Chroma from Luma (CfL) in AV1
[*Work in progress*]

PVQ CfL was replaced with normal CfL. As such, this document is superseded by

https://nbviewer.jupyter.org/github/luctrudeau/VideoExperiments/blob/master/cfl/gain/Chroma%20from%20Luma%20(CfL).ipynb

Background

The recent integration of PVQ into the AV1 video codec now allows for other technologies from the Daala video codec, like Chroma from Luma (CfL), to be integrated into AV1. In Daala, PVQ, TF and CfL are combined into PVQ-CfL, where the prediction vector used by PVQ for Chroma coefficients is derived from the reconstructed transformed Luma coefficients.

Introduction

In this living[1] document, we outline the work performed so far regarding the integration of CfL into AV1. The scope of this document is not to explain either PVQ, CfL or AV1. The aim is to document the work accomplished and the findings resulting from the implementation. A special attention is also given to alterations required to CfL, PVQ and AV1 in order to achieve the integration.

To quickly access an outline of the current progress of this project the Tasks section contains a bullet list, where the completed tasks are ~~strikethroughed~~. In the General Approach section, we describe general concepts of how CfL can be performed in AV1. There is more than one way to implement CfL in AV1, as such, the Alternate Implementation section describes concepts related to a specific variant. The Submitted to Gerrit section lists the current work on CfL that is under review. The Merged into AV1 section details the work that has been included upstream into AV1 behind an experimental flag. The Progress section describes the results and findings of the work done so far. This information is also used to enhance the previous sections.

[1] Constructive comments are welcomed.

[*Work in progress, comments are welcomed*]

Tasks

- Prototype 1:
  - ~~Disable ADST~~
    - ~~Configure flag DCT_ONLY to enable DCT only AV1~~
    - ~~Add assert if bitstream writes out anything other than DCT_DCT~~
  - ~~Disable Chroma intra prediction signaling~~
    - ~~Remove Chroma intra from RDO~~
    - ~~Hardcode Chroma intra to DC_PRED~~
- Prototype 2:
  - Add CFL_PRED as a Chroma Intra mode
  - Perform DC_PRED when CFL_PRED is selected
  - Generate probabilities for entropy coding of the CFL_PRED mode
- Prototype 3:
  - ~~Store reconstructed pixel values (to avoid TF)~~
  - ~~Forward DCT the CfL prediction (same as PVQ)~~
  - ~~PVQ Refactorings to facilitate CFL~~
  - ~~Refactor is_keyframe code~~
  - ~~Disable UV_mode signaling~~
  - ~~RDO support~~
  - ~~Compute DC from DC_PRED (no DCT)~~
  - ~~Merge CFL_CTX and cfl_ctx~~
  - ~~Add flip bit support~~
  - Remove gain prediction
  - ~~Average 2x2 luma pixel blocks in 4:2:0 Chroma prediction~~
    - Investigate better filters
  - Support 4:4:4
  - Support intra in inter
- General CfL:
  - ~~Enable keyframes~~
    - ~~Add Flip bit to CfL~~
    - ~~Remove gain prediction~~
    - ~~Validate skipping behavior when Keyframe code is enabled~~
  - ~~Add CfL (Barebone)~~
    - ~~Store Luma dequantized coeffs~~
    - ~~Load dequantized coeffs as Chroma AC prediction~~
    - ~~Apply TF Merge~~
    - ~~Manage skips~~
  - ~~Perform CfL during RDO~~
    - ~~Successive TF Merges~~
    - ~~Store best luma coeffs~~
  - ~~ADST~~

        - ~~Experimentation to TF-merge ADST transformed blocks.~~
        - ~~Experiment with transforms allowing to apply TF-merge on ADST transformed blocks.~~
        - ~~Inverse transform ADST and forward transform to DCT for storage~~
    - Intra in Inter
        - Adapt keyframe code to inter frame (currently Intra is always assumed)
        - Add CfL to Intra mode decision in Inter blocks
    - 4:4:4
        - Adjust offsets used to compensate for Chroma subsampling
    - ~~Fast Transform for DC_PRED~~
        - ~~DCT_DCT~~

Submitted to Gerrit

CfL

8603: [PVQ_CFL] Chroma prediction is average of Luma values in 4:2:0
8571: [PVQ_CFL] Removed Gain Prediction
8566: [PVQ_CFL] Flip bit
8511: [PVQ_CFL] Replace CFL DC by DC_PRED DC
8564: [PVQ_CFL] Merge CFL_CTX with cfl_ctx
8541: [PVQ_CFL] CfL during RDO
7943: [PVQ_CFL] Disable uv_mode signaling
8442: PVQ_CFL Experiment flag
8510: Chroma from Luma (CfL) for Intra in PVQ

PVQ Refactorings

8372: Replace is_keyframe with is_skip_copy in PVQ encoder
8289: Replace is_keyframe with is_skip_copy in PVQ decoder
8502: [PVQ] Don't transform if block skipped
8469: [PVQ] Stop passing NULL reference for PVQ_INFO

Merged into AV1

PVQ Refactorings

8514 (Merged as b6e94d9): Uniform way of accessing mbmi in av1_xform_quant
8130 (Merged as 9e6a960): Merging robust and nodesync in PVQ
8128 (Merged as 9a834c0): Remove frame type dependency in od_pvq_rate
8129 (Merged as 9638228): Remove H/V considerations in pvq_theta
8029 (Merged as 005feb6): Add get_plane_type() helper function.
7166 (Merged as e1c0929): Convert PVQ skip variable to enum
6911 (Merged as 472f63f): Replace Skip with AC/DC coded in PVQ

General Approach

DC Prediction

CfL only applies to AC coefficients, another approach must be used to predict the DC. We propose to use the DC_PRED[2] intra prediction mode. This approach offers numerous advantages:

- Low complexity
- Already in AV1
    - Hardware acceleration available (see the aom_dc_predictor_NxN() functions)
- DC does not require DCT (see Fast DCT for DC_PRED)
- Widely known mechanism (also In VP8 and VP9)

Fast DCT for DC_PRED

The DC_PRED intra prediction mode produces fills the predicted block with a single value. Let the get_tx_scale function return 1 if the transform size (tx_size) equals 32x32 and 0 otherwise. For 32x32 transforms, the DC is obtained by multiplying the single value by 128. For transform sizes smaller than 32x32, the DC is obtained by multiplying the single value by 8 * the block size (tx_blk_size). As demonstrated in the following code snippet:

```
DC = pred[0] * ((get_tx_scale(tx_size)) ? 128 : tx_blk_size * 8);
```

AC Prediction

The reconstructed Luma pixels are used as the pixel domain Chroma Intra prediction. This approach offers numerous advantages:

- Low complexity
    - Does not require TF merging
    - Does not require compensating for the varying transform scaling
- Compatible with all transform type combinations found in AV1
- Better quality prediction
    - No loss incurred by TF merging

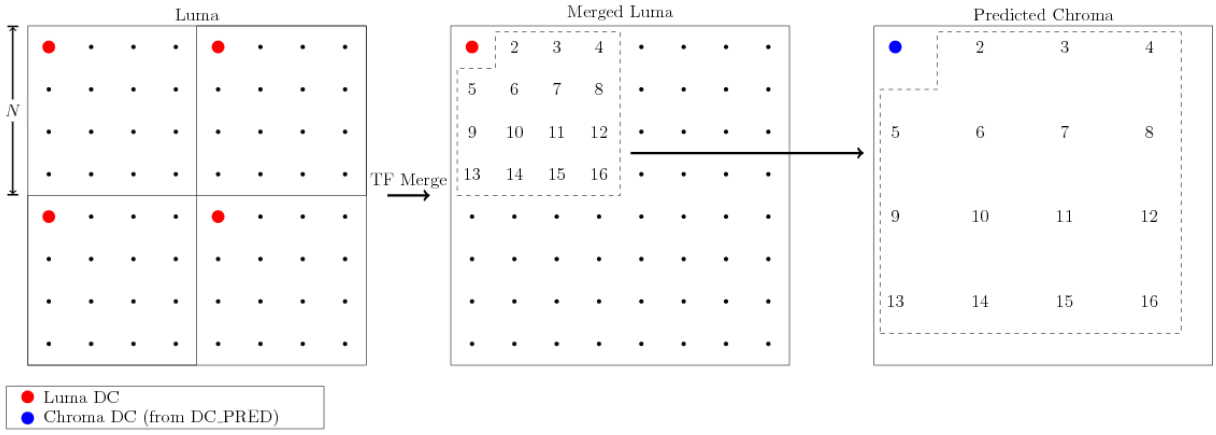
When 4:2:0 Chroma subsampling is used, the Chroma pixel is the average of the 4 reconstructed Luma pixels corresponding to that Chroma pixel.

[2] **DC_PRED (DC prediction)**. Fills the block with a single value using the average of the pixels in the row above *A* and the column to the left of *L*
http://blog.webmproject.org/2010/07/inside-webm-technology-vp8-intra-and.html

As for the AC coefficients, they are dequantized Luma coefficients. However, the Luma transform size and Chroma transform size can differ. In table 2, we indicate how the Chroma prediction is built based on the offset between the Luma transform size and the Chroma transform size.

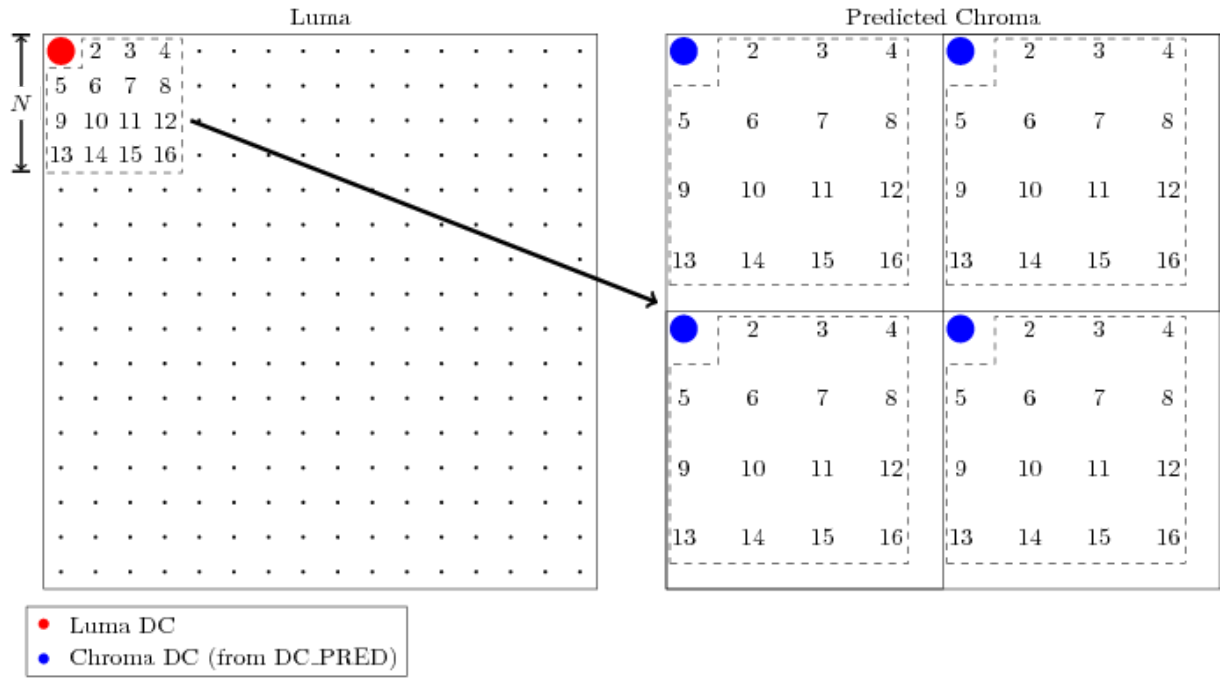
Table 2: Chroma Prediction based on transform size offset

Case 1: The Luma transform size equals Chroma transform size
Since the 4:2:0 Chroma subsampling is used, the transform sizes are equal but the covered region in the frame is not. The NxN Chroma transform coefficients cover a 2Nx2N region in the Luma plane. As such, the quartet of NxN Luma blocks corresponding the 2Nx2N region are TF merged together resulting in a 2Nx2N prediction, of which the upper left NxN coefficients are used.



Case 2: The Luma transform size is greater than the Chroma transform size
Multiple small NxN Chroma transforms cover the same region as the luma transform size. Every Chroma block in this region are predicted using the upper left NxN luma coefficients.

[*Work in progress, comments are welcomed*]



Case 3: The Luma transform size is less than the Chroma transform size
The Luma transform size is NxN but the Chroma transform size is CxC, where C = M x N. Perform a TF merge on every non-overlapping 2Nx2N quartet of NxN luma blocks. As a results, N := 2N. Continue TF merging quartets of NxN Luma blocks until N == C. When N == C, perform case 1.

If PVQ determines that the AC coefficients are skipped, the AC coefficients of the Luma Intra prediction are stored instead. To allow the decoder to compensate for the eventuality of a late skip, the decoder stores the AC coefficients of the Luma Intra prediction even when the block is skipped. When noref is used, the dequantized AC coefficients are zeroed out.

When RDO is performed on the encoder side, multiple Luma modes and transform sizes are tested followed by Chroma modes and transform sizes. An extra PVQ encode is required between the Luma and Chroma plane RDO. This extra PVQ encode allows CfL to store the dequantized luma coefficients with the best encoding parameters.

Assumptions

- Luma intra modes may not be reused for Chroma Intra prediction.
- AV1's experimental rectangular transforms are disabled.

Uncertainties

- Skipping
    - Special skipping for CfL that also manages gain signaling (see with jmspeex)

Alternate Implementations

Prototype 1 (CfL Always on):

The Intra prediction mode DC_PRED is always used to predict the DC coefficients for Chroma Intra blocks. Therefore, the Chroma Intra prediction mode is no longer signaled in the bitstream.

Status

As shown here, TF is not compatible with the ADST. As such, when a transform type other than DCTxDCT is selected, we must perform the inverse transform and a forward DCT transform.

The current BD-Rate of this prototype is

AV1 (master, with PVQ) vs CFL (with RDO, Slow ADST support)

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
0.7153	-3.8530	-1.2560	0.6849	0.7341	0.6705	-0.3704

https://arewecompressedyet.com/?job=av1_master_pvq%402017-01-27T03%3A26%3A26.934Z&job=CfL_RDO%402017-02-01T18%3A50%3A05.719Z

Further research is required to allow TF when ADST is used, either by adapting TF to the ADST or converting the ADST to DCT.

Approach

The Intra prediction mode for Chroma is always DC_PRED. As such, we remove the Chroma Intra prediction mode from the Intra block mode info syntax. The resulting syntax is described in table 1.

Table1: Intra block mode info syntax (adapted from the VP9 spec)

intra_block_mode_info( ) {	Type
ref_frame[0] = INTRA_FRAME	
ref_frame[1] = NONE	
if (MiSize >= BLOCK_8X8) {	
intra_mode	T
y_mode = intra_mode	
for(b = 0; b < 4; b++)	
sub_modes[b] = y_mode	
} else {	

[*Work in progress, comments are welcomed*]

num4x4 w = num_4x4_blocks_wide_lookup[MiSize]	
num4x4h = num_4x4_blocks_high_lookup[MiSize]	
for(idy = 0; idy < 2; idy += num4x4h) {	
for (idx = 0; idx < 2; idx += num4x4w) {	
sub_intra_mode	T
for (y2 = 0; y2 < num4x4h; y2++)	
for(x2 = 0; x2 < num4x4w; x2++)	
sub_modes[(idy+y2) * 2 + idx + x2] = sub_intra_mode	
}	
}	
y_mode = sub_intra_mode	
}	
uv_mode = DC_PRED	
}	

As shown here, TF is not compatible with the ADST. As such, when a transform type other than DCTxDCT is selected, we must perform the inverse transform and a forward DCT transform before storing the coefficient. This way only DCTxDCT transformed coefficients are used for CfL. This also requires that the transform type for Chroma be DCTxDCT.

Prototype 2 (CfL as a Chroma Intra Prediction Mode):

An extra Intra prediction mode CFL_PRED is added to Chroma Intra prediction. When this mode is chosen the prediction is built using DC_PRED and CfL is used to populate the ACs when the PVQ function is invoked.

Status

Work as started on adding the CFL_PRED mode. However, managing all the probabilities required for entropy coding is complicated. Used dummy probabilities to start but the whole thing segfaults. *** DEBUGGING ***

Prototype 3 (Spatial Prediction)

In this prototype the CfL prediction is in the pixel domain prediction instead of the frequency domain prediction. In the context of AV1, a CfL prediction in the pixel domain is simpler to implement and to maintain and also results in a better rate-distortion ratio.

The reason for the later is that the transform size between Chroma and Luma may be considerably different. This requires multiple TF merge operations resulting in error propagation in the frequency domain prediction.

In the pixel domain, TF is not needed as adjacent pixel can be concatenated in order to produce a prediction of the desired size. Concatenation of adjacent pixel values does not add error to the prediction.

In Daala, the motivation for TF is that predictions were already in the frequency domain. This is not the case for AV1. Since PVQ in AV1 already requires that spatial predictions be transformed into the frequency domain, we argue that, when combined with PVQ, pixel domain CfL does not incur any supplemental transform operations with the added benefit of not adding error to the prediction.

Another benefit of this approach is that it facilitates compatibility with the multiple transform types, transform scales, transform shapes and transform sizes found in AV1.

Status

Work has begun in porting prototype 1 to produce a pixel domain prediction.

[*Work in progress, comments are welcomed*]

Progress

Establishing a baseline

In Disable ADST, we modify AV1 to remove ADST, all blocks are transformed using DCT_DCT. This will serve as our baseline for development. Transform type signaling is not removed as ADST support will be added later on.

A configure flag named dct_only is added to build AV1 using only the DCT. Asserts are added to both the encoder and decoder to validate the only DCT is used.

The BD-Rates of AV1 and AV1 without ADST for subset1 are as follows:

AV1 vs AV1 without ADST

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
3.9636	2.9882	2.9406	3.3386	3.8011	3.7404	3.3981

https://arewecompressedyet.com/?job=av1_master_pvq%402016-12-23&job=DCT_only%402016-12-23T14%3A59%3A21.443Z

Preparing for CfL

In Remove Chroma Intra, we remove Chroma Intra prediction signaling in the bitstream and force the use of DC_PRED.

A configure flag named cfl is added to build AV1 using DC_PRED only (and other CfL features in later releases). Asserts are added to both the encoder and decoder to validate the only DC_PRED is used.

The resulting BD-Rates for subset1 are as follows:

AV1 without ADST vs Removing Chroma Intra signaling

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
1.5922	2.4575	3.7760	1.6213	1.5675	1.5131	2.2787

https://arewecompressedyet.com/?job=Remove_Chroma_Intra_Pred%402016-12-23&job=DCT_only%402016-12-23T14%3A59%3A21.443Z

[*Work in progress, comments are welcomed*]

Enable key frames

In Remove Gain Prediction, we remove gain prediction for the Chroma planes. To do so, we enable the keyframe variable in PVQ. However, we disable PVQ features like CfL and skipping the DC.

Enabling keyframe code also enables skip zero, which zeros out the prediction when quantized gain is 0.

The resulting BD-Rates for subset1 are as follows:

Removing Chroma Intra signaling vs Removing gain prediction

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.0193	0.2353	0.0368	-0.0117	-0.0125	-0.0150	0.0574

https://arewecompressedyet.com/?job=Remove_Chroma_Intra_Pred%402016-12-23&job=No_Chroma_No_Gain%402017-01-09T20%3A53%3A17.436Z

CfL Flip Bit

In Flip Bit, we signal a flip bit to indicate whether to flip the sign of the AC coefficients. Checks are made to ensure that flipping does not occur during RDO as CfL is not currently applied during RDO.

When CfL is enabled, the dot product of the CfL prediction and the reference is measured this allows to determine if flipping the sign of the prediction results in a better prediction than using the original prediction.

Barebone CfL vs CfL with bit flipping

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.7497	-5.5682	-3.4322	-0.6781	-0.7819	-0.7453	-2.3553

https://arewecompressedyet.com/?job=Barebone-CfL%402017-01-10T16%3A30%3A48.410Z&job=CfL-FlipBit%402017-01-10T19%3A21%3A31.742Z

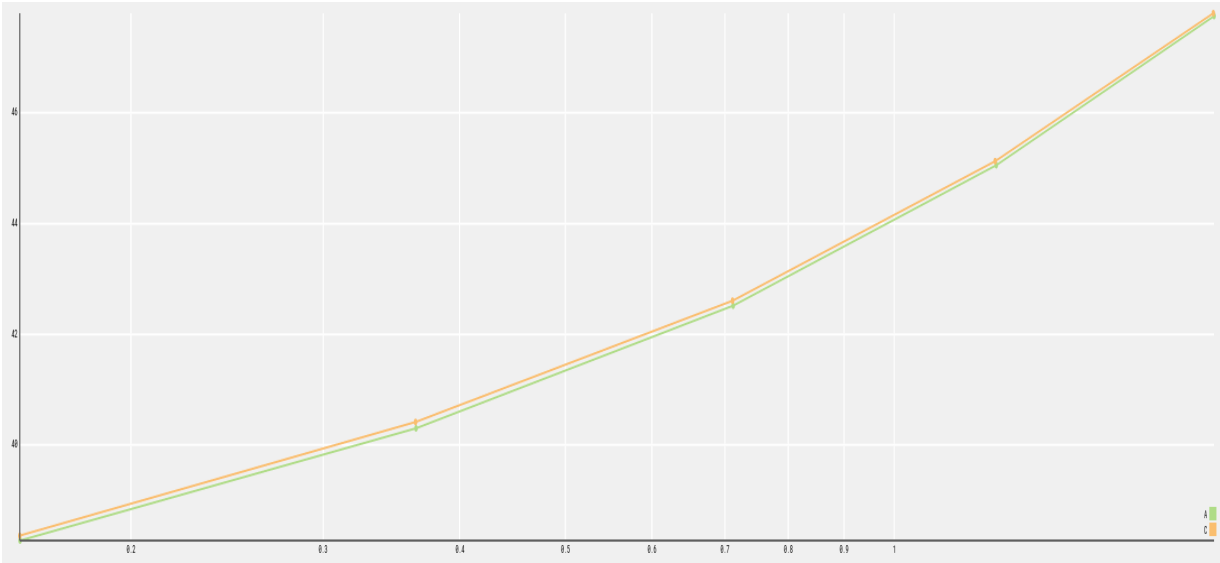
[*Work in progress, comments are welcomed*]

CfL

In Barebone CfL, we replace the AC coefficients of the forced DC_PRED prediction of Chroma Intra blocks with the AC coefficients of the dequantized transformed luma coefficients.

The resulting BD-Rates are as follows:

Removing gain prediction vs Barebone-CfL

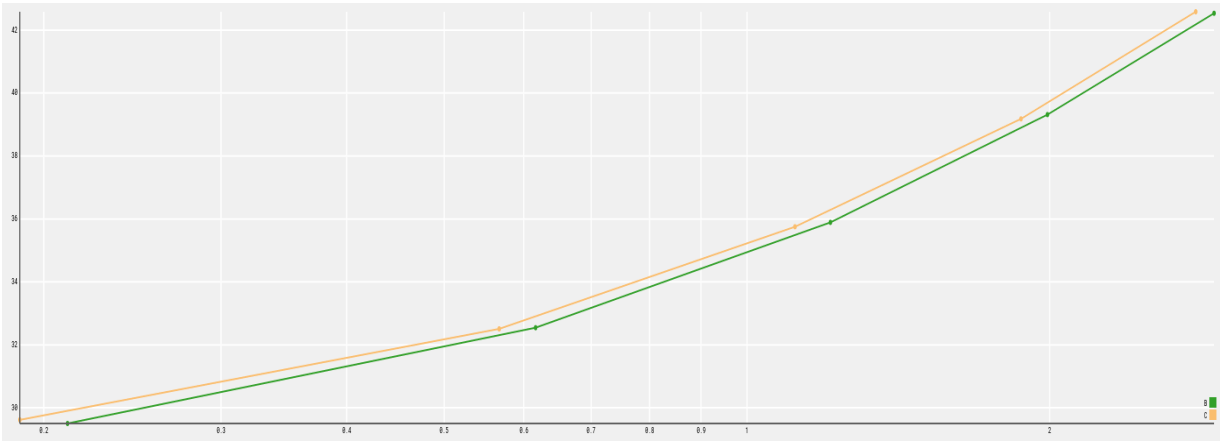


PSNR Cr: (A) removing gain prediction (C) Barebone-CfL

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-0.2032	-1.3154	-2.2157	-0.2731	-0.2246	-0.2454	-0.4316

https://arewecompressedyet.com/?job=No_Chroma_No_Gain%402017-01-09T20%3A53%3A17.436Z&job=Barebone-CfL%402017-01-10T14%3A41%3A08.607Z

Barebone-Cfl with and without gain prediction



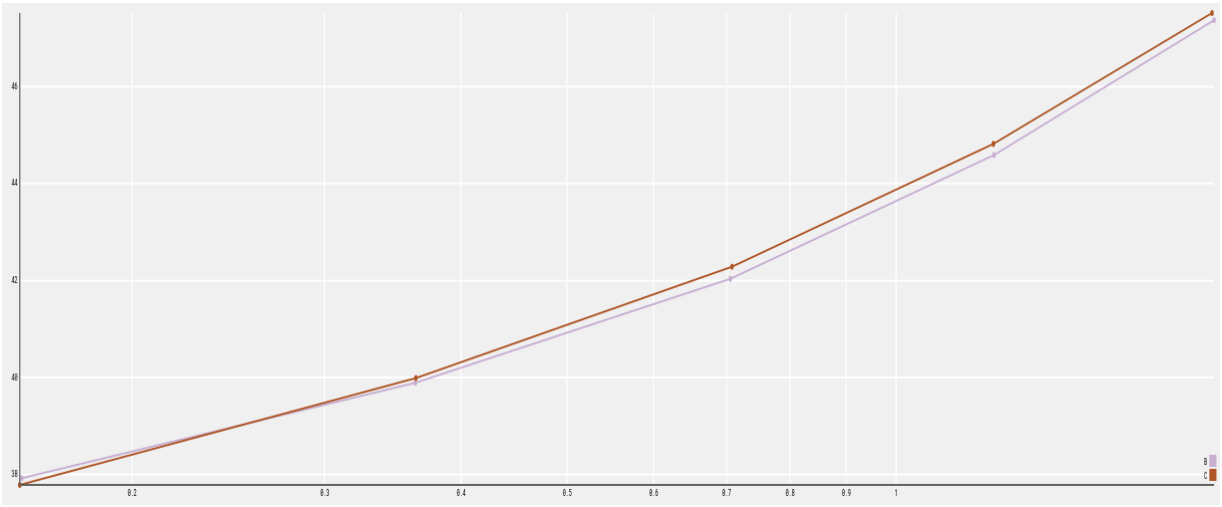
CIEDE 2000: (B) Barebone-CfL with gain prediction, (C) Barebone-CfL without gain prediction

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
-7.8430	-5.1001	-5.2241	-8.2927	-7.8688	-7.9671	-7.5249

https://arewecompressedyet.com/?job=Barebone-CfL%402017-01-06T18%3A46%3A12.624Z&job=Barebone-CfL%402017-01-10T14%3A41%3A08.607Z

[*Work in progress, comments are welcomed*]

AV1 Without ADST vs CFL with bit flipping



PSNR Cb: (B) AV1 without ADST, (C) CfL with bit flipping

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
0.6084	-4.4272	-2.0530	0.6673	0.5464	0.5234	-0.5253

https://arewecompressedyet.com/?job=DCT_only%402016-12-23T14%3A59%3A21.443Z&job=CfL-FlipBit%402017-01-10T19%3A21%3A31.742Z

CfL is performed every time PVQ is called for Chroma

To add CfL in RDO section of AV1 an extra PVQ encode is required. This PVQ encode is performed after all Luma intra modes have been tested, but before Chroma intra modes are tested. The PVQ encode uses the best Luma Intra mode and stores the dequantized Luma coefficient as prediction for Chroma during Chroma Intra mode testing.

The buffer used to store the CfL prediction is now zeroed out when on the first luma store of a partition. It was discovered that certain edge condition caused stale memory reads. Zeroing out these value solves this problem.

The BD-Rates of AV1 without ADST and CfL (with RDO) for subset1 are as follows:

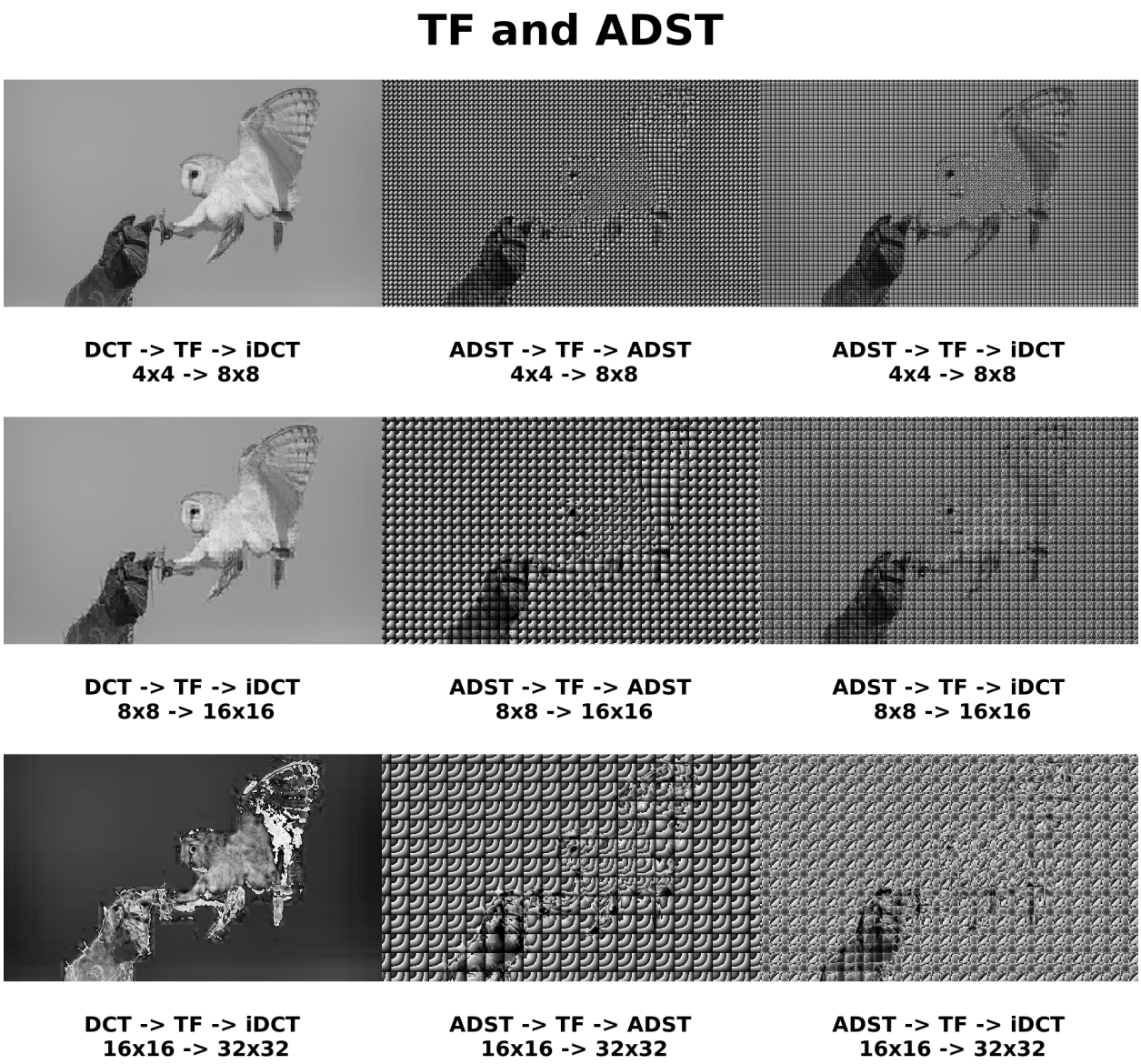
AV1 Without ADST vs CFL (with RDO)

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
0.4033	-5.7431	-3.0341	0.3989	0.3933	0.3254	-1.1425

https://arewecompressedyet.com/?job=DCT_only%402016-12-23T14%3A59%3A21.443Z&job=CfL_RDO%402017-01-20T17%3A26%3A01.851Z

ADST and TF

In this experiment, we transformed blocks using AV1's ADST and TF-merged 2Nx2N quartets of NxN blocks. It can be seen from the following image that ADST and TF are not compatible. More research is required to determine if an approximation is possible in order to allow TF on ADST or convert the ADST to a DCT.



[*Work in progress, comments are welcomed*]

Fast DCT for DC_PRED

As explained here, the DC of a DC_PRED prediction can be computed using $Y_0 = 128\, x_0$ for 32x32 blocks and $Y_0 = 8\, N\, x_0$ for smaller block sizes. In DC_PRED, we implement this approach in AV1 master and obtain the following results

AV1 master vs Fast DCT for DC_PRED

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
0.1266	0.0056	-0.0100	0.1836	0.1733	0.1642	0.0992

https://arewecompressedyet.com/?job=av1_master_pvq%402017-01-27T03%3A26%3A26.934Z&job=Fast_Transform_For_DCPRED%402017-01-27T13%3A32%3A38.574Z

This regression is caused by two factors:

1. Bias is added to the forward transform, to reduce the rounding error. However, the resulting transform does not have all ACs at zero all the time for DC_PRED.
2. AV1 applies a $\frac{1}{\sqrt{(2)}}$ scaling factor to the DC. This is done by multiplying by $\cos(\frac{\pi}{4})$ to the DC which results in rounding error. This rounding error is compounded in a 2D DCT as this scaling factor is performed with on the DC.

Fixing the DC Rounding Error

Rounding error on the DC should be limited to the rounding error of $\frac{1}{2}$ since $(\frac{1}{\sqrt{(2)}})^2 = \frac{1}{2}$. In FIX_DC, we compute a separate DC which is scaled by $\frac{1}{2}$ and we replace the DC of the 2D DCT with it. FIX_DC is implemented in AV1 master and the following BD-Rate is obtained

AV1 master vs Fix Rounding Error on DC

PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM	CIEDE 2000
0.0129	0.0085	-0.1475	0.0592	0.0220	0.0518	-0.0122

https://arewecompressedyet.com/?job=av1_master_pvq%402017-01-27T03%3A26%3A26.934Z&job=DCT_DC_plus1%402017-01-31T20%3A48%3A09.043Z

Fixing the DC causes a slight regression. *Best guess for now is that the rounding error on the DC acts like a quantizer...To be continued.*

[*Work in progress, comments are welcomed*]

Slow ADST support

In Slow_ADST, when a transform type other than DCTxDCT is chosen, we perform the inverse transform and forward DCTxDCT before storing the coefficient. This way, the stored transformed Luma coefficients are always transformed with the DCTxDCT. As for the Chroma transform, it remains an implicit DCTxDCT.

AV1 (master, with PVQ) vs CFL (with RDO, Slow ADST support)

```
  PSNR | PSNR Cb | PSNR Cr | PSNR HVS |   SSIM | MS SSIM | CIEDE 2000
0.7153 | -3.8530 | -1.2560 |   0.6849 | 0.7341 |  0.6705 |    -0.3704
```

https://arewecompressedyet.com/?job=av1_master_pvq%402017-01-27T03%3A26%3A26.934Z&job=CfL_RDO%402017-02-01T18%3A50%3A05.719Z

Converting from ADST to DCTxDCT

Since we know that we will perform an inverse ADST (iADST) and a forward DCT (fDCT) we can combine both transform into a single transform. The motivation is that, once combined, it might be possible to factor out some parts of these functions.

In AV1, the 1D transforms can be mixed together to form 2D transforms resulting in the following 2D transforms: ADSTxADST, ADSTxDCT, DCTxADST and DCTxDCT. Let ADST_2_DCT be a transform that perform both an iADST and a fDCT, the following table show the steps required to go from a given transform type to the DCTxDCT.

Tx type	Step 1	Step 2	Step 3
ADSTxADST	iADST	ADST_2_DCT	fDCT
ADSTxDCT	iDCT	ADST_2_DCT	fDCT
DCTxADST	ADST_2_DCT		
DCTxDCT			

In the following experiment, we show that compared to the combined number of operations of the iADST and fDCT, combining them allows to remove about 20% of those operations.

Further up work is needed on this experiment, as Unlord indicated the for planar rotations can be achieved using 3 multiplies instead of 4.