Motion Estimation methods

Review and comparison

Motion Prediction Models

Translational Model

the prediction signal for each block is a block of same size MXN from other frames
the prediction block is specified by the translational motion vector and reference frame(s) index.

Affine Model

motion in 3-D is described by affine transformations (a composition of scaling, rotation, sheering and translation).

Translational Model

Translational model maps rectangle to rectangle of same size and it is non-adequate for 3-D motion (e.g. imagine the case when a rectangular object quickly approaching to the camera, its size is increasing). This model works good for small motions:

- Subdivide current frame into blocks.
- Find one displacement vector for each block:

Within a search range, find a “best match” that minimizes an error measure.

In Translational model all pixels with a current block are displaced by same vector from the reference block. It’s like the reference block is displaced (without rotation and scaling) to a new position.

Affine Model

- Displacement of each pixel (x,y) is computed as follows:

$$d_x = a_0 + a_1 x + a_2 x$$
$$d_y = b_0 + b_1 x + b_2 x$$

Subdivide current frame into blocks.

Find 6 parameters (a_0, a_1, a_2, b_0, b_1, b_2) for each block:

With these 6 parameters use the above equation to compute prediction signal.

Taxonomy of Motion Estimation Methods

Pixel Domain Methods

- Matching algorithms
  - Block Matching (**most popular**): full-search, Three Step, diamond etc.
  - Feature Matching: Integral Projection matching, Successive Elimination
- Gradient-based algorithms
  - pel-recursive
  - block-recursive

Frequency Domain Methods

- Phase correlation
- matching in wavelet domain
- matching in DCT domain

Motion estimation Parameters

Search Area

in case of significant (or fast) motion large search area impacts significantly on Motion Estimation effectiveness. On the other hand ME complexity increases.

Sub-pixel mode

Motion is not limited to pixel granularity, therefore sub-pixel prediction (with accuracy up to 1/8 of pixel) is applied

Motion estimation error measures

- SSD (sum squared differences):

$$SSD(d_x, d_y) = \sum_{y=1}^{By} \sum_{x=1}^{Bx} [s(x, y, t) - s'(x - d_x, y - d_y, t - \Delta t)]^2$$

SAD (sum of absolute differences, less complex than SSD and slightly worse):

$$SAD(d_x, d_y) = \sum_{y=1}^{By} \sum_{x=1}^{Bx} | s(x, y, t) - s'(x - d_x, y - d_y, t - \Delta t) |$$

R-D metrics:

$$D_K(d_x, d_y) + \lambda_m \cdot R(d_x, d_y)$$

Where

D_K *is SSD or SAD or other distortion metric and* $R(d_x, d_y)$ is bit-size estimation of residuals.

Block Matching Motion Estimation Parameters

Hierarchical Architecture:

To reduce complexity and/or to pipeline Motion Estimation two hierarchical levels are commonly used:

- First stage: coarse motion estimation (usually on decimated search region)
- Second stage: fine motion estimation tuning around "best" coarse motion vectors obtained in the previous stage.

Speed Up Techniques:

- Early termination – exclude current candidate if its preliminary cost exceeding the minimal cost (already obtained).
- Exclude candidates – not all candidates are checked (e.g. logarithmic search schemas).

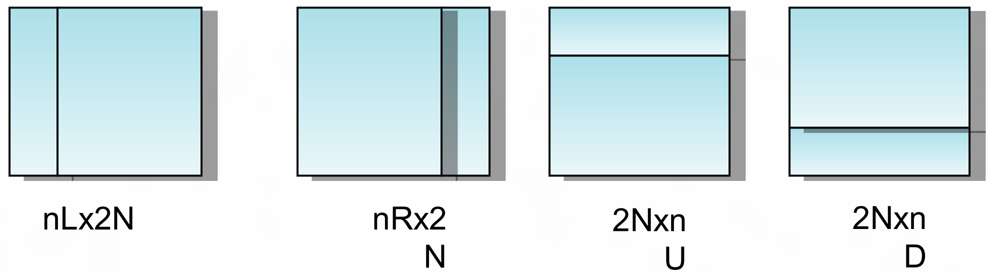
Inter Prediction Comparison: AVC/H.264, HEVC/H.265, VP9, AV1

Features	AVC/H.264	HEVC/H.265	VP9	AV1
Square blocks Only	yes	no	yes	no
Weighted prediction	yes	yes	no	yes
Bi-Prediction	yes	yes	Yes as superframe*	Yes as superframe*
Number of references	Up to 16 (depending on level)	Up to 16 (depending on level)	3	Up to 7
Sub-pixel Precision	¼-pel for luma 1/8-pel for chroma	¼-pel for luma 1/8-pel for chroma	1/8-pel for luma 1/16-pel for chroma	1/8-pel for luma 1/16-pel for chroma

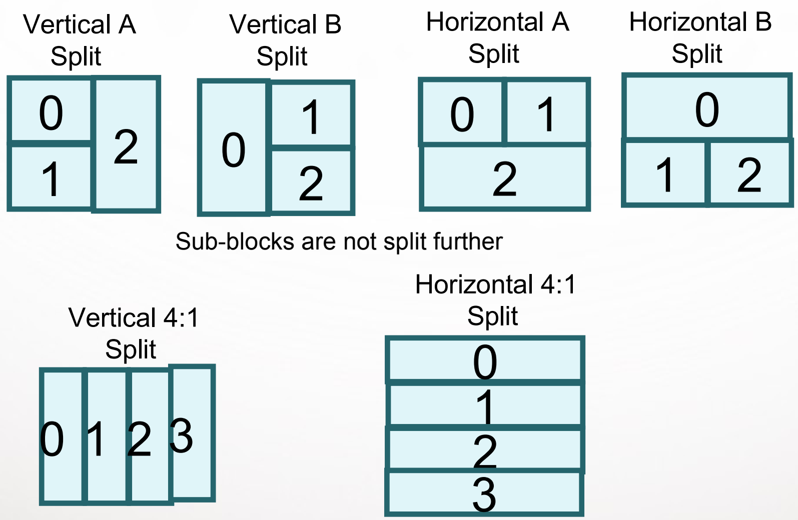
*To avoid patent infringements B-frame is coded as a couple of non-displayable frame plus displayable frame consisting of skip blocks. This pair of frames is called 'superframe'

Rectangular Prediction Blocks in HEVC/H.265 and AV1

- HEVC/H.265



- AV1



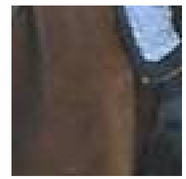
Rectangular Prediction Blocks in HEVC/H.265 and AV1 (cont.)

)Benefits of rectangular partitioning (HEVC

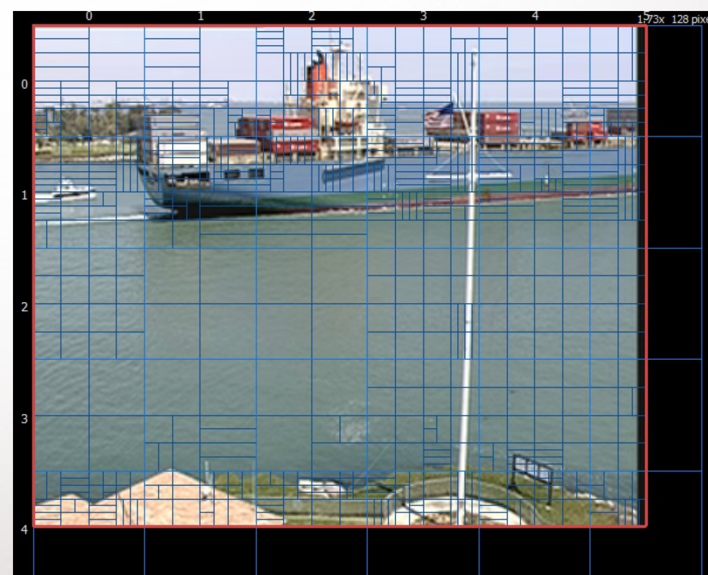

2NxnU


2NxnD


nLx2N


nRx2N

)Benefits of rectangular partitioning (AV1



Sub-pixel Precision in AVC/H.264, HEVC/H.265, VP9 and AV1

- **AVC sub-pel precision**:
- is ¼ for luma and ⅛ for chroma respectively (4:2:0).

The interpolation filters for both luma and chroma are fixed (non-adaptive).

- For luma the interpolation is pipelined and it is executed in two non-balanced serial stages for each direction (horizontal and vertical):
  - ✓ 6-tap filter for half-pels (high complex)
  - ✓ bilinear filter for quarter-pels (low complex)
- For chroma a fixed 4-tap filter is used for all fractional positions (similar to HEVC).

- **HEVC sub-pel precision**: ¼ for luma and ⅛ for chroma respectively

The interpolation filters for generating sub-pel data for both luma and chroma are fixed (non-adaptive):

- For luma pixels a fixed 8-tap filter is applied for both half-pels and quarter-pels.
  The luma interpolation process is pipelined, it consists of two stages: horizontal and vertical
  filtering.
- For chroma a fixed 4-tap filter is used for all fractional positions.

Sub-pixel Precision in H.264, H.265, VP9 & AV1

VP9 sub-pel precision

¼ for luma and ⅛ for chroma respectively (if 4:2:0).

The interpolation filters for generating sub-pel data can be adaptively chosen at frame-level, available filters kernels:

- Normal
- Smooth - slightly smooths or blurs the prediction block
- Sharp - slightly sharpens the prediction block.

- Interpolation filtering is pipelined: firstly a corresponding horizontal filter is used to build up a temporary array, and then at the second stage this array is vertically filtered to obtain the final prediction.

- **Note:** important advantage of HEVC over VP9 is a separation of filters for half and quarter pel (can be realized in stages, friendly for HW).

Sub-pixel Precision in H.264, H.265, VP9 & AV1

AV1 sub-pel precision

Up-to 1/8-pel sub-pel precision for luma (1/8 and 1/16 precision for chroma respectively due to 4:2:0), the precision level is specified at frame level.

There are four interpolation kernels (up to 8 taps), filter can be block-level adaptive:

EIGHTTAP, EIGHTTAP_SMOOTH, EIGHTTAP_SHARP, BILINEAR

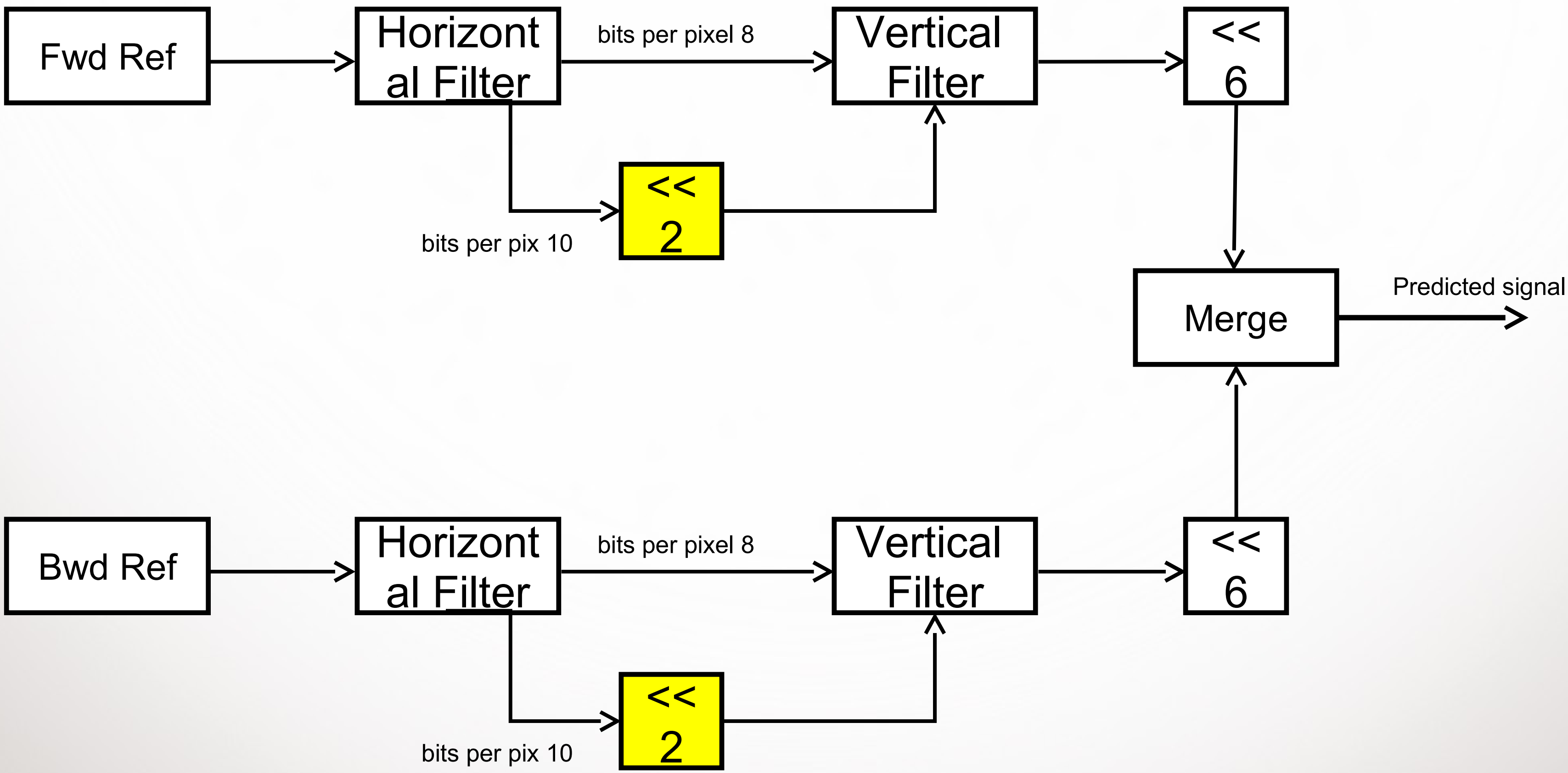
Each filter is separable (i.e. filtering process is pipelined): firstly horizontal filtering is performed and then vertical filtering.

- ✓ Interpolation filter can be fixed within a frame, in such case one of four kernels is selected at frame header.
- ✓ Interpolation filter can be switchable at block-level
- ✓ There is a special mode - **dual filtering**, where kernel for each direction can be different. Justification for dual filtering - signals can possess distinctive statistics in vertical and horizontal directions.

Use Case: HEVC/H.265 Motion Estimation Details

- Variable inter-prediction block sizes – from 8x8 to 64x64, including non-square sizes like 32x16 (actually 4x8 and 8x4 blocks are also permitted with some constraints).
- Chroma block sizes mimic luma, for 4:2:0 case with the scaling factor 1/2 (although for small luma blocks the scaling factor is 1).
- Bi-directional prediction: two prediction blocks from previous and future pictures are mixed (averaged) to produce the final prediction signal (it's a kind of interpolation).
- weighted prediction (e.g. to compensate fading).
- Sub-pixel precision: up to 1/4-th for luma and up to 1/8 for chroma

Weighted Prediction in HEVC/H.265



AV1 Motion Estimation Details

- AV1 supports Global Motion mode which is divided into the following categories:
  - ✓ Translation (panning video)
  - ✓ Rotation
  - ✓ Zoom
  - ✓ Affine (suitable for 3D motion)
- AV1 supports OBMC (Overlapped Block Motion Compensation)
- AV1 supports Warped motion per superblock

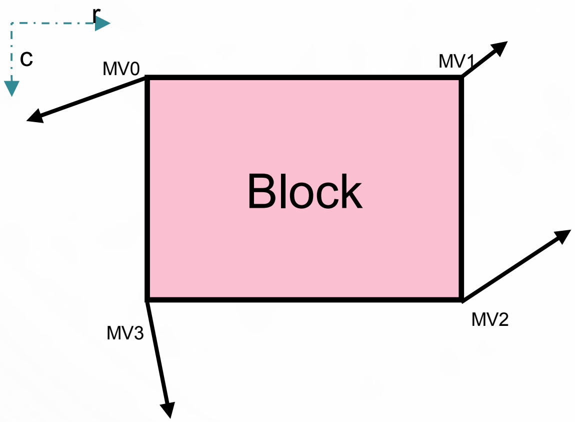
Examples:

In case of translation a global Motion Vector is applied for the whole frame.

In case of Zoom and Rotation Motion Vector is depending on block location

AV1 Motion Estimation Details – General Idea of OBMC

Justification of **OBMC** - MV is most reliable in the center of the block (where prediction errors tend to be smaller than those at the corners). For a block it's better to assign several MVs (its own and nearby blocks) and to blend reference samples:



Mathematical formulation:

Let **L** motion vectors $\{MV\}_{i=1}^{L}$ are assigned to a block. Let denote as **I(s)** - intensity of s-th pixel. Then we search the weight vector $\vec{W}^*$*,where* $\sum_{i=1}^{L} w^*_i = 1$

$$\vec{W}^* = argmin\, E(\, I(s) - \sum_{i=1}^{L} w_i \cdot I(s + MV_i))$$

Illustration: For each pixel I(r,c) in the block (here 'r' and 'c' are normalized coordinates in the range [0..1]) we assign four weights (each is associated with its own corner) as follows:

$$W0[r,c] = (1-r)(1-c)\,,\ \ W1[r,c] = r(1-c),\ \mathrm{W3[r,c]} = (1-\mathrm{r})\mathrm{c},\ \ \mathrm{W2[r,c]} = \mathrm{r} \cdot c$$

Predicted pixel $P[r,c] = W_0 \cdot P_{MV0}\,[r,c] + W_1 \cdot P_{MV1}\,[r,c] + W_2 \cdot P_{MV2}\,[r,c] + W_3 \cdot P_{MV3}\,[r,c]$

OBMC is observed as reducing blockiness artifacts.

AV1 Motion Estimation Details – Technical Details of OBMC

In AV1 OBMC predicted block is associated with a single vector MV0 corresponding to the block's center while corner MVs are taken from causal (already decoded) neighbors.

Blending is executed in two separable stages: firstly according to vertical direction and then according to horizontal direction (the filter coefficients are pre-defined in the AV1 spec.)