Predicting Chroma from Luma with Frequency Domain Intra Prediction

Nathan E. Egge and Jean-Marc Valin

Mozilla, Mountain View, USA

Xiph.Org Foundation

ABSTRACT

This paper describes a technique for performing intra prediction of the chroma planes based on the reconstructed luma plane in the frequency domain. This prediction exploits the fact that while RGB to YUV color conversion has the property that it decorrelates the color planes globally across an image, there is still some correlation locally at the block level.[1] Previous proposals compute a linear model of the spatial relationship between the luma plane (Y) and the two chroma planes (U and V).[2] In codecs that use lapped transforms this is not possible since transform support extends across the block boundaries[3] and thus neighboring blocks are unavailable during intra-prediction. We design a frequency domain intra predictor for chroma that exploits the same local correlation with lower complexity than the spatial predictor and which works with lapped transforms. We then describe a low-complexity algorithm that directly uses luma coefficients as a chroma predictor based on gain-shape quantization and band partitioning. An experiment is performed that compares these two techniques inside the experimental Daala video codec and shows the lower complexity algorithm to be a better chroma predictor.

Keywords: Intra Prediction, Lapped Transforms, Color Image Coding, Chroma Correlation, Regression, Gain-Shape Quantization, Perceptual Vector Quantization

1. INTRODUCTION

Still image and video codecs typically consider the problem of intra-prediction in the spatial domain. A predicted image is generated on a block-by-block basis using the previously reconstructed neighboring blocks for reference, and the residual is encoded using standard entropy coding techniques. Modern codecs use the boundary pixels of the neighboring blocks along with a directional mode to predict the pixel values across the target block (e.g., AVC, HEVC, VP8, VP9, etc.). These directional predictors are cheap to compute (often directly copying pixel values or applying a simple linear kernel), exploit local coherency (with low error near the neighbors) and predict hard to code features (extending sharp directional edges across the block). In Figure 1 the ten intra-prediction modes of WebM are shown for a given input block based on the 1 pixel boundary around that block.

In codecs that use lapped transforms these techniques are not applicable (e.g., VC-1, JPEG-XR, Daala, etc.). The challenge here is that the neighboring spatial image data is not available until *after* the target block has been decoded and the appropriate unlapping filter has been applied across the block boundaries. Figure 2 shows the decode pipeline of a codec using lapped transforms with a single block size. The support used in spatial intra prediction is exactly the region that has not had the unlapping post-filter applied. Note that the pre-filter has the effect of decorrelating the image along block boundaries so that the neighboring pixel values before unlapping are particularly unsuitable for use in prediction.

Work has been done to use intra prediction with lapped transforms. Modifying AVC, de Oliveira and Pesquet showed that it was possible to use the boundary pixels just outside the lapped region to use 4 of the 8 directional intra prediction modes with lapped transforms.[4] The work of Xu, Wu and Zhang considers prediction as a transform and proposes a frequency domain intra prediction method using non-overlapped blocks.[5] An early experiment with the Daala video codec extended this idea using machine learning to train sparse intra predictors.[6] However this technique is computationally expensive (4 multiplies per coefficient) and not easily vectorized.

Send correspondence to Nathan E. Egge <negge@xiph.org>.

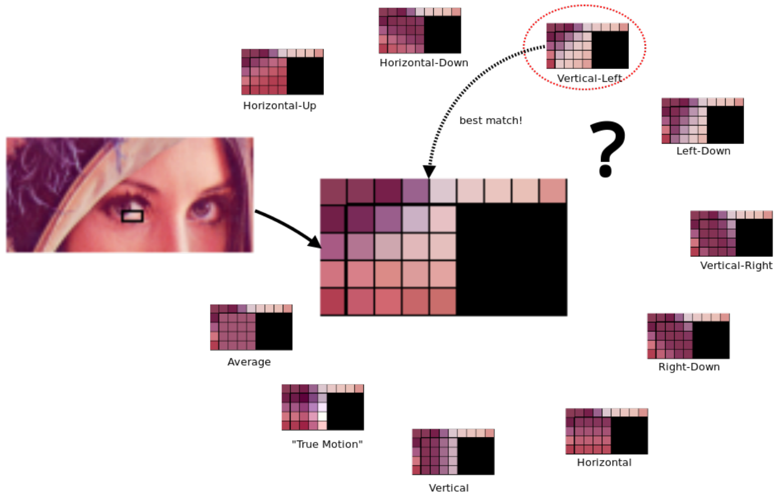


Figure 1. The 10 intra-prediction modes for 4x4 blocks in WebM (VP8).

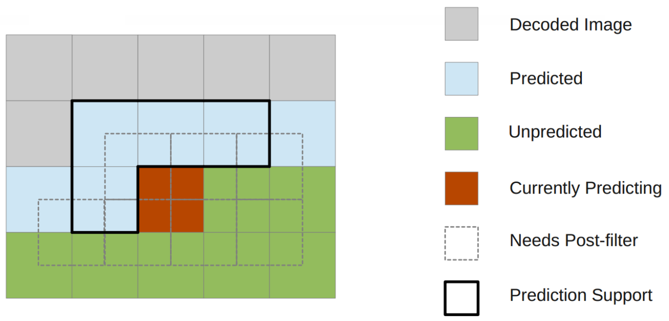


Figure 2. State of blocks in the decode pipeline of a codec using lapped transforms. Immediate neighbors of the target block (bold lines) cannot be used for spatial prediction as they still require post-filtering (dotted lines).

A promising technique was proposed by Lee and Cho to predict the chroma channels using the spatially coincident reconstructed luma channel.[1] This was formally proposed for use in HEVC by Chen et al[2] however was ultimately not selected due to the increased encoder and decoder running time of 20-30%. We propose a similar technique that adapts the spatial chroma-from-luma intra prediction for use with frequency-domain coefficients. We call this algorithm frequency-domain chroma-from-luma (FD-CfL).

More recently, work on the Daala video codec has included replacing scalar quantization with gain-shape quantization.[7] We show that when prediction is used with gain-shape quantization, it is possible to design a frequency-domain chroma-from-luma predictor without the added encoder and decoder overhead. An experimental evaluation between FD-CfL and the proposed PVQ-CfL algorithm shows this reduction in complexity comes with no penalty to quality and actually provides an improvement at higher rates.

2. CHROMA FROM LUMA PREDICTION

In spatial-domain chroma-from-luma, the key observation is that the local correlation between luminance and chrominance can be exploited using a linear prediction model. For the target block, the chroma values can be estimated from the reconstructed luma values as

$$C(u,v) = \alpha \cdot L(u,v) + \beta \tag{1}$$

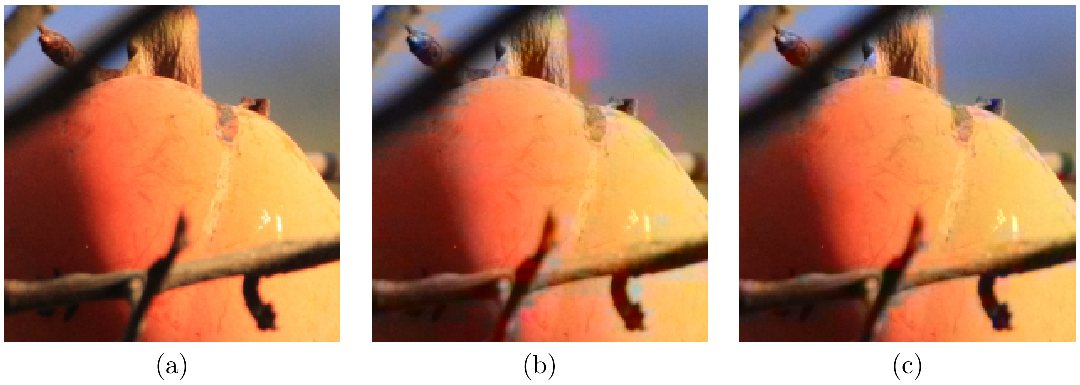


Figure 3. Comparison of (a) the original uncompressed image with composite images of (b) reconstructed luma and predicted chroma using experimental Daala intra modes and (c) reconstructed luma and predicted chroma using FD-CfL.

where the model parameters α and β are computed as a linear least-squares regression using N pairs of spatially coincident luma and chroma pixel values along the boundary:

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

When α and β are sent explicitly in the bitstream, the pairs (L_i, C_i) are taken from the original, unmodified image. However, the decoder can also compute the same linear regression using its previously decoded neighbors and thus α and β can be derived *implicitly* from the bitstream. Additional computation is necessary when the chroma plane is subsampled (e.g., 4:2:0 and 4:2:2 image data) as the luma pixel values are no longer coincident and must be resampled. In the next section we adapt the algorithm to the frequency-domain and show that this issue does not exist at most block sizes.

2.1 EXTENSION TO FREQUENCY-DOMAIN

In codecs that use lapped transforms, the reconstructed pixel data is not available. However the transform coefficients in the lapped frequency domain are the product of two linear transforms: the linear pre-filter followed by the linear forward DCT. Thus the same assumption of a linear correlation between luma and chroma coefficients holds. In addition, we can take advantage of the fact that we are in the frequency domain to use only a small subset of coefficients when computing our model.

The chroma values can then be estimated using frequency-domain chroma-from-luma (FD-CfL):

$$C_{DC} = \alpha_{DC} \cdot L_{DC} + \beta_{DC} \tag{3}$$
$$C_{AC}(u,v) = \alpha_{AC} \cdot L_{AC}(u,v) \tag{4}$$

where the α_{DC} and β_{DC} are computed using the linear regression in Equation 2 with the DC coefficients of the three neighboring blocks: up, left and up-left. When estimating $C_{AC}(u,v)$ we can omit the constant offset β_{AC} as we expect the AC coefficients to be zero mean. Additionally, we do not include all of the AC coefficients from the same three neighboring blocks when computing α_{AC}.

It is sufficient to use the three lowest AC coefficients from the neighboring blocks. This means that the number of input pairs N is constant regardless of the size of chroma block being predicted. Moreover, the input AC coefficients have semantic meaning: we use the strongest horizontal, vertical and diagonal components. This has the effect of preserving features across the block as edges are correlated between luma and chroma, see the fruit and branch edges in Figure 3 (c).

2.2 TIME-FREQUENCY RESOLUTION SWITCHING

When image data is 4:4:4 or 4:2:0, the chroma and luma blocks are aligned so that the lowest 3 AC coefficients describe the same frequency range. In codecs that support multiple block sizes (or that support 4:2:2 image data) it is the case that the luma blocks and the chroma blocks are not aligned. For example, in the Daala video codec the smallest block size supported is 4x4. In 4:2:0, when an 8x8 block of luma image data is split into four 4x4 blocks, the corresponding 4x4 chroma image data is still coded as a single 4x4 block.

This is a problem for FD-CfL as it requires the reconstructed luma frequency-domain coefficients to cover the same spatial extent. In Daala this is overcome by borrowing a technique from the Opus audio codec.[8] Using Time-Frequency resolution switching (TF) it is possible to trade off resolution in the spatial domain for resolution in the frequency domain. Here the four 4x4 luma blocks are *merged* into a single 8x8 block with half the spatial resolution and twice the frequency resolution. We apply a 2x2 Hadamard transform to corresponding transform coefficients in four 4x4 blocks to merge them into a single 8x8 block. The low frequency (LF) coefficients are then used with FD-CfL, see Figure 4.

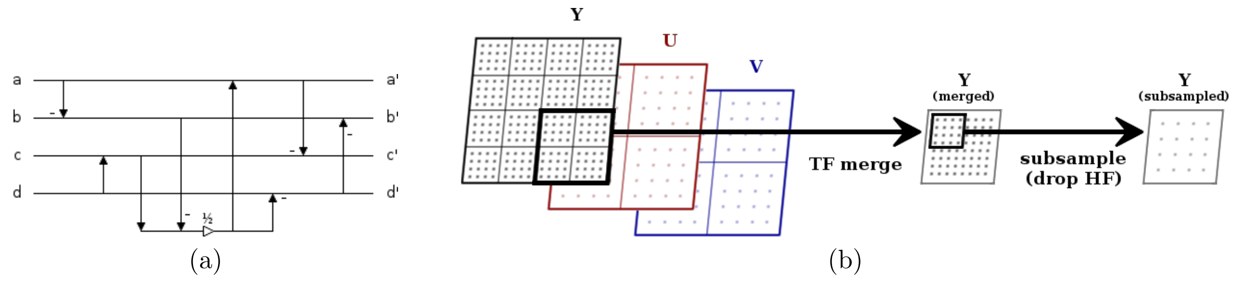


Figure 4. (a) Circuit diagram of the 2x2 TF-merge algorithm that uses 7 adds and 1 shift. (b) Using TF-merge to convert four 4x4 frequency domain luma blocks (Y) into a single 8x8 frequency domain block for use with TD-CfL when image data is 4:2:0. Note that only the lower frequency (LF) portion is actually necessary and thus only that portion of the TF-merge need be computed.

2.3 COMPLEXITY COMPARISON

Both the spatial and frequency domain chroma-from-luma techniques have the property that once the model parameters α and β have been determined the entire chroma block can be computed with one multiply and one add per pixel. This is far better than the frequency domain intra prediction used to predict the luma plane in Daala which uses 4 multiplies and 4 adds.[6] An important question to answer is how does the computational complexity of the model fitting step in FD-CfL compare to the HEVC proposal.[2] Note that the HEVC proposal required resampling the entire luma block for 4:2:0 image data, a process which is roughly as expensive as a TF-merge and is necessary for all luma block sizes (not just at the smallest block size).

Block Size	Spatial Domain		Freq. Domain	
	Mults	Adds	Mults	Adds
$N \times N$	$4 * N + 2$	$8 * N + 3$	$2 * 12 + 5$	$4 * 12 + 5$
4×4	18	35	29	53
8×8	34	67	29	53
16×16	66	131	29	53

Table 1. Comparison of cost to fit the model in Equation 2 using spatial and frequency domain chroma-from-luma.

To answer this question, Table 1 is provided which compares just the cost of fitting the chroma-from-luma model. An interesting feature of the FD-CfL algorithm is that the cost to fit the model is independent of the block size, as we only consider a small portion of the frequency domain coefficients. In the frequency domain we use 12 (L_i, C_i) pairs (the 4 LF coefficients from the 3 neighboring blocks), versus $2 * N$ pairs in the spatial domain (N from each of the bordering left and up blocks). This means that for all chroma block sizes larger than 4x4, model fitting in the frequency domain is actually cheaper than in the spatial domain.

3. GAIN-SHAPE QUANTIZATION

Most modern video and still image codecs use scalar quantization as a “lossy” way of reducing the amount of information needed to code a block. After the block coefficients have been transformed into the frequency-domain, they are each quantized to an integer index which is entropy coded. In the decoder the transform coefficients are reconstructed by reversing the quantization. For a coefficient C_i and quantization step size Q_i, the quantization index γ_i and reconstructed coefficient $\hat{C}_i$ can be found by

$$\gamma_i = \lfloor C_i / Q_i \rfloor \tag{5}$$

$$\hat{C}_i = \gamma_i \cdot Q_i \tag{6}$$

Note that when the quantization step size is large (e.g., at low rates) the smaller, high frequency coefficients go to zero. While good for compression (many codecs will run-length encode a string of zeros), this has the effect of low-passing the block and removing much of the texture from the image.

An alternative to scalar quantization is to use vector quantization (VQ). Here the quantization index γ no longer represents a single coefficient, but rather an entire vector of coefficients. The idea is to take, for example, the entire block of coefficients and treat them as an n-dimensional vector. Quantization then amounts to finding the index γ of the nearest codeword (n-dimensional vector) in a possibly infinite VQ-codebook. The density of codewords in the codebook around the n-dimensional vector we are quantizing dictates the quantization error. However, it has been shown that even for a fixed set of input vectors, designing an optimal VQ-codebook is an NP-hard problem. Moreover, searching for the optimal quantization index γ requires computing the distance between the input vector and every VQ-codeword to select the closest.

In the Daala video codec we use gain-shape quantization.[9] A vector of coefficients $\mathbf{x}$ is separated into two intuitive components: its magnitude (*gain*) and its direction (*shape*). The gain $g = \|\mathbf{x}\|$ represents how much energy is contained in the block, and the shape $\mathbf{u} = \mathbf{x} / \|\mathbf{x}\|$ indicates where that energy is distributed among the coefficients. The gain is then quantized using scalar quantization, while the shape is quantized by finding the nearest VQ-codeword in an algebraically defined codebook. This has the advantage of not needing to explicitly store the VQ-codebook in the decoder as well as allowing the encoder to search only a small set of VQ-codewords around the input vector. Given the gain quantization index γ_g, the shape vector quantization index γ_u and an implicitly defined VQ-codebook CB, the reconstructed gain $\hat{g}$ and shape $\hat{\mathbf{u}}$ can be found by

$$\hat{g} = \gamma_g \cdot Q \tag{7}$$

$$\hat{\mathbf{u}} = CB[\gamma_u] \tag{8}$$

and reconstructed coefficients $\hat{\mathbf{x}}$ are thus

$$\hat{\mathbf{x}} = \hat{g} \cdot \hat{\mathbf{u}} \tag{9}$$

By explicitly signaling the amount of energy in a block, and roughly where that energy is located, gain-shape quantization is texture preserving. Because the algebraic codebook used in Daala is based on the pyramid vector quantizer described by Fisher,[10] this technique is referred to as Perceptual Vector Quantization (PVQ). A complete description of PVQ usage in Daala and its other advantages over scalar quantization is outside the scope of this paper and and is described in detail by Valin.[7]

3.1 PREDICTION WITH PVQ

In block based codecs, both intra- and inter-prediction can often construct a very good predictor for the block that will be decoded next. In the encoder, this predicted block is typically subtracted from the input image and the residual is transformed to the frequency domain, quantized and entropy coded. When the transform is linear, as is the case with codecs based on lapped transforms, this is equivalent to transforming the predictor and computing the difference in the frequency domain. However, if one were to simply quantize the frequency domain residual using PVQ, the texture preservation property described in the previous section would be lost. This is because the energy that would be preserved is no longer that of the block being coded, but instead the

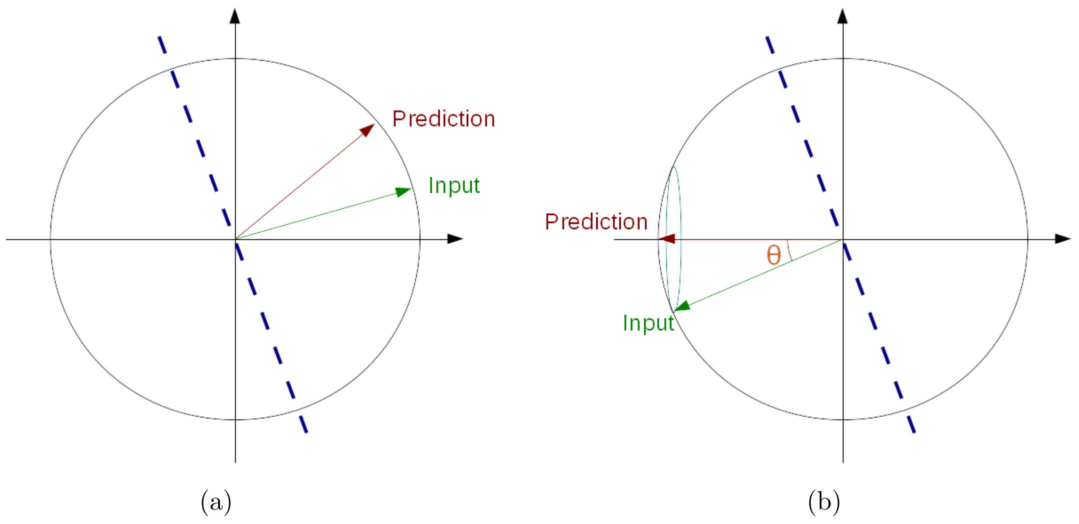


Figure 5. (a) A Householder reflection plane is computed that aligns the prediction vector so that its largest component is along an axis. (b) The input vector is reflected across the plane and the angle θ is computed and coded using scalar quantization. The axis on which the predictor lies is eliminated, the remaining $n-1$ dimensions are coded using PVQ.

gain represents how much the image is different from its predictor. In Daala, this is avoided by explicitly *not* computing a residual, but instead extracting another intuitive parameter in gain-shape quantization.

Ideally, when the predictor is good we would like the cost of coding the gain and shape to be low. That is, we would like the entropy of the symbols we code to be as small as possible. We can achieve this and retain the energy preserving properties of PVQ by using a Householder reflection. Considering the predictor as another n-dimensional vector, a reflection plane is computed that aligns the predictor with one of the axes in our n-dimensional vector space making all but one of the components in the predictor equal zero. The encoder can then reflect the input vector $\mathbf{x}$ across this reflection plane in a way that is perfectly reproducible in the decoder, see Figure 5.

Let $\mathbf{r}$ be the n-dimensional vector of predictor coefficients. Then the normal to the reflection plane can be computed as

$$\mathbf{v} = \frac{\mathbf{r}}{\|\mathbf{r}\|} + s \cdot \mathbf{e}_m \tag{10}$$

where $s \cdot \mathbf{e}_m$ is the signed unit vector in the direction of the axis we would like to reflect $\mathbf{r}$ onto. The input vector $\mathbf{x}$ can then be reflected across this plane by computing

$$\mathbf{z} = \mathbf{x} - 2\frac{\mathbf{v}^T\mathbf{x}}{\mathbf{v}^T\mathbf{v}}\mathbf{v} \tag{11}$$

We can measure how well the predictor $\mathbf{r}$ matches our input vector $\mathbf{x}$ by computing the cosine of the angle θ between them as

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{r}}{\|\mathbf{x}\|\,\|\mathbf{r}\|} = \frac{\mathbf{z}^T\mathbf{r}}{\|\mathbf{z}\|\,\|\mathbf{r}\|} = -s\frac{z_m}{\|\mathbf{z}\|} \tag{12}$$

We are free to choose any axis in our n-dimensional space and we select $\mathbf{e}_m$ to be the dimension of the largest component of our prediction vector $\mathbf{r}$ and $s = \operatorname{sgn}(r_m)$. Thus the largest component lies on the m-axis after reflection. When the predictor is good, we expect that the largest component of $\mathbf{z}$ will also be in the $\mathbf{e}_m$ direction and θ will be small. If we code $\hat{\theta}$ using scalar quantization, we can remove the largest dimension of $\mathbf{z}$ and reduce the coding of $\mathbf{x}$ to a gain-shape quantization of the remaining $n-1$ coefficients where the gain has been reduced to $\sin\theta \cdot g$. Given a predictor $\mathbf{r}$, the reconstructed coefficients $\hat{\mathbf{x}}$ are computed as

$$\hat{\mathbf{x}} = \hat{g}\big(-s \cdot \cos\hat{\theta} \cdot \mathbf{e}_m + \sin\hat{\theta} \cdot \hat{\mathbf{u}}\big) \tag{13}$$

When the predictor is poor, θ will be large and the reflection is unlikely to make things easier to code. Thus when θ is greater than 90° we code a flag and use PVQ with no predictor. Conversely when $\mathbf{r}$ is exact, $\hat{\theta}$ is zero and no additional information needs to be coded. In addition, because we expect $\mathbf{r}$ to have roughly the same amount of energy as $\mathbf{x}$, we can get additional compression performance by using $\|\mathbf{r}\|$ as a predictor for g:

$$\hat{g} = \gamma_g \cdot Q + \|\mathbf{r}\| \tag{14}$$

3.2 CHROMA FROM LUMA USING PVQ PREDICTION

Let us now return to the frequency-domain chroma-from-luma (FD-CfL) algorithm from Section 2.1 and consider what happens when it is used with gain-shape quantization. As an example, consider a 4x4 chroma block where the 15 AC coefficients are coded using gain-shape quantization with the FD-CfL predictor from Equation 4. The 15-dimensional predictor $\mathbf{r}$ is simply a linearly scaled vector of the coincident reconstructed luma coefficients:

$$C_{AC}(u,v) = \alpha_{AC} \cdot L_{AC}(u,v) \implies \mathbf{r} = \alpha_{AC} \cdot \hat{\mathbf{x}}_L \tag{15}$$

Thus the shape of the chroma predictor $\mathbf{r}$ is exactly that of the reconstructed luma coefficients $\hat{\mathbf{x}}_L$ with one exception:

$$\frac{\mathbf{r}}{\|\mathbf{r}\|} = \frac{\alpha_{AC} \cdot \hat{\mathbf{x}}_L}{\|\alpha_{AC} \cdot \hat{\mathbf{x}}_L\|} = \operatorname{sgn}(\alpha_{AC})\frac{\hat{\mathbf{x}}_L}{\|\hat{\mathbf{x}}_L\|} \tag{16}$$

Because the chroma coefficients are sometimes inversely correlated with the coincident luma coefficients, the linear term α_{AC} can be negative. In these instances the *shape* of $\hat{\mathbf{x}}_L$ points in exactly the wrong direction and must be flipped.

Moreover, consider what happens to the gain of $\mathbf{x}_C$ when it is predicted from $\mathbf{r}$. The PVQ prediction technique assumes that $\|\mathbf{r}\| = \alpha_{AC} \cdot \|\hat{\mathbf{x}}_L\|$ is a good predictor of $g_C = \|\mathbf{x}_C\|$. Because α_{AC} for a block is learned from its previously decoded neighbors, often it is based on highly quantized or even zeroed coefficients. When this happens, $\alpha_{AC} \cdot \|\hat{\mathbf{x}}_L\|$ is no longer a good predictor of g_C and the cost to code $\|\mathbf{x}_C\| - \alpha_{AC} \cdot \|\hat{\mathbf{x}}_L\|$ using scalar quantization is actually greater than the cost of just coding g_C alone.

Thus we present a modified version of PVQ prediction that is used just for chroma-from-luma intra prediction called PVQ-CfL. For each set of chroma coefficients coded by PVQ, the prediction vector $\mathbf{r}$ is exactly the coincident luma coefficients. Note that for 4:2:0 video we still need to apply the Time-Frequency resolution switching (TF) described in Section 2.2 to merge the reconstructed coefficients of 4x4 luma blocks to get the coincident predictor $\hat{\mathbf{x}}_L$ for the corresponding 4x4 chroma block. We determine if we need to flip the predictor by computing the sign of the cosine of the angle between $\hat{\mathbf{x}}_L$ and $\mathbf{x}_C$:

$$f = \operatorname{sgn}(\hat{\mathbf{x}}_L^T \mathbf{x}_C) \tag{17}$$

A negative sign means the angle between the two is greater than 90° and flipping $\hat{\mathbf{x}}_L$ is guaranteed to make the angle less than 90°.

We then code f using a single bit[‡], and the gain $\hat{g}_C$ using scalar quantization with no predictor. The shape quantization algorithm for $\mathbf{x}_C$ is unchanged except that $\mathbf{r} = f \cdot \hat{\mathbf{x}}_L$. This algorithm has the advantage over FD-CfL of being both lower complexity (neither the encoder nor decoder need to compute a linear regression per block) and providing better compression (the chroma gain g_C is never incorrectly predicted).

3.3 PVQ WITH FREQUENCY BANDS

Up to this point we have only examined the case when all of the AC coefficients for an $N \times N$ block are considered together as a single input vector for PVQ prediction. In practice, it may be better to consider portions of the AC coefficients together so partitions of the block where $\hat{g} = 0$ or $\hat{\theta} = 0$ are coded more efficiently. Consider the frequency band structure currently used by Daala in Figure 6. The PVQ-CfL technique in the previous section is trivially modified to work with any arbitrary partitioning of block coefficients into bands.

[‡]It is not strictly necessary to code a bit for f. Instead the parameter α_{AC} could be found using least-squares regression and the sign extracted. However, using a single bit to code f is (1) better overall than relying on least-squares regression which can be wrong and (2) reduces the complexity significantly.

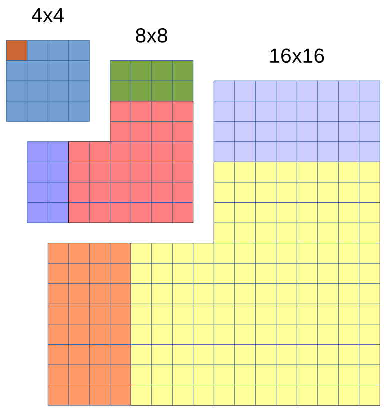


Figure 6. The band structure of 4x4, 8x8 and 16x16 blocks in Daala.

Instead of considering whether to flip the direction of $\mathbf{x}_L$ for each band partition individually (a signaling cost of 7 bits per 16x16 block), simply look at the lowest 4x4 AC partition and use the flip decision there for the entire block. The assumption is that having those larger low frequency coefficients predicted well is far more important than getting it exactly right at higher frequencies. When the quantization step size is large, the high frequency coefficients will be sent to zero regardless.

4. EXPERIMENTAL EVALUATION

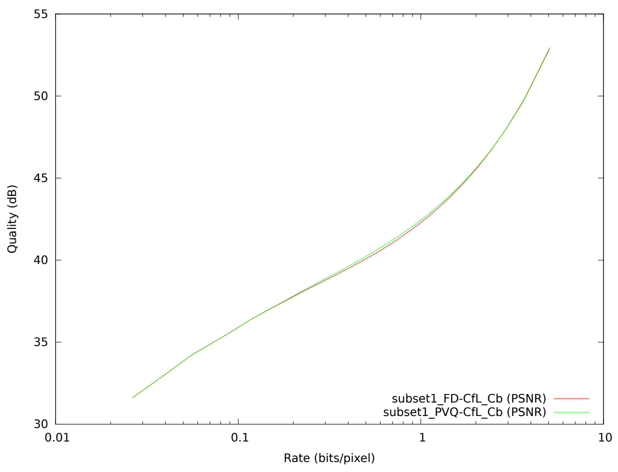
The two frequency-domain intra-prediction techniques described in this paper for predicting chroma coefficients from reconstructed luma coefficients were evaluated within the framework of the experimental Daala video codec. To make the comparison fair, only the AC chroma coefficients were predicted using chroma-from-luma with the DC chroma coefficients being predicted as the average of its neighbors. Only gain-shape quantization was used to code the transform coefficients, with implicitly predicted FD-CfL coefficients being passed as the predictor to PVQ in one instance, and the PVQ-CfL algorithm being used in the other. All other video coding techniques used were identical.

A sample of 50 still images taken from Wikipedia and downsampled to 1 megapixel were compressed at varying quantization levels, and the resulting rate-distortion curves were computed on both the Cb and Cr chroma planes using four different metrics. The Bjontegaard distance[11] was computed to measure the average improvement in both Rate (at equivalent quality) and SNR (at equivalent size) between the two techniques. The result of this experiment is shown in Table 2.

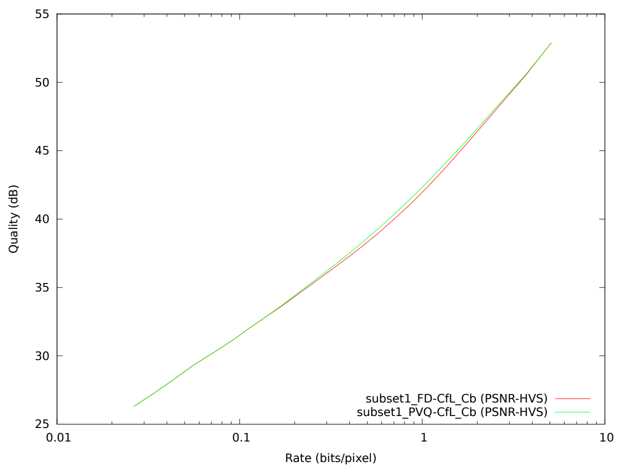
Metric	Cb (plane 1)		Cr (plane 2)	
	Δ Rate (%)	Δ SNR (dB)	Δ Rate (%)	Δ SNR (dB)
PSNR	-1.27280	0.05171	-0.57558	0.02941
PSNR-HVS	-2.67808	0.13703	-1.24695	0.07459
SSIM	-2.51125	0.07116	-1.66549	0.05779
FastSSIM	-3.18398	0.06969	-2.79776	0.06737

Table 2. Computation of the Bjontegaard distance (improvement) between the two rate-distortion curves, moving from FD-CfL to PVQ-CfL, based on four quality metrics.

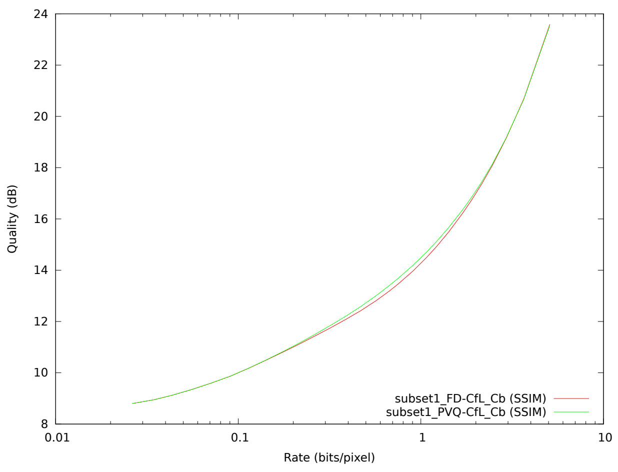
For every quality metric we considered, the PVQ-CfL technique did a better job of predicting Cb and Cr chroma coefficients both delivering higher quality at the same rate (up to 0.13 dB improvement for PSNR-HVS[12]) and better compression for equivalent distortion (3.2% smaller files for FastSSIM[13]). Looking at the actual rate-distortion curves in Figure 7, we see that the largest improvements are found at higher rates. However, the reduced complexity of PVQ-CfL over FD-CfL with no rate or quality penalty at lower rates means this technique is clearly superior and thus has been adopted in Daala.



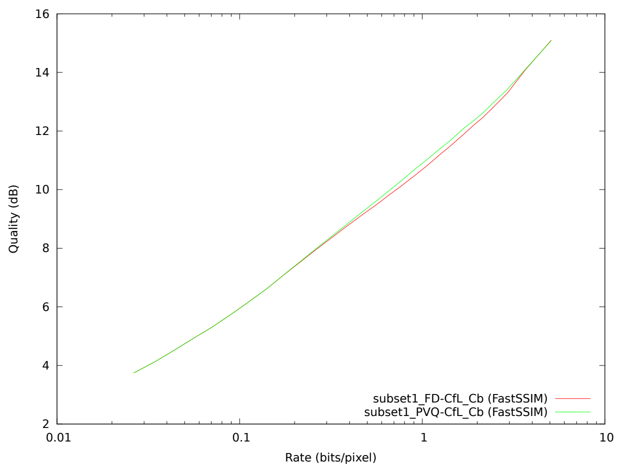
(a)



(b)



(c)



(d)

Figure 7. A comparison of the rate-distortion curves for FD-CfL and PVQ-CfL on the same set of 50 still images using quality metrics (a) PSNR, (b) PSNR-HVS, (c) SSIM and (d) FastSSIM.

5. CONCLUSIONS

We have presented two new techniques for doing chroma-from-luma intra-prediction in the frequency domain for use with video and still image codecs that employ lapped transforms. The first technique (FD-CfL) is suitable for use in codecs based on scalar quantization and provides a reduction in complexity when compared to spatial domain CfL as per block complexity stays constant with block size increases. The second technique (PVQ-CfL) extends the PVQ prediction technique in codecs using gain-shape quantization to allow for better chroma prediction with no additional per block complexity from model fitting.

This work is part of the Daala project.[14] The full source code, including both of the algorithms described in this paper is available in the project git repository.[15]

REFERENCES

[1] Lee, S.-H. and Cho, N.-I., "Intra prediction method based on the linear relationship between the channels for yuv 4:2:0 intra coding," in [*Image Processing (ICIP), 2009 16th IEEE International Conference on*], 1037–1040 (Nov 2009).

[2] Kim, J., Park, S., Choi, Y., Jeon, Y., and Jeon, B., "New intra chroma prediction using inter-channel correlation," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG (July 2010).

[3] Tran, T., Liang, J., and Tu, C., "Lapped transform via time-domain pre- and post-filtering," *Signal Processing, IEEE Transactions on* **51**, 1557–1571 (June 2003).

[4] de Oliveira, R. and Pesquet-Popescu, B., "Intra-frame prediction with lapped transforms for image coding," in [*Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*], 805–808 (May 2011).

[5] Xu, J., Wu, F., and Zhang, W., "Intra-predictive transforms for block-based image coding," *Signal Processing, IEEE Transactions on* **57**, 3030–3040 (Aug 2009).

[6] "Introducing daala part 2: Frequency domain intra prediction." http://people.xiph.org/~xiphmont/demo/daala/demo2.shtml.

[7] Valin, J.-M. and Terriberry, T. B., "Perceptual vector quantization for video coding," *SPIE Proceedings* **9410** (2015).

[8] Valin, J.-M., Maxwell, G., Terriberry, T. B., and Vos, K., "High-quality, low-delay music coding in the opus codec," in [*Audio Engineering Society Convention 135*], (Oct 2013).

[9] "Daala, part 6: Perceptual vector quantization." https://people.xiph.org/~jm/daala/pvq_demo/.

[10] Fischer, T. R., "A pyramid vector quantizer," *IEEE Trans. on Information Theory* **32**, 568–583 (1986).

[11] Bjontegaard, G., "Calculation of average psnr differences between rd-curves," Tech. Rep. VCEG-M33, ITU-T SG16/Q6, Austin, TX, USA (Apr 2001).

[12] Ponomarenko, N., Silvestri, F., Egiazarian, K., Carli, M., Astola, J., and Lukin, V., "On between-coefficient contrast masking of dct basis functions," in [*Proceedings of the Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics*], (January 2007).

[13] Chen, M.-J. and Bovik, A. C., "Fast structural similarity index algorithm," in [*Proc. ICASSP*], 994–997 (march 2010).

[14] "Daala website." https://xiph.org/daala/.

[15] "Daala git repository." https://git.xiph.org/daala.git.