Perceptual Vector Quantization For Video Coding

Jean-Marc Valin
Timothy B. Terriberry

Perceptual Vector Quantization

- Separate “gain” (contrast) from “shape” (spectrum)
  - Vector = Magnitude × 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!

Simple Case: PVQ without a Predictor

- Scalar quantize gain
- Shape: place *K* unit pulses in *N* dimensions

$$\mathbf{y} \in \mathbb{Z}^N : \sum_{i=0}^{N-1} |y_i| = K$$

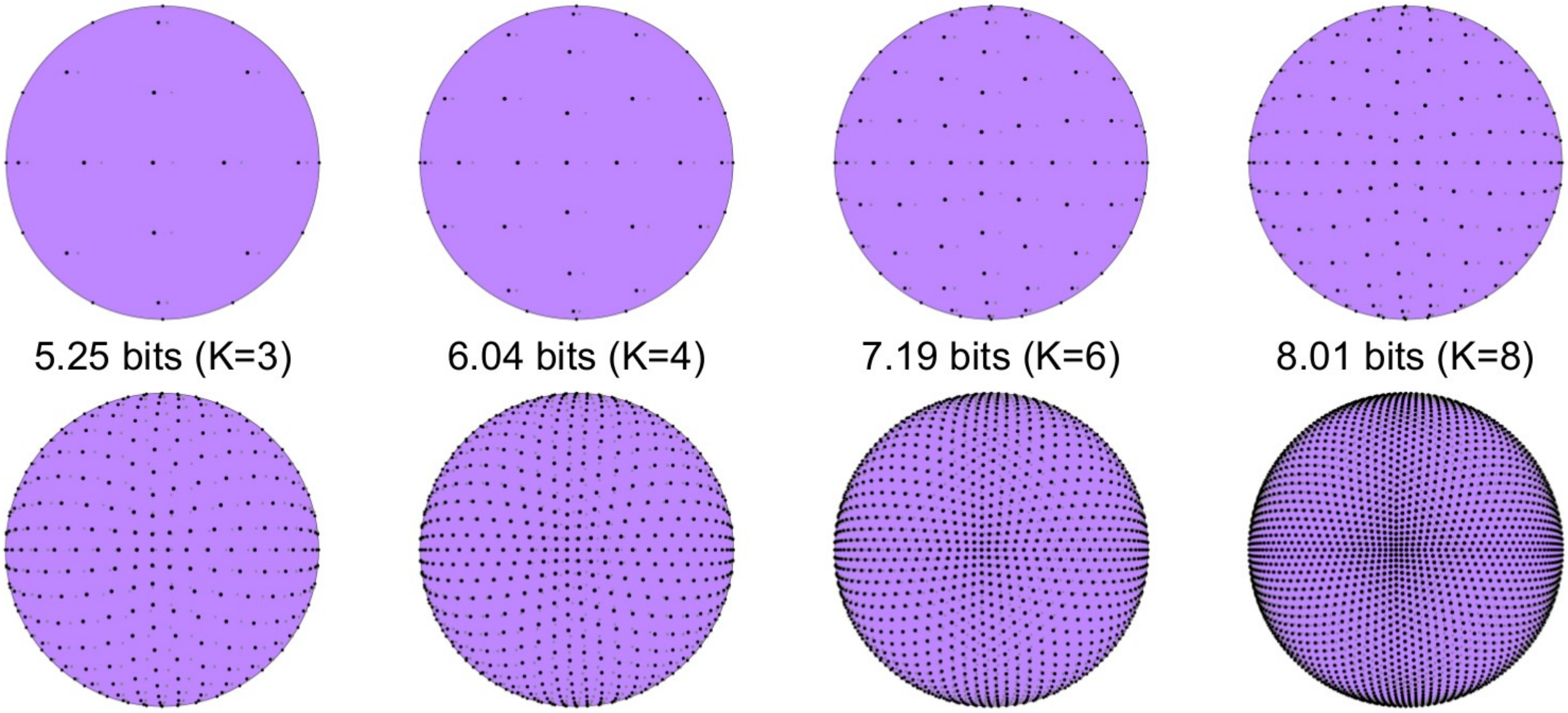
  - Only has (*N* - 1) degrees of freedom

- Normalize to unit L_2 norm

$$\mathbf{u} = \mathbf{y} / \left\| \mathbf{y} \right\|_{L2}$$

- *K* is derived implicitly from the gain

Codebook for *N*=3 and different *K*



5.25 bits (K=3)

6.04 bits (K=4)

7.19 bits (K=6)

8.01 bits (K=8)

8.92 bits (K=11)

10.00 bits (K=16)

11.05 bits (K=23)

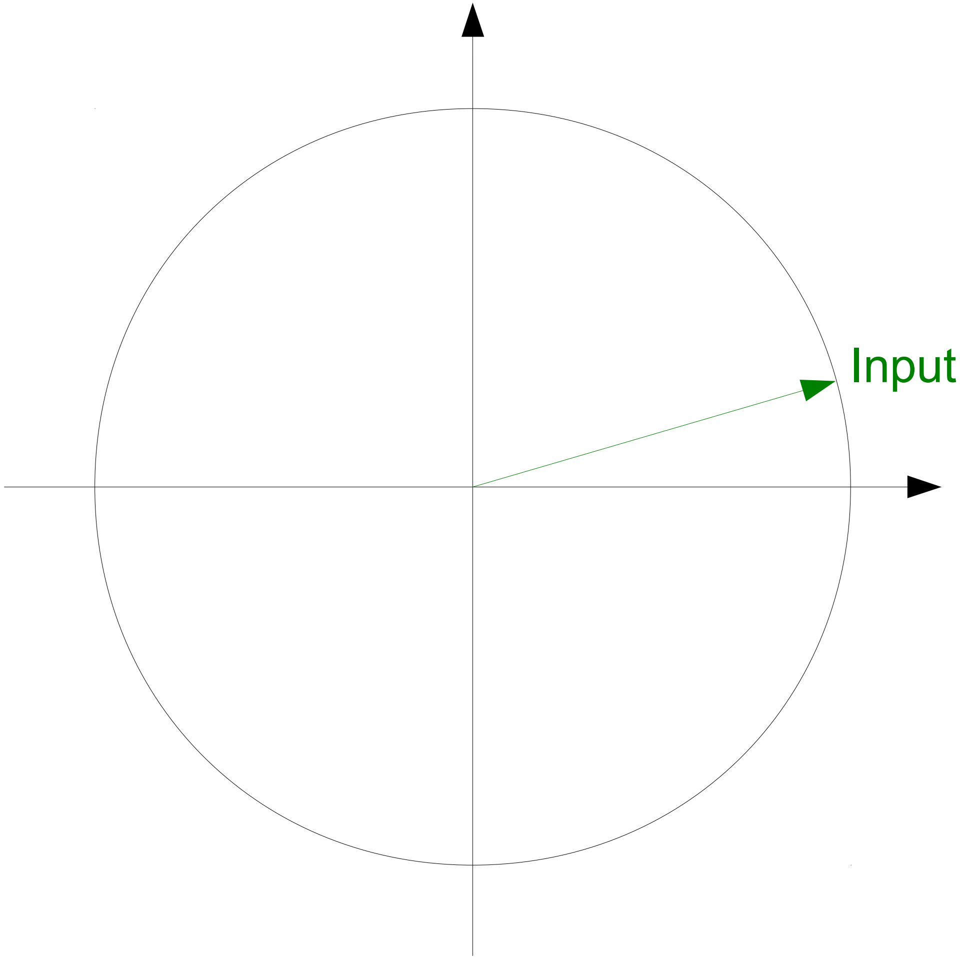
12.00 bits (K=32)

Using 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  - The “gain” no longer represents the contrast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
  - Predicting gain is easy
  - Warping codebooks or probability distributions on the surface of a hyper-sphere is hard
- Solution: transform the space to make it easier

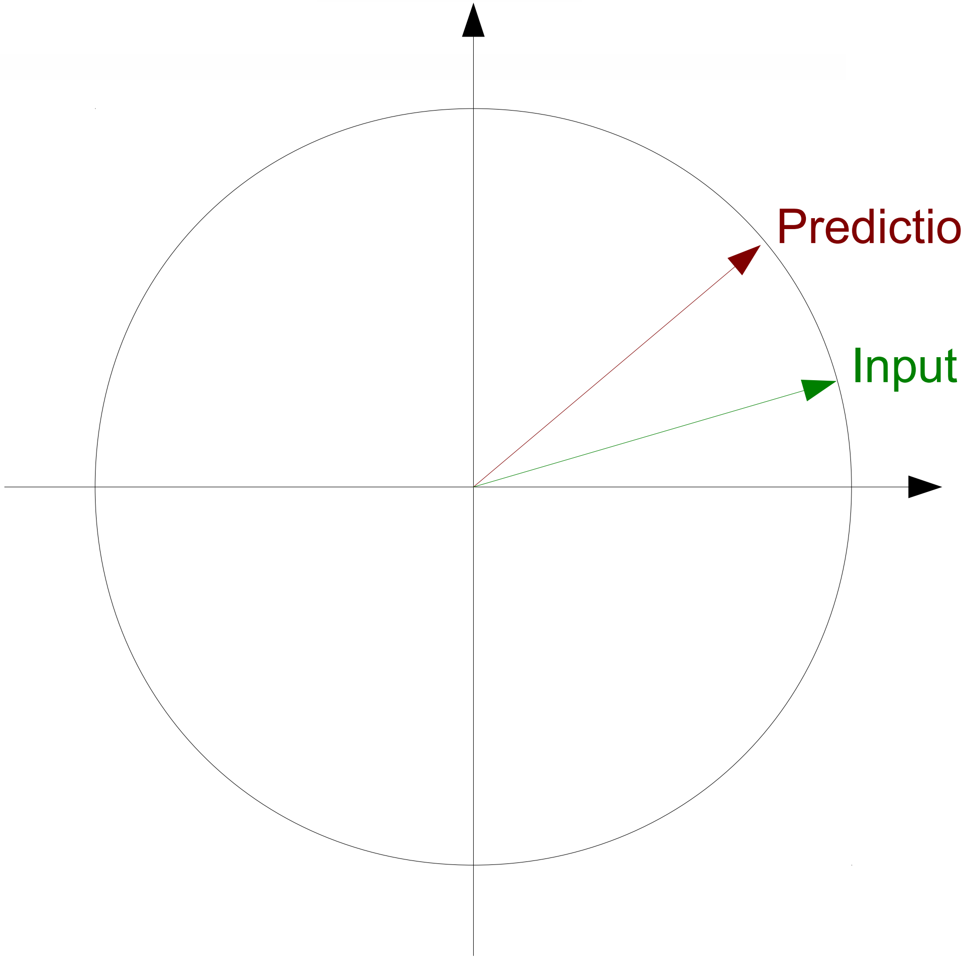
2-D Projection Example

- Input



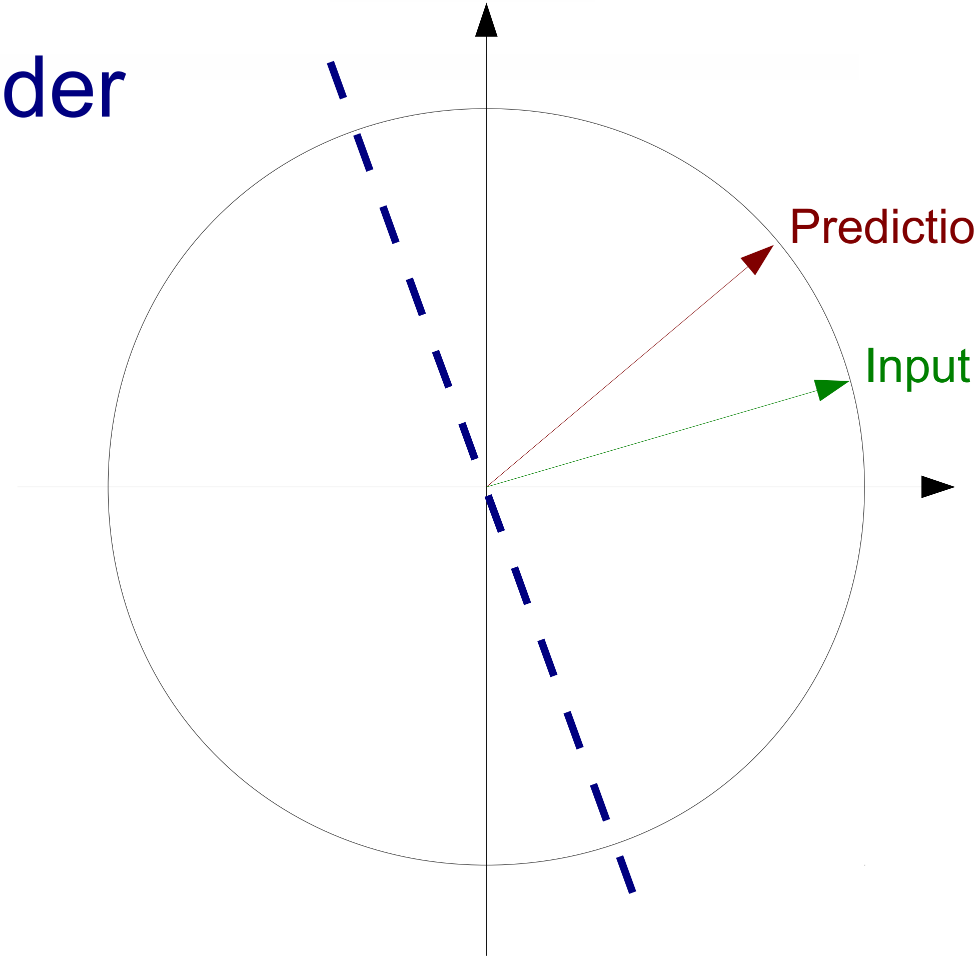
2-D Projection Example

- Input + Prediction



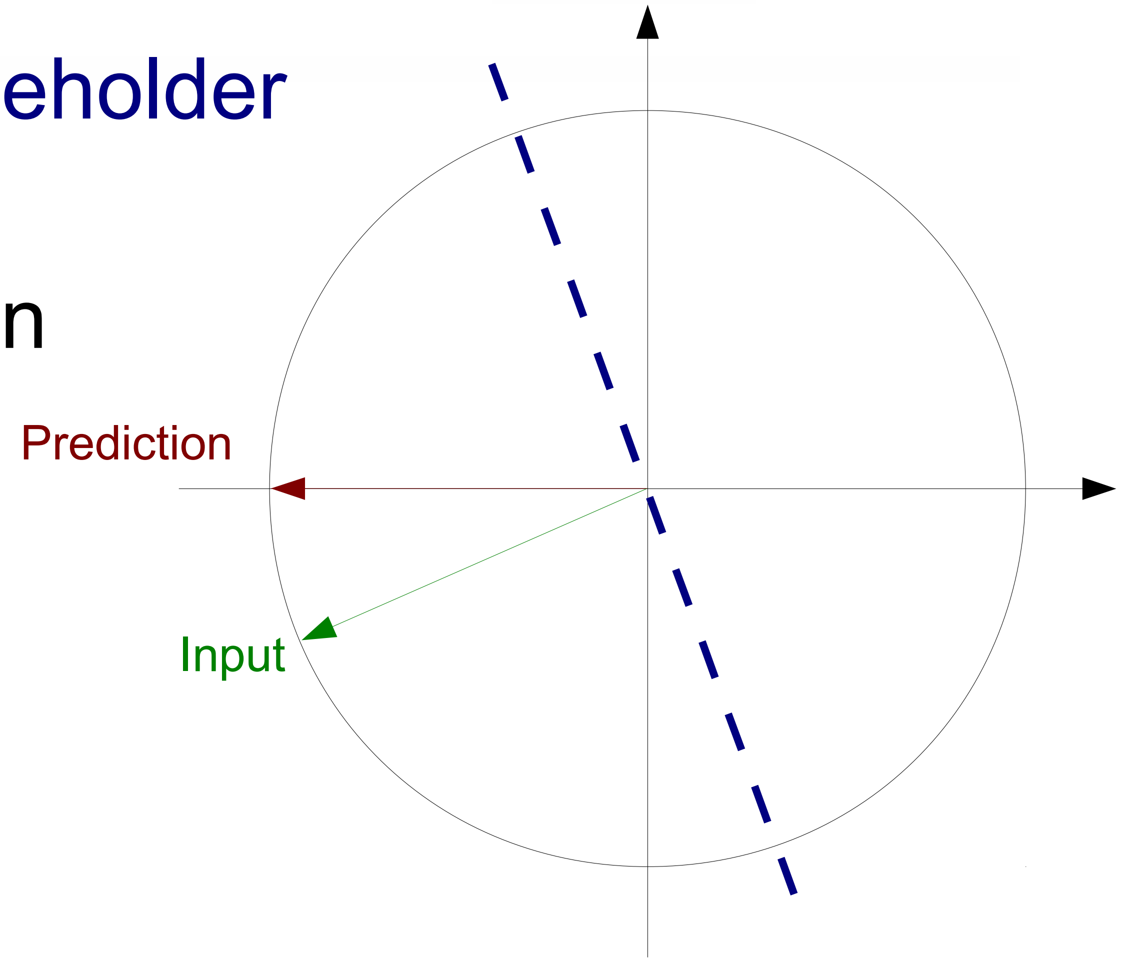
2-D Projection Example

- Input + Prediction
- Compute Householder Reflection



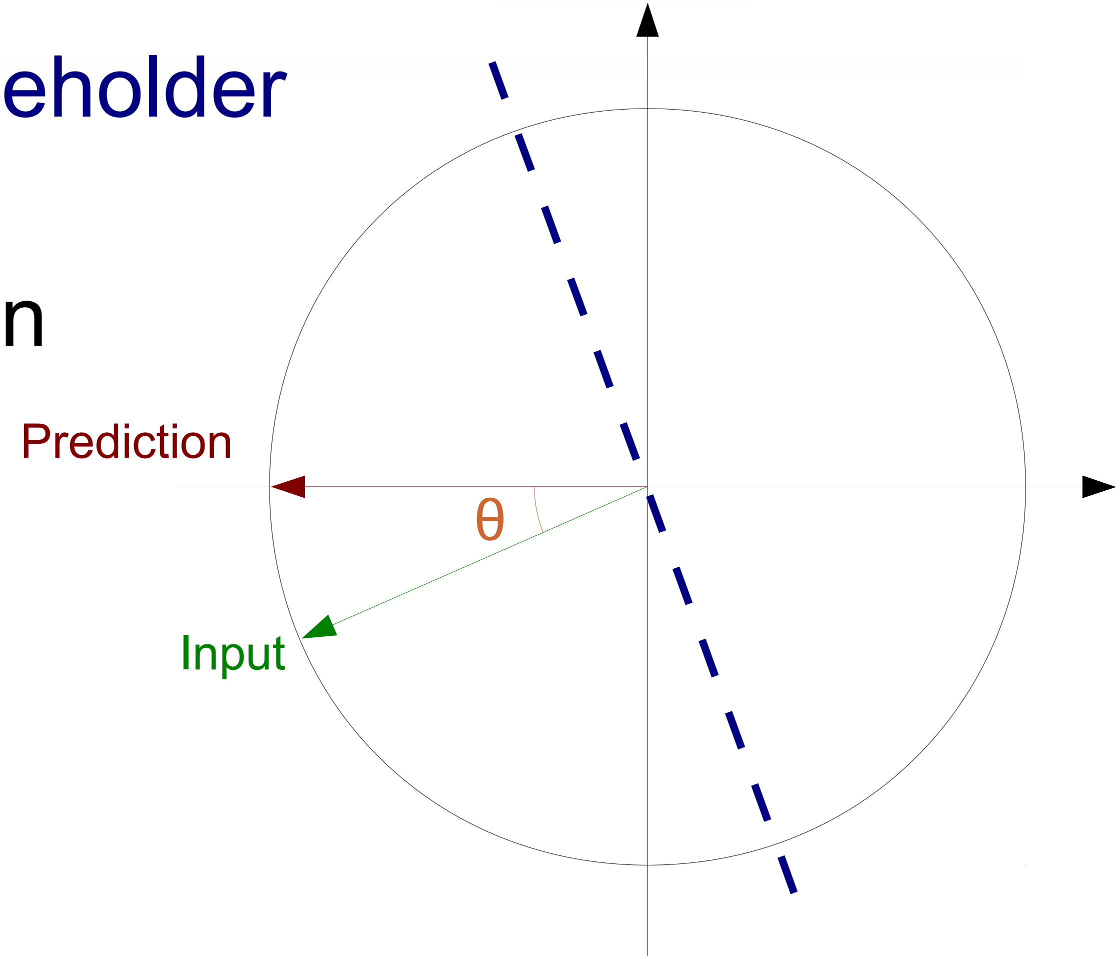
2-D Projection Example

- Input + Prediction
- Compute Householder Reflection
- Apply Reflection



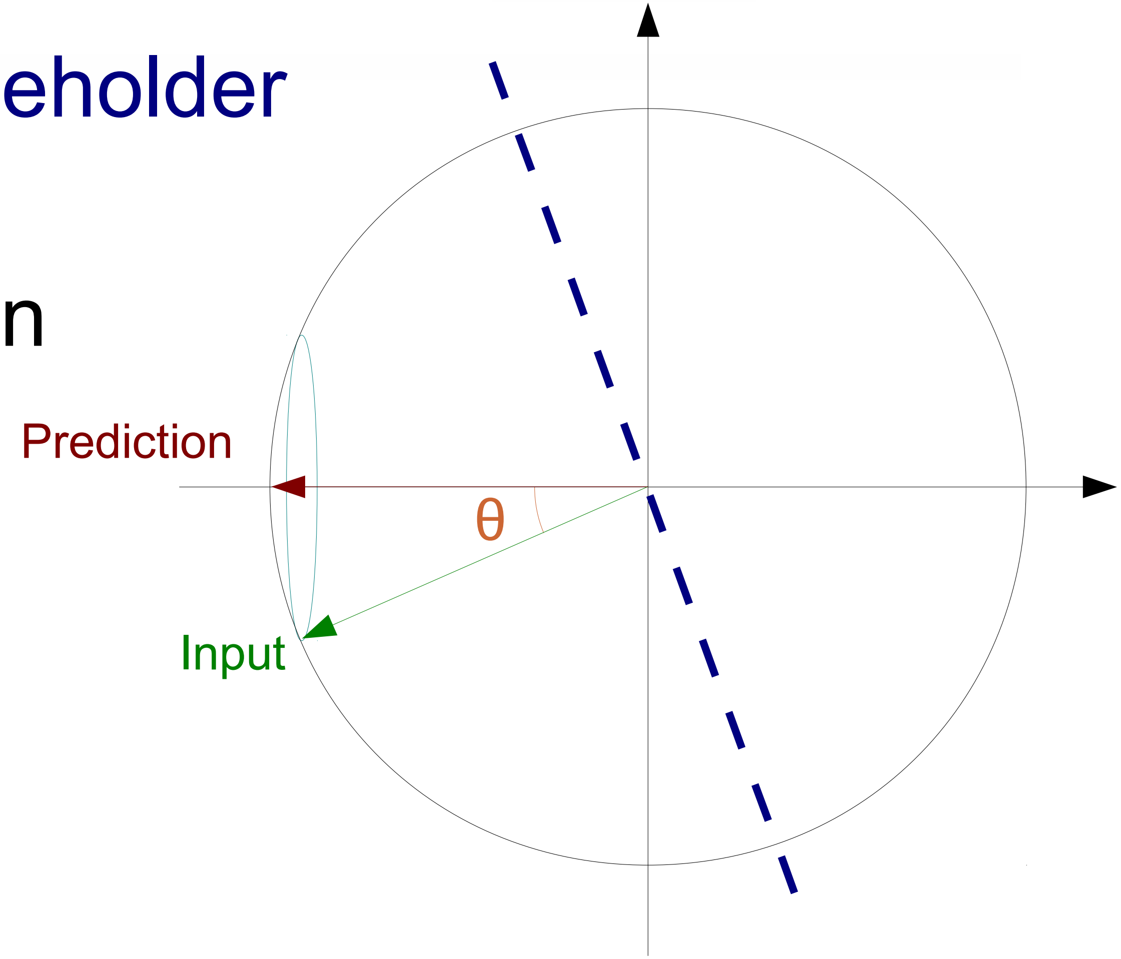
2-D Projection Example

- Input + Prediction
- Compute Householder Reflection
- Apply Reflection
- Compute & code angle



2-D Projection Example

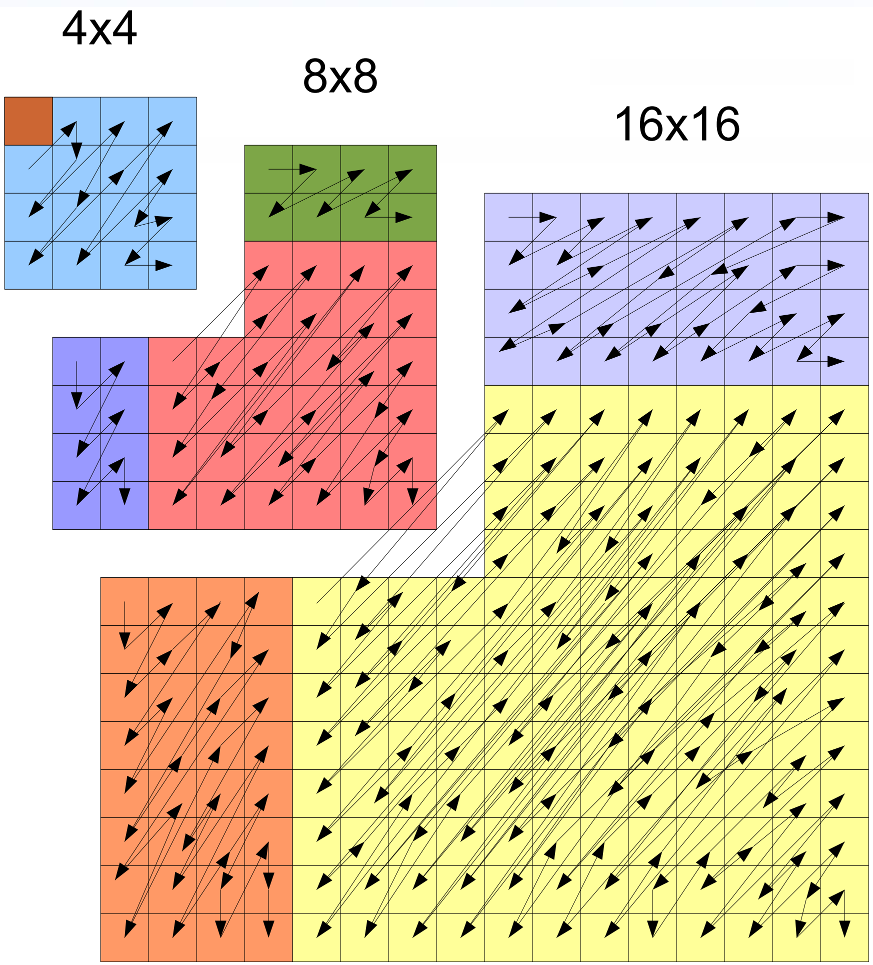
- Input + Prediction
- Compute Householder Reflection
- Apply Reflection
- Compute & code angle
- Code other dimensions



What does this accomplish?

- Creates another “intuitive” parameter, θ
  - “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 gain*sin(θ)
  - Only has (N - 2) degrees of freedom
- Instead of subtraction (translation), we’re scaling and reflecting

Band Structure



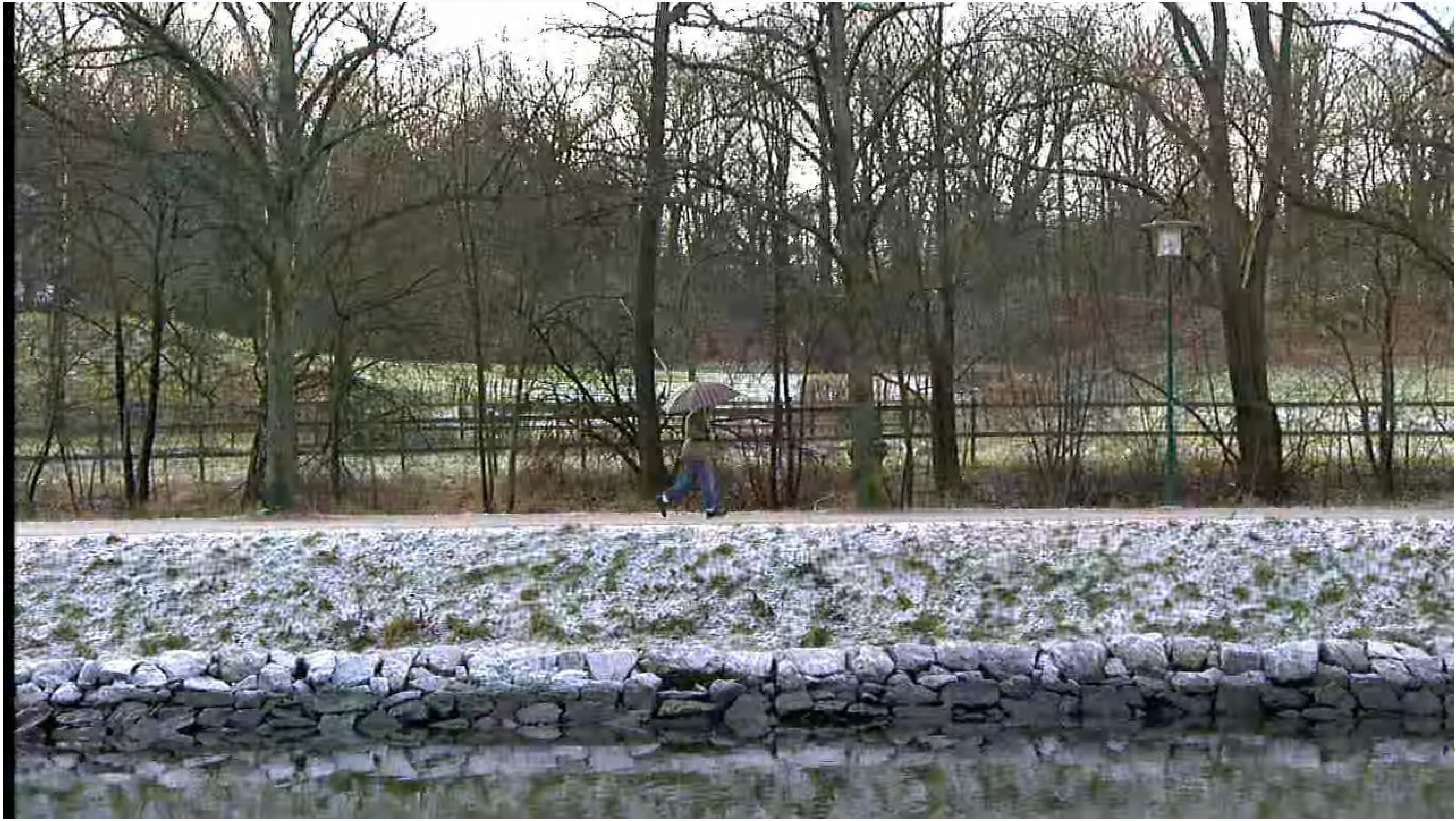
To Predict or Not to Predict...

- $\theta \geq \pi/2$ → Prediction not helping
  - Could code large θ's, but doesn't seem that useful
  - Need to handle zero predictors anyway
- Current approach: code a “noref” flag
  - Jointly coded with gain and θ

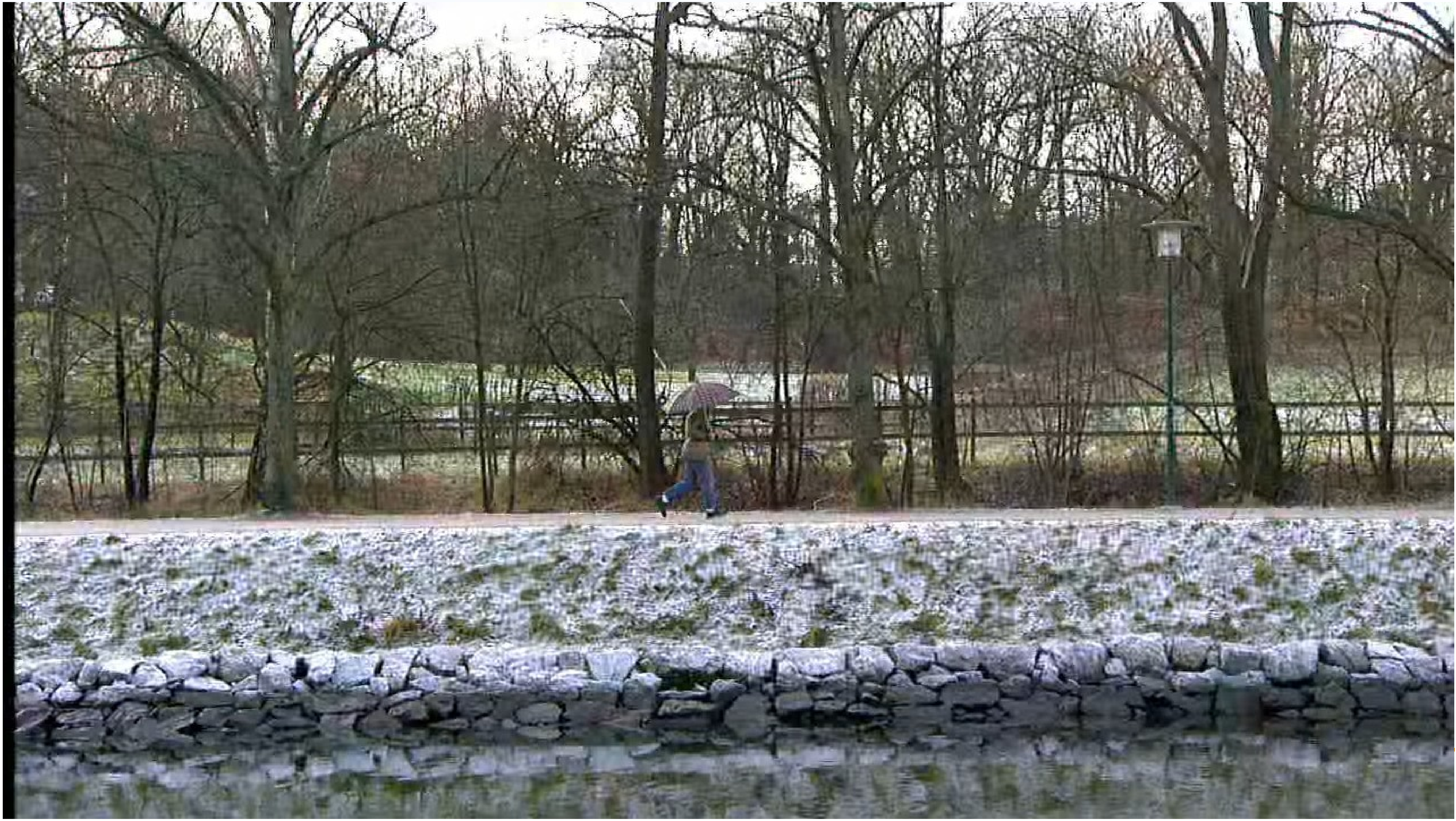
“Perceptual”: Activity Masking

- Goal: Use better resolution in flat areas
  - Most codecs require explicit QP signaling (MB)
  - PVQ allows implicit signaling based on gain (band)
- Use non-uniform quantization of the gain
- Change how *K* is computed from the gain

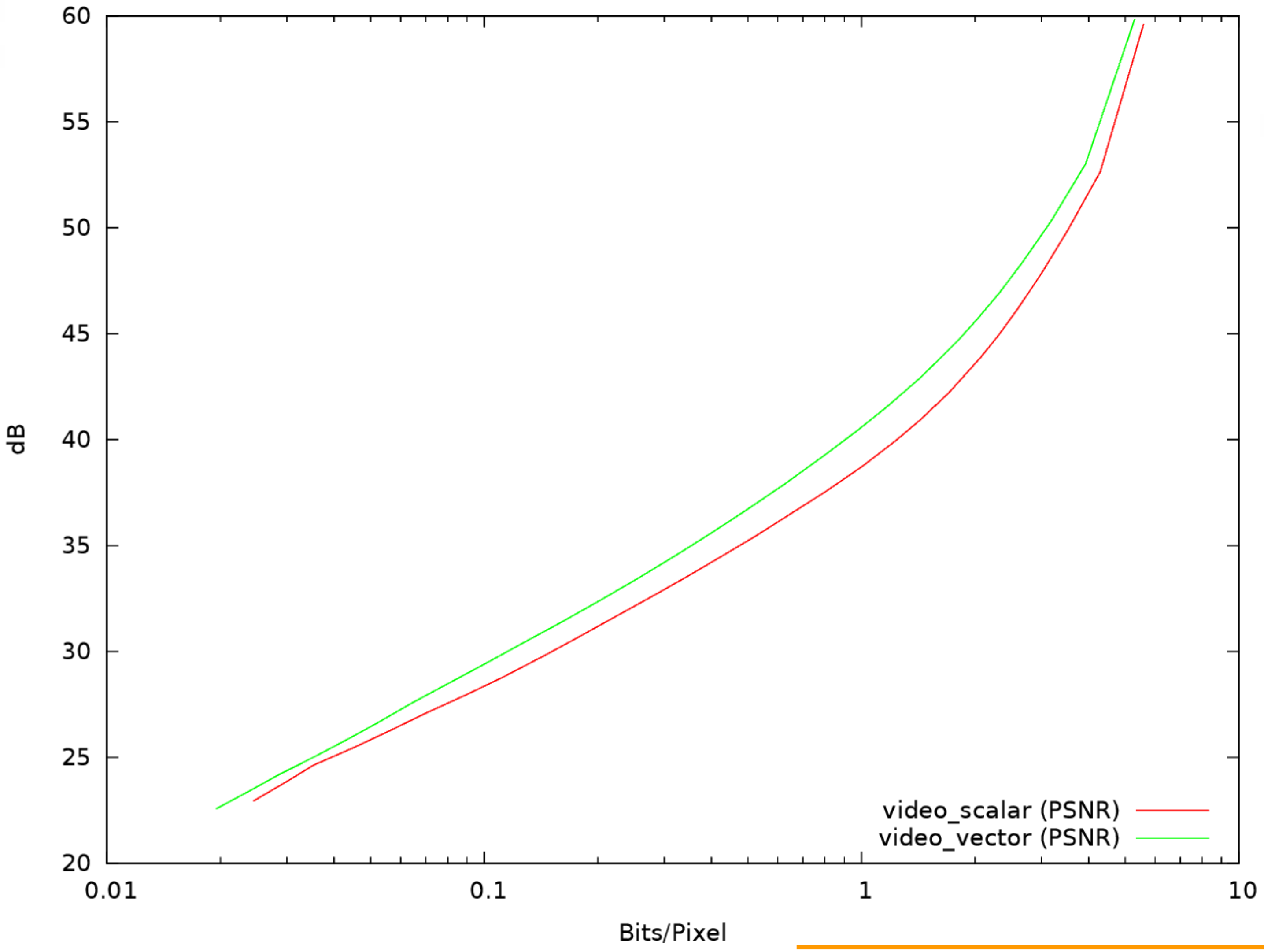
No Activity Masking (54 kB)



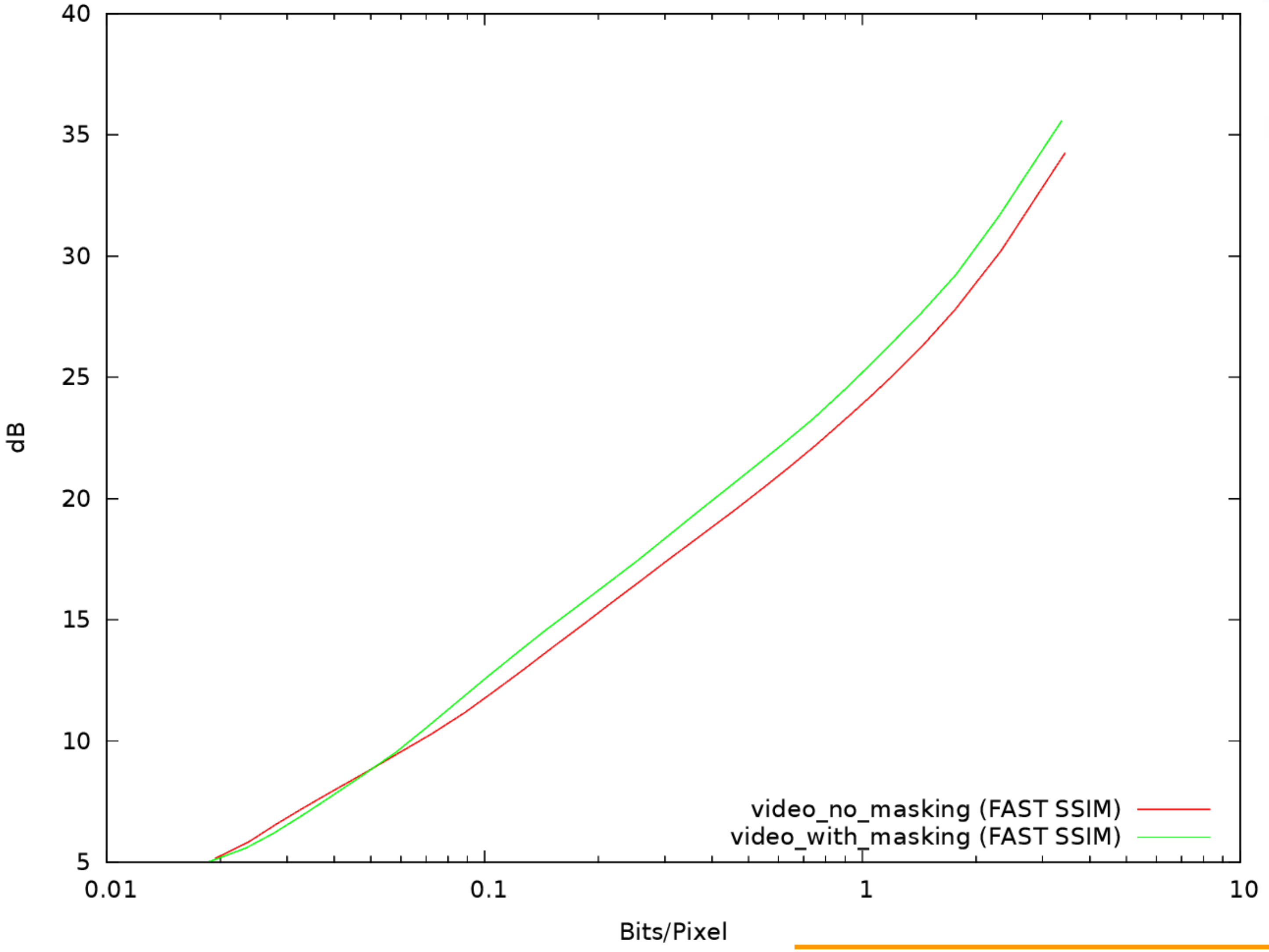
Activity Masking (54 kB)



Results (PVQ vs Scalar)



Results (Activity Masking)



Open Issues

- Better entropy coding
  - Take advantage of correlation in gain/θ/noref/etc.
  - Both spatially and across bands
- Better RDO
  - Some rate estimates very approximate
- Perceptual noise injection
- “Motion-blur” masking
- Bit-exact implementation, tuning, etc.

Resources

- Daala codec website: https://xiph.org/daala/
- PVQ Demo: https://people.xiph.org/~jm/daala/pvq_demo/
- Git repository: https://git.xiph.org/
- IRC: #daala channel on irc.freenode.net
- Mailing list: daala@xiph.org

Questions?