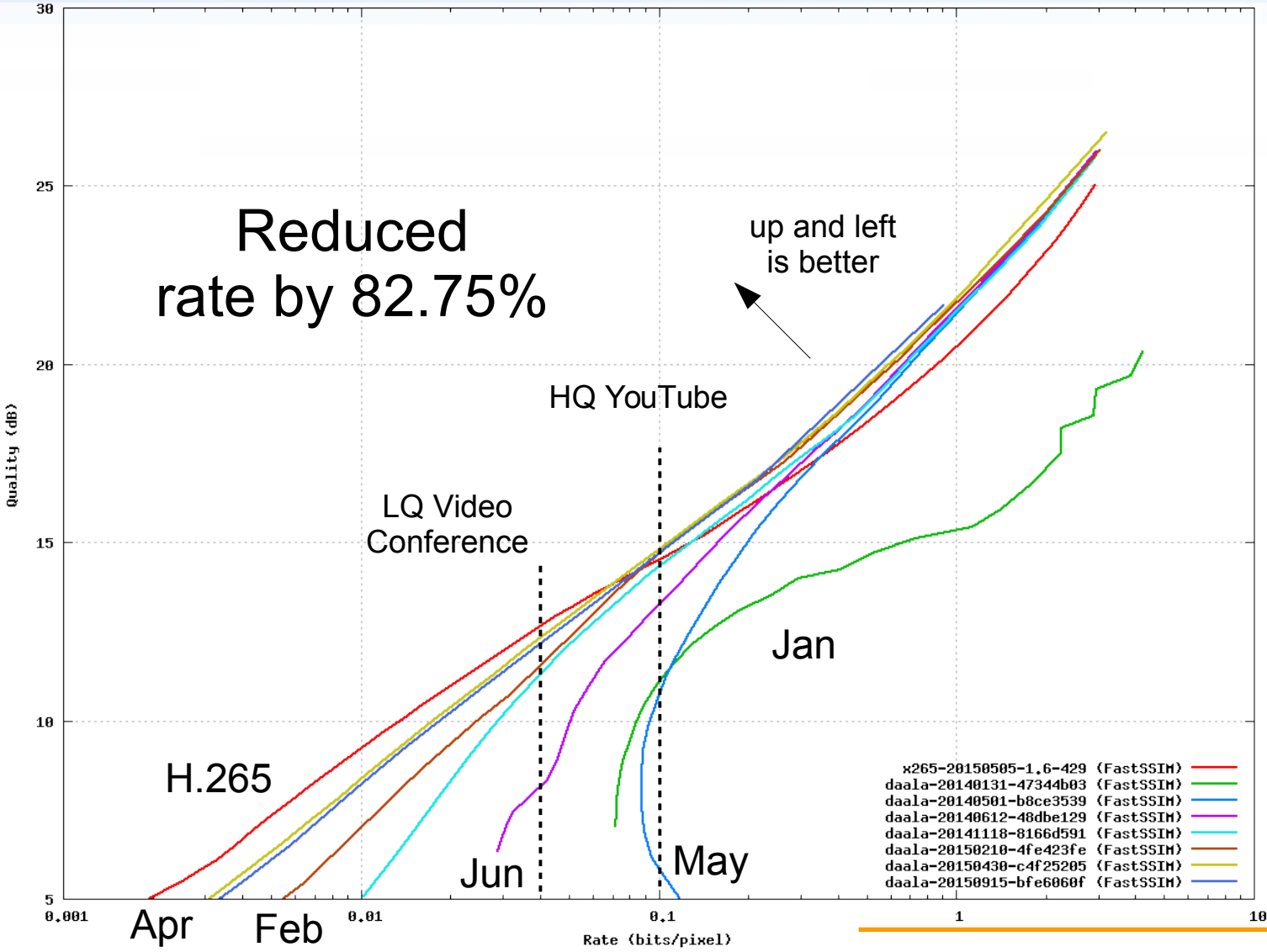
Progress and Metrics

PCS2015
Still Image Challenge

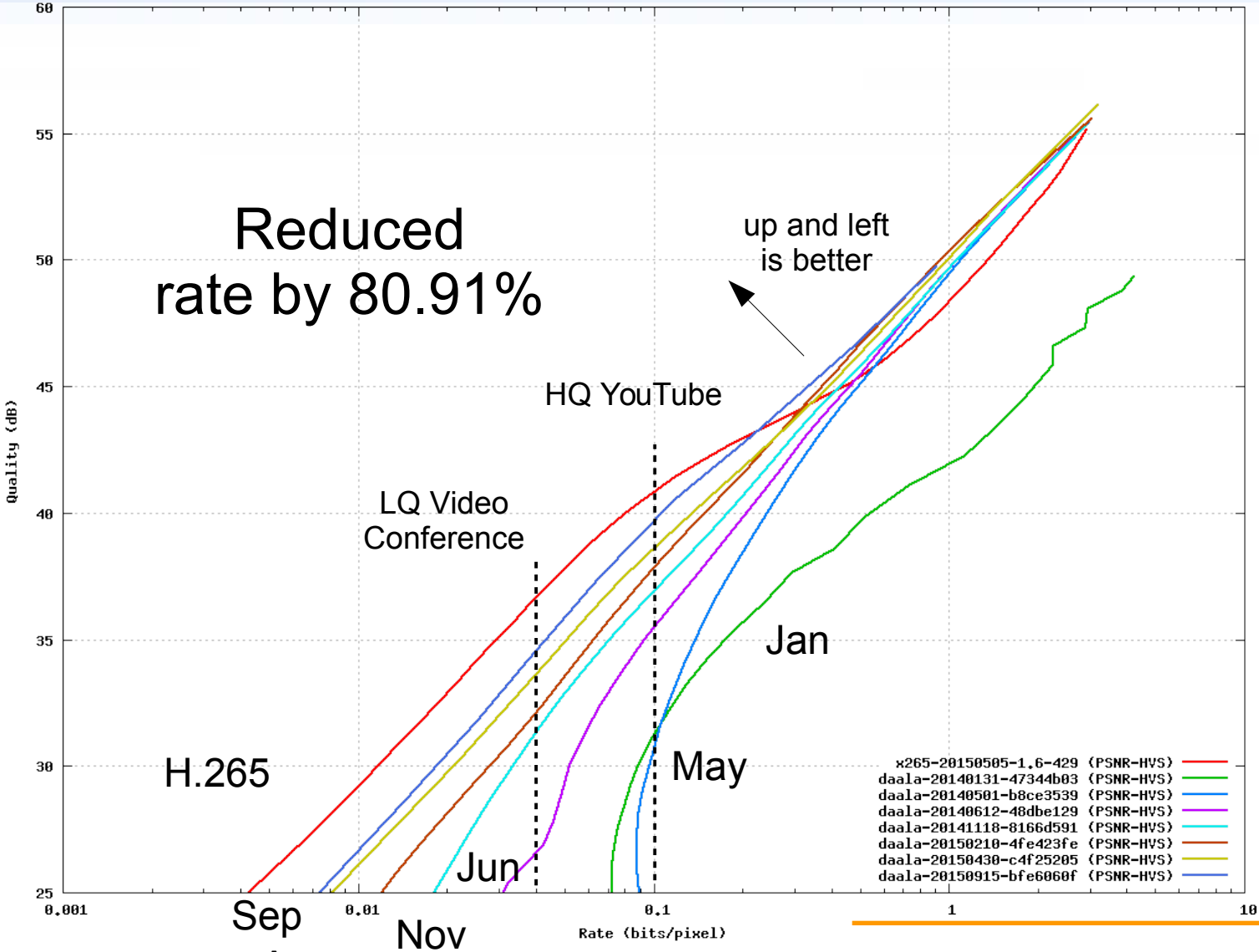




FastSSIM Progress
Jan. 2014 to Sep. 2015



PSNR-HVS Progress: Jan. 2014 to Sep. 2015



Questions?