Overview of Coding Tools Under Consideration in AVM

Debargha Mukherjee, Onur Guleryuz, **Google**

Van Luong Pham, Yeqing Wu, **Apple**

Liang (Leo) Zhao, Madhu Peringassery Krishnan, **Tencent**

On behalf of AOM Codec Working Group

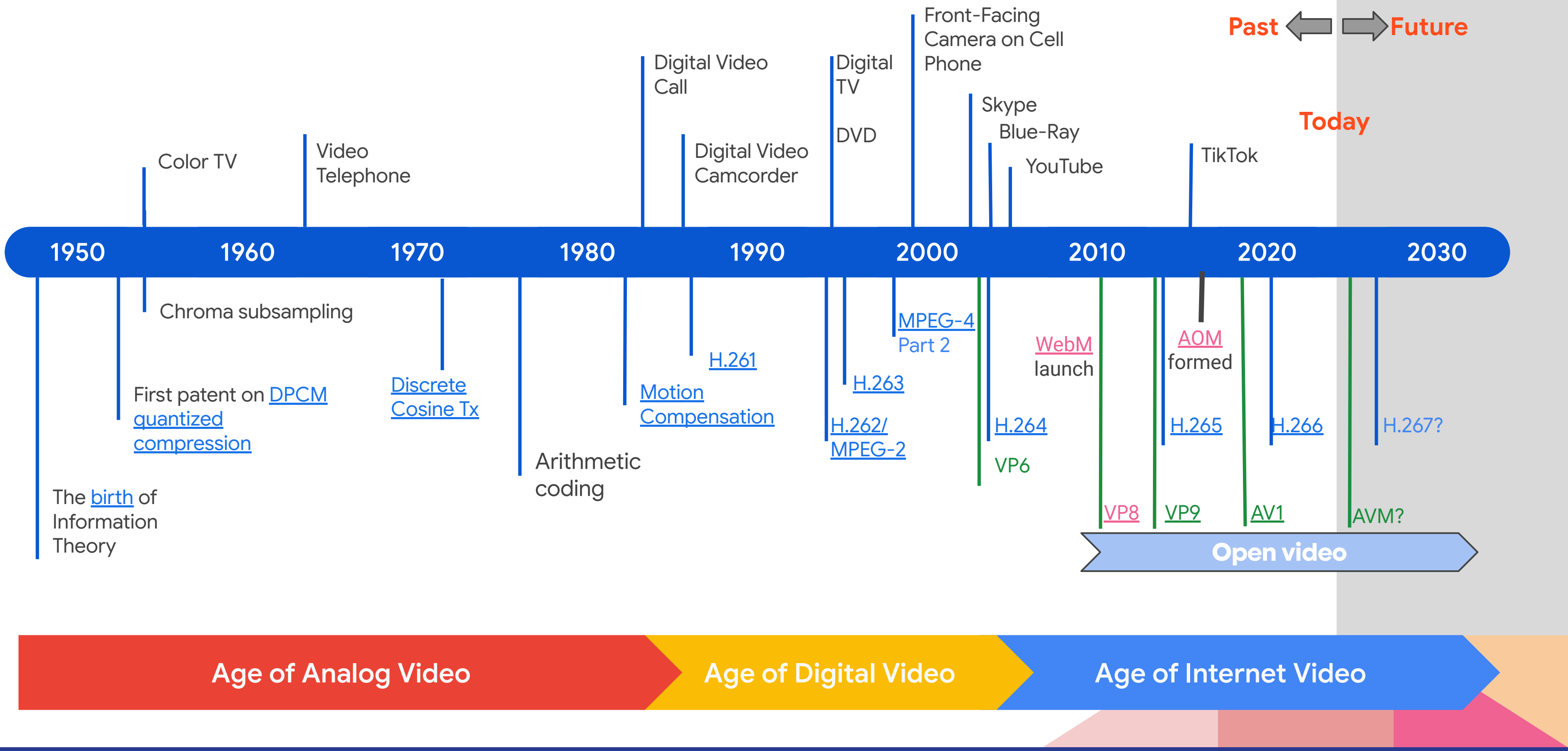
AOM Workshop @ ICIP 2024, Abu Dhabi, UAE, October 30, 2024

Outline

- Introduction
- Tools Overview
  - Quantization
  - Partition tools
  - Intra coding tools
  - Inter coding tools
  - Transform coding tools
  - Entropy coding tools
  - In-Loop filtering tools
- Conclusion

Introduction

Introduction



Next-Gen AOM Video Codec (AVM)

- In 2020, AOM started exploratory work towards a **next-gen video codec**
- Requirements for next-gen codec:
  - Should achieve ~40% lower bandwidth than AV1 at equivalent perceptual quality
    - PSNR-wise target could be lower
  - Should not have a decoder side complexity/area of more than 2X that of AV1
- Initial phase of development (2020-21):
  - Formalized Processes to be followed during development
  - Testing Sub Working Group: Selected Test Sequences; defined Common Test Conditions (CTC)
  - Starting Codebase: *A version of* libaom - open-source AV1 codec library selected
    - Inception of **AOM Video Model** (**AVM**)
  - Setup of repository on gitlab: https://gitlab.com/AOMediaCodec/avm
  - Selection of Chairs for various working groups and sub-working groups, SW co-ordinators

Introduction

Progress so far

- In the last three years, many new tools have gone through the CWG review process and have been chosen as candidates in AVM
  - A new version of AVM is released ~ every 4 months
- **Rigorous scrutiny** for hardware decoding complexity
- Current status:
  - Latest anchor: AVM-v8.0.0
  - A good way towards our goal in terms of objective metrics
- New codec finalization schedule
  - TBD
- Disclaimer:
  - None of the tools presented here are guaranteed to exist in the final next-gen AOM codec.
  - Only covers tools that are in AVM v-8.0.0

Coding Tool Contributors and Collaborators

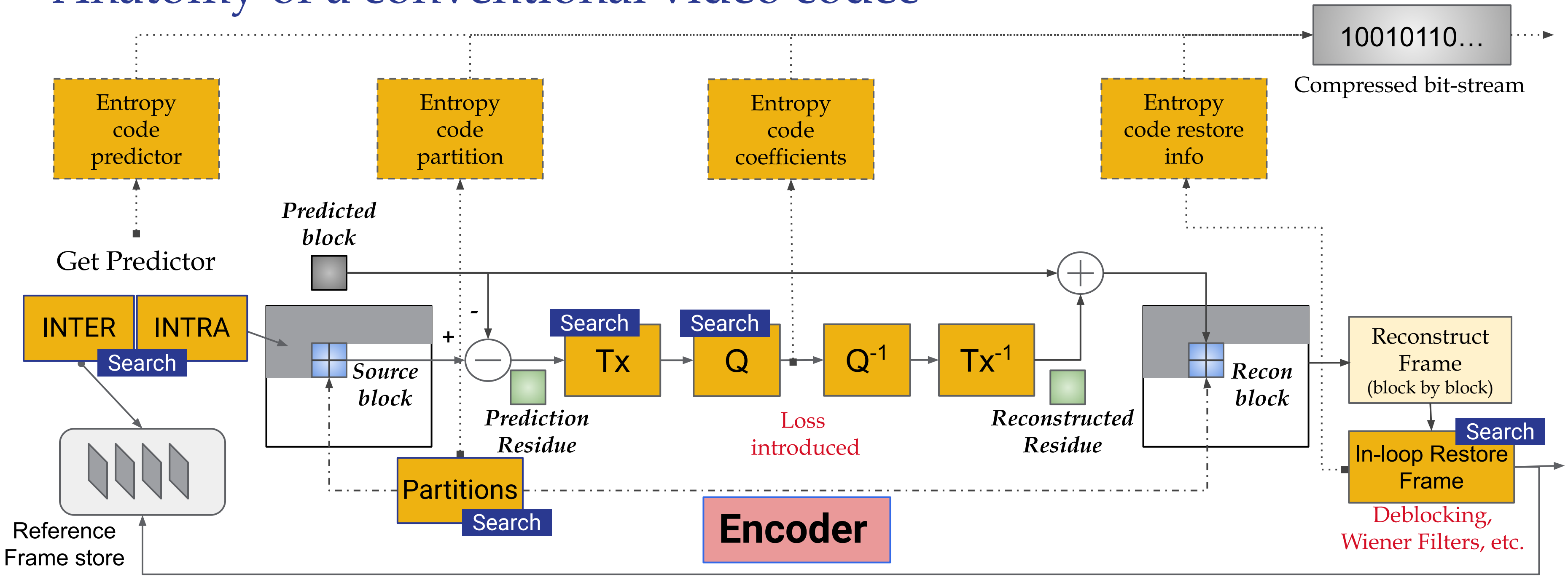
- Industry Contributors:
  - Amazon
  - Apple
  - Cisco
  - Google
  - Intel
  - Ittiam
  - Meta
  - Netflix
  - Oppo
  - Samsung
  - Tencent
  - Videolan
  - …
- Academic Collaborators:
  - UCSB
  - USC
  - NYU
  - Hangzhou University
  - …

- AVM is an open-source project!
- We are always welcoming new contributors

Tools Overview

Tools Overview

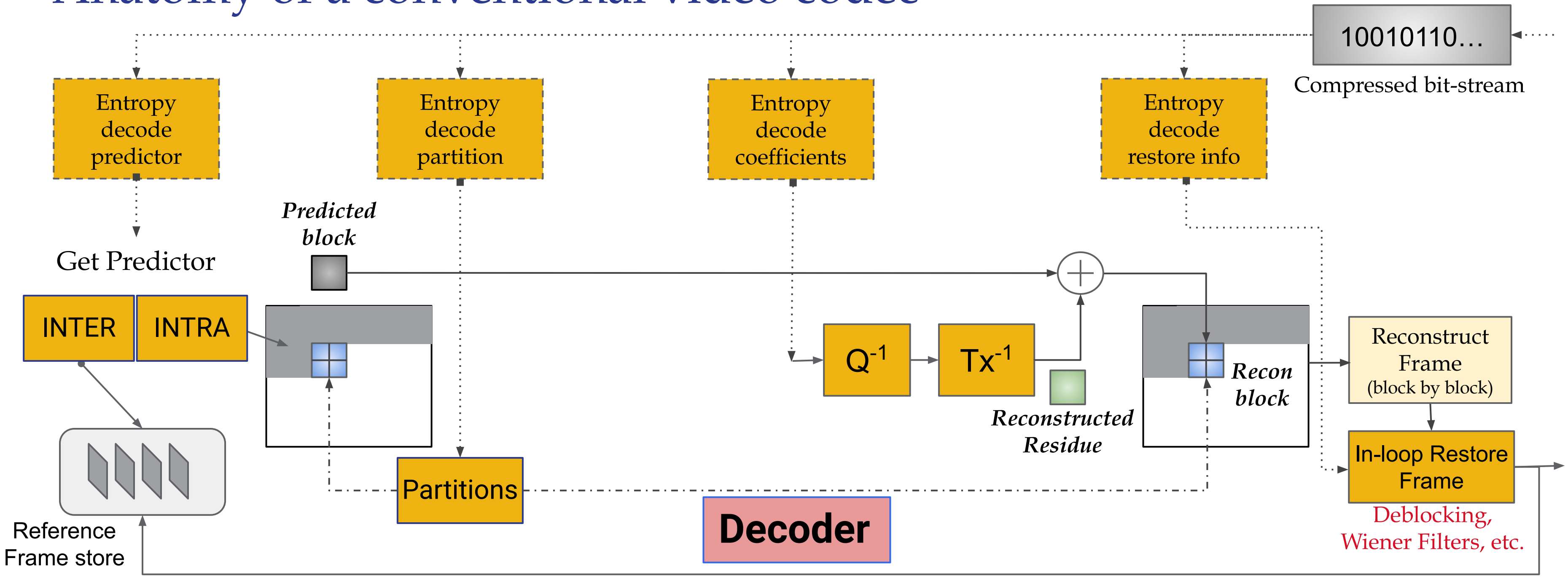
Anatomy of a conventional video codec



Framework has not changed much in the last 35 years !

Tools Overview

Anatomy of a conventional video codec



Framework has not changed much in the last 35 years !

Tools Overview

- Intra Predictor
  - Multiple Reference Line Selection
  - Adaptive Intra Mode Coding
  - Intra Bi-Prediction
  - Local Intra Block Copy
  - Improved CfL Prediction
- Inter Predictor
  - Reference Framework
  - Temporal Interpolated Frame
  - Optical Flow MV Refinement
  - MV Prediction
  - MV Coding
  - Enhanced Warp Prediction
  - Extended Wedge Compound Mode
  - Block Adaptive Weighted Prediction
  - Skip Mode Enhancement
- Quantization
  - Extended Quantization
- Partitioning
  - Extended Recursive Partitioning
  - Semi-Decoupled Partitioning
- Transform Coding
  - New Transform Partitioning
  - Primary Transforms
  - Secondary Transforms
  - Cross Component Transform
- Entropy Coding
  - Parity Hiding & Forward Skip Coding
  - Coefficient Coding Improvements
  - Arithmetic Coding Improvements
- In-Loop Filtering
  - Improved Deblocking Filters
  - Cross Component Sample Offset
  - Extended Loop Restoration Filters

Tools Overview

Quantization

Quantization: Extended Quantization Redesign

- AV1 Quantizer
  - Quantization Index (QP) to step-size mapping are given by predefined look-up tables
  - 256 quantization levels
  - Different tables used for 8/10/12-bit content as well as AC/DC
- AVM New Quantizer
  - Quantization Index (QP) to step-size mapping given by a unified exponential formula
  - 10-bit, 12-bit quantizers and AC/DC modifiers are shifts on indices
    - Steps double for every 24
  - 256/304/352 levels for 8/10/12-bit content
  - Range expanded to cover wider range of qualities

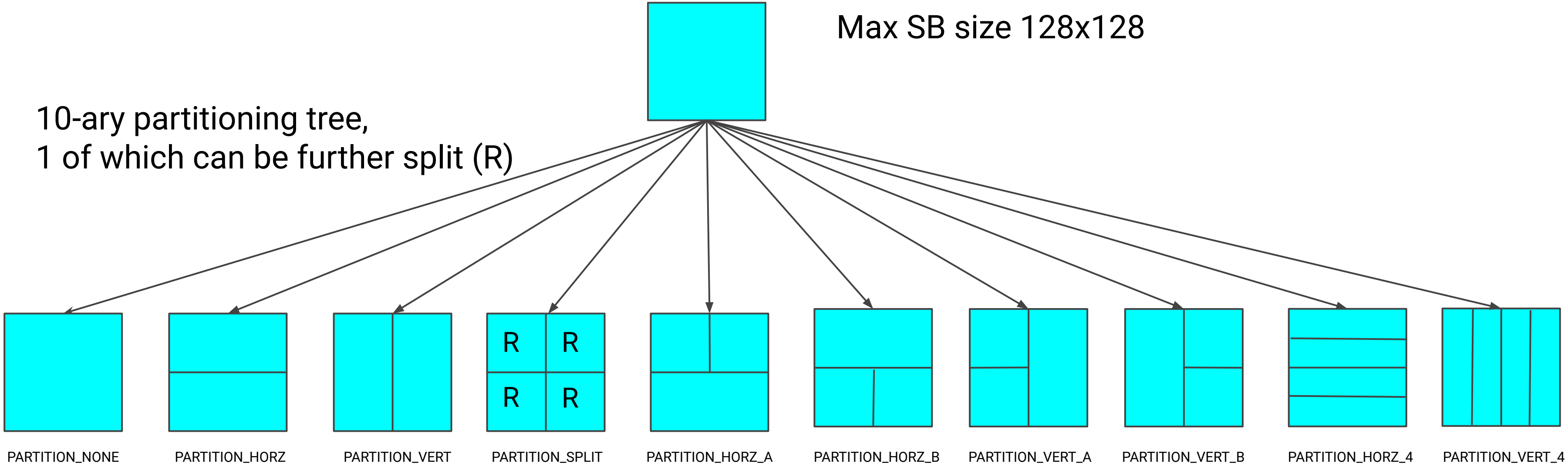
$$Q_{step} = \begin{cases} 32 & q_{index} = 0 \\ round\left(2^{\frac{q_{index}+127}{24}}\right) & q_{index}\, in\, [1, 24] \\ Q_{step}[(q_{index} - 1)\,\%\,24) + 1] * (2^{\left\lfloor\frac{q_{index}-1}{24}\right\rfloor}) & q_{index} \geq 25 \end{cases}$$

Partitioning Tools

Tools Overview

Partitioning: Extended Recursive Partitions (ERP)

- AV1 partitioning types



R = Recursive

Tools Overview

Partitioning: Extended Recursive Partitions (ERP)

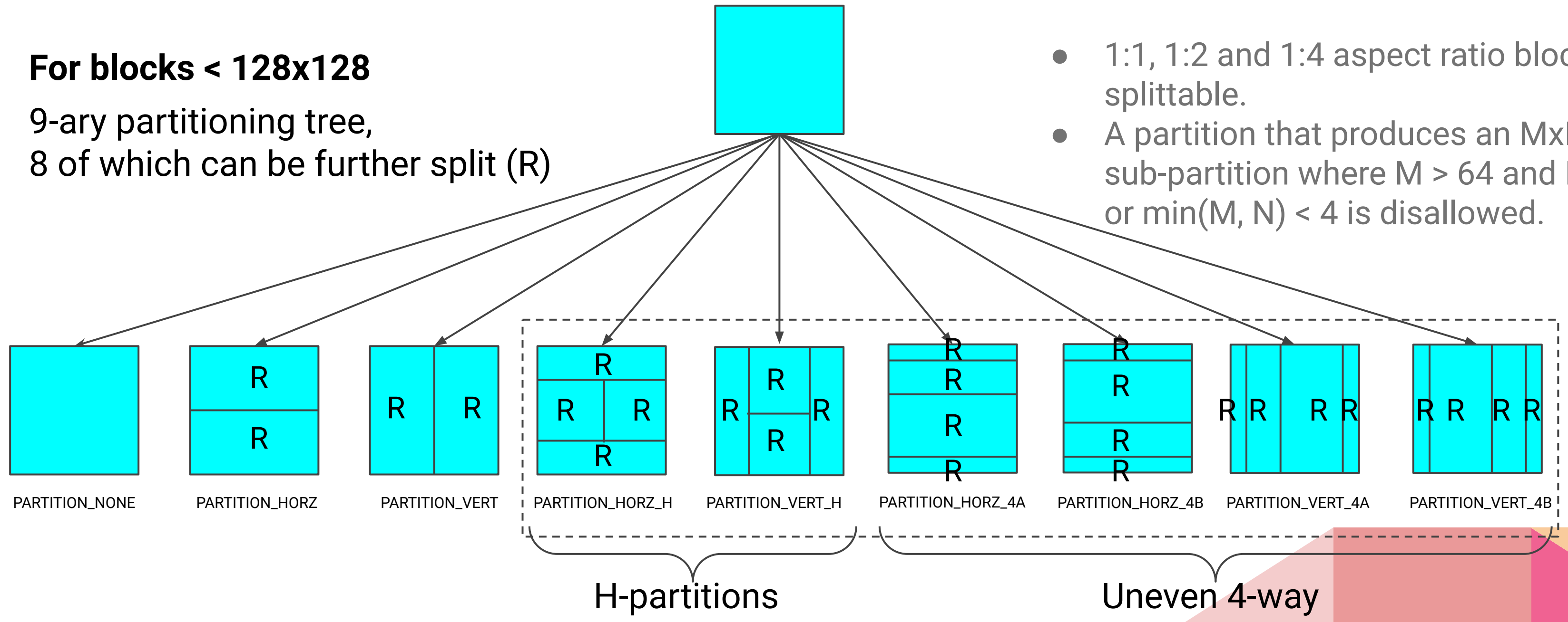
- AVM partitioning tree (so far)

Max SB size 256x256, processing unit is still 64x64

For blocks < 128x128

9-ary partitioning tree,
8 of which can be further split (R)

- 1:1, 1:2 and 1:4 aspect ratio blocks are splittable.
- A partition that produces an MxN sub-partition where M > 64 and N < 64, or min(M, N) < 4 is disallowed.



Tools Overview

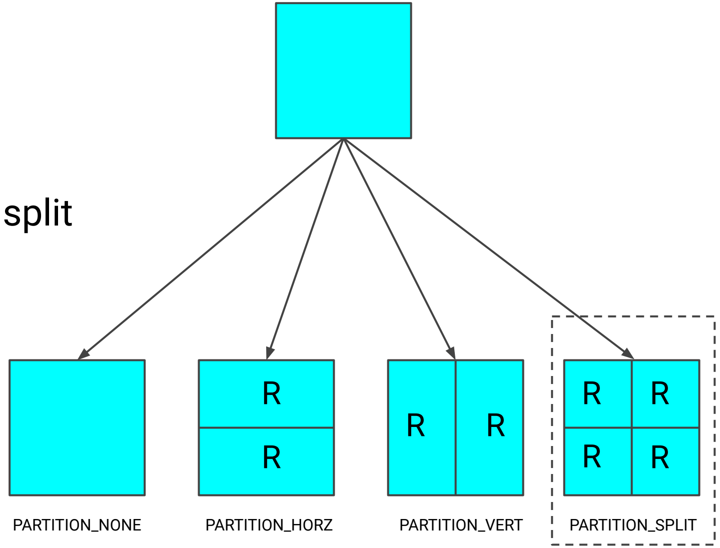
Partitioning: Extended Recursive Partitions (ERP)

- AVM partitioning tree (so far)

Max SB size 256x256, processing unit is still 64x64

For 256x256, 128x128,
256x128, 128x256

4-ary partitioning tree,
3 of which can be further split



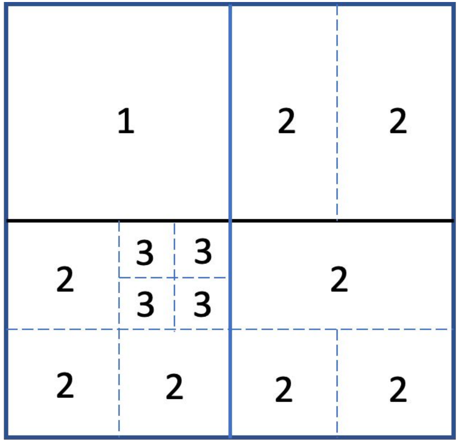
Partitioning: New Rectangular Block Sizes and Tx Sizes

- Coding block sizes:
  - Square: 4x4, 8x8, 16x16, 32x32, 64x64, 128x128, **256x256**
  - Rectangular 1:2/2:1: 4x8, 8x4, 8x16,16x8,16x32, 32x16, 32x64, 64x32, 64x128, 128x64, **128x256, 256x128**
  - Rectangular 1:4/4:1: 4x16,16x4, 8x32, 32x8, 16x64, 64x16
  - **Rectangular 1:8/8:1: 4x32, 32x4, 8x64, 64x8**
  - **Rectangular 1:16/16:1: 4x64, 64x4**
- Transform block sizes:
  - Square: 4x4, 8x8, 16x16, 32x32, 64x64
  - Rectangular 1:2/2:1: 4x8, 8x4, 8x16, 16x8, 16x32, 32x16, 32x64, 64x32
  - Rectangular 1:4/4:1: 4x16, 16x4, 8x32, 32x8, 16x64, 64x16
  - **Rectangular 1:8/8:1: 4x32, 32x4, 8x64, 64x8**
  - **Rectangular 1:16/16:1: 4x64, 64x4**

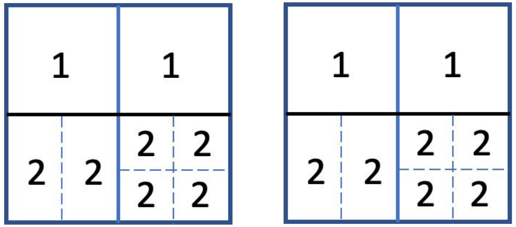
■ **New blocksizes added**

Partitioning: Semi Decoupled Partitioning (SDP)

- In AV1, partition structure is shared between luma and chroma
- SDP
  - <u>Same</u> luma & chroma partition structure up to BLOCK_64X64.
  - <u>Independent</u> luma & chroma partition structure at smaller block sizes.



Luma coding block partition

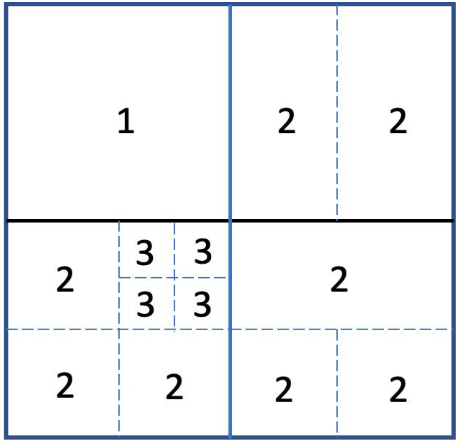


Chroma coding block partition

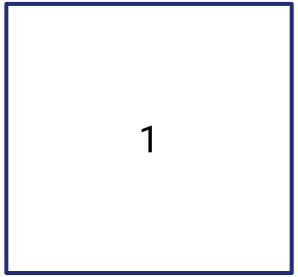
Tools Overview

Partitioning: Semi Decoupled Partitioning (SDP) in Inter Frames

- Coding blocks <= BLOCK_64X64 may use semi-decoupled partitioning
- If SDP is used for a coding block:
    - luma channel can be further partitioned recursively
    - chroma channel cannot be partitioned further.
    - All sub-blocks within this coding block will use intra prediction only



Luma coding block partition

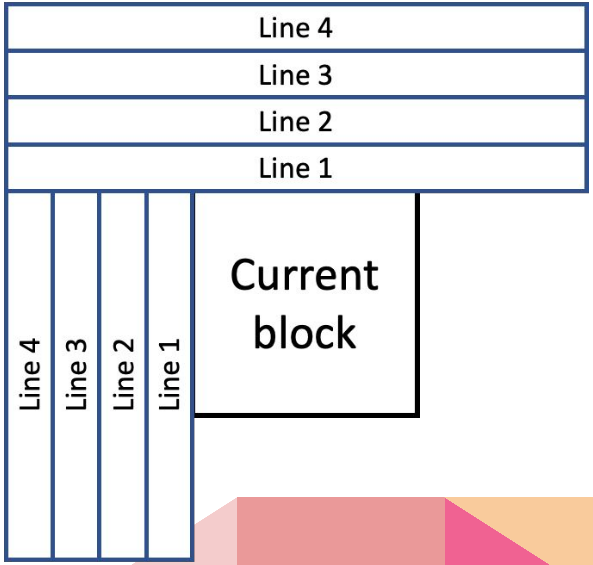


Chroma coding block partition

Intra Coding Tools

Intra: MRLS & AIMC

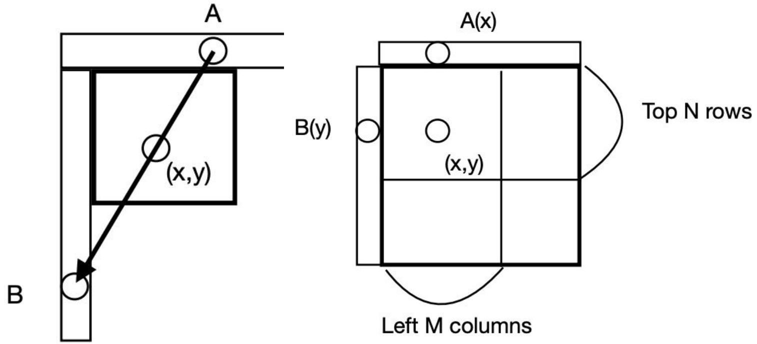
- Multiple Reference Line Selection (MRLS)
  - Select and signal one among 4 top/left reference lines for intra prediction
  - Applied to luma component only
  - Applied to all directional prediction modes
  - For blocks at SB boundary, 1 top and 4 left reference lines
  - Angle refinement {-1, 0, 1} is applied to different MRL
- Adaptive Intra Mode Coding (AIMC)
  - Rank available intra modes based on those of neighboring/colocated intra mode info
  - Split the ranked modes into mode sets, then signal
    - mode set idx
    - mode idx within mode set
  - Top ranked sets/modes are cheaper to signal.



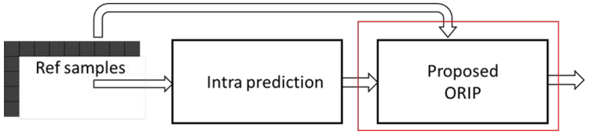
Tools Overview

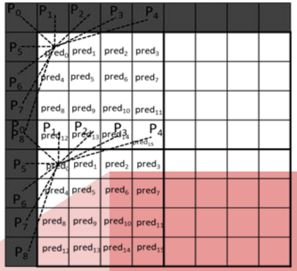
Intra: Intra Prediction

- Intra Bi-Prediction (IBP)
  - Prediction is the weighted average of A, and B.
  - Weights defined based on (mode, bsize, x, y)



- Offset based refinement of intra prediction (ORIP)
  - The offsets are generated from top and left neighboring prediction samples.
  - The refinement is applied to top-most 4x4 left-most 4x4 sub-blocks

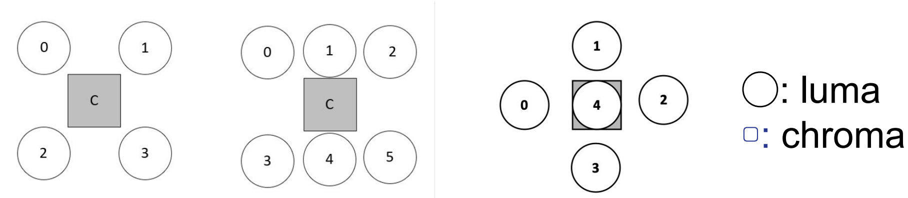




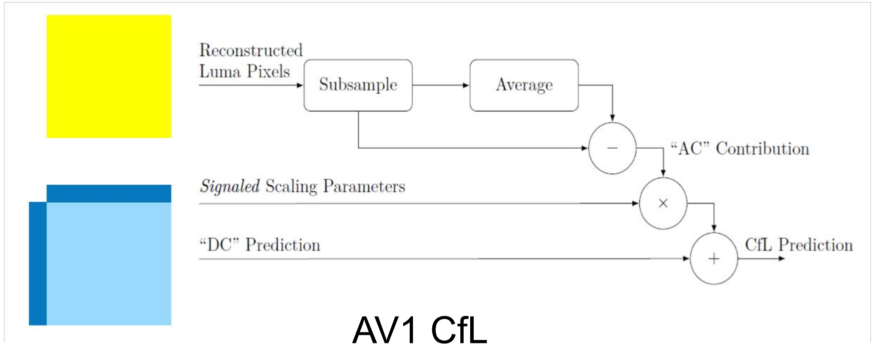
Tools Overview

Intra: Improved Chroma from Luma (CfL) Prediction

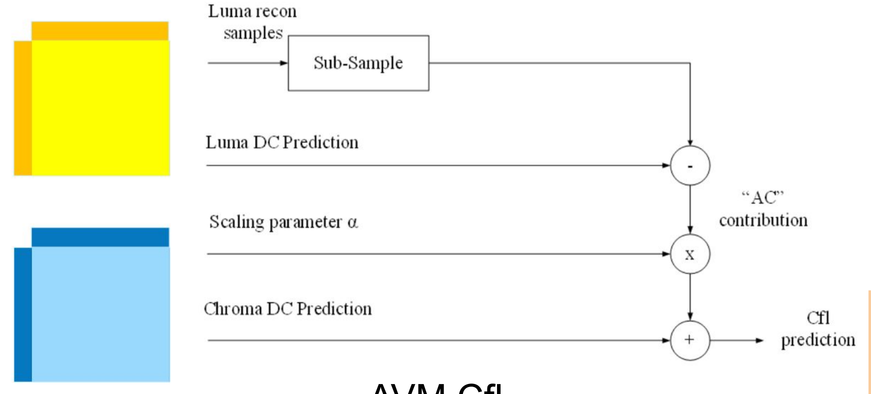
- AV1 CfL
  - Fixed 4-tap luma downsampling filter
  - Pred_c = *α* (Rec_y - DC_y) + DC_c
  - *α* is searched by encoder and signaled in the bitstream
- Improved CfL
  - Luma downsampling filter
    - Three filter types (4-, 5- or 6-tap)
    - Filter selection signaled in sequence header



  - Additional CfL mode with implicitly derived α
    - *α* = argmin sum(Rec_c – *α* * Rec_y)^2 over the L-shape region



AV1 CfL

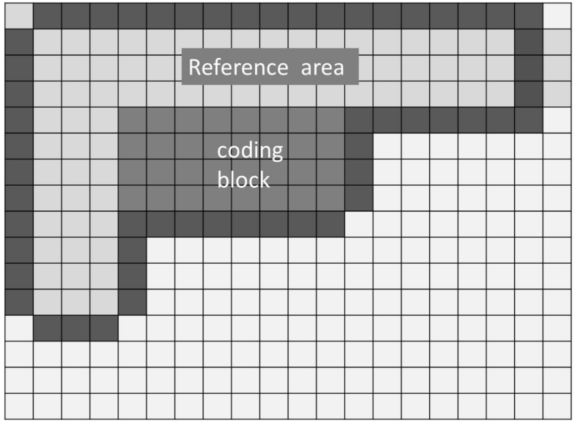


AVM CfL

Tools Overview

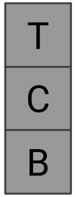
Intra: Multi Hypothesis Cross Component Prediction (MHCCP)

- Extend the reference lines for deriving the model parameters



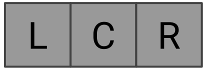
Two shapes are supported in MHCCP, and 5 parameters are derived in each shape

$$E = (C^2 + F) >> bit\_depth$$
$$F = 1 << (bit\_depth - 1)$$



Vertical shape:

$$chroma = w_0.C + w_1.T + w_2.B + w_3.E + w_4.F$$



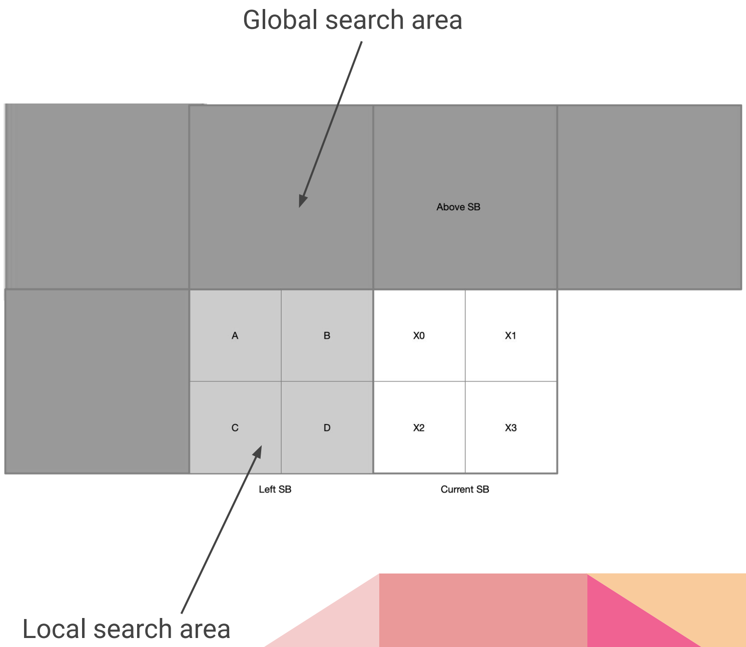
Horizontal shape:

$$chroma = w_0.C + w_1.L + w_2.R + w_3.E + w_4.F$$

Tools Overview

Intra block copy improvement

- Block vector prediction improvement
- Existing (hardware) on-chip 128x128 buffer for nearby pixels as an additional search area.
- Supports additional mode to generate linear predictor from top and left template.
- Modified signaling to enable for natural camera capture content.
    - Signaling of frame level intrabc flag does not depends on the “allow_screen_content_tool” flag.



Inter Coding Tools

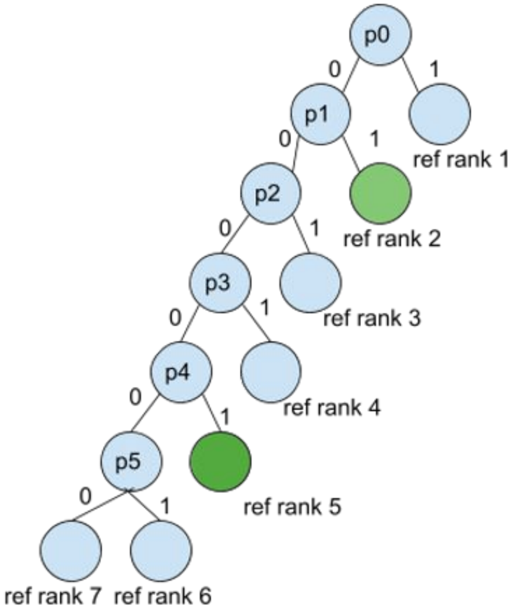
Tools Overview

Inter Coding: Reference Frame Framework

- New Reference Signaling
  - Replace named references by ranked references
    - LAST/LAST2/LAST3/GOLDEN/BWDREF/ALTREF2/ALTREF ⇒ ref rank indices 0-6
  - No signaling overhead for reference mapping
    - Ranks and num_total_refs are implicitly derived.
    - Ranking based on 1) temporal distance to reference frame (*d*) and 2) quantizer (*q*) of reference frame

$$\text{cost} = 64 * d + q$$

  - Bitstream syntax
    - Unary codeword for reference frame index signaling



Example: compound with refs ranked 2 & 5

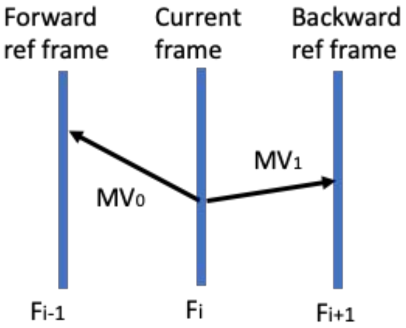
- 010001, if num_refs>5
- 01000, if num_refs=5

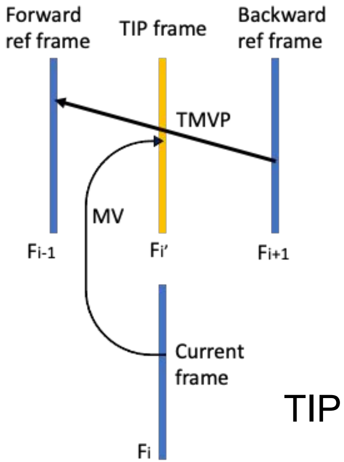
Tools Overview

Inter Coding: Compound Modes

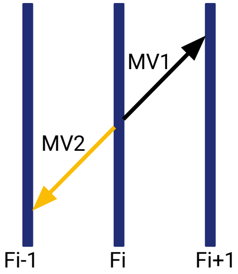
- Temporal Interpolated Prediction (TIP) frame
  - Interpolated intermediate frame between forward and backward reference frames
- TIP mode at block level
  - Only one motion information required
  - On-the-fly decoder implementation

Traditional compound prediction





TIP



Joint MVD

- Joint MVD coding
  - Signal MVD_0, and derive MVD_1 by scaling MVD_0
  - Scaling factor can be either derived from temporal distances or explicitly signaled.

Tools Overview

Inter Coding: Optical Flow MV Refinement

- MVs are refined for each (8x8 or 4x4) subblock in bi-directional prediction block
  - P0, P1: Reference blocks after MC.
  - Cur: Current source block.
  - (MVi_x, MVi_y) ← (MVi_x + di * vx, MVi_y + di * vy)

$$0 = \frac{\partial I}{\partial t} + v_x \frac{\partial I}{\partial x} + v_y \frac{\partial I}{\partial y}$$

P0:

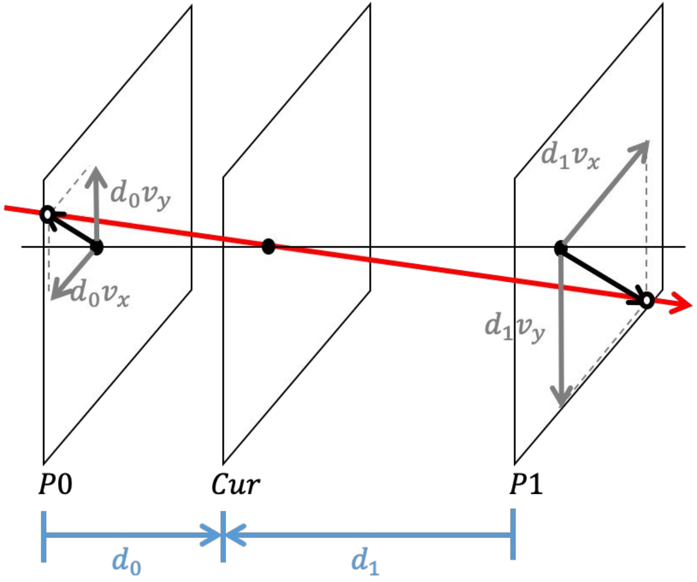
$$0 = P0 - Cur + v_x \frac{\partial P0}{\partial x} + v_y \frac{\partial P0}{\partial y},$$

P1:

$$0 = P1 - Cur + v_x \frac{\partial P1}{\partial x} + v_y \frac{\partial P1}{\partial y}$$

subtraction

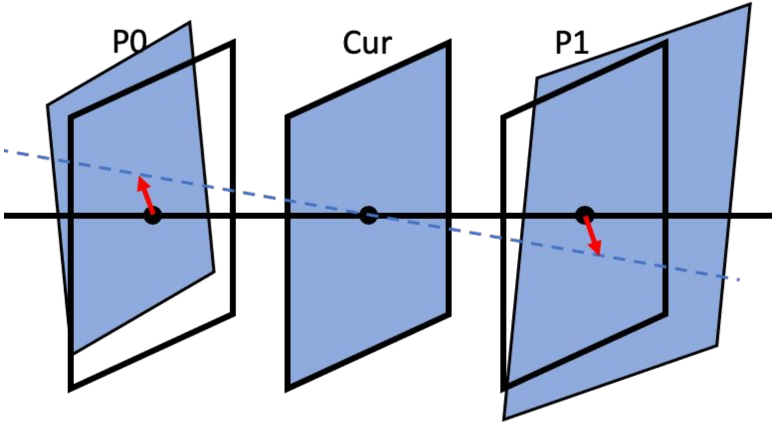
$$0 = (P0 - P1) + v_x \left(\frac{\partial P0}{\partial x} + \frac{\partial P1}{\partial x} \right) + v_y \left(\frac{\partial P0}{\partial y} + \frac{\partial P1}{\partial y} \right)$$



Tools Overview

Inter Coding: Decoder Side Affine Motion Refinement (DAMR)

- In optical flow refinement
  - Solve two translational parameters per subblock
- DAMR
  - 1st step: solve 4 affine parameters (rotation ϕ, scaling α, translational tx & ty) for whole block
    - Formulation based on the optical flow equation
    - Parameter are assumed to be small, which allows 4-parameter least squares solution
  - 2nd step: solve 2 translational parameters per subblock
    - Same as optical flow refinement
  - Recon step: compound warped prediction per subblock using parameters from two steps combined

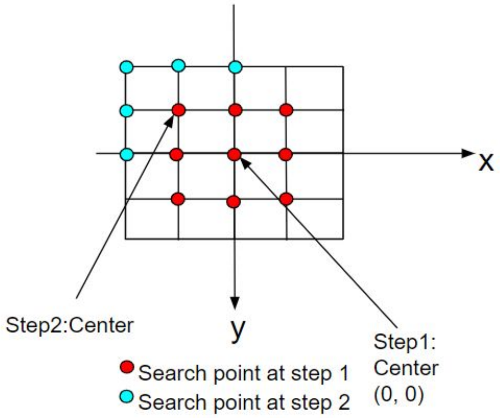


$$\begin{pmatrix} x' \\ y' \end{pmatrix} = (1+\alpha)^{\Delta t} \begin{pmatrix} \cos(\Delta t\phi) & -\sin(\Delta t\phi) \\ \sin(\Delta t\phi) & \cos(\Delta t\phi) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \Delta t \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$
$$\approx \begin{pmatrix} 1+\alpha\Delta t & -\Delta t\phi \\ \Delta t\phi & 1+\alpha\Delta t \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \Delta t \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Tools Overview

Inter Coding: Sub-Block Based MV Refinement

- Compound mode
- Divide a block into N 16x16 sub-blocks
- For each sub-block:
  - Search 5x5 region with MV0, MV1 (initial MVs) as center
  - For each offset (ΔMV)
    - Generate first predictor P0 using (MV0 + ΔMV) .
    - Generate second predictor P1 using (MV1 - ΔMV)
    - Compute SAD between P0 and P1
  - Find best offset (ΔMVbest) with minimum SAD
  - Final motion compensation using (MV0 + ΔMVbest) and (MV1 - ΔMVbest)



Inter Coding: Compound Weighted Prediction (CWP)

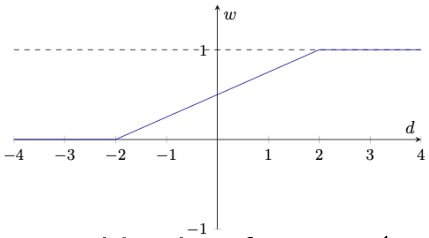
- Adaptive weighting compound mode
  - P0, P1: Reference blocks after MC
  - $P=(W\times P0+(16-W)\times P1+8)\gg 4$

	Weighting factors (W)
Two ref frames are from different directions	8, 10, 6, 12, 4
Two ref frames are from same directions	8, 12, 4, 20, -4

- Only applies to:
  - COMPOUND_AVERAGE mode
  - Some inter modes (NEAR_NEARMV, JOINT_NEWMV, or JOINT_AMVDNEWMV)

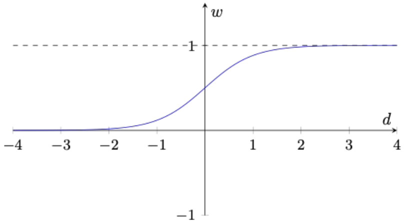
Inter Coding: Extended Wedge Compound Mode

- Explicit wedge mask signaling
  - Wedge mode is extended from 32x32 to 64x64 block
  - Wedge mode is extended from 16 to 68 modes
  - Blending function based on sigmoid function

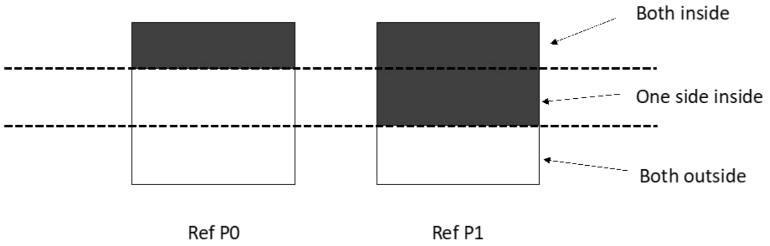


Linear blending function (AV1)

- Implicit wedge mask
  - Handling prediction samples outside frame boundary



Sigmoid blending function (AVM)



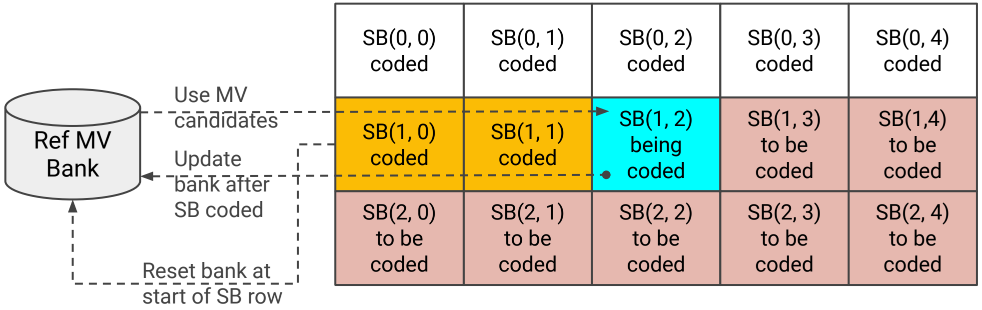
Weights = {32, 32}

Weights = {0, 64}

Weights = {32, 32}

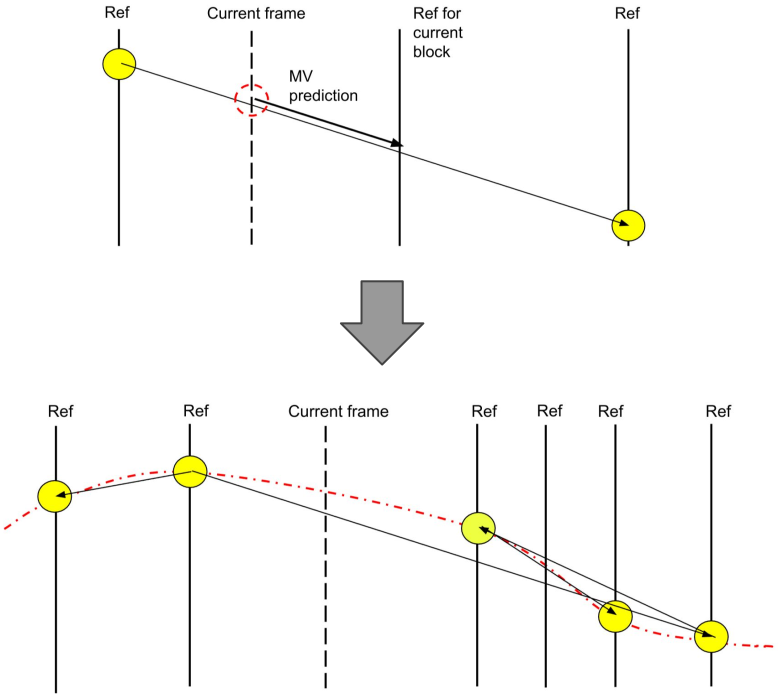
Inter Coding: MV Prediction

- Temporal and Spatial MV Prediction (TMVP/SMVP) Improvements
- Reference MV Bank



Inter Coding: MV trajectory tracking

- In addition to linear projection, track the trajectory of blocks by concatenating the decoded MVs in the reference frames.
  - Stored as a two-way mapping table
    - Block ⇔ trajectory ID
  - Improves both TMVP and SMVP
- Utilizes past motion information more efficiently
- Adaptive to non-linear motion
  - e.g. camera shake, objects moving along a curved path



Inter Coding: Motion Vector Resolution

- Adaptive MV Difference (AMVD)
  - Implicit adaptive MVD resolution
  - Only applies to some inter modes (NEAR_NEW, NEW_NEAR, JMVD_AMVD, AMVD_NEWMV)
  - A look-up table based approach where a set of predefined absolute MVD values are allowed.
    - For MVD magnitude > 1 pel, allow MV magnitudes of 2, 4, 8, ..., 2048 only, leading to reduction in the signaling cost of motion vector difference (down to 8 bits)

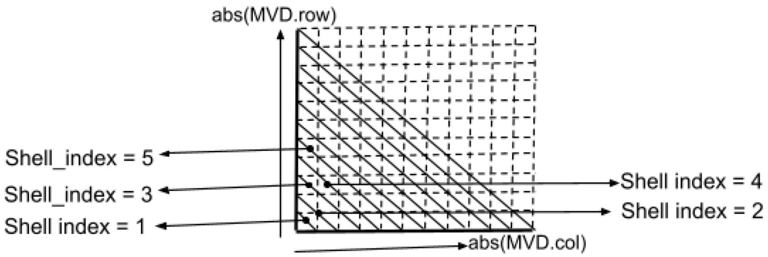
amvd set	index	0	1	2	3	4	5	6	7	8
	row/col in unit of ⅛ pel	0	2	4	6	8	16	32	64	128

- Adaptive MV Resolution (AMVR)
  - Signaled adaptive MVD resolution
  - AMVD precision from {8, 4, 2, 1, ½, ¼, ⅛} pixel selected and signaled.

Inter Coding: Joint Motion Vector Coding

- Joint coding of absolute row and column
    - Decode shell_index
    - Decode absolute value of col
- Sign derivation for MVD
    - Sign is predicted from the sum of absolute values row and col.
    - Sign of last non-zero MVD component is equal to the parity of the sum of absolute values of the all mvd components.

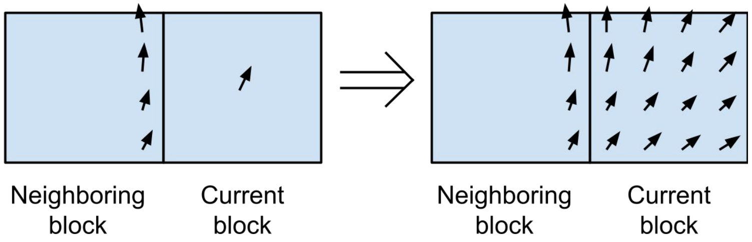
$$shell\_index = |MVD.row| + |MVD.col|$$



Tools Overview

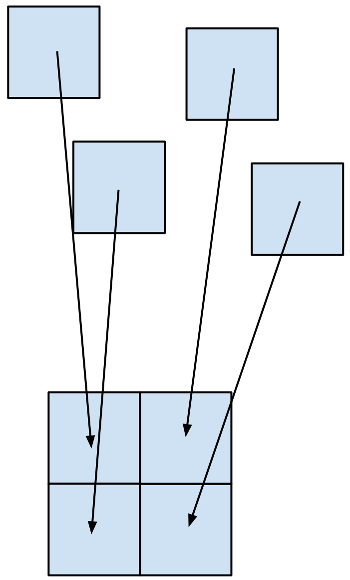
Inter: Extended Warp Prediction

- Non-translation prediction
  - Affine model, up to 6 parameters
- WARP_EXTEND motion mode
  - Keep continuity with neighbour at block edge
  - Match signaled MV at block center
- WARP_DELTA motion mode
  - Take a nearby warp model, and select/signal the delta w.r.t. the nearby warp model
  - Warp reference list (WRL): constructed by adding warp models from
    - Warp model derive from the 3 corner MVs
    - Models from the spatial neighboring blocks
    - Warp parameter bank: circular buffer to store previously coded models in the SB row
    - (If there are still available slots) Global motion and pre-defined models
- WARPMV mode
  - Supports WARP_CAUSAL and WARP_DELTA mode
  - MV is predicted from the warp reference list (instead of DRL)



Inter Coding: New Warp Filter

- AV1 warp filter: Break an affine transformation up into two shears
  - Very efficient, but can only handle small amounts of warping
- New warp filter for stronger warps
  - Split block into 4x4 units
  - Translate each 4x4 unit independently
  - Deblock boundaries between 4x4 units
- Greatly extends the range of warps we can perform



Tools Overview

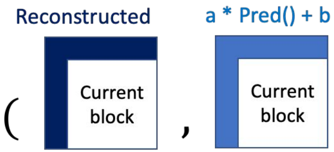
Inter Coding: BAWP and Skip Mode

- Block Adaptive Weighted Prediction (BAWP)
  - Illuminance compensation in the form

    $$ax + b$$

    Parameters ***a*** can be implicitly derived or explicit signaled, parameter ***b*** is implicitly derived
    - Implicit derivation: minimal SSE between reconstructed and predicted L-shape causal neighbors

      $$(a^*, b^*) = \underset{(a,b)}{\operatorname{argmin}} \; SSE(\text{Reconstructed}, \; a * Pred() + b)$$

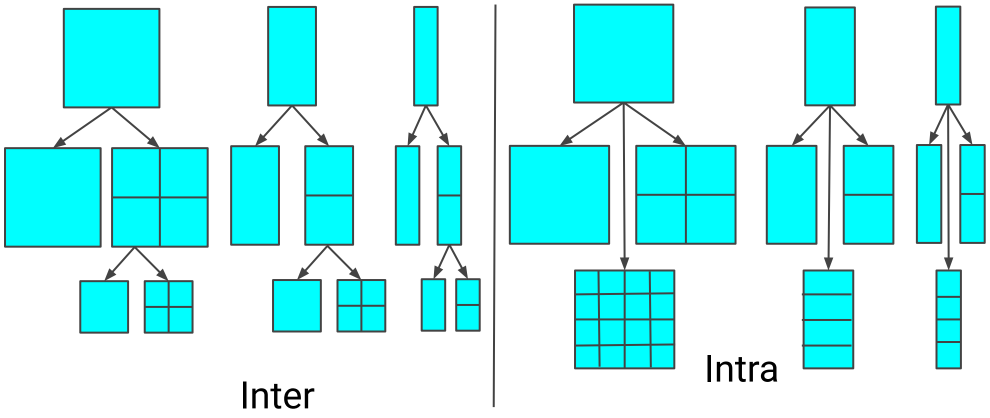
      

    - Explicit signaling: selection from a set of candidate values of ***a*** is signaled
- Skip mode enhancement
  - Allow MV candidate selection (DRL) and residual signaling

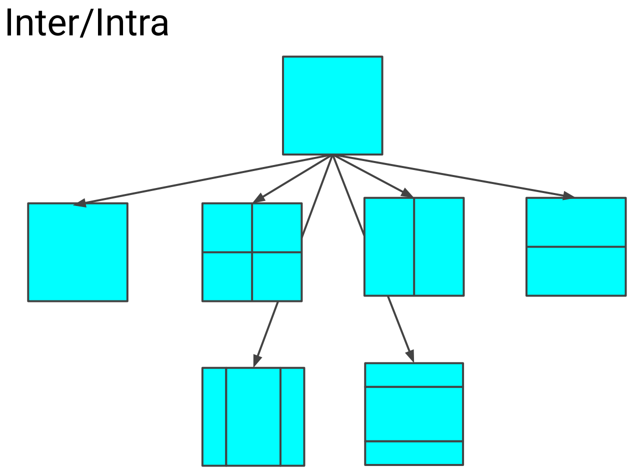
Transform Coding Tools

Tools Overview

Transform Coding: New Transform Partitions



AV1 transform partition



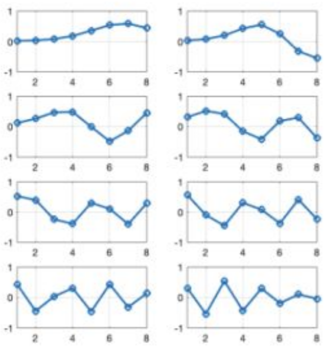
AVM: New transform partition (NTP)

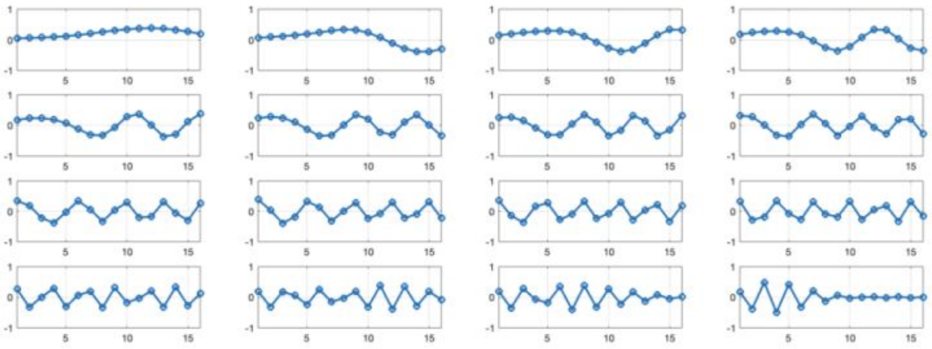
- Removal of recursive scheme used in AV1
- Allowed partitions: PARTITION_NONE, PARTITION_SPLIT, PARTITION_VERT, PARTITION_HORZ, PARTITION_HORZ_M, PARTITION_VERT_M

Tools Overview

Transform Coding: Primary Transforms

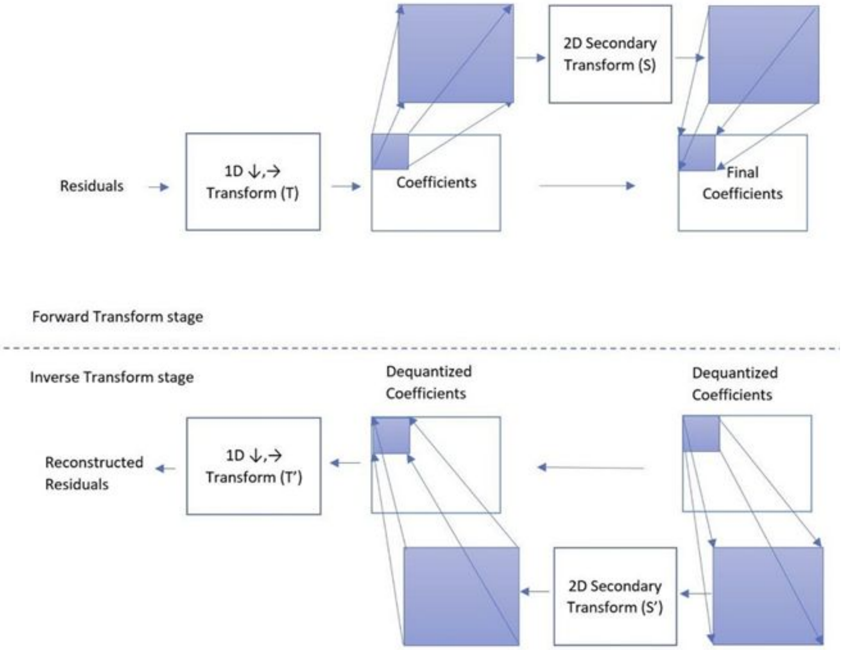
- ADST type replacement for intra residuals
  - Replace 4-point ADST of type DST-VII to DST-IV.
  - Replace 8-point ADST of type DST-IV with a generalized unitary transform.
  - Replace 16-point ADST of type DST-IV to DST-VII.
- ADST type replacement for inter residuals with Data-driven primary transform kernels (DDTX)
  - Used for 8-point and 16-point only.





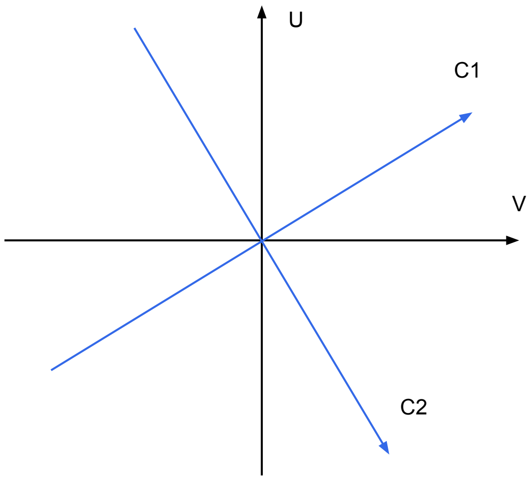
Transform Coding: Secondary Transforms

- Secondary transform on low frequency transform coefficients.
- Enabled when primary transform is
  - DCT_DCT or ADST_ADST (Intra).
  - DCT_DCT (Inter).
- 14 transform sets, each set with 3 offline trained kernels.
- Signaled transform set and kernel index (Intra).
- Only kernel index signaled for inter blocks.



Transform Coding: Cross chroma component transform (CCTX)

- 2D rotations applied to the colocated u and v transform coefficients. (U,V) → (C1,C2).
- 7 rotation angles (0, 30, 45, 60, -30, -45, -60 degrees) for selection.



Entropy Coding Tools

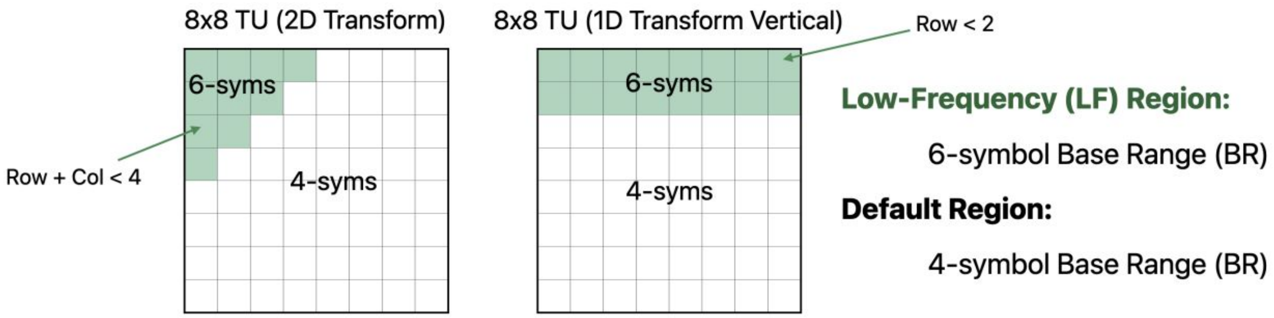
Entropy Coding: Parity Hiding & Forward Skip Coding

- Parity Hiding
  - Luma DC coefficient parity can be implicitly derived.
  - Is applied when there are 3+ non-zero AC coefficients.
  - Parity is obtained from the sum of base range (BR) and low range (LR) AC coefficient levels.
- Forward Skip Coding (FSC)
  - Separate residual coding scheme is applied for prediction residuals using identity transform
    - Forward coefficient scan for BR, LR and sign coding passes.
    - Refined context modeling.
    - Implicit end of block (EOB) derivation.
    - Context coded sign values.

Tools Overview

Entropy Coding: Coefficient Coding Improvements

- Region based coefficient coding.
- 6-ary base range (BR) symbol in LF region.
- 4-ary low range (LR) symbol for both default and LF region.
- Truncated Rice coding of high range (HR) symbol.
- Overall reduction and refinement of contexts for luma and chroma coefficient coding.
- Unify the scan order → always use up-right diagonal scan.



Entropy Coding: Arithmetic Coding Improvements

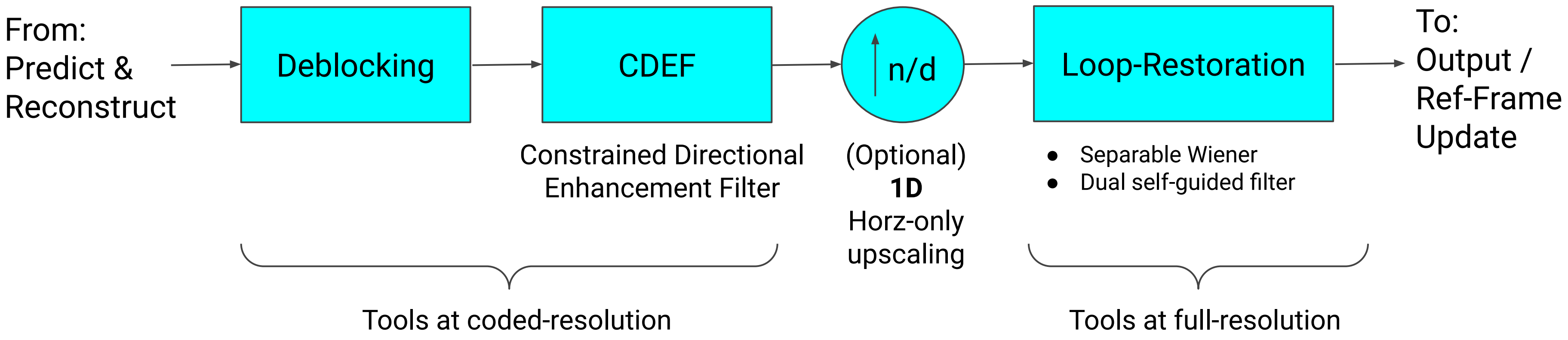
- Uses a multi-symbol arithmetic coder
  - Supports M-ary symbols, where M can range from 2 to 16.

- AVM Improvements
  - Probability Adjustment Rate Adaptation (PARA)
    - Improves symbol probability estimation by flexible adjustment of update rate.
    - Offset is jointly optimized for different symbol groups and symbol frequency.
  - Bypass coding improvement
    - Avoids complex scaling and range normalization between bypass symbols.
    - Allows processing of 8+ consecutive bypass symbols in one cycle, compared with 1 or 2 symbols per cycle currently possible with AV1 hardware implementations.

In-loop Filtering Tools

Tools Overview

In-loop Filtering: AV1 pipeline

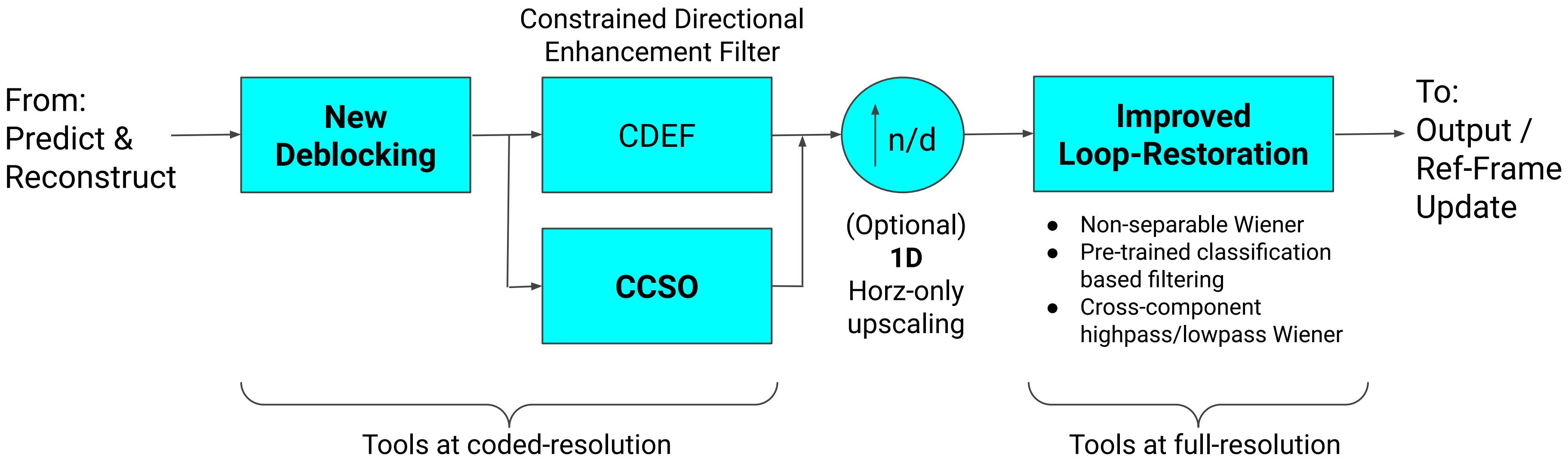


Super-resolution mode:

- Each frame could be coded at a few lower resolution options, then upscaled in-loop with filtering

Tools Overview

In-loop Filtering: Intended AVM pipeline



Super-resolution mode:

- Each frame could be coded at a few lower resolution options, then upscaled in-loop with filtering
- 2D upscaling is under development

Tools Overview

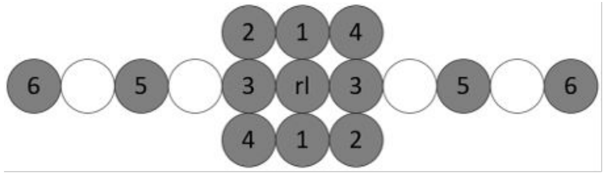
In-loop Filtering: New Deblocking Filter

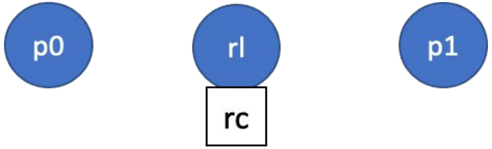
- Several different filters in av1 are replaced with one generalized filter
  - One formula for any filter length, but carefully designed decision logic
- Some decision logics (at very high level)
  - Boundaries are examined based on signal smoothness
    - Decision thresholds are q-dependent
  - Many more details…
- Number of samples modified at each side of block boundary
  - AV1: 1,2,3, or 6 for luma, 1 or 2 for chroma
  - NEW_DF: 1-12 for luma, 1-5 for chroma

In-loop Filtering: Cross-Component Sample Offset (CCSO)

- CCSO
  - Main idea: use luma pixels to refine both luma and chroma reconstruction pixels
  - Nonlinear filtering process applied on all three color components. Steps:
    - d0, d1: quantized luma delta values in {-1, 0, 1}, p0, p1 - locations determined by filter index
    - Get component band index b, where #bands = {1, 2, 4, 8}
    - s = LUT(d0, d1, b)
    - Component output rc' = clip(rc + s)
  - #Bands, quantization step, filter shape and LUT entries are all signaled at the frame level

6 Filter shapes
(locations of p0, p1)





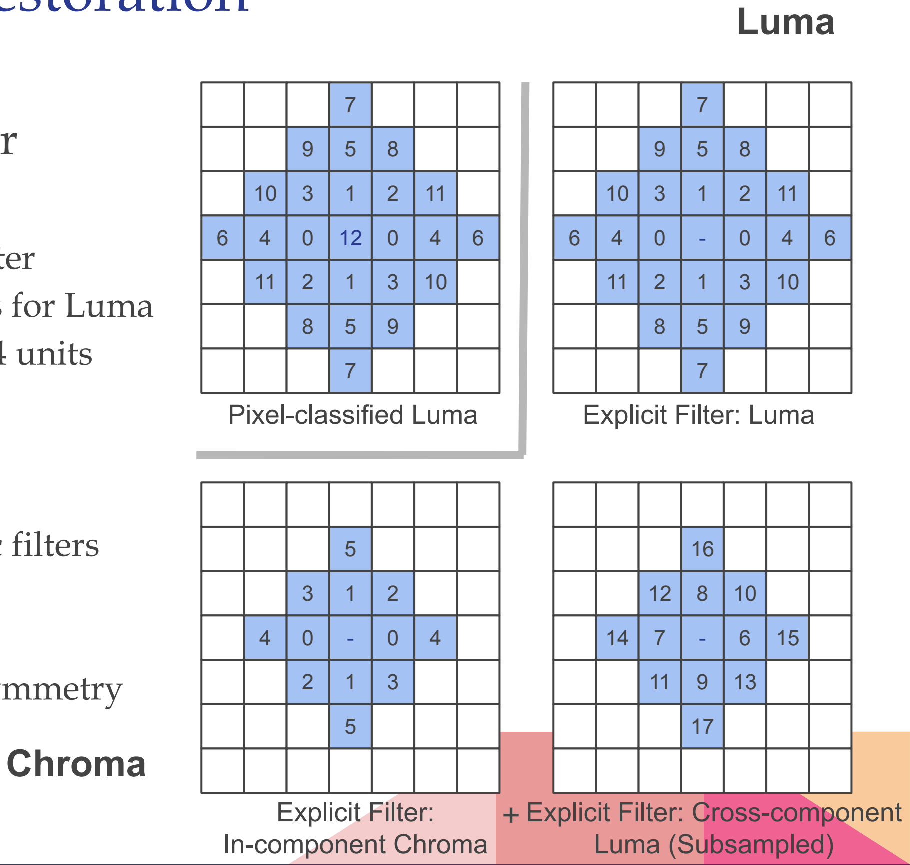
◯: luma
▢: luma/chroma

d0=Quantize(val(p0)-val(rl))
d1=Quantize(val(p1)-val(rl))

Tools Overview

In-loop Filtering: Improved Loop Restoration

- AV1 LR: Separable Wiener + Guided Filter
- New filters for AVM
  - Pixel Classification based pre-trained Wiener filter
    - Pre-trained 13 tap diamond-shaped filters for Luma
    - Gradient based 64-ary classification in 4x4 units
    - No side-info overheads!
  - Explicitly signaled non-separable Wiener filter
    - Explicit signaling of coefficients
    - Luma: diamond-shaped 12 tap symmetric filters
    - Chroma: small diamond shaped
      - In-component + cross-component
      - Cross-component portion has no symmetry constraints



In-loop Filtering: Improved Loop Restoration (cont'd)

- Frame level explicitly-signaled classified Wiener filters
  - Flexible number of classes derived from master 64-ary classifier
  - Explicitly signal filters for each class at frame level
- Syntax Optimization
  - Allows multiple RUs to share the same side information
    - Circular bank of previously used filters can be used as predictor for the next RU
  - Flexible mechanism to turn on/off tools at sequence/frame level

Conclusion

Conclusion

- Development of coding tools for next-gen royalty-free codec from AOM continues as AVM
- Status
  - More than 100 candidate coding tools available
  - ~25.70% (YUV-PSNR 14:1:1) coding gain compared to AV1 with candidate tools; ~27.55% (YUV-PSNR 6:1:1)
  - All tools in AVM-8.0 rigorously vetted for decoder hardware complexity
  - Encoder - very slow right now !
- Plan & future work
  - Extend objective gain to 30% and perceptual coding gain to 40% bitrate reduction over AV1.
  - Optimization and encoder-complexity reduction

Thank You!