Overview of inter prediction and the inter evolution of ivc in recent years

Zhuoyuan Li

invited partner: Yao Li

2022.10.21

Content

1 Brief introduction of the inter prediction process (10 min)

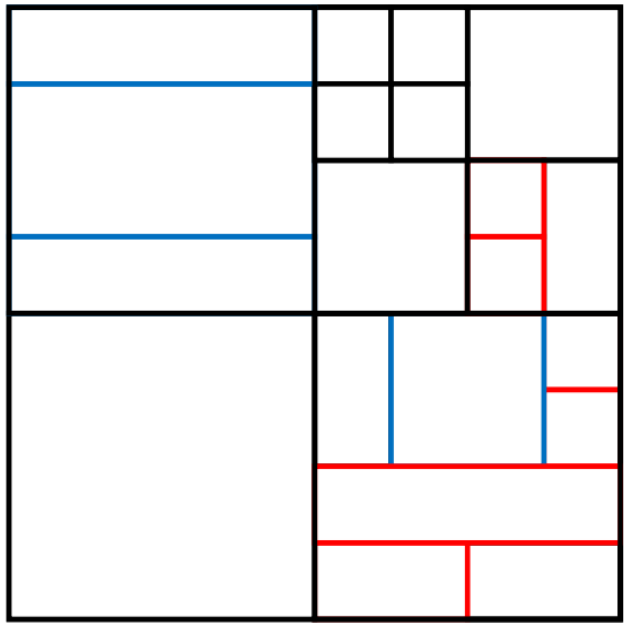
2 Different ideology in inter prediction (10 min)

3 Explain typical related papers (60 min)

4 Question (10 min)

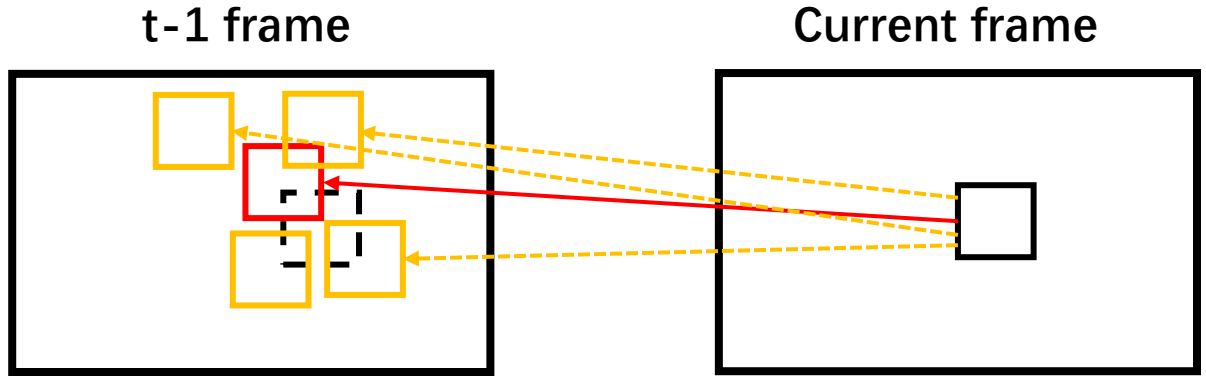
introduction

Block Partition



Block-based warp

Per Block

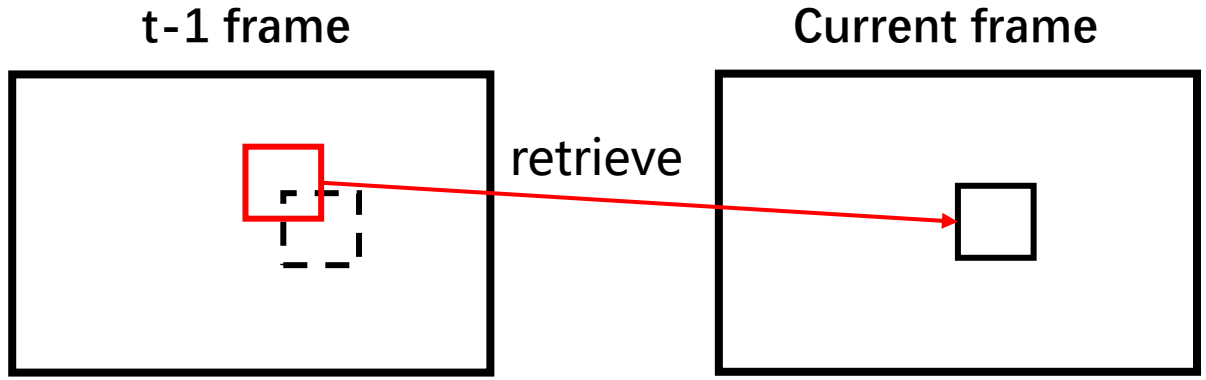


Motion Estimation

Warp step: 1 Motion Estimation (ME) for Encoder

2 Motion Compensation (MC) for codec

ME for optimal search MC for retrieve prediction



Motion Compensation

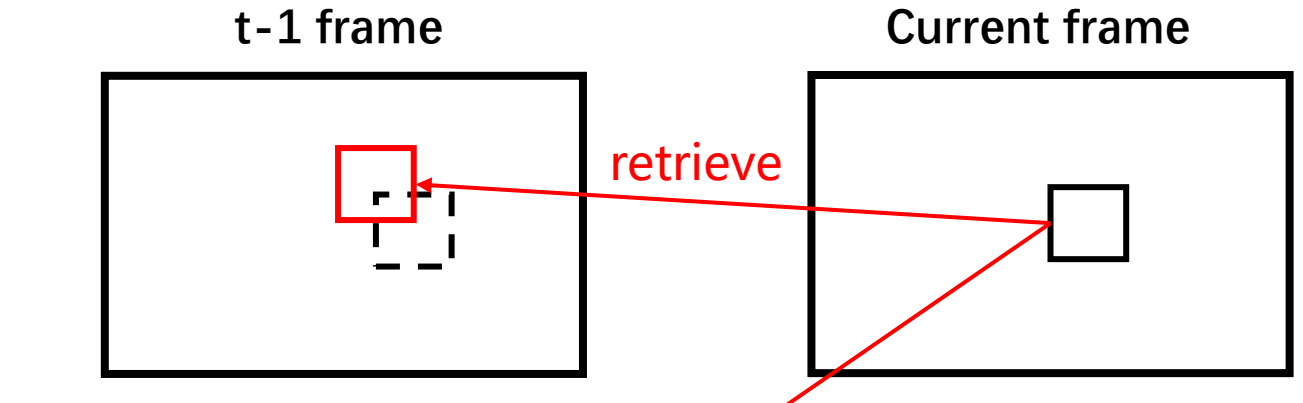
Basic tool for Motion Compensation

Two step: 1 full pixel

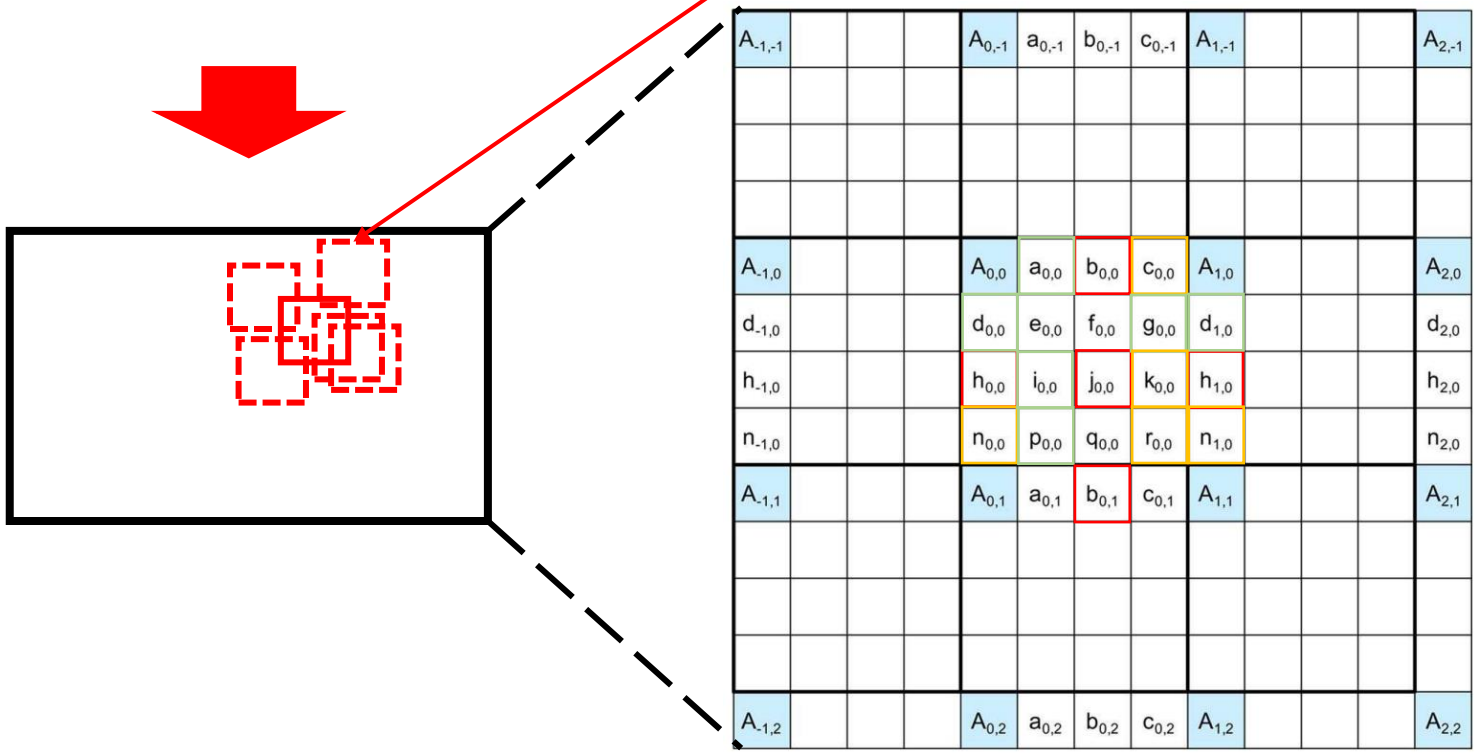
2 Fractional pixel (1/2, 1/4)

Fractional pixel reason: Due to the inherent spatial discretization of digital video, block translation may not happen to be aligned with pixels.

Finite impulse response (FIR) filters are used for luma and chroma interpolation in HEVC and VVC. The coefficients of the FIR filters are designed using a Fourier decomposition of the discrete cosine transform (DCTIF). Three kinds of tap are supported: 8-tap, 4-tap.



Full pixel Motion Compensation

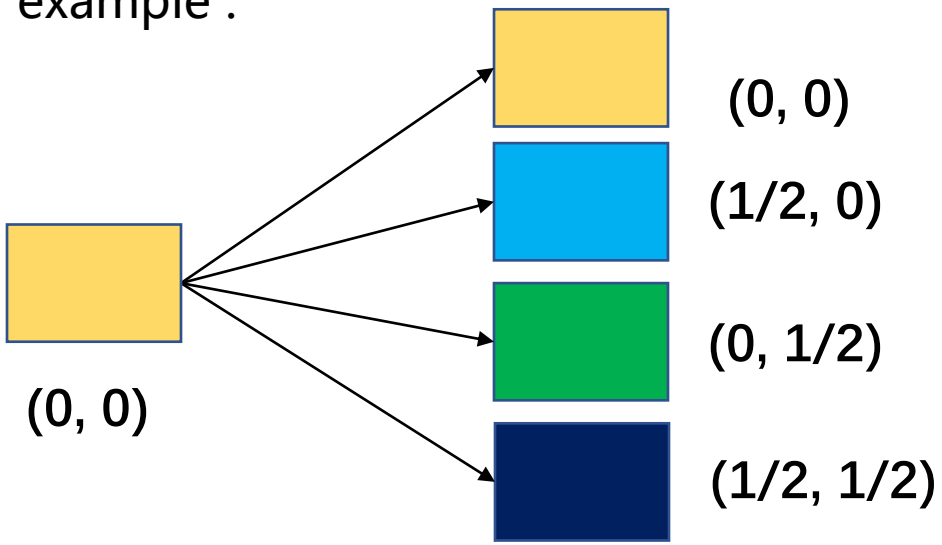


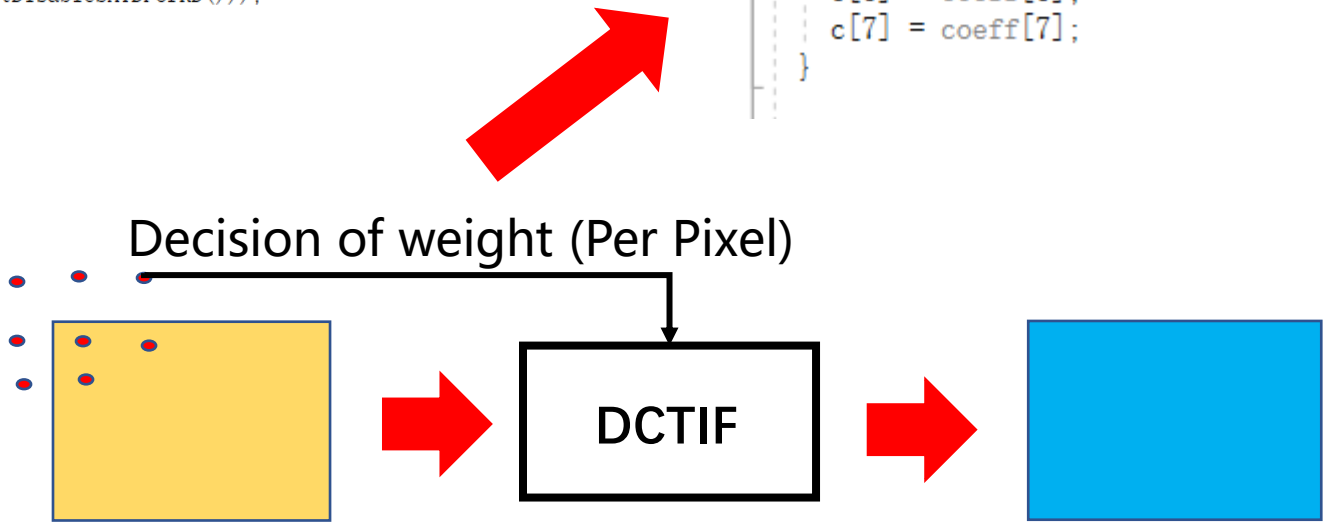
Fractional pixel Motion Compensation

```
//  Half-pel refinement
m_pcRdCost->setCostScale(1);
xExtDIFUpSamplingH(&cPatternRoi, cStruct.useAltHpelIf);

rcMvHalf = rcMvInt;   rcMvHalf <<= 1;    // for mv-cost
Mv baseRefMv(0, 0);
ruiCost = xPatternRefinement(cStruct.pcPatternKey, baseRefMv, 2, rcMvHalf, (!pu.cs->slice->getDisableSATDForRD()));

//  quarter-pel refinement
if (cStruct.imvShift == IMV_OFF)
{
  m_pcRdCost->setCostScale(0);
  xExtDIFUpSamplingQ(&cPatternRoi, rcMvHalf);
  baseRefMv = rcMvHalf;
  baseRefMv <<= 1;

  rcMvQter = rcMvInt;
  rcMvQter <<= 1;   // for mv-cost
  rcMvQter += rcMvHalf;
  rcMvQter <<= 1;
  ruiCost = xPatternRefinement(cStruct.pcPatternKey, baseRefMv, 1, rcMvQter, (!pu.cs->slice->getDisableSATDForRD()));
}
```

```
Pel c[8];
c[0] = coeff[0];
c[1] = coeff[1];
if( N >= 4 )
{
  c[2] = coeff[2];
  c[3] = coeff[3];
}
if( N >= 6 )
{
  c[4] = coeff[4];
  c[5] = coeff[5];
}
if( N == 8 )
{
  c[6] = coeff[6];
  c[7] = coeff[7];
}
```

For example :

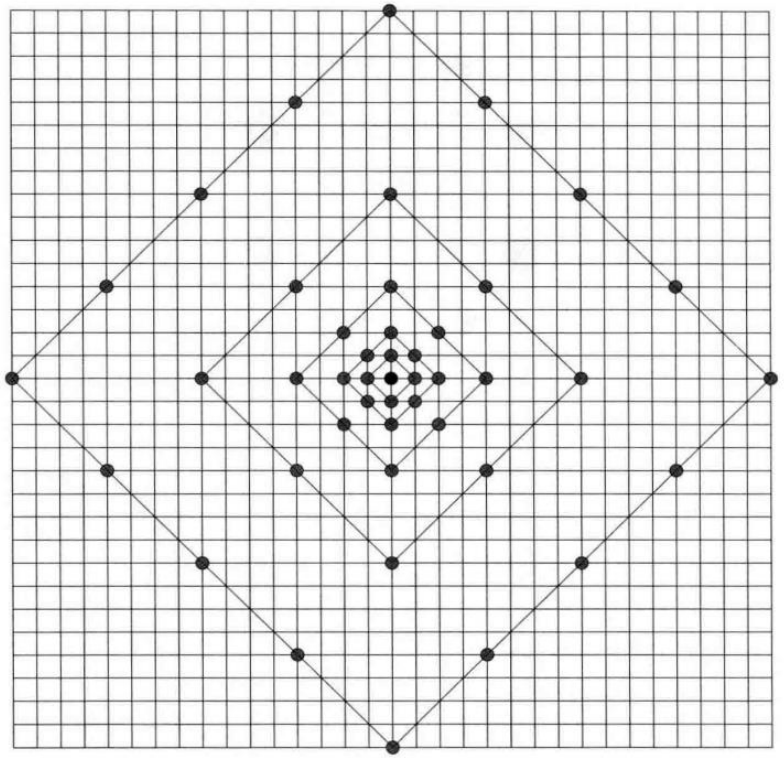


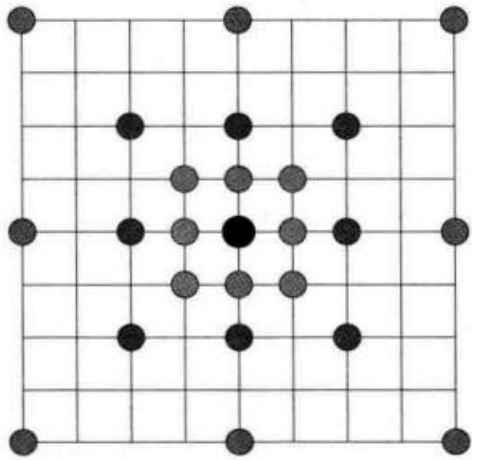


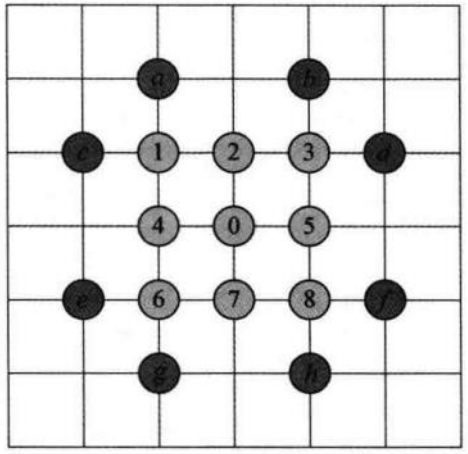
Basic tool for Motion Estimation

- Search range
- Full search guarantees the best MV (reference block), but leads to unfordable computation consumptions
- Fast search algorithms provide a trade off
  - log search, three-step search, **TZSearch**

- **TZSearch**
  - ① 使用MVP作为搜索起点
  - ② 从步长1开始，按右图的菱形模板或正方形模板进行搜索，比较各点对应MV的RDCost；步长按2的整数次幂增长
    - 若得到的最优点的步长为1，则还需在该点进行两点搜索
    - 若得到的最优点的步长大于某一阈值，则还需在该点附近进行全搜索
  - ③ 以②得到的最优点为起点，重复②，进行细化搜索
    - ➢ 相邻两次细化搜索结果相同时算法停止







Basic tool for Motion Estimation

Full pixel to fractional pixel: More accurate MV and prediction

- Interpolation is required to obtain the reference pixel values at fractional pixel position
- Supported motion vector resolution:
  - Regular MV: 4pel, 1pel, 1/2pel, 1/4pel
  - Affine MV: 1/16pel, 1/4pel, 1 pel
- FIR

Merge: Remove the redundancy of MVs of adjacent blocks

- Basic idea: Deduce a list of probable MVs using decoded adjacent blocks' motion information (no bits transmitted), then signal a index to indicate which one is used
- MVP candidates setup
  - Spatial, Temporal, Previous coded …

Inter prediction math paradigm

In a video codec, the encoder is now responsible for searching the best matching motion vector $\boldsymbol{t}^*$ for each block that leads to the best possible quality at the lowest possible rate in a process called rate-distortion optimization [23]. A motion compensated or predicted image $\boldsymbol{I}_{\text{pred}} \in \mathbb{R}^{U \times V}$ can then be obtained both at the encoder and at the decoder by extracting the motion compensated pixel values from the reference image as

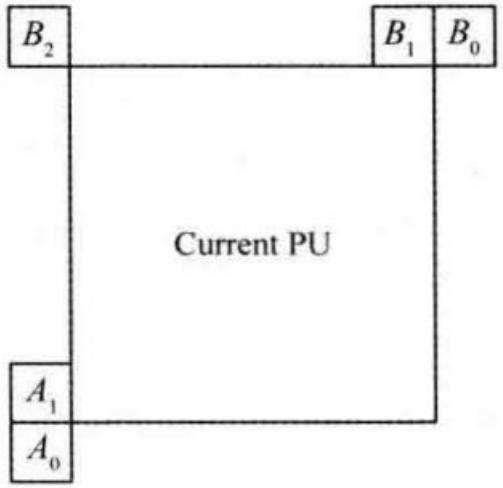
$$\boldsymbol{I}_{\text{pred}}(\boldsymbol{p}) = \boldsymbol{I}_{\text{ref}}(\boldsymbol{m}_{\text{t}}(\boldsymbol{p}, \boldsymbol{t}^*)) \quad \forall\, \boldsymbol{p} \in \mathcal{B} \tag{3}$$

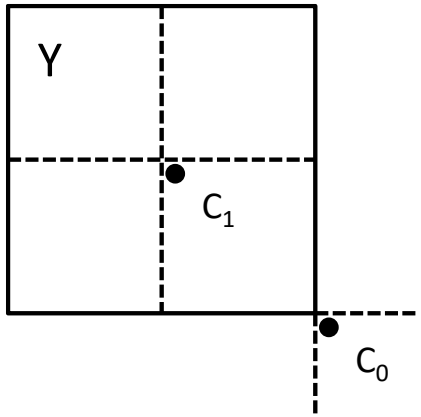
for all blocks in the image, where $\mathcal{B}$ denotes the set of pixel coordinates within the regarded block $\boldsymbol{B}_{\text{pred}}$, and $\boldsymbol{I}_{\text{ref}} \in \mathbb{R}^{U \times V}$ describes the reference image. Thereby, $\boldsymbol{I}(\boldsymbol{p})$ yields the pixel value of image $\boldsymbol{I}$ at pixel coordinate $\boldsymbol{p}$. Internally, a suitable interpolation method is required in order to access pixel values at fractional pixel positions.

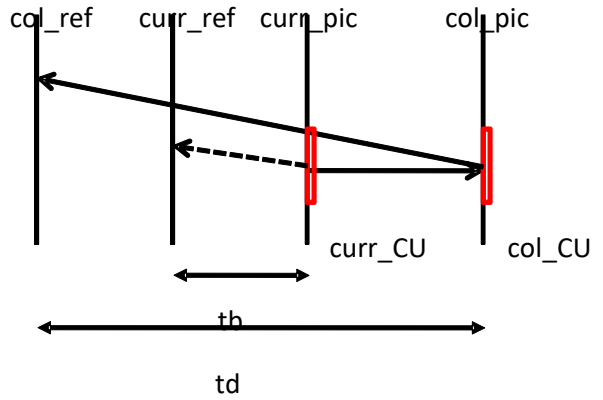
Basic tool for Motion Estimation

Merge: Remove the redundancy of MVs of adjacent blocks

- MVP candidates setup (6 MVPs)
  1. Spatial candidates
     - B1->A1->B0->A0->B2
     - 4 MVPs at most
  2. Temporal candidates
     - C0->C1 (in ColPic), to get more info form bottom right corner
     - Temporal scaling is required
  3. HMVP: Previous coded CUs
  4. Pair-wise average candidates
  5. 0 MV
- Skip
  - Skip the residual coding, only for merge mode
- MMVD: Merge mode with MVD
  - MVD indicated simply by direction index and offset index







Inter prediction math paradigm

a crucial component of any modern hybrid video codec, where the term hybrid refers to a combination of predictive and transform-based coding. Typically, the current frame $\boldsymbol{I}_{\text{cur}} \in \mathbb{R}^{U \times V}$ of size $U \times V$ pixels to be coded is subdivided into individual blocks $\boldsymbol{B}_{\text{cur}} \in \mathbb{R}^{M \times N}$ of size $M \times N$ pixels and each block is coded individually. In a first step, a prediction $\boldsymbol{B}_{\text{pred}} \in \mathbb{R}^{M \times N}$ is formed for each block based on its causal spatial and temporal neighborhood. Most video codecs allow either intra (spatial) or inter (temporal) prediction for a given block.

In a second step, the residual signal $\boldsymbol{B}_{\text{res}} \in \mathbb{R}^{M \times N}$ between the predicted block $\boldsymbol{B}_{\text{pred}}$ and the actual block $\boldsymbol{B}_{\text{cur}}$

$$\boldsymbol{B}_{\text{res}} = \boldsymbol{B}_{\text{cur}} - \boldsymbol{B}_{\text{pred}}. \tag{1}$$

is converted to a transform domain and the resulting signal is quantized and entropy coded [1], [2], [4].

At the decoder, the prediction is formed analog to the prediction procedure at the encoder, before the decoded residual is added to the prediction yielding the reconstructed block $\widehat{\boldsymbol{B}}_{\text{cur}} \in \mathbb{R}^{M \times N}$. To ensure that both the encoder and the decoder are able to arrive at the same prediction, additional side information is signaled to control the prediction procedure. As such, a flag indicates whether intra or inter prediction is used and further control mechanisms specify additional prediction information. In traditional inter prediction, this prediction information includes motion information that is shared by all pixels within the block. The precise motion information that needs to be signaled depends on the applied motion model.

Using the translational motion model

$$\boldsymbol{m}_{\text{t}}(\boldsymbol{p}, \boldsymbol{t}) = \boldsymbol{p} + \boldsymbol{t}, \tag{2}$$

the motion is described by the motion vector $\boldsymbol{t} = (\Delta u, \Delta v)^T \in \mathbb{R}^2$ that shifts the pixel coordinate $\boldsymbol{p} = (u, v)^T \in \mathbb{R}^2$ by Δu pixels in horizontal u-direction and Δv pixels in vertical v-direction.

Different ideology about Paper

1 Motion modeling (L. Li & Y. Li)

2 Geometric Partitioning (Me)

3 360-Degree coding (L. Li， Y.F Wang)

4 Fractional enhancement & fast algorithm (N. Yan)

5 NN-based enhancement (S. Huo)

6 Reference Generation & Clip & Background (S. Huo, C.Y Ma, F.D Chen)

7 Multi-hypothesis Prediction

8 Motion Vector Refinement

9 Texture Synthesis (K. Yang)

10 Energy-Aware Quality Optimization

11 Template Matching

Different ideology about Paper

1 Motion modeling

- An Efficient Four-Parameter Affine Motion Model for Video Coding** (Li Li)
- Global Homography Motion Compensation for Versatile Video Coding** (Yao Li)

2 Geometric Partitioning

- Geometric Partitioning Mode in Versatile Video Coding** (Han Gao, Tencent America)
- Object Segmentation-Assisted Inter Prediction for Video Coding** (Zhuoyuan Li)

3 360-Degree coding

- Motion-Plane-Adaptive Inter Prediction in 360-Degree Video Coding**(Andre Kaup)

4 Fractional enhancement

- Cnn-based invertible half-pixel interpolation filter for video coding (Ning Yan)
- Invertibility-driven interpolation filter for video coding (Ning Yan)

6 NN-based enhancement

- Cnn-based motion compensation refinement for video coding** (Shuai Huo)
- Deep Affine Motion Compensation Network for Inter Prediction in VVC** (Dengchao Jin, TJU)

6 Reference Generation

- Deep Network-Based Frame Extrapolation With Reference Frame Alignment** (Huo Shuai)

7 Multi-hypothesis Prediction

- Multi-Hypothesis Prediction for Video Coding (Zhao Wang, PKU)

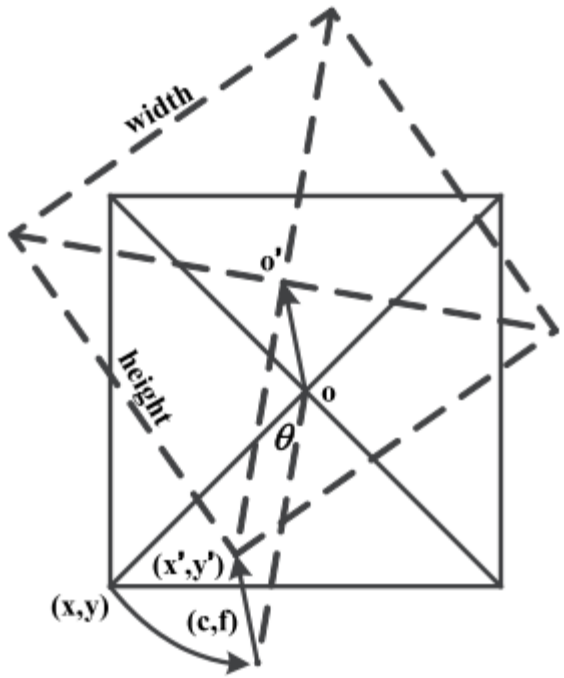
8 Energy-Aware Quality Optimization

- Optimized Decoding-Energy-Aware Encoding in Practical VVC Implementations (A. Kaup, FAU)

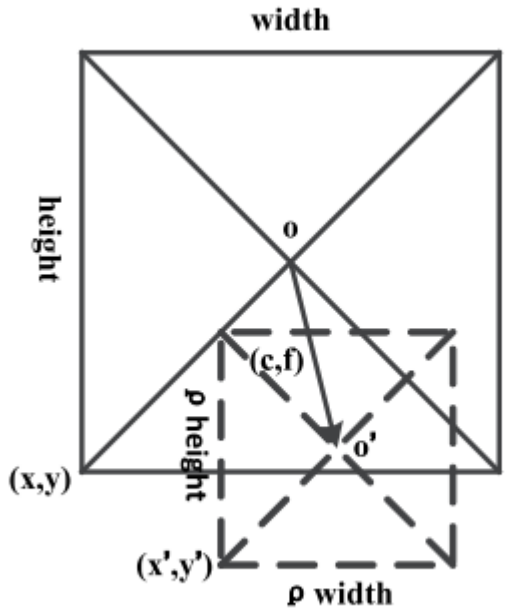
Motion modeling

An Efficient Four-Parameter Affine Motion Model for Video Coding

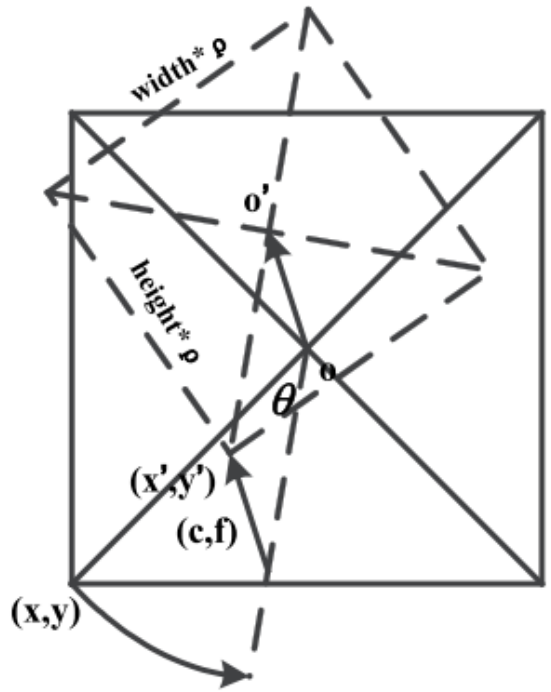
Li Li, Houqiang Li, *Senior Member, IEEE*, Dong Liu, *Member, IEEE*, Haitao Yang, Sixin Lin, Huanbang Chen, and Feng Wu, *Fellow, IEEE*



(a) rotation + translation



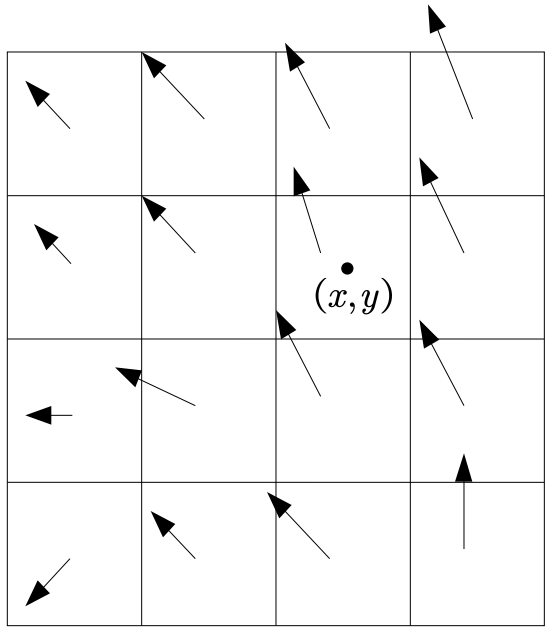
(b) zooming + translation



(c) rotation + zooming + translation

Affine Motion Model

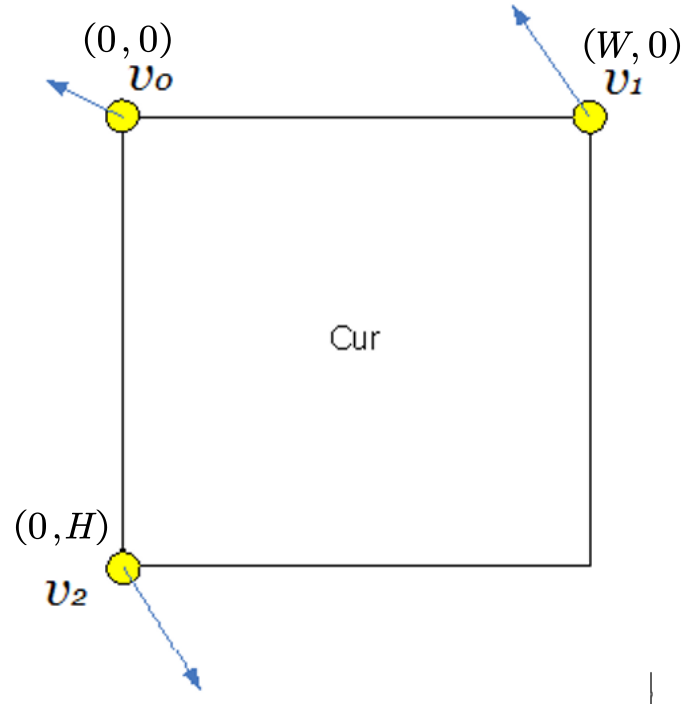
- Affine函数：最高次数为1的多项式函数
- Affine运动模型：映射关系可以由关于坐标(x, y)的Affine函数表示



$$\begin{cases} x' = ax + by + c \\ y' = dx + ey + g \end{cases}$$

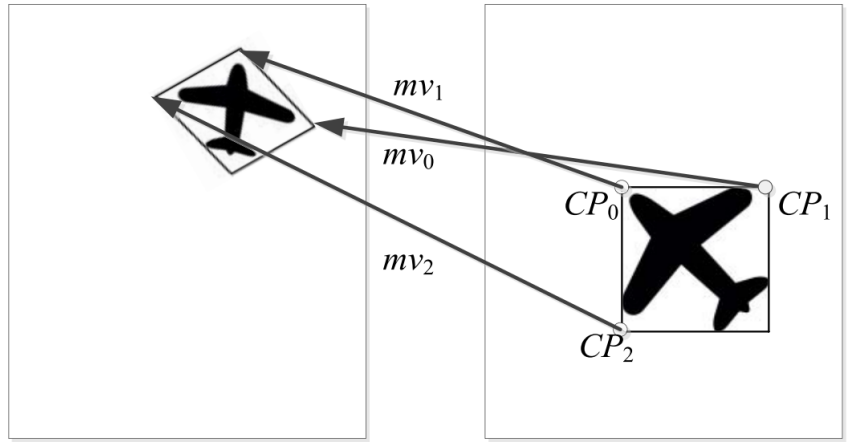
$$\begin{cases} mv_x = ax + by + c \\ mv_y = dx + ey + g \end{cases}$$

代入3个顶点的MV



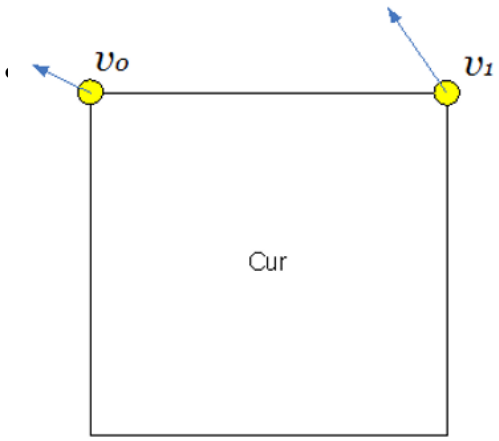
$$\begin{cases} mv_x = \dfrac{mv_{1x} - mv_{0x}}{W} x + \dfrac{mv_{2x} - mv_{0x}}{H} y + mv_{0x} \\ mv_y = \dfrac{mv_{1y} - mv_{0y}}{W} x + \dfrac{mv_{2y} - mv_{0y}}{H} y + mv_{0y} \end{cases}$$

Motion Model



- Affine可描述的运动类型
  - translation, zooming, rotation, shear mapping
- 当仅考虑translation, zooming, rotation时

$$\begin{cases} x' = \rho\cos\theta \cdot x + \rho\sin\theta \cdot y + c \\ y' = -\rho\sin\theta \cdot x + \rho\cos\theta \cdot y + f \end{cases} \quad \begin{cases} MV^{h}_{(x,y)} = x' - x = ax + by + c \\ MV^{v}_{(x,y)} = y' - y = -bx + ay + f \end{cases}$$



$$\begin{cases} mv_x = \dfrac{mv_{1x} - mv_{0x}}{W}x + \dfrac{mv_{1y} - mv_{0y}}{W}y + mv_{0x} \\ mv_y = \dfrac{mv_{1y} - mv_{0y}}{W}x + \dfrac{mv_{1y} - mv_{0x}}{W}y + mv_{0y} \end{cases}$$

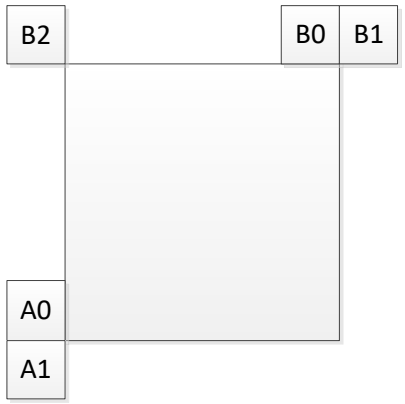
mv_0，mv_1 等被称为CPMV(Control Point Motion Vector)

Motion Vector Coding

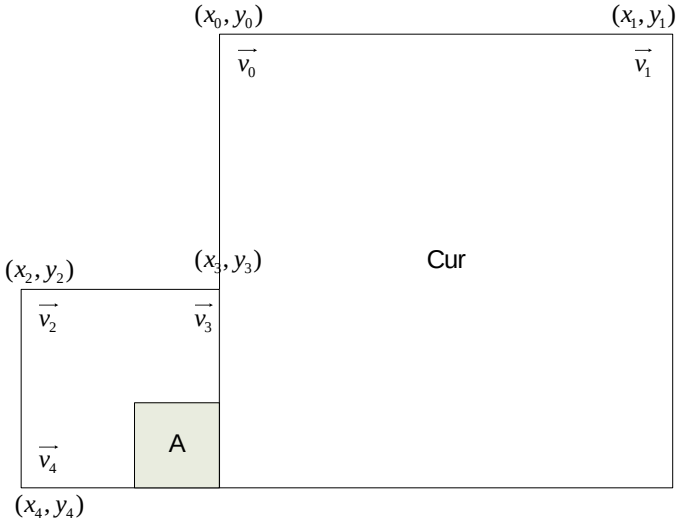
1. Merge

➢MVP列表的建立（5项MVP）

- 1.1 空域相邻的 Affine模式CU CPMV继承候选



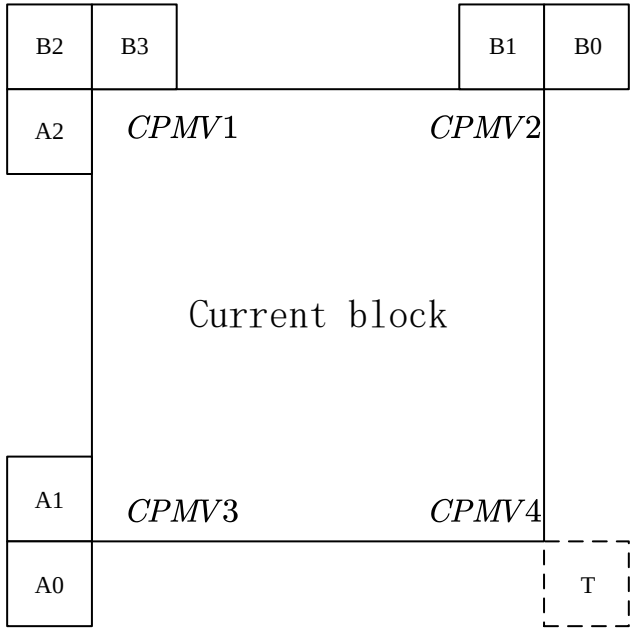
- A0->A1
- B0->B1-B2



Motion Vector Coding

1. Merge

- MVP列表的建立
  - 1.2 空/时域相邻的 普通CU 平移MV构造候选



    - CPMV1 ~ CPMV4各自进行merge
      - CPMV1: B2->B3->A2
      - CPMV2: B1->B0
      - CPMV3: A1->A0
      - CPMV4: 从前面帧相应位置进行merge

表 5.7 候选 CPMVP

索 引	左上控制点 MV	右上控制点 MV	左下控制点 MV
1	CPMV1	CPMV2	CPMV3
2	CPMV1	CPMV2	CPMV4+CPMV1−CPMV2
3	CPMV1	CPMV4+CPMV1−CPMV3	CPMV3
4	CPMV2+CPMV3−CPMV4	CPMV2	CPMV3
5	CPMV1	CPMV2	—
6	CPMV1	f(CPMV1, CPMV3)	—

  - 1.3 零值MV填充

Motion Vector Coding

1. **AMVP:** MVP + MVD

➢MVP列表的建立 （2项MVP）

- 2.1 空域相邻的 Affine模式CU CPMV继承候选

  与merge基本一致

- 2.2 空域相邻的 普通CU 平移MV构造候选

  与merge基本一致，但不再使用时域相邻CU进行merge得到CPMV4

- 2.3 空 / 时域相邻的 普通CU 平移MV填充

  使用 {CPMV3->CPMV2->CPMV1->时域预测MVP} 中的1个来填充所有CPMV（得到了一个仅有平移运动的Affine模型）

- 2.4 零MV填充

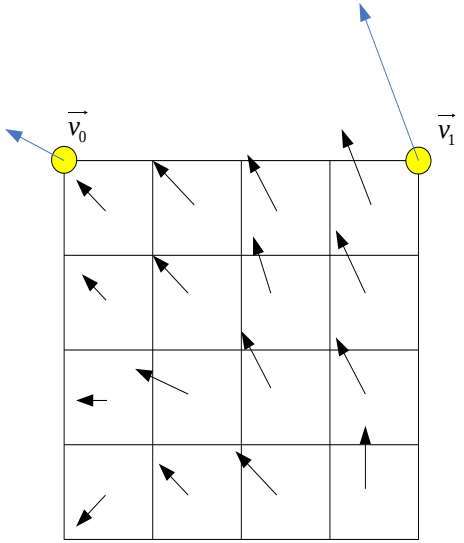
➢MVD的编码

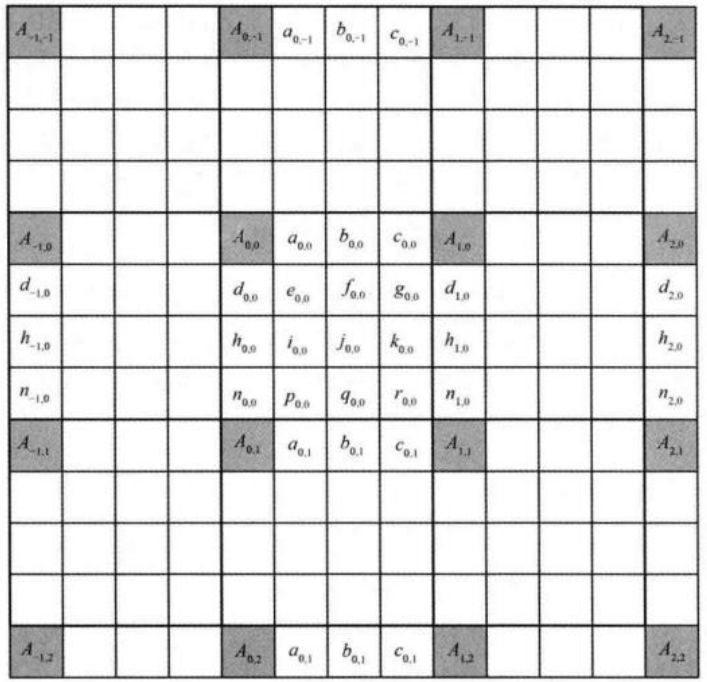
- 整像素、1/4像素、1/16像素精度

Motion Compensation

- 基于4x4子块进行运动补偿
  - 复杂度与补偿精度间的trade off

➢分像素位置插值



- 

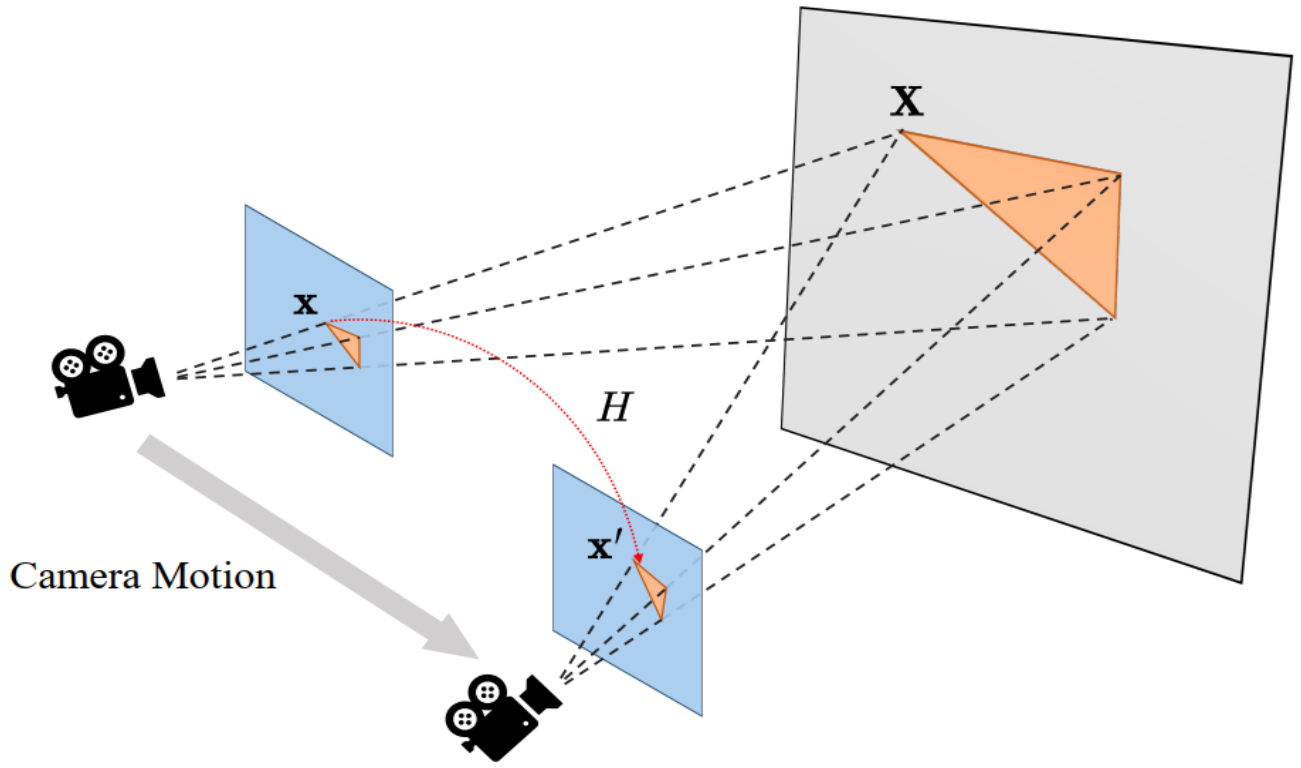
(¼像素精度)

MV	Tap filter					
0	0	0	64	0	0	0
1/16	1	-3	63	4	-2	1
1/8	1	-5	62	8	-3	1
3/16	2	-8	60	13	-4	1
1/4	3	-10	58	17	-5	1
5/16	3	-11	52	26	-8	2
3/8	2	-9	47	31	-10	3
7/16	3	-11	45	34	-10	3
1/2	3	-11	40	40	-11	3
9/16	3	-10	34	45	-11	3
5/8	3	-10	31	47	-9	2
11/16	2	-8	26	52	-11	3
6/8	1	-5	17	58	-10	3
13/16	1	-4	13	60	-8	2
7/8	1	-3	8	62	-5	1
15/16	1	-2	4	63	-3	1

Motion modeling

Global Homography Motion Compensation for Versatile Video Coding

Yao Li, Zhuoyuan Li, Li Li*, Dong Liu, Houqiang Li



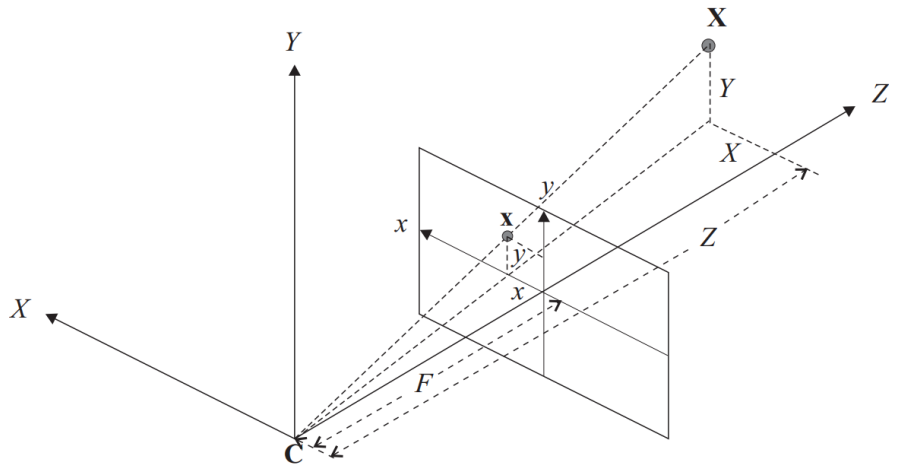
Motion Model

- 摄相机的刚体运动模型

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

- 摄像机的透视投影成像模型

$$x = F\frac{X}{Z},\ y = F\frac{Y}{Z}$$



透视投影成像中的相似三角形关系

- 任意相机运动下像 **x** 坐标的变换关系

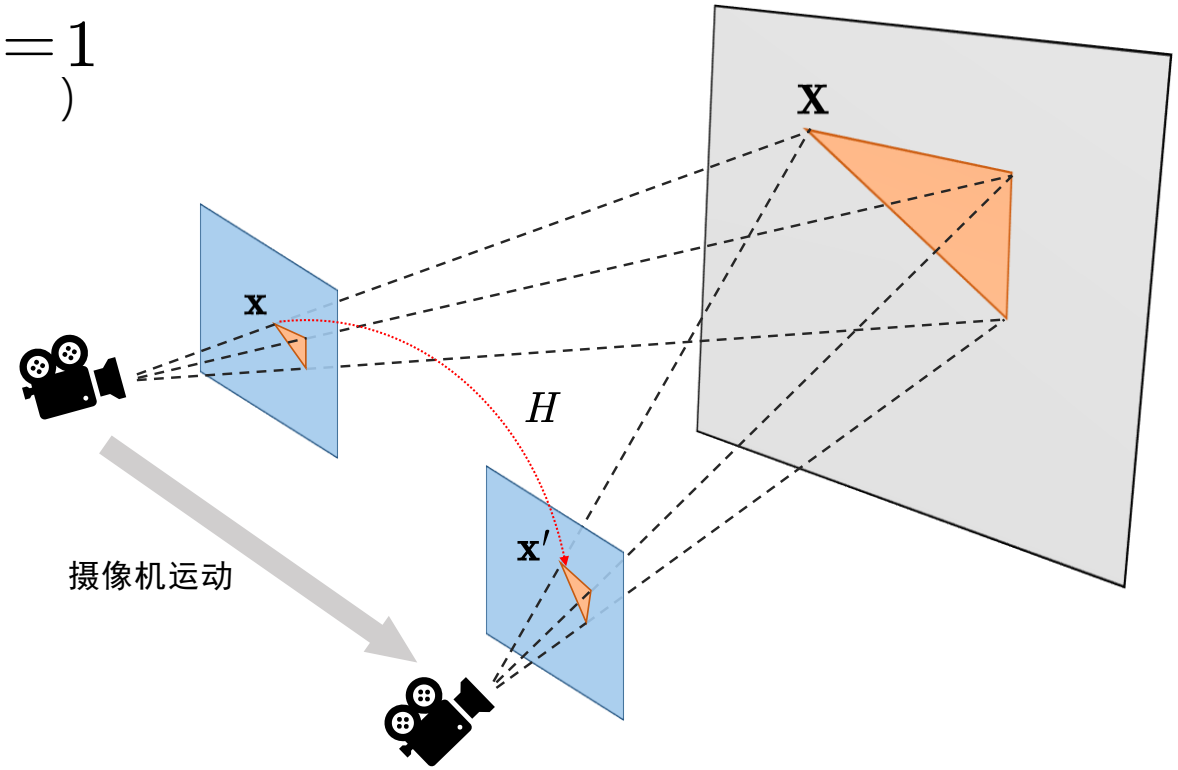
$$x' = F\frac{(r_1 x + r_2 y + r_3 F)Z + T_x F}{(r_7 x + r_8 y + r_9 F)Z + T_z F},\ y' = F\frac{(r_4 x + r_5 y + r_6 F)Z + T_y F}{(r_7 x + r_8 y + r_9 F)Z + T_z F}$$

Motion Model

- 对于场景中的平面物体（$aX+bY+cZ=1$）

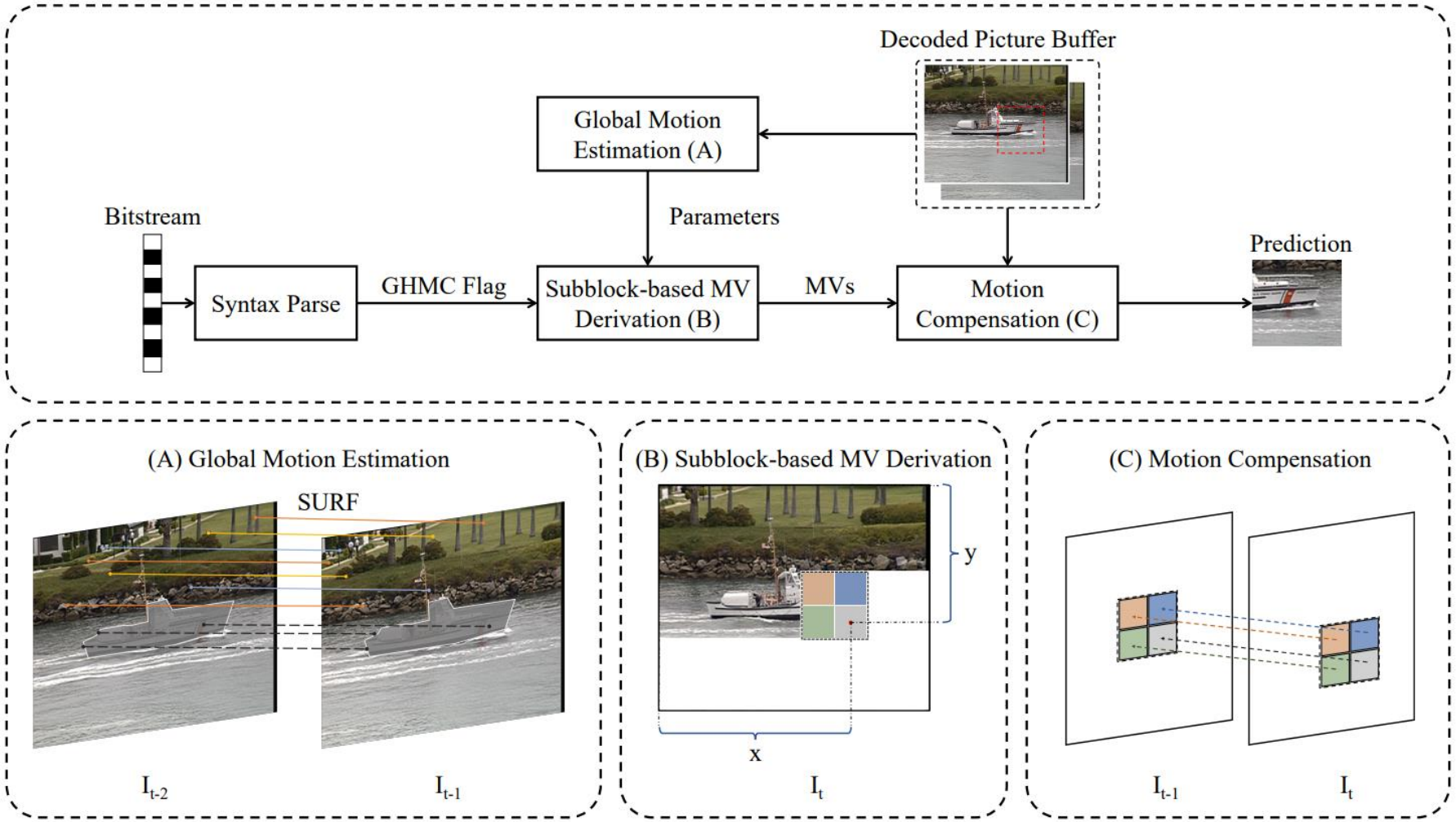
$$x'=\frac{a_0+a_1x+a_2y}{1+c_1x+c_2y},\quad y'=\frac{b_0+b_1x+b_2y}{1+c_1x+c_2y}$$

- 运动参数编码
  - ~~显示地传输~~
  - 利用参考帧进行预测
- 基于4x4子块进行运动补偿（Affine）



同一平面物体在视频前后两帧的像的映射关系可以用 H 表征

Motion Estimation & Motion Compensation

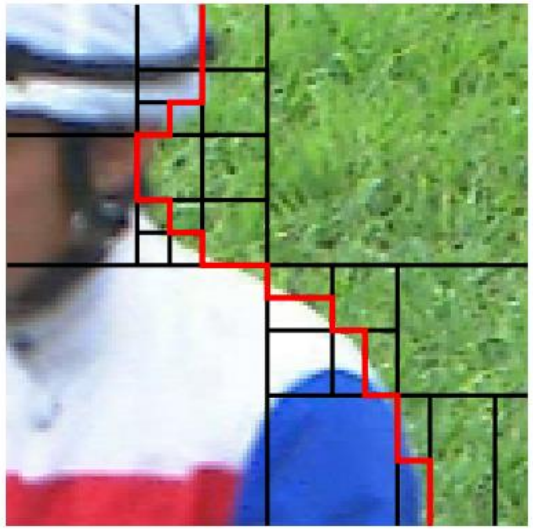


Geometric Partitioning

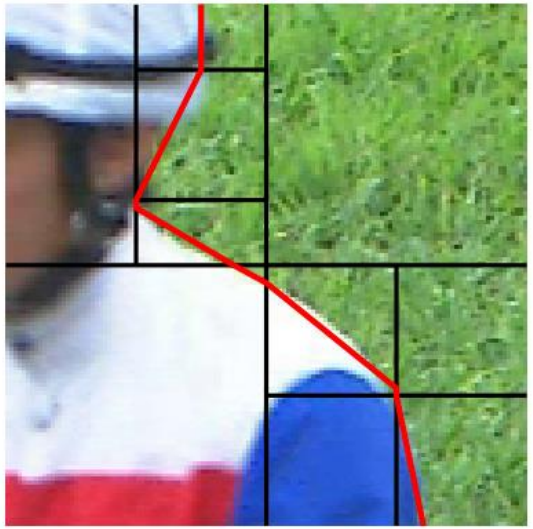
Geometric Partitioning Mode in Versatile Video Coding: Algorithm Review and Analysis

Han Gao, Semih Esenlik, Elena Alshina, and Eckehard Steinbach, *Fellow, IEEE*

(b) 128 × 128 CTU



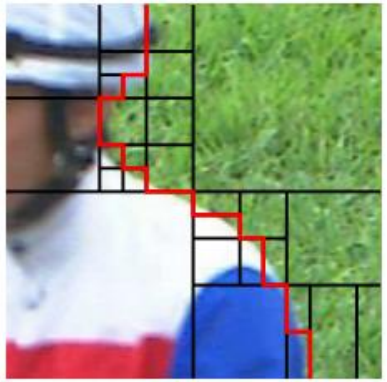
(c) Rectangular partition



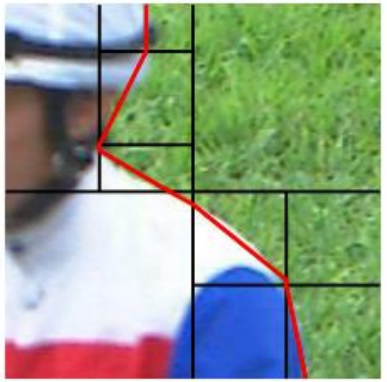
(d) Geometric partition

(a) Picutre and motion field

(b) 128 × 128 CTU



(c) Rectangular partition



(d) Geometric partition

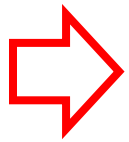
1、自然序列中的对象边界很少遵循矩形块模式。

2、传统方法增加为这些块划分和预测语法元素发送信号的速率开销。近似分割边界很少遵循实际运动场边界。因此预测误差较高，增加了残差信号的比特率。

This is because objects typically exhibit movement relative to a static background or other moving objects, and object boundaries in natural sequences rarely adhere to rectangular block patterns. As shown in Figure, finer block partitions are required when approximating moving object boundaries using rectangular blocks, which increases the rate overhead for signaling the partition and the prediction syntax elements for these blocks. In addition, the approximated partitioning boundary rarely follows the actual motion field boundaries. Consequently, the prediction error is higher, which increases the bitrate for residual signaling.

Partition Rule

Because the intrapicture predicted CUs in VVC can use angular and non-linear prediction modes, the non-horizontal and non-vertical edges are handled well. Therefore, the presented GPM focuses on the interpicture predicted CUs. When GPM is applied to a CU, this CU is split into two parts by a straight partitioning boundary. The location of the partitioning boundary is mathematically defined by an angle parameter φ and an offset parameter ρ. These parameters are quantized and combined into a GPM partitioning index lookup table. The GPM partitioning index of the current CU is coded into the bitstream. In total, 64 partitioning modes are supported by GPM in VVC for a CU with a size of $w \times h = 2^k \times 2^l$ (in terms of luma samples) with $k, l \in \{3 \ldots 6\}$. Moreover, GPM is disabled on a CU that has an aspect ratio larger than 4:1 or smaller than 1:4, because narrow CUs rarely contain geometrically separated patterns.



Definition of the partitioning boundary

Quantization of angle parameter

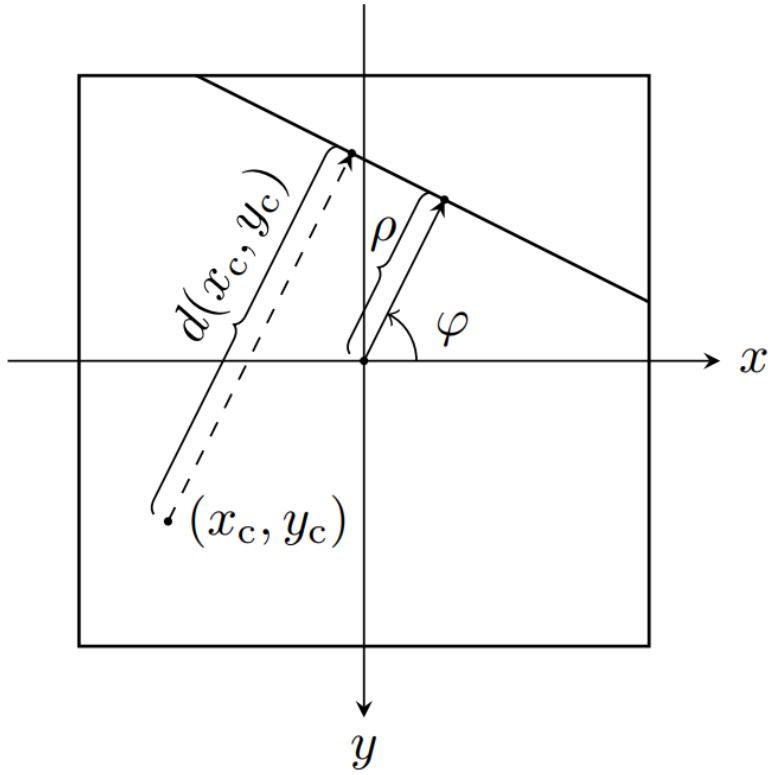
Quantization of offset parameter

Partition Rule (Definition of the partitioning boundary)

1) Definition of the partitioning boundary: The partitioning boundary is defined as a straight line that is geometrically located inside the CU. The equation of this line is expressed in Hessian normal form as

$$x_c \cos(\varphi) - y_c \sin(\varphi) + \rho = 0, \tag{3}$$

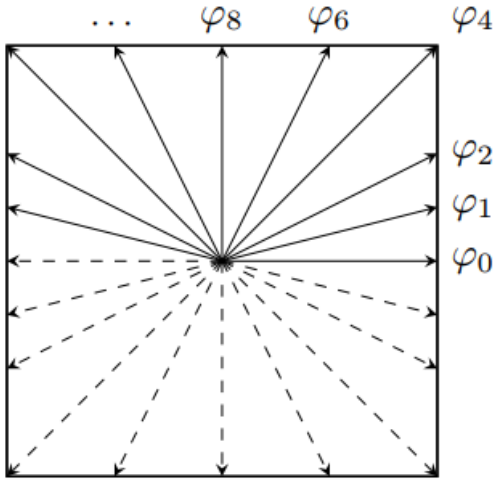
with (x_c, y_c) defining continuous positions relating to the central position of the CU. In the example illustrated in Fig. 6(a), the angle parameter φ in (3) describes the anti-clockwise angle from the x-axis to the normal vector of the partitioning boundary, whereas the offset parameter ρ in (3) is the displacement of the partitioning boundary from the origin that is defined at the center of the CU. Note that the y-axis is reversed to simplify the discretization of the partitioning boundary.

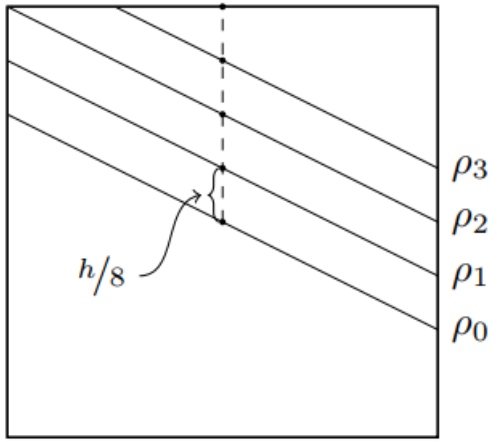


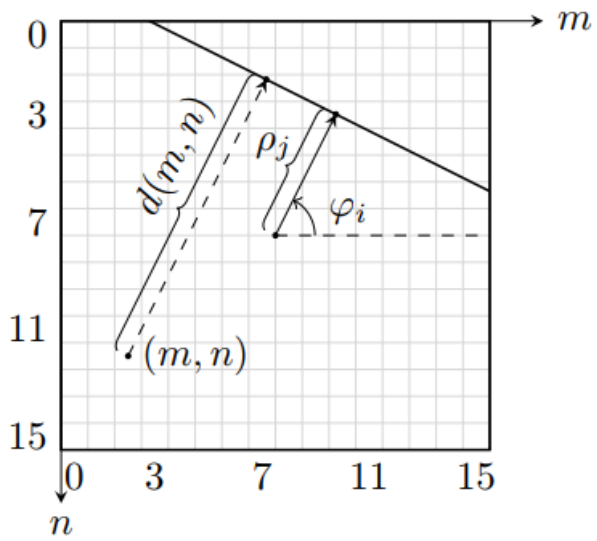
(a)

(a) Example of a Hessian normal form-based partitioning boundary

Partition Rule (Definition of the partitioning boundary)







discretization of the partitioning boundary.

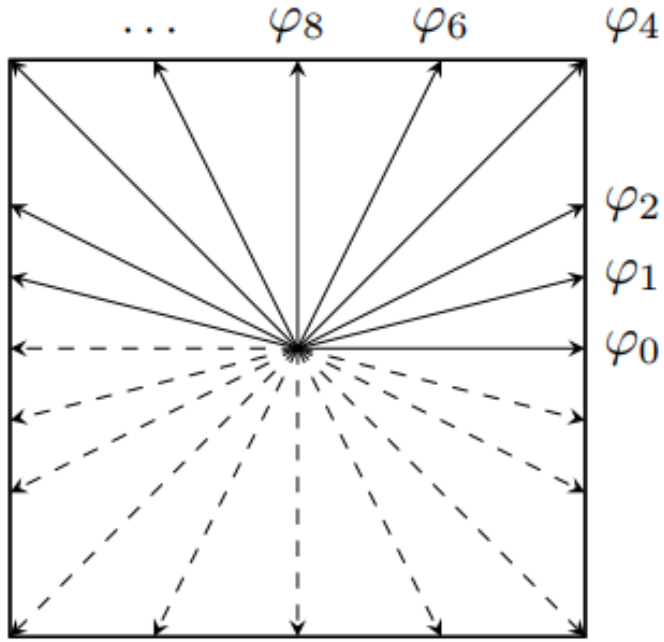
The displacement d of an arbitrary position (x_c, y_c) relative to the partitioning boundary is used to derive the blending matrices W_0 and W_1. Based on (3), the value of d is given as

$$d(x_c, y_c) = x_c \cos(\varphi) - y_c \sin(\varphi) + \rho. \quad (4)$$

As displacement is directional, the value of d can be positive or negative. The sign of d indicates which partition the position (x_c, y_c) belongs to, whereas the magnitude of d is equal to the distance of (x_c, y_c) to the partitioning boundary.

Partition Rule (Quantization of angle parameter)

2) Quantization of angle parameter φ: To achieve a reasonable number of defined partitioning boundaries, the angle parameter φ and offset parameter ρ have to be quantized. As shown in Fig. 6(b), the angle parameter φ is quantized into 20 discrete angles φ_i, with the range of $[0, 2\pi)$ symmetrically divided. Because the diagonal or anti-diagonal partitioning boundaries are most frequently used in GPM, the quantized φ_i is therefore designed with fixed $\tan(\varphi_i)$ values that are equal to the possible aspect ratios of CU where GPM is applied. For example, if GPM is applied on a CU with an aspect ratio $w/h = 1/2$, the diagonal partitioning boundary is aligned with the line of (3) defined by $\varphi_i = \arctan(1/2)$ and $\rho = 0$. In the presented GPM algorithm, $\tan(\varphi_i)$ is limited as a value in $\{0, \pm 1/4, \pm 1/2, \pm 1, \pm 2, \infty\}$. Note that the tangent values of ± 4, which yield a near horizontal partitioning boundary, are not included, because the near horizontal partitioning of a motion field is less frequently used for natural video content.



20种□ 高比或□ 高比

Partition Rule (Quantization of offset parameter ρ)

3) Quantization of offset parameter ρ: Fig. 6(c) shows an example of the offset parameter ρ quantization. The offset parameter ρ is quantized into ρ_j depending on the CU width w and height h. The offset index j is defined in $\{0 \ldots 3\}$. To avoid unequally distributed partitioning boundary offsets between different block sizes, the ρ_j is first factorized with

4种位移

$$\rho_j = \rho_{x,j} \cos(\varphi_i) - \rho_{y,j} \sin(\varphi_i) \tag{7}$$

or

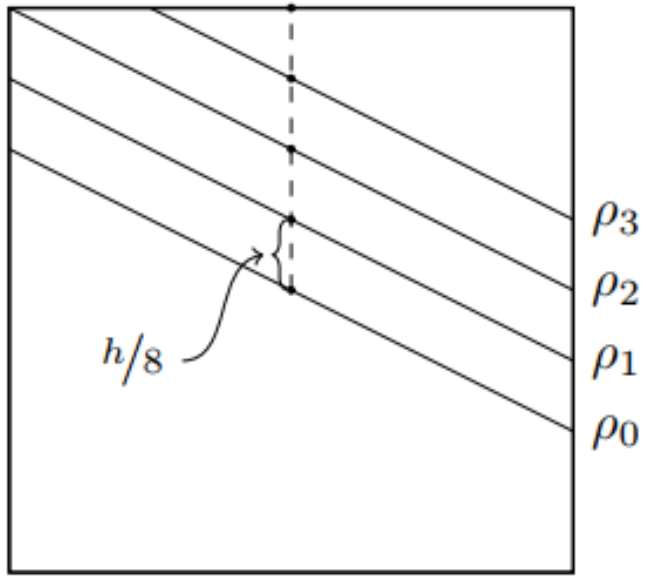
$$\begin{aligned} \rho_j = {} & (\rho_{x,j} \cdot \mathrm{cosLut}[i]) \gg 3 \\ & + (\rho_{y,j} \cdot \mathrm{cosLut}[(i+8)\%32]) \gg 3. \end{aligned} \tag{8}$$

The $\rho_{x,j}$ and $\rho_{y,j}$ are then coupled with w and h using

$$\rho_{x,j} = \begin{cases} 0 & i\%16 = 8 \text{ or } (i\%16 \neq 0 \text{ and } h \geqslant w) \\ \pm j \cdot {}^{w}/_{8} & \text{otherwise} \end{cases} \tag{9}$$

and

$$\rho_{y,j} = \begin{cases} \pm j \cdot {}^{h}/_{8} & i\%16 = 8 \text{ or } (i\%16 \neq 0 \text{ and } h \geqslant w) \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$



All types of partition & signal

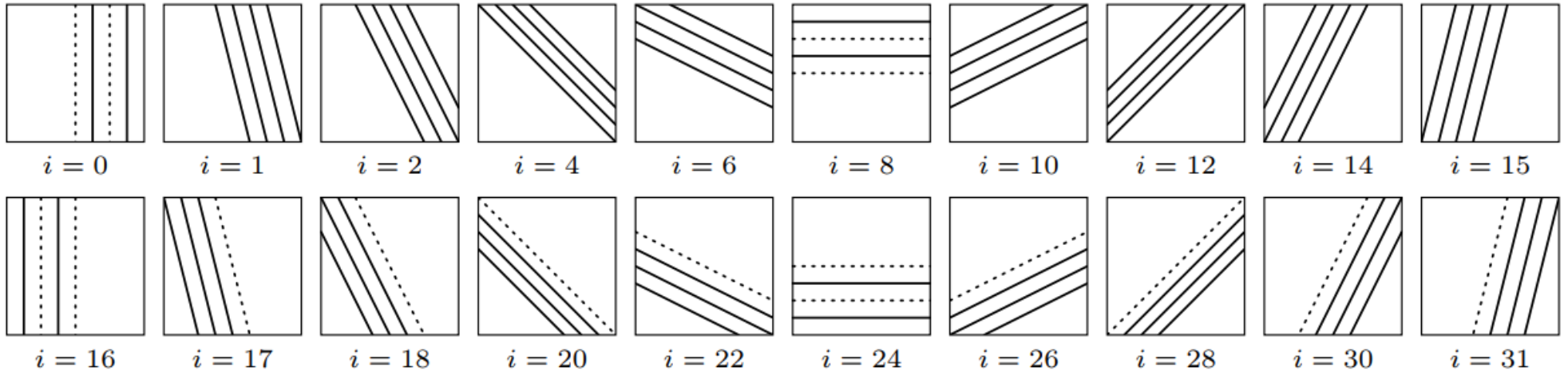


Fig. 7. Visualized examples of the 64 supported GPM partitioning modes grouped by identical angle index i; the offset indices j in each subfigure vary in the range of $\{0 \ldots 3\}$; the removed redundant quantized offsets are illustrated by dotted lines.

Several redundant quantized offsets, which are listed in Table II, are removed in the presented GPM algorithm. Therefore, the total number of GPM partitioning modes is $N_{\mathrm{GPM}} = N_\varphi N_\rho - {}^{N_\varphi}/_2 - 2 - 4 = 64$ with $N_\varphi = 20$ and $N_\rho = 4$. The GPM partitioning modes are visualized in Fig. 7, grouped by identical angles index i. The dotted partitioning lines in Fig. 7 demonstrate the redundant modes that are not included in the presented GPM design. The 64 supported GPM partitioning modes are indexed by the syntax element *merge_gpm_partition_idx* that is coded into the bitstream using the VVC entropy coding engine.

TABLE II
REDUNDANT QUANTIZED OFFSETS AND CORRESPONDING REASONS

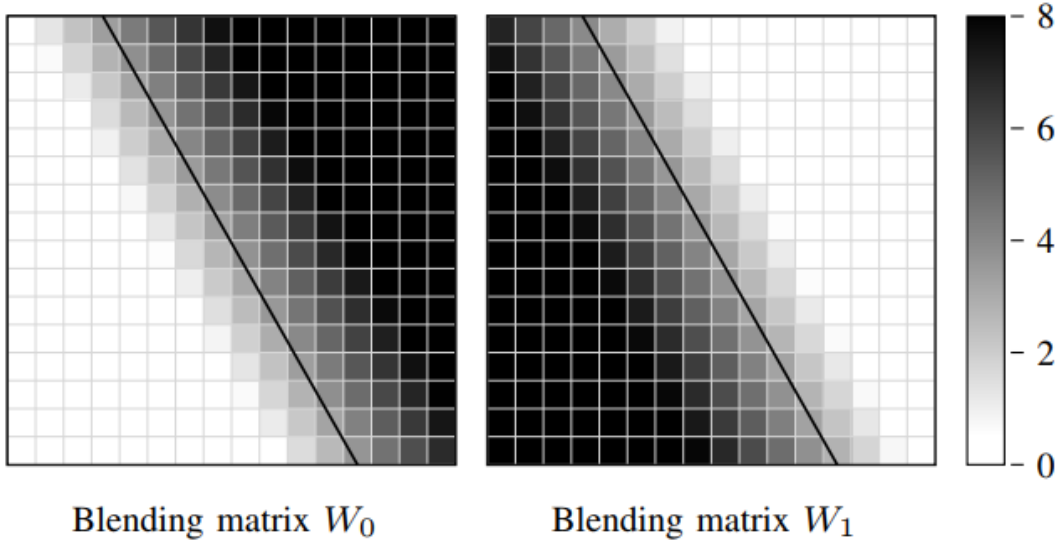
Modes	**Overlapped with**	**Numbers**
$i \geqslant 16,\ j = 0$	$i < 16,\ j = 0$	${}^{N_\varphi}/_2$
$i \in \{0, 8\},\ j = 0$	BT split line	2
$i \in \{0, 8, 16, 24\},\ j = 2$	One of the TT split lines	4

Motion Compensation

In the presented GPM algorithm, a soft blending process is applied. That is, as shown in Fig. 8, ramped weighting values in the range of $(0, 8)$ are used when the sample is located inside the soft blending area (i.e., between the dashed lines); otherwise, a full weighting value of 0 or 8 is selected. The weighting values of individual sample positions in one of the blending matrices are given by a ramp function as

$$\gamma_{x_c,y_c} = \begin{cases} 0 & d(x_c, y_c) \leqslant -\tau \\ \frac{8}{2\tau}(d(x_c, y_c) + \tau) & -\tau < d(x_c, y_c) < \tau \\ 8 & d(x_c, y_c) \geqslant \tau, \end{cases} \quad (14)$$

where τ defines the width of the blending area. In the presented GPM algorithm, τ is selected as two samples. Based



Blending matrix W_0 Blending matrix W_1

Distance:

$$d(x_c, y_c) = x_c \cos(\varphi) - y_c \sin(\varphi) + \rho. \quad (4)$$

$$d(m,n) = ((m + \rho_{x,j}) \ll 1 - w + 1) \cdot \text{cosLut}[i] + ((n + \rho_{y,j}) \ll 1 - h + 1) \cdot \text{cosLut}[(i + 8)\%32], \quad (12)$$

Motion Estimation

Inspired by these, the GPM merge list is derived from the regular merge list by the parity of the GPM merge index. That is, for a candidate with an even value of the GPM merge index, the MV_0 from reference picture list L0 of the regular merge list with corresponding regular merge index is used as the GPM merge candidate. If MV_0 is not available, MV_1 from the reference picture list L1 is used instead. Conversely, MV_1 is chosen as the default GPM merge candidate for an odd value of the GPM merge index, and if MV_1 is unavailable, MV_0 is used instead. Compared with other merge list construction methods studied in [48], the index parity-based method directly extracts the GPM merge candidates from the regular merge list without pruning, which minimizes the implementation complexity.

TABLE III
EXAMPLES OF REGULAR MERGE LIST AND GPM MERGE LIST

merge_idx	0	1	2	3	4	5
Cand.	MV_0	MV_0	–	–	MV_0	MV_0
	MV_1	MV_1	MV_1	MV_1	MV_1	MV_1

(a) Regular merge list

GPM_idx$_{0,1}$	0	1	2	3	4	5
Cand.	MV_0	MV_1	MV_1	MV_1	MV_0	MV_1

(b) GPM merge list

Motion Estimation

Instead of a texture-based or statistic-based geometric partitioning search method such as those presented in [11], [17]–[20], a rate distortion optimization (RDO)-based partitioning encoder search is applied in the presented GPM algorithm. Compared with the texture-based search methods, the RDO-based search more easily catches the geometrically separated motion field that contains similar textures. Moreover, the RDO-based search is more versatile than statistic-based methods for different video content.

Assuming the signaled maximum number of GPM merge candidates is six, each partitioning mode has in total $6 \times 5 = 30$ motion information combinations because the $\mathrm{GPM\_idx}_0$ and $\mathrm{GPM\_idx}_1$ are not the same. As $N_{\mathrm{GPM}} = 64$ partitioning modes are designed in GPM, one out of $6 \times 5 \times 64 = 1920$ combined GPM candidates of partitions and motion information is to be selected by the encoder.

Entropy Coding

GPM SYNTAX ELEMENTS TABLE

Syntax	Descriptor
merge_data() {	
...	
if(!ciip_flag){	
merge_gpm_partition_idx	ae(v)*
merge_gpm_idx0	ae(v)
if(MaxNumGpmMergeCand > 2)	
merge_gpm_idx1	ae(v)
}	
...	
}	

* Context-adaptive arithmetic entropy-coded syntax element.

Motion Estimation

1) Stage 1: For each unidirectional motion information candidate of the GPM merge list, full CU MCPs over the six GPM merge candidates are performed to obtain six rectangular predictors the same size as the current CU. The sum of absolute differences (SAD) in luma component between the predictors and the original signal is calculated as $\mathrm{SAD}_{\mathrm{CU},k}$ with index $k \in \{0 \ldots 5\}$. Identical entries of the GPM merge candidate list are excluded for this stage. If all motion information in the merge candidate list is identical, the entire GPM encoder search is aborted.

Candidate Prepare & Selection

Motion Estimation

2) Stage 2: For each GPM partitioning mode, the parts predicted by $\mathrm{GPM\_idx}_0$ of the six rectangular predictors are masked out, using a hard blending matrix (i.e., only 0 or 8 is selected as a weighting value, depending on the sign of the displacement $d(m,n)$.). The SAD in luma of these parts are computed and denoted as $\mathrm{SAD}_{\mathrm{P0},k,l}$ with $k = 0 \ldots 5$ and $l = 0 \ldots 63$. Since the hard blending matrix (no blending area) is used, the SAD of the parts predicted by $\mathrm{GPM\_idx}_1$ is obtained by

$$\mathrm{SAD}_{\mathrm{P1},k,l} = \mathrm{SAD}_{\mathrm{CU},k} - \mathrm{SAD}_{\mathrm{P0},k,l}. \tag{20}$$

A combined rate-distortion (RD)-cost $J_{\alpha,\beta,l}$, for the partitioning mode with index l, is given by

$$J_{\alpha,\beta,l} = \mathrm{SAD}_{\mathrm{P0},\alpha,l} + \mathrm{SAD}_{\mathrm{P1},\beta,l} + \lambda(R_\alpha + R_\beta), \tag{21}$$

where α and β denote the predictor indices $\mathrm{GPM\_idx}_0$ and $\mathrm{GPM\_idx}_1$, and R_α and R_β denote the corresponding estimated motion rates. Both α and β belong to $\{0 \ldots 5\}$, but $\alpha = \beta$ cases are excluded in the encoder search. The combined GPM candidates are sorted by $J_{\alpha,\beta,l}$. Moreover, GPM candidates with $J_{\alpha,\beta,l} > \mathrm{SAD}_{\mathrm{CU},k} + \lambda R_k$ are excluded, where R_k is the estimated motion rate of the k-th GPM merge candidate.

Rough Estimation

Motion Estimation

3) Stage 3: The best 60 (or less) combined GPM candidates from stage 2 are used to conduct the soft blending process as described in Section III-C to generate P_{G} in luma component. The sum of absolute transformed differences (SATD) of luma between GPM predictor P_{G} and the original signal for each combined candidate is computed as $\mathrm{SATD}_{\mathrm{PG},l}$. The RD-cost is updated as

$$J'_{\alpha,\beta,l} = \mathrm{SATD}_{\mathrm{PG},l} + \lambda(R_\alpha + R_\beta + R_l), \qquad (22)$$

where R_l represents the estimated rate of the partitioning index. The combined GPM candidates are sorted again by $J'_{\alpha,\beta,l}$. In this stage, the lowest SATD costs of previously tested coding tools, such as regular merge or affine, are used for early termination.

Rough Estimation of SATD

Motion Estimation

4) Stage 4: The corresponding chroma component of the best eight (or less) candidates from stage 3 is generated with the soft blending process. Residual transform coding (if there is a residual) and CABAC-based rate estimation are applied on these candidates to obtain the accurate rate cost R_{GPM} that includes the rate for motion, partitioning mode, and residual coding. The distortion over three components between these candidates and the original signal is measured by the sum of squared differences (SSD) as $\mathrm{SSD}_{\mathrm{PG},l}$. Finally, the GPM candidate with the lowest overall RD-cost

$$J''_{\alpha,\beta,l} = \mathrm{SSD}_{\mathrm{PG},l} + \lambda R_{\mathrm{GPM}}, \tag{23}$$

is selected as the final GPM mode.

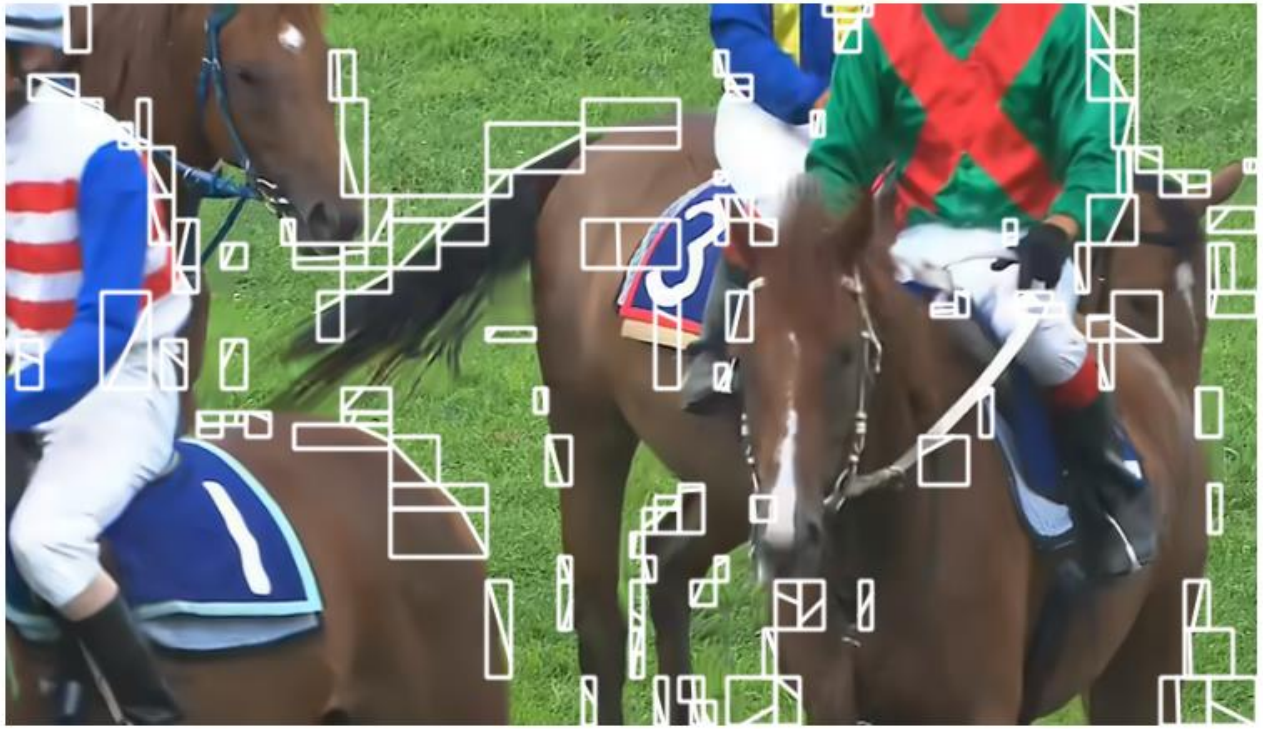
Refine Estimation of SSE

Performance

Sequence	*GPM_Tool_Off* (anchor: VTM 8.0)										*GPM_Tool_On* (anchor: VTM 8.0 w/o VVC tools)									
	RA (%)					**LB** (%)					**RA** (%)					**LB** (%)				
	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT
Tango	0.65	1.36	1.18	98	100						−1.17	−2.83	−2.60	123	99					
FoodMarket	0.43	0.55	0.54	97	98						−0.77	−0.93	−0.93	124	100					
Campfire	0.21	0.15	0.63	97	99						−0.72	−0.80	−2.66	128	101					
Average A1	**0.43**	**0.69**	**0.78**	**97**	**99**						**−0.89**	**−1.52**	**−2.06**	**125**	**100**					
CatRobot	0.68	0.99	1.11	97	99						−1.66	−2.87	−2.65	128	99					
DaylightRoad	0.32	0.31	0.41	98	99						−1.26	−1.77	−1.43	127	98					
ParkRunning	0.54	0.76	0.75	96	100						−1.16	−1.39	−1.49	130	99					
Average A2	**0.51**	**0.68**	**0.76**	**97**	**100**						**−1.36**	**−2.01**	**−1.86**	**129**	**99**					
MarketPlace	0.42	0.77	1.05	97	97	0.88	1.29	0.91	95	99	−0.77	−0.83	−1.37	130	100	−1.51	−1.79	−2.04	130	97
RitualDance	0.56	0.86	1.17	97	97	1.01	1.11	1.30	95	102	−1.14	−1.88	−2.56	129	100	−1.73	−2.37	−2.92	131	99
Cactus	0.66	0.76	0.95	97	98	1.45	1.71	1.92	94	103	−1.21	−1.46	−1.75	131	102	−2.87	−3.11	−2.91	128	93
BasketballDrive	0.27	0.94	0.87	97	97	0.86	1.70	1.45	95	98	−1.01	−2.39	−1.90	130	100	−1.75	−3.45	−2.96	128	97
BQTerrace	0.30	0.28	0.31	97	98	0.59	0.64	−0.22	96	99	−1.07	−0.67	−0.56	131	101	−1.59	−0.96	−1.57	126	98
Average B	**0.44**	**0.72**	**0.87**	**97**	**97**	**0.96**	**1.29**	**1.07**	**95**	**100**	**−1.04**	**−1.45**	**−1.63**	**131**	**101**	**−1.89**	**−2.34**	**−2.48**	**129**	**97**
BasketballDrill	1.16	1.88	1.91	96	102	2.39	3.45	3.03	93	98	−2.17	−3.35	−3.48	137	104	−3.68	−5.06	−5.50	136	102
BQMall	2.13	3.43	3.46	97	100	2.53	3.54	2.82	94	101	−4.18	−5.28	−5.38	137	104	−4.76	−6.04	−5.61	137	97
PartyScene	0.71	1.07	1.23	96	99	1.14	1.34	1.73	93	100	−1.84	−1.88	−2.14	138	105	−2.87	−3.01	−3.18	142	101
RaceHorsesC	1.40	2.30	2.07	96	101	1.73	2.45	2.86	93	100	−3.04	−5.93	−5.62	138	103	−3.44	−5.29	−5.69	144	104
Average C	**1.35**	**2.17**	**2.17**	**96**	**100**	**1.95**	**2.69**	**2.61**	**93**	**100**	**−2.81**	**−4.11**	**−4.16**	**137**	**104**	**−3.69**	**−4.85**	**−5.00**	**140**	**101**
BasketballPass	0.80	2.53	1.45	96	99	1.65	3.38	3.09	93	102	−2.11	−4.29	−4.15	140	103	−3.94	−7.03	−6.24	148	109
BQSquare	0.13	0.13	1.18	97	98	0.84	1.69	0.10	95	103	−1.49	−1.10	−1.32	135	103	−3.13	−3.15	−1.42	143	106
BlowingBubbles	0.73	1.13	1.19	96	99	1.81	2.15	2.63	92	105	−2.36	−2.22	−2.50	139	101	−4.55	−4.56	−4.98	142	102
RaceHorses	1.44	2.67	2.00	95	100	1.95	3.12	3.32	92	101	−3.49	−6.50	−5.63	143	100	−4.77	−6.78	−6.24	152	106
Average D*	**0.77**	**1.62**	**1.45**	**96**	**99**	**1.56**	**2.59**	**2.28**	**93**	**103**	**−2.36**	**−3.53**	**−3.40**	**140**	**102**	**−4.10**	**−5.38**	**−4.72**	**146**	**106**
FourPeople						1.84	1.83	1.37	95	101						−4.23	−4.24	−4.74	128	97
Johnny						2.51	0.95	1.45	98	100						−6.09	−5.29	−5.32	126	104
KristenAndSara						1.71	0.62	0.97	96	101						−5.52	−4.20	−3.60	129	101
Average E						**2.02**	**1.13**	**1.26**	**96**	**101**						**−5.28**	**−4.57**	**−4.55**	**128**	**101**
BasketballDrillText	1.16	1.97	1.53	97	99	2.30	2.36	2.88	94	99	−2.22	−3.14	−3.44	105	104	−3.71	−4.05	−5.13	104	101
ArenaOfValor	0.97	1.29	0.97	98	99	1.71	2.17	2.43	95	103	−2.62	−3.19	−2.92	102	97	−3.72	−4.66	−4.34	98	95
SlideEditing	0.04	−0.03	−0.01	99	99	−0.10	0.07	−0.05	98	103	−0.59	−0.28	−0.30	109	102	0.07	−0.96	−0.84	108	100
SlideShow	0.19	0.27	0.38	98	99	−0.04	−0.32	−0.85	96	97	−0.43	−0.89	−0.92	108	102	−1.01	−2.21	−1.49	108	104
Average F*	**0.59**	**0.88**	**0.72**	**98**	**99**	**0.97**	**1.07**	**1.10**	**96**	**101**	**−1.47**	**−1.87**	**−1.89**	**106**	**101**	**−2.09**	**−2.97**	**−2.95**	**104**	**100**
Overall	**0.70**	**1.09**	**1.18**	**97**	**99**	**1.55**	**1.72**	**1.63**	**95**	**100**	**−1.54**	**−2.28**	**−2.44**	**130**	**101**	**−3.34**	**−3.73**	**−3.84**	**132**	**99**

* Not included in the overall average results based on [39].

Selection



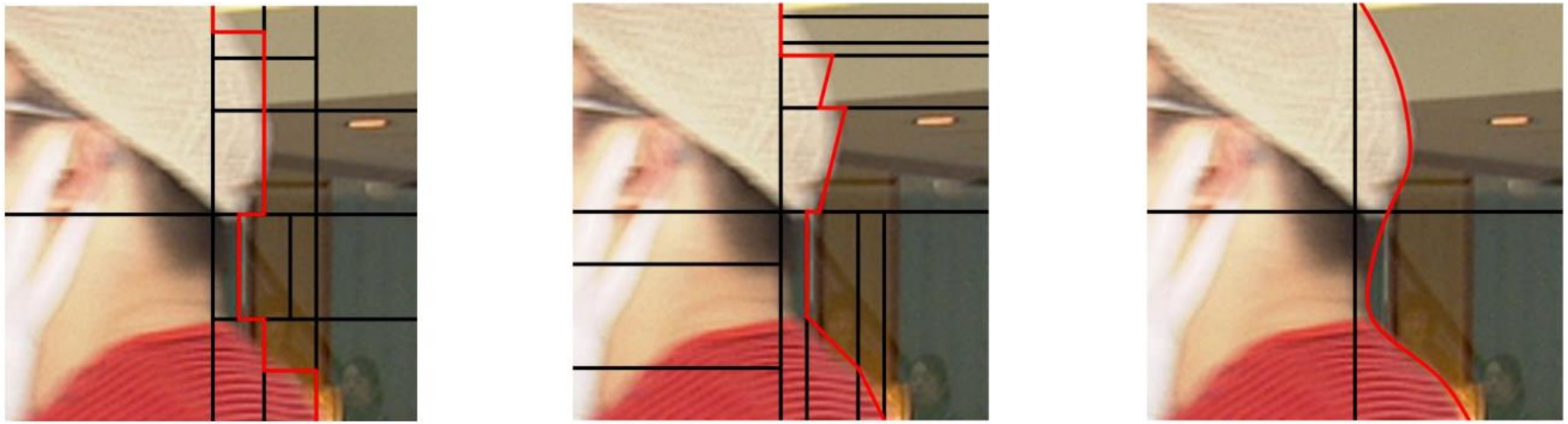
(a) GPM predicted CU example of the 31st picture of *RaceHorsesC*.

(b) GPM predicted CU example of the 92nd picture of *BQMall*.

Geometric Partitioning

Object Segmentation-Assisted Inter Prediction for Video Coding

Zhuoyuan Li[1*], Zikun Yuan[2*], Dong Liu[1], *Senior Member, IEEE,* Li Li[1], *Member, IEEE,* Xiaohu Tang[2], *Senior Member, IEEE,* and Feng Wu[1], *Fellow, IEEE*

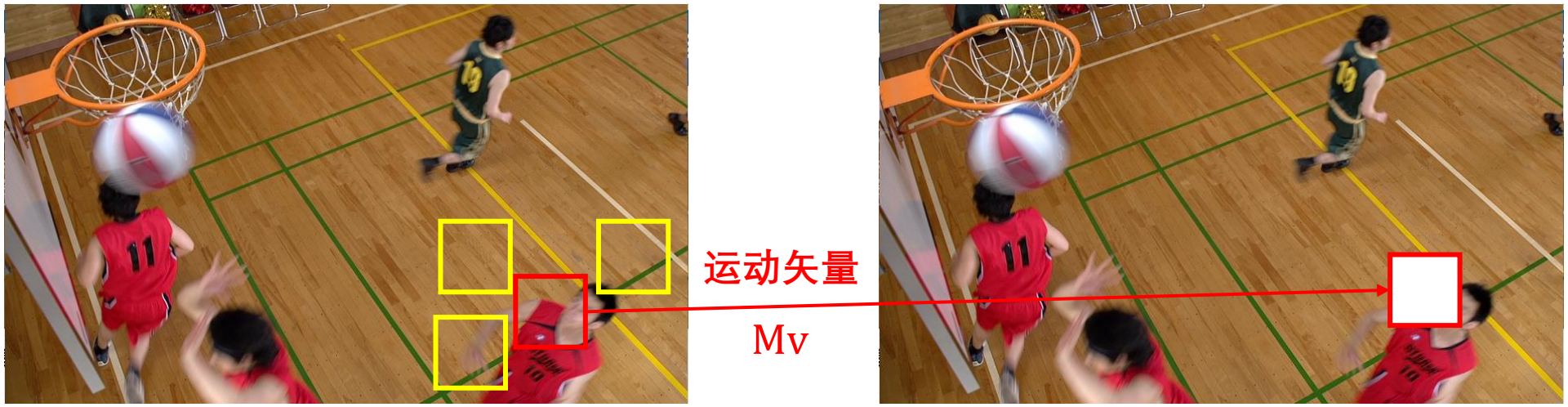


(a) Rectangular partition (RP) (b) Line-based geometric partition (GP) (c) Segmentation-based partition (SP)

Fig. 1. The partition results of different methods in the actual coding process, where (a) uses the rectangular partition (RP), (b) uses the RP + line-based geometric partition (GP), (c) uses the RP + GP + segmentation-based partition (SP). The block is from the 71-th frame of the VVC ClassC *BQMall* sequence.

基于块的帧间预测方法 Strong Assumption 基于块的运动假设

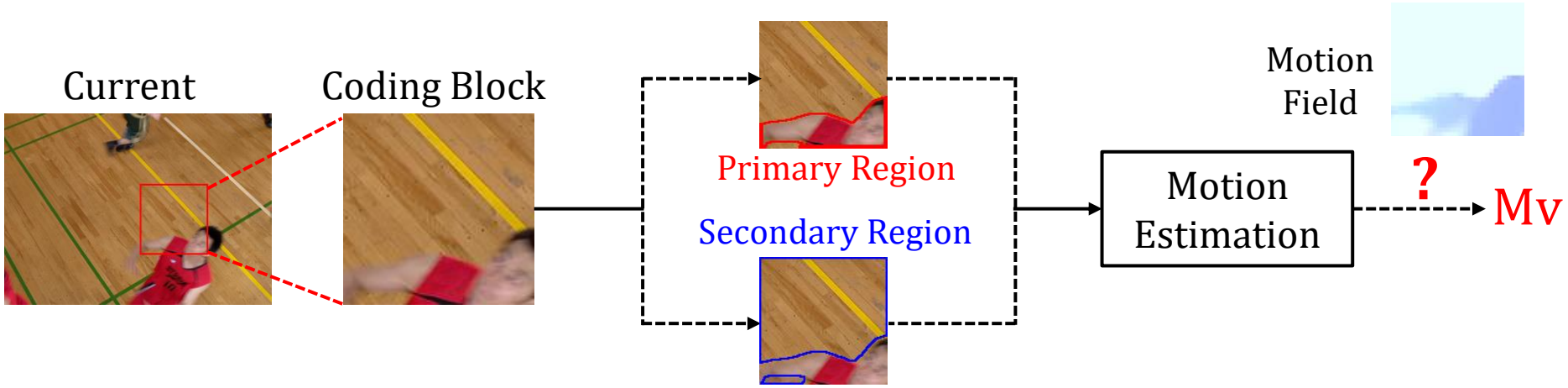
参考帧



当前帧

基于块的帧间预测技术通常是在假设块内像素的运动趋于均匀的情况下进行。

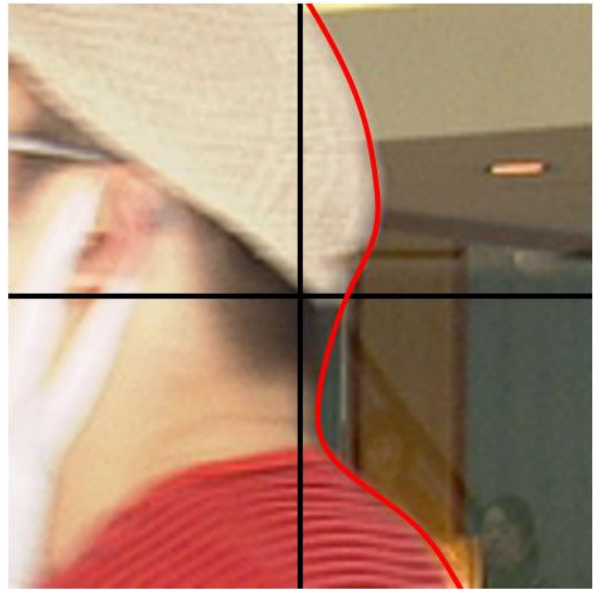
每个块内的运动用一个运动矢量对其运动场进行描述。



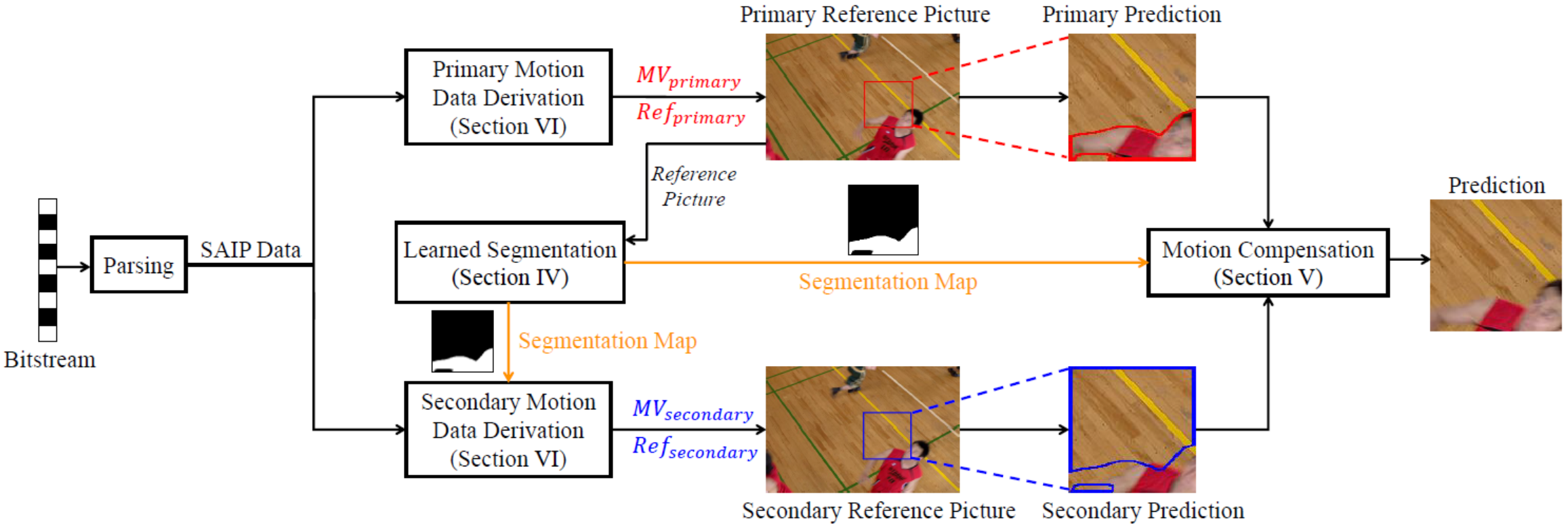
Is there Mv can represent the motion of Primary and Secondary regions ?

Segmentation-based partition Method

- Traditional segmentation algorithm → Rough estimation
- Motion Compensation → Boundary artifacts
- Motion Vector Coding → Extra bits consumption
- Motion Estimation → Optimal motion vectors



Framework (Decoder):



We propose an object segmentation-assisted inter prediction (SAIP) method that does not restrict the partitioning shape and further improves the prediction accuracy by utilizing segmentation to assist the entire prediction process. To the best of our knowledge, we are the first to introduce deep learning-based segmentation technologies to improve the performance of inter prediction in traditional video coding. (Blackboard)

Learned Segmentation

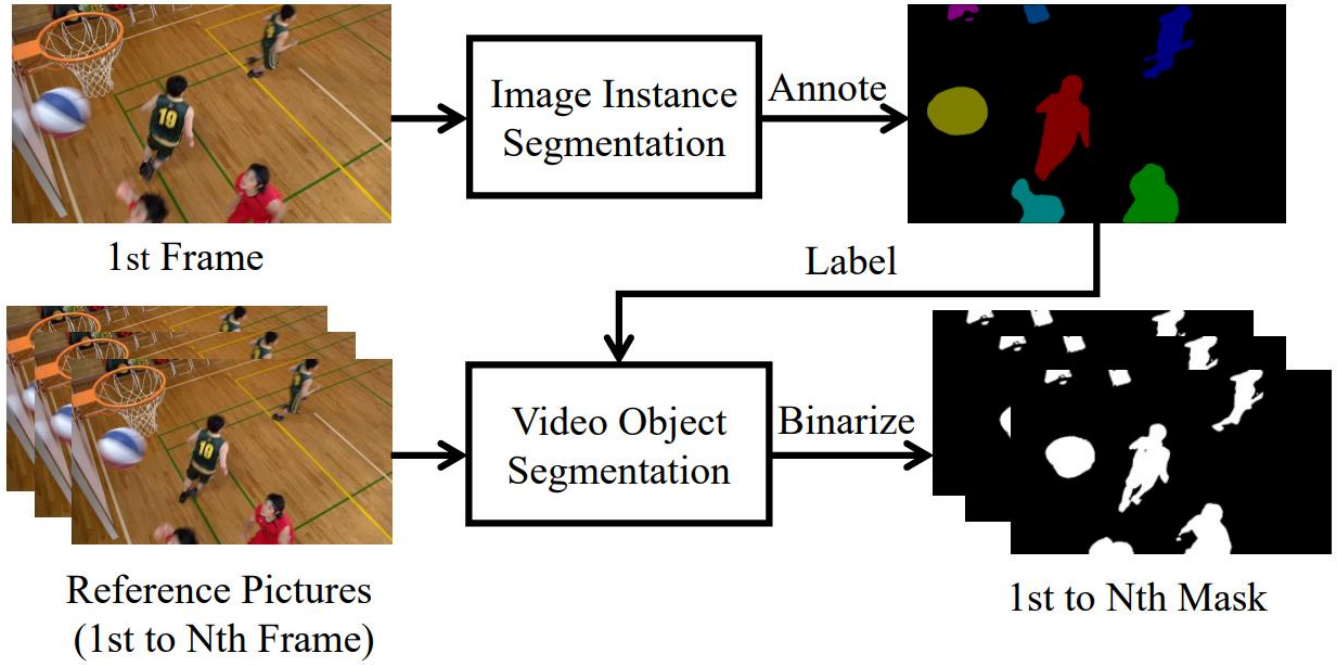
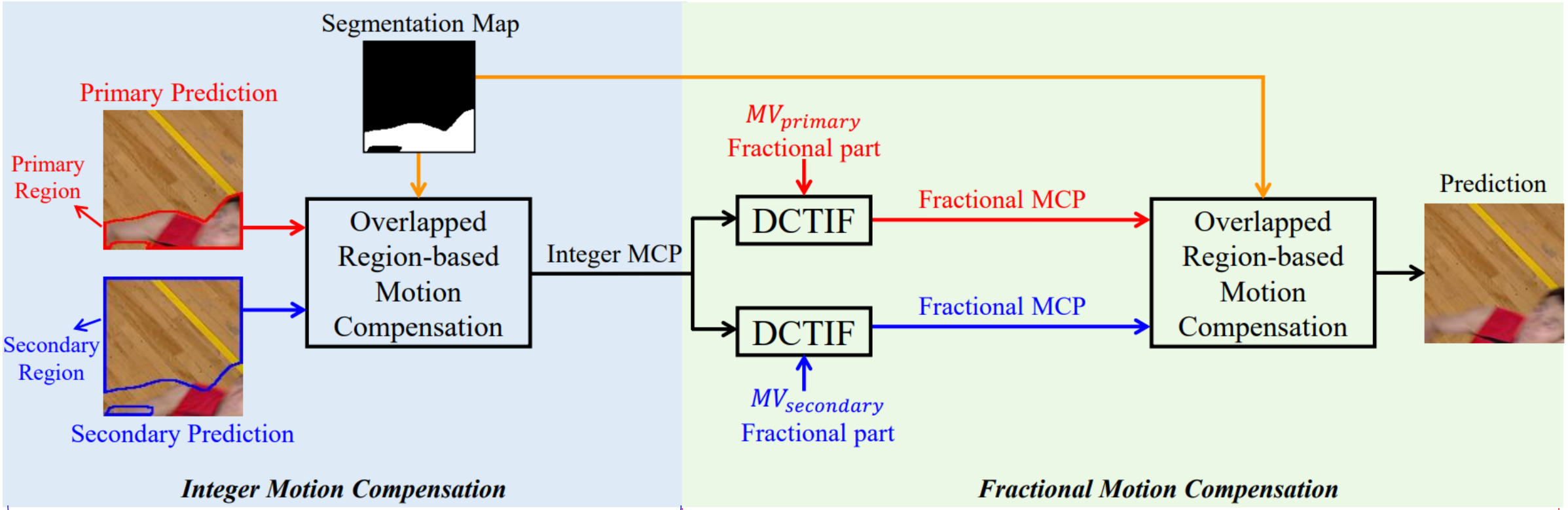


Image Instance Segmentation for the coding of high quality frame

Video Object Segmentation for the coding of other frame

Motion Compensation



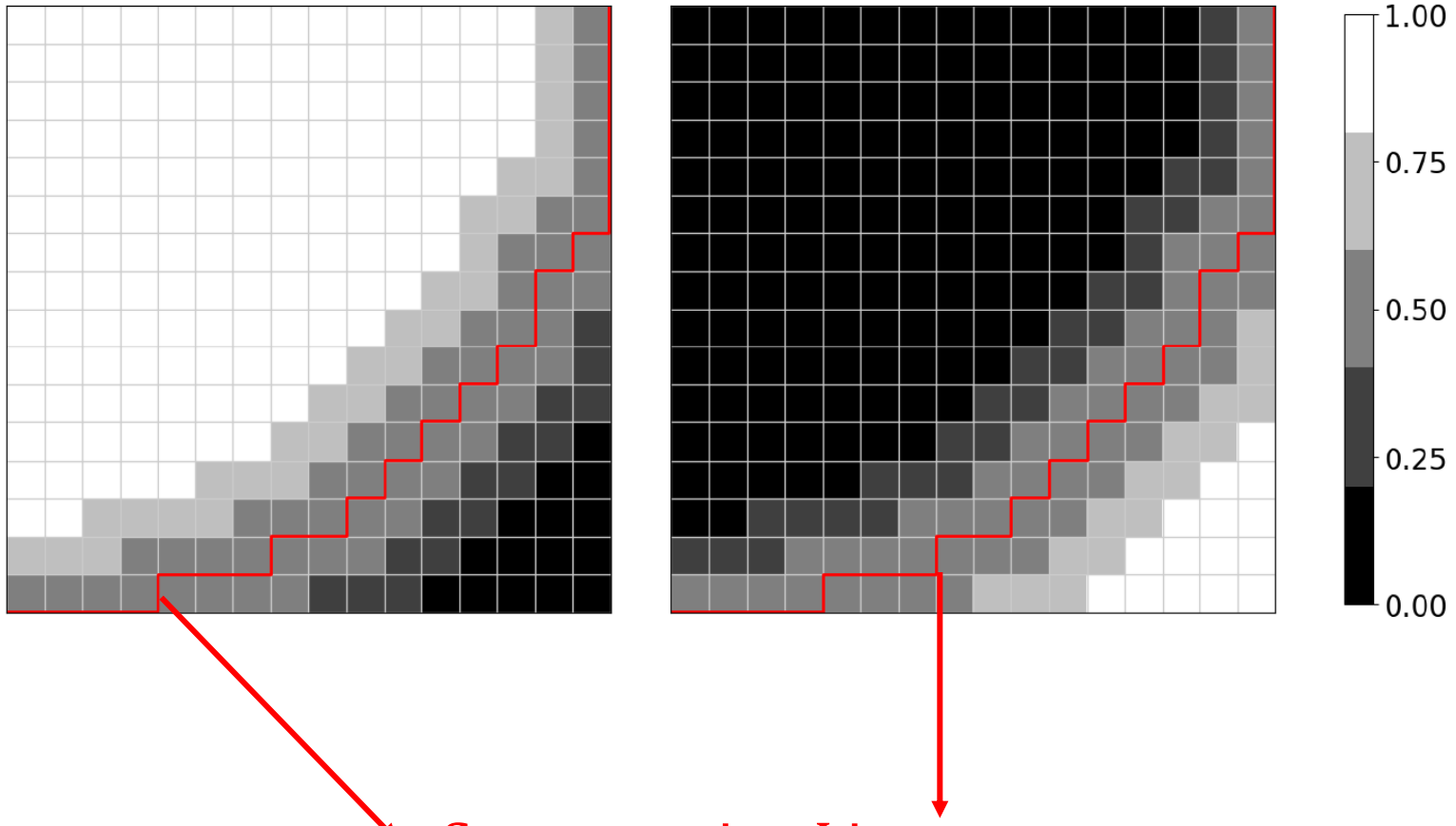
基于分割图的整像素运动补偿

基于分割图的分像素运动补偿

利用分割图引导多区域的**多阶段**运动补偿

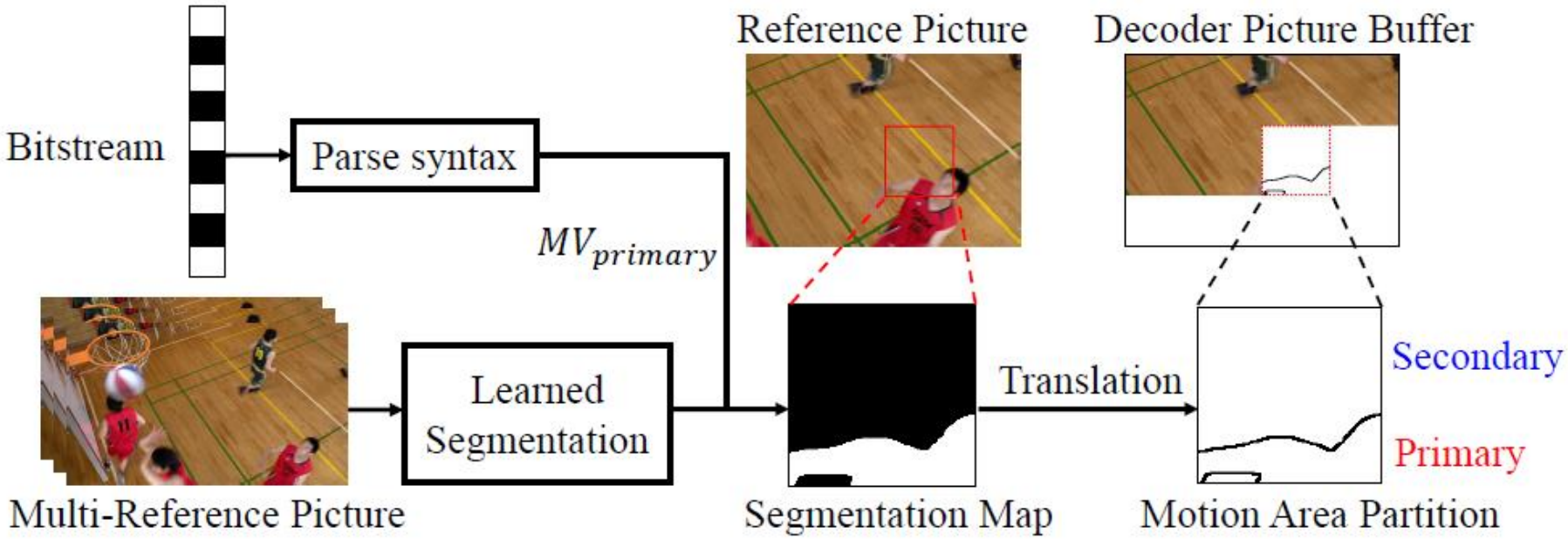
Motion Compensation

(distance-based adaptive weighted strategy)



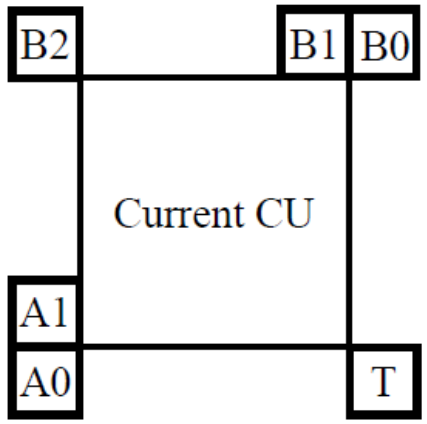
Motion Vector Coding

Partition Derivation

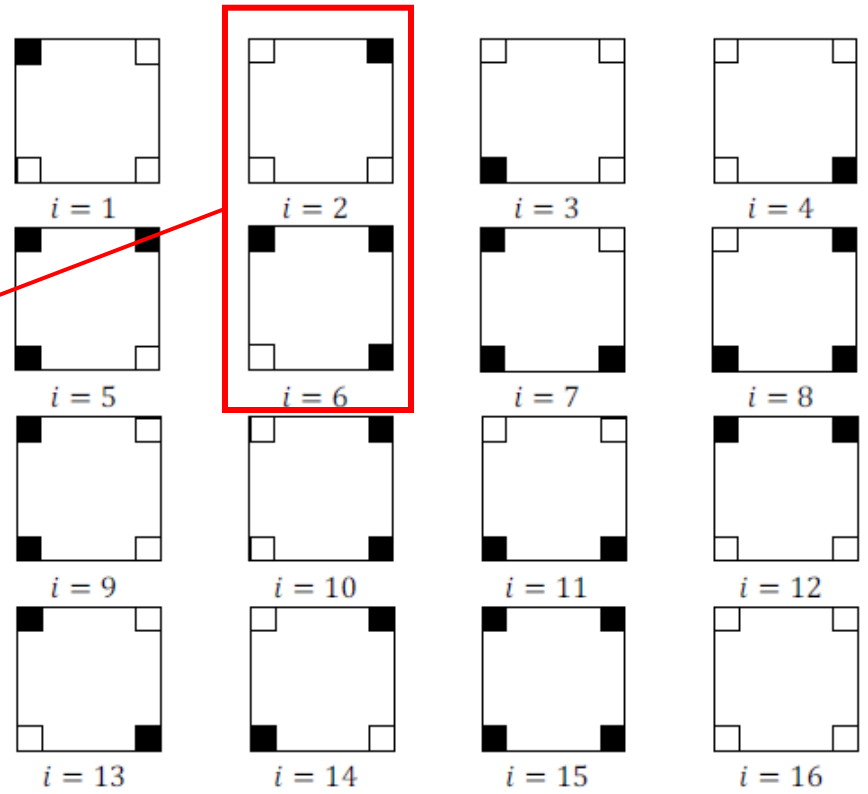


Motion Vector Coding

Primary & Secondary Motion Vector Coding



Positions of spatial and temporal merge candidates



All types of the distribution of motion-inconsistent regions

Type(i)	1/5	2/6	3/7	4/8	9	10	11	12	13	14	15	16
Cand[0]	B2[1]	B0	A0	T	A1	B0	A0	B1	B2	B1	**0**[2]	B1
Cand[1]	B1	B2	B2	B0	B2	T	T	B2	T	A1	B1	A1
Cand[2]	A1	T	T	A0	A0	B1	A1	B0	B1	B0	A1	B0
Cand[3]	B0	B1	A1	B1	**0**	**0**	**0**	**0**	A1	A0	B0	A0

[1] Neighbor CUs in Fig. 6.
[2] Zero vector (0,0).

Candidate list derivation for secondary region

(Blackboard)

Motion Estimation

1 Rough motion vector estimation:

With the primary and secondary candidates constructed, the best combination of these candidates is determined by two stages.

1) Stage 1: The combined primary and secondary candidates are used to conduct the SAMC to generate the prediction of the coding block. The sum of the absolute transformed differences (SATD) of luma between the prediction and the original signal is computed as $\mathrm{SATD}_{\alpha,j,\beta}$. The rate-distortion (RD) cost $J_{\alpha,j,\beta}$ for α, j and β can be sorted by

$$J_{\alpha,j,\beta} = \mathrm{SATD}_{\alpha,j,\beta} + \lambda(R_\alpha + R_j + R_\beta), \tag{11}$$

where R_α, R_j and R_β denote the estimated rates for the primary candidate index, judge index, and secondary candidate index, respectively.

2 Refined motion vector estimation:

2) Stage 2: From stage 1, the best four combined candidates further apply the residual transform coding and CABAC-based rate estimation to obtain the accurate rate cost $R_{\alpha,j,\beta}$. The distortion over three components between these candidates and the original signal is measured by the sum of squared differences (SSD) as $\mathrm{SSD}_{\alpha,j,\beta}$. Finally, the optimal α, j, and β can be selected by

$$J_{\alpha,j,\beta} = \mathrm{SSD}_{\alpha,j,\beta} + \lambda R_{\alpha,j,\beta}. \tag{12}$$

Performance

Class	Sequence	Low-delay P			Low-delay B		
	Name	Y	U	V	Y	U	V
ClassB (1920x1080)	*MarketPlace*	-0.38%	0.34%	-0.89%	-0.42%	-0.61%	-0.74%
	RitualDance	-0.18%	-0.06%	-0.43%	-0.18%	0.09%	0.02%
	Cactus	-0.26%	-0.36%	-0.52%	-0.26%	-0.23%	-0.12%
	BasketballDrive	-0.27%	-0.15%	-0.36%	-0.12%	-0.65%	-0.05%
	BQTerrace	0.06%	-0.56%	0.73%	-0.11%	0.22%	0.30%
	B Average	**-0.21%**	**-0.16%**	**-0.30%**	**-0.22%**	**-0.24%**	**-0.12%**
ClassC (832x480)	*BasketballDrill*	-1.77%	-1.07%	-2.06%	-0.99%	-0.49%	-1.13%
	BQMall	-1.97%	-2.78%	-2.41%	-0.78%	-1.31%	-0.72%
	PartyScene	-0.20%	-0.33%	-0.35%	-0.14%	-0.28%	-0.18%
	RaceHorsesC	-0.94%	-1.26%	-0.24%	-0.18%	0.27%	-0.29%
	C Average	**-1.22%**	**-1.36%**	**-1.27%**	**-0.52%**	**-0.45%**	**-0.58%**
ClassE (1280x720)	*FourPeople*	-0.78%	0.01%	-0.07%	-0.32%	0.34%	-0.33%
	Johnny	-1.79%	-1.41%	-1.76%	-1.11%	-1.99%	0.50%
	KristenAndSara	-0.53%	-1.61%	0.32%	-0.24%	-1.10%	0.09%
	E Average	**-1.04%**	**-1.00%**	**-0.50%**	**-0.56%**	**-0.91%**	**0.09%**
B, C, E Average		**-0.75%**	**-0.77%**	**-0.67%**	**-0.41%**	**-0.48%**	**-0.22%**
ClassD (416x240)	*BasketballPass*	-0.54%	-0.59%	-0.81%	-0.33%	-1.01%	-0.02%
	BQSquare	-0.74%	-2.54%	2.67%	-0.28%	-2.06%	2.44%
	BlowingBubbles	-0.94%	-1.91%	-1.82%	-0.53%	-0.57%	0.69%
	RaceHorses	-0.74%	-0.27%	-2.24%	-0.01%	0.20%	-0.89%
	D Average	**-0.74%**	**-1.33%**	**-0.55%**	**-0.29%**	**-0.86%**	**0.56%**

Ablation

Motion Compensation

ABLATION STUDY ON SAMC BASED ON VTM-12.0

Low-delay P configuration

Class	ONE-STEP			TWO-STEP			SAMC (TWO-STEP+ORMC)		
	Y	U	V	Y	U	V	Y	U	V
B	-0.05%	-0.34%	0.27%	-0.04%	-0.11%	-0.06%	-0.21%	-0.16%	-0.30%
C	-0.36%	-0.94%	-0.72%	-0.77%	-0.99%	-0.94%	-1.22%	-1.36%	-1.27%
E	-0.37%	-0.37%	-1.07%	-0.54%	-0.50%	-0.45%	-1.04%	-1.00%	-0.50%
Avg.	**-0.23%**	**-0.55%**	**-0.40%**	**-0.41%**	**-0.50%**	**-0.45%**	**-0.75%**	**-0.77%**	**-0.67%**
D	-0.55%	-0.46%	-0.45%	-0.70%	-0.56%	-1.17%	-0.74%	-1.33%	-0.55%

Low-delay B configuration

Class	ONE-STEP			TWO-STEP			SAMC (TWO-STEP+ORMC)		
	Y	U	V	Y	U	V	Y	U	V
B	-0.01%	-0.09%	0.11%	-0.06%	-0.18%	0.15%	-0.22%	-0.24%	-0.12%
C	-0.12%	-0.36%	-0.10%	-0.33%	-0.50%	-0.65%	-0.52%	-0.45%	-0.58%
E	-0.24%	-0.11%	-0.02%	-0.27%	0.06%	0.02%	-0.56%	-0.91%	0.09%
Avg.	**-0.10%**	**-0.19%**	**0.01%**	**-0.20%**	**-0.23%**	**-0.15%**	**-0.41%**	**-0.48%**	**-0.22%**
D	-0.19%	0.06%	0.63%	-0.40%	-0.16%	0.04%	-0.29%	-0.86%	-0.56%

Motion Vector Coding

ABLATION STUDY ON SAMVC BASED ON VTM-12.0

Low-delay P configuration

Class	w/o SAMVC			w/ SAMVC		
	Y	U	V	Y	U	V
B	-0.09%	-0.32%	0.20%	-0.21%	-0.16%	-0.30%
C	-0.95%	-1.20%	-1.13%	-1.22%	-1.36%	-1.27%
E	-0.54%	-0.88%	-0.49%	-1.04%	-1.00%	-0.50%
Avg.	**-0.49%**	**-0.76%**	**-0.58%**	**-0.75%**	**-0.77%**	**-0.67%**
D	-0.63%	-1.04%	-0.22%	-0.74%	-1.33%	-0.55%

Low-delay B configuration

Class	w/o SAMVC			w/ SAMVC		
	Y	U	V	Y	U	V
B	-0.07%	-0.17%	-0.16%	-0.22%	-0.24%	-0.12%
C	-0.20%	-0.35%	-0.33%	-0.52%	-0.45%	-0.58%
E	-0.21%	-0.04%	-0.17%	-0.56%	-0.91%	0.09%
Avg.	**-0.15%**	**-0.20%**	**-0.22%**	**-0.41%**	**-0.48%**	**-0.22%**
D	-0.15%	-0.13%	0.20%	-0.29%	-0.86%	-0.56%

Selction & Time complexity

TIME COMPLEXITY

Class	Low-delay P		Low-delay B	
	EncT	DecT	EncT	DecT
B	424%	114%	409%	111%
C	398%	109%	370%	108%
D	398%	95%	356%	109%
E	454%	87%	400%	100%
Overall	419%	101%	384%	107%

*:the rule of learning-based coding paper

(a) GEO (BQMall, QP: 37, POC : 68)

(b) GEO + SAIP (BQMall, QP : 37, POC: 68)

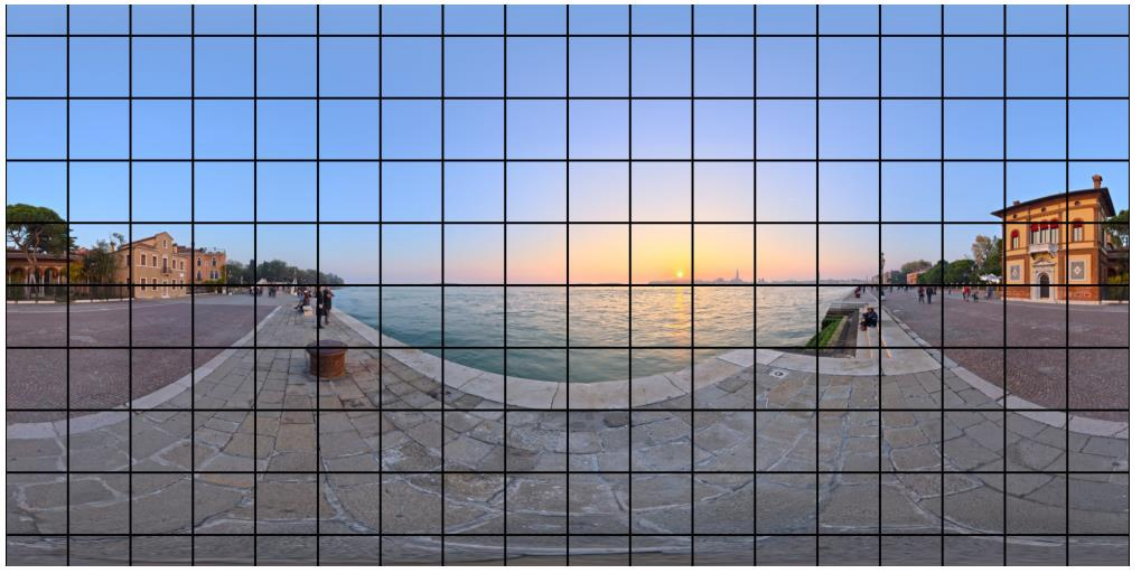
360-Degree coding

Motion-Plane-Adaptive Inter Prediction in 360-Degree Video Coding

Andy Regensky, Christian Herglotz, *Member, IEEE*, and André Kaup, *Fellow, IEEE*

(b) sphere mapping (front)

(c) sphere mapping (back)



(a) ERP-projected 360-degree image

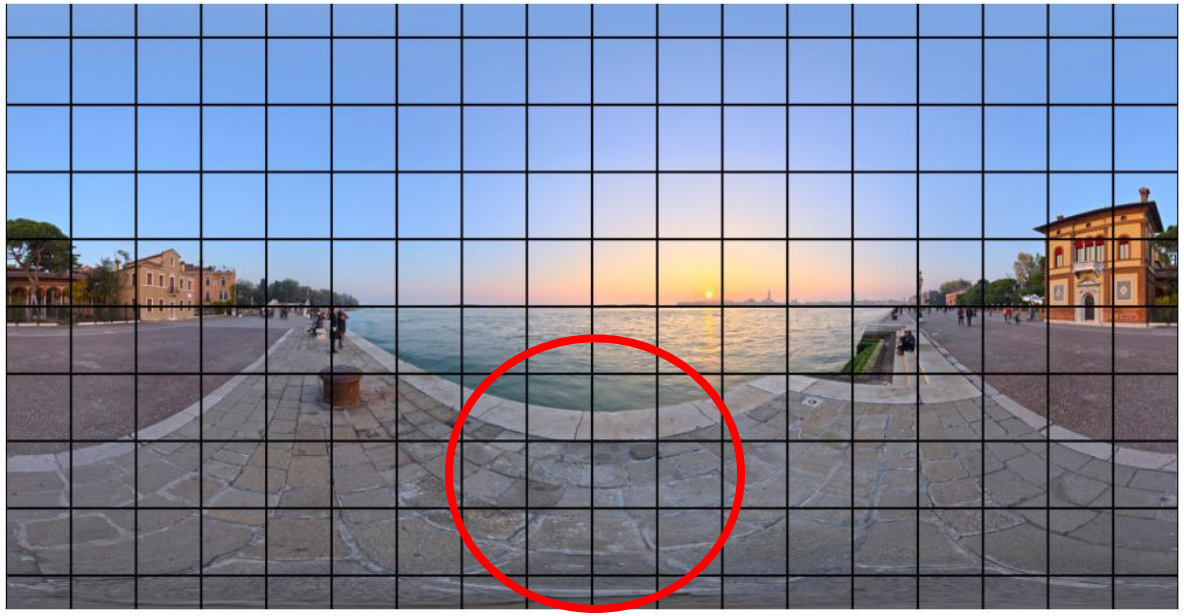
(b) sphere mapping (front)

(c) sphere mapping (back)

In this paper, we propose a motion-plane-adaptive inter prediction technique (MPA) for 360-degree video that takes the spherical characteristics of 360-degree video into account. Based on the known projection format of the video, MPA allows to perform inter prediction on different motion planes in 3D space instead of having to work on the - in theory arbitrarily mapped - 2D image representation directly. We furthermore derive a motion–plane-adaptive motion vector prediction technique (MPA-MVP) that allows to translate motion information between different motion planes and motion models.

传统360的编码方法 ⇨ 3D的编码方法

The key question of sphere video coding



(a) ERP-projected 360-degree image

Typically, a 2D representation of the 360-degree video is required to allow compression using existing video coding techniques such as the H.264/AVC, the H.265/HEVC or the H.266/VVC, video coding standards. Fig(a) shows an example of a 360-degree image mapped to the 2D image plane through an equirectangular projection (ERP). While this is one of the most common 360-degree projection formats, there exist a plethora of other formats including various variations of cubemap projections, segmented and rotated sphere projections, or octa- and isocahedron projections to name only a few [5].

While the black lines of constant azimuthal (vertical) and polar (horizontal) angles form a block structure in the ERP-projected image, this is not the case in the spherical domain. It is clearly visible that the different blocks become increasingly distorted with a higher distance to the equator.

In this work, we propose a novel *motion-plane-adaptive* inter prediction technique (MPA) for 360-degree video that allows to perform inter prediction on different motion planes in 3D space. Any motion on these planes is modeled purely using horizontal and vertical shifts while the motion planes themselves can be oriented freely in 3D space. In this way, MPA takes both the spherical characteristics of 360-degree video and the translational nature of most camera and object motion into account. MPA thus is able to more accurately reproduce the resulting pixel shifts in the 2D projection domain than classical translational techniques are able to. Due to their narrow field of view, such 3D space considerations are not necessary for conventional perspective video. To further improve the performance of MPA and make it compatible to existing inter prediction techniques, we additionally derive an efficient method to transfer motion information between different motion planes and motion models.

1 如何利用投影来在2D上进行更准确的3D帧间预测

2 不改变投影方式，如何完成3D的帧间预测

3 如何将2D的帧间预测在3D上更好的发挥

1 An overview over related approaches to improving 360-degree video coding is given.

2 Briefly recapitulates the traditional inter prediction procedure,

3 Introduce the proposed MPA. Within this section, the projection functions required for motion-plane adaptivity are introduced including a generalized formulation of the perspective projection, the motion-plane-adaptive motion model is presented, an adapted motion vector prediction method is derived.

Related work:

Y. Wang, L. Li, D. Liu, F. Wu, and W. Gao, "A New Motion Model for Panoramic Video Coding," in ICIP., Sep 2017, pp. 1407–1411.

Y. Wang, D. Liu, S. Ma, F. Wu, and W. Gao, "Spherical Coordinates Transform-Based Motion Model for Panoramic Video Coding," IEEE JETCAS., vol. 9, no. 1, pp. 98–109, Mar 2019.

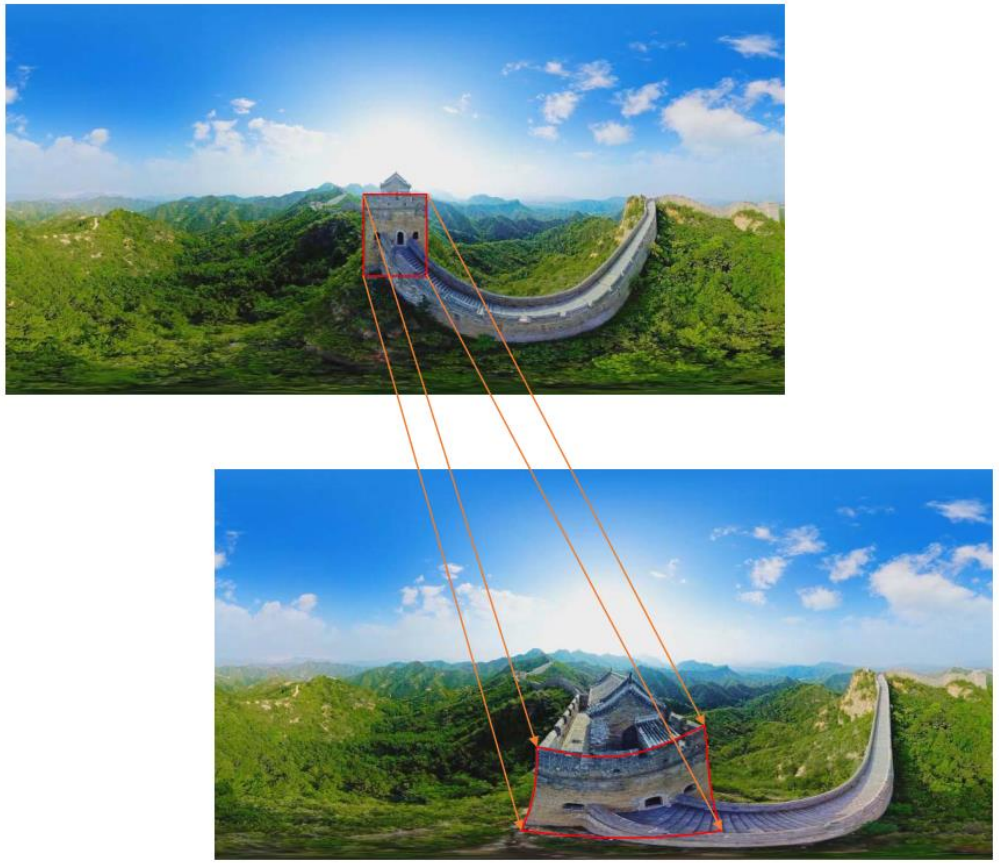
L. Li, Z. Li, M. Budagavi, and H. Li, "Projection Based Advanced Motion Model for Cubic Mapping for 360-Degree Video," in ICIP., Sep 2017, pp. 1427–1431.

L. Li, Z. Li, X. Ma, H. Yang, and H. Li, "Advanced Spherical MotionModel and Local Padding for 360° Video Compression," TIP., vol. 28, no. 5, pp. 2342–2356, May 2019.

Related work

Wang et al. propose a 3D translational motion model, where all pixels in a regarded block on the sphere are shifted in 3D space according to a 3D motion vector derived from the original 2D motion vector.

A similar approach is followed by *Li et al.* in [12],[13], where the 3D motion vector is derived based on the assumption that two neighboring blocks adhere to the same motion in 3D space (local padding and frame padding).



Y.F Wang 's Method

(a) (b)

Assumption: Motion in sphere is translation.

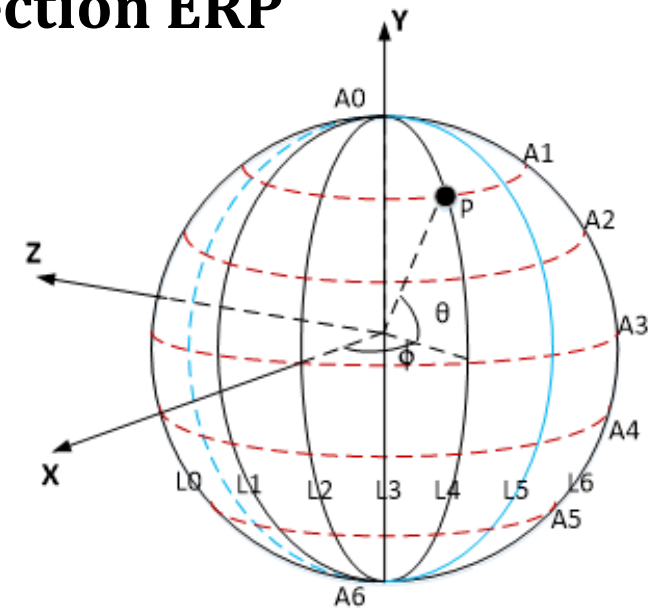
L.Li 's Method

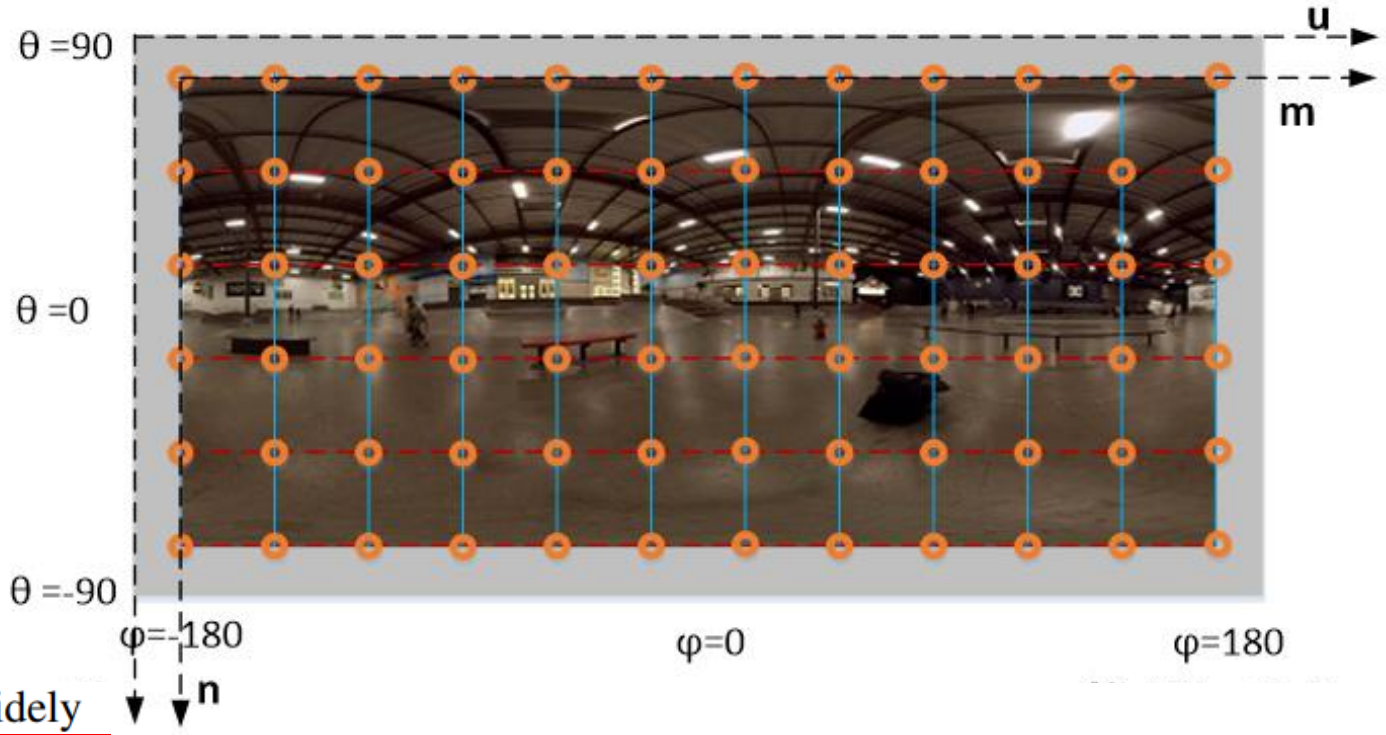
Projection ERP

As MPA is based on the known mappings between the 2D image plane and the 3D space representations of a 360-degree video, a general formulation of these mappings in the form of projection functions is required. Any valid projection function $\boldsymbol{\xi} : \mathcal{S} \rightarrow \mathbb{R}^2$ is invertible and describes the relation between a 3D space coordinate $\boldsymbol{s} = (x, y, z)^T \in \mathcal{S}$ on the unit sphere and the corresponding pixel coordinate $\boldsymbol{p} = (u, v)^T \in \mathbb{R}^2$ on the 2D image plane, where $\mathcal{S} = \{\boldsymbol{s} \in \mathbb{R}^3 \mid \|\boldsymbol{s}\|_2 = 1\}$ describes the set of all coordinates on the unit sphere. The inverse projection function $\boldsymbol{\xi}^{-1} : \mathbb{R}^2 \rightarrow \mathcal{S}$ maps the 2D image plane coordinate back to the unit sphere in 3D space.

任何的sphere投影都是一种可逆的过程

Projection ERP





The equirectangular projection $\boldsymbol{\xi}_{\text{erp}}$ is a popular and widely applied example of a general 360-degree projection. It maps the polar angle $\theta \in [0, \pi]$ to the vertical v-axis and the azimuthal angle $\varphi \in [0, 2\pi]$ to the horizontal u-axis of the 2D image plane. For projecting a 3D space coordinate $\boldsymbol{s}$ on the unit sphere to the 2D image plane, its spherical angles (θ, φ) according to Fig. 2 need to be obtained first through

$$\theta = \arccos(z), \tag{4}$$

$$\varphi = \arctan2(y, x), \tag{5}$$

where $\arctan2$ describes the four-quadrant arctangent. The spherical angles are then projected to the 2D image plane yielding the pixel coordinate $\boldsymbol{p}_{\text{erp}} = (u_{\text{erp}}, v_{\text{erp}})^T \in \mathbb{R}^2$ as

$$u_{\text{erp}} = \frac{\varphi}{2\pi} \cdot U, \tag{6}$$

$$v_{\text{erp}} = \frac{\theta}{\pi} \cdot V, \tag{7}$$

where U describes the width and V the height of the 2D image plane in pixels. Typically, $U = 2V$ in case of the equirectangular projection as the azimuthal angle φ has twice the angular range compared to the polar angle θ. The equirectangular projection function $\boldsymbol{\xi}_{\text{erp}}$ combines steps (4)-(7) in a concise expression.

Projection ERP

For the inverse equirectangular projection $\boldsymbol{\xi}_{\text{erp}}^{-1}$, the pixel coordinate on the 2D image plane is projected back to the spherical domain through

$$\varphi = \frac{u_{\text{erp}}}{U} \cdot 2\pi, \tag{8}$$

$$\theta = \frac{v_{\text{erp}}}{V} \cdot \pi, \tag{9}$$

before the final pixel coordinate on the unit sphere is obtained as

$$x = \sin(\theta)\cos(\varphi), \tag{10}$$

$$y = \sin(\theta)\sin(\varphi), \tag{11}$$

$$z = \cos(\theta). \tag{12}$$

Projection Generalized Perspective Projection

The orientation of the applied 3D coordinate sytem (x, y, z) is visualized in black in Fig. 2. Thereby, y is oriented horizontally, z is oriented vertically and x is oriented perpendicular to y and z. The default camera is positioned at the origin and oriented in negative x-direction. The rotated blue coordinate system (x', y', z') is an intermediate system for the generalized perspective projection, which will be introduced later.

For MPA, two projection functions are important. First, the employed 360-degree projection $\boldsymbol{\xi}_{\mathrm{o}}$ of the given video, and second, the perspective projection $\boldsymbol{\xi}_{\mathrm{p}}$ for representing motion on the desired motion planes in 3D space.

只需要坐标系的转换

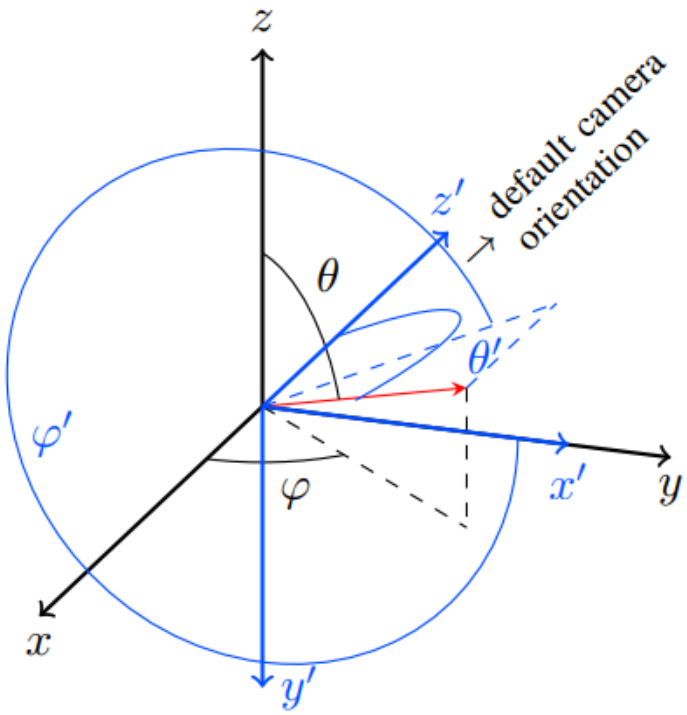


Fig. 2. The employed 3D coordinate systems. The black coordinate system (x, y, z) describes the main system orientation where y is oriented horizontally, z is oriented vertically, and x is oriented perpendicular to y and z. The default camera is positioned at the origin and oriented in negative x-direction. θ and φ denote the corresponding polar and angular angles in spherical coordinates. The blue coordinate system (x', y', z') describes an intermediate system used for the perspective projection where the virtual perspective camera is oriented in positive z'-direction. The corresponding polar and angular angles θ' and φ' are given in blue.

Projection Generalized Perspective Projection

The inverse projection then entails the following steps. First, the polar coordinates (r_p, φ') need to be calculated from the pixel coordinate $\boldsymbol{p}_\mathrm{p}$

$$r_\mathrm{p} = \sqrt{u_\mathrm{p}^2 + v_\mathrm{p}^2}, \tag{19}$$

$$\varphi' = \operatorname{arctan2}(v_\mathrm{p}, u_\mathrm{p}). \tag{20}$$

The corresponding incident angle is then obtained using

$$\theta' = \begin{cases} \arctan\left(r_\mathrm{p}/f\right) & \text{if } b_\mathrm{vip} = 0, \\ \pi - \arctan\left(r_\mathrm{p}/f\right) & \text{if } b_\mathrm{vip} = 1. \end{cases} \tag{21}$$

Finally, the pixel coordinate $\boldsymbol{s}$ in the spherical domain results in

$$x = -\cos(\theta'), \tag{22}$$

$$y = \sin(\theta')\cos(\varphi'), \tag{23}$$

$$z = -\sin(\theta')\sin(\varphi'). \tag{24}$$

转换方式

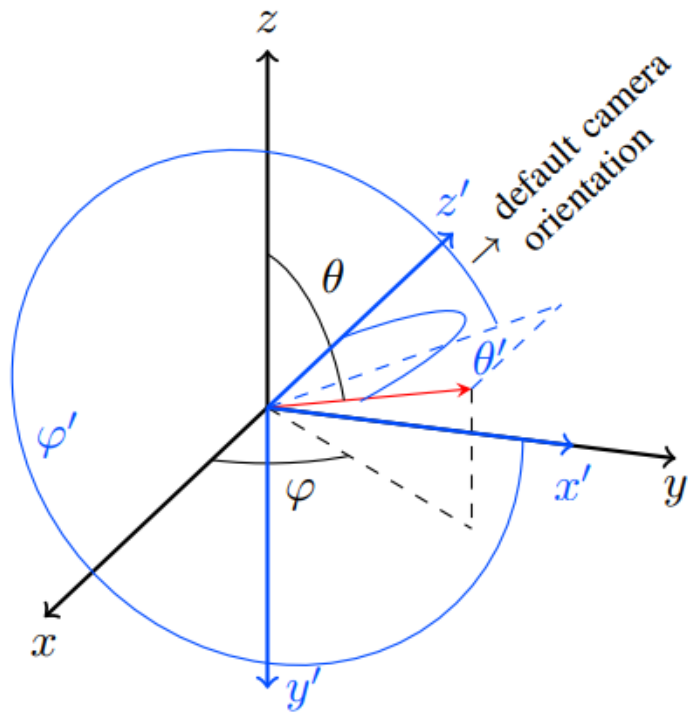
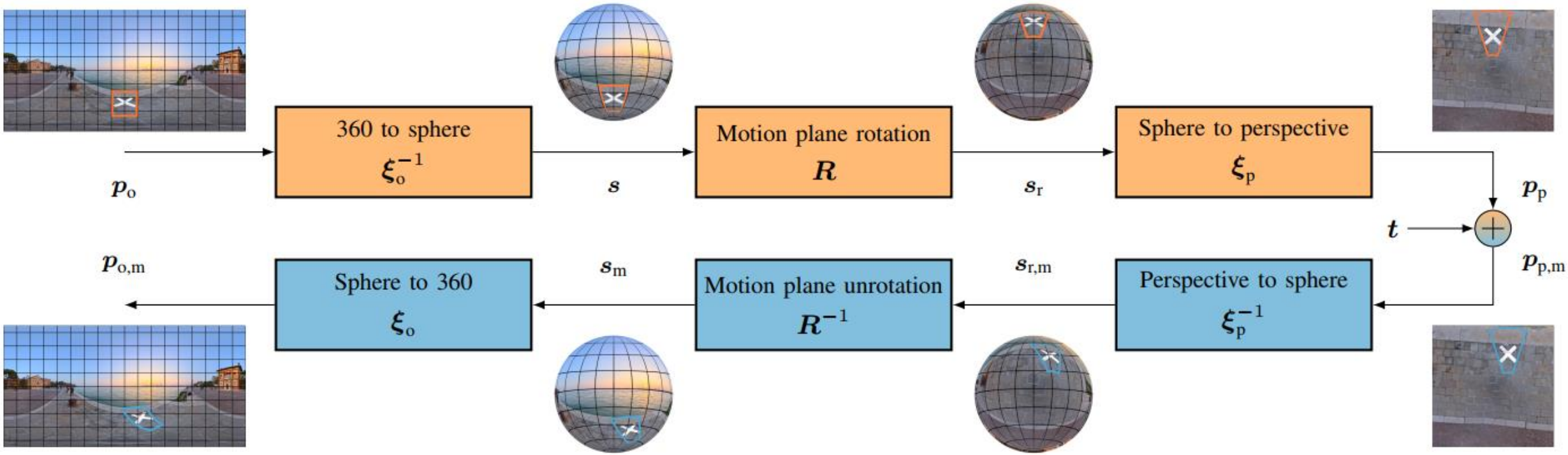


Fig. 2. The employed 3D coordinate systems. The black coordinate system (x, y, z) describes the main system orientation where y is oriented horizontally, z is oriented vertically, and x is oriented perpendicular to y and z. The default camera is positioned at the origin and oriented in negative x-direction. θ and φ denote the corresponding polar and angular angles in spherical coordinates. The blue coordinate system (x', y', z') describes an intermediate system used for the perspective projection where the virtual perspective camera is oriented in positive z'-direction. The corresponding polar and angular angles θ' and φ' are given in blue.

Motion model



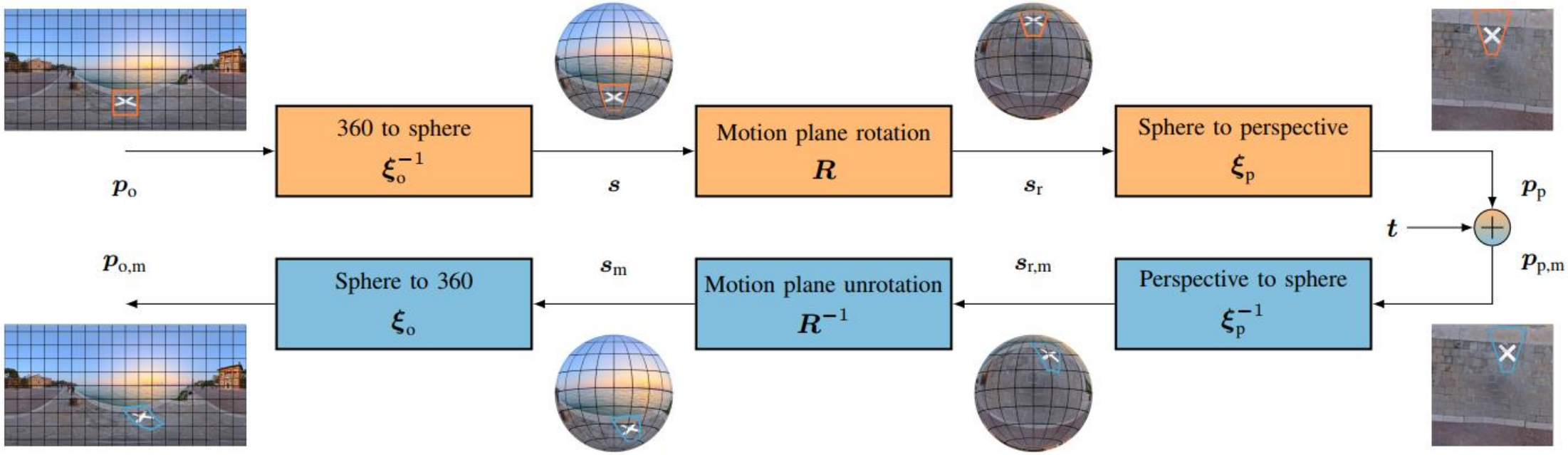
Based on the generalized perspective projection function ξ_p and a given 360-degree projection function ξ_o such as the equirectangular projection function ξ_{erp} described in Section IV-A, the motion-plane-adaptive motion model is derived as follows.

In a first step, the original 360-degree pixel coordinate p_o (in, e.g., the ERP domain) is projected to the pixel coordinate p_p on the desired motion plane using

$$p_p = \xi_p \left(R \xi_o^{-1}(p_o) \right), \tag{25}$$

where ξ_o^{-1} projects the original 360-degree pixel coordinate p_o to the unit sphere, the motion plane rotation matrix $R \in \mathbb{R}^{3\times3}$ rotates the pixel coordinate on the unit sphere according to the desired motion plane orientation, and ξ_p then projects the pixel coordinate onto the motion plane.

Motion model



In a second step, the translational motion according to the motion vector $\boldsymbol{t}$ is performed on the obtained motion plane yielding the moved pixel coordinate on the motion plane

$$\boldsymbol{p}_{\mathrm{p,m}} = \boldsymbol{p}_{\mathrm{p}} + \boldsymbol{t}. \tag{26}$$

In the final third step, the moved pixel coordinate $\boldsymbol{p}_{\mathrm{p,m}}$ on the motion plane is projected back to the original 360-degree format to obtain the moved pixel coordinate in the 360-degree projection

$$\boldsymbol{p}_{\mathrm{o,m}} = \boldsymbol{\xi}_{\mathrm{o}} \left(\boldsymbol{R}^{-1} \boldsymbol{\xi}_{\mathrm{p}}^{-1} (\boldsymbol{p}_{\mathrm{p,m}}) \right), \tag{27}$$

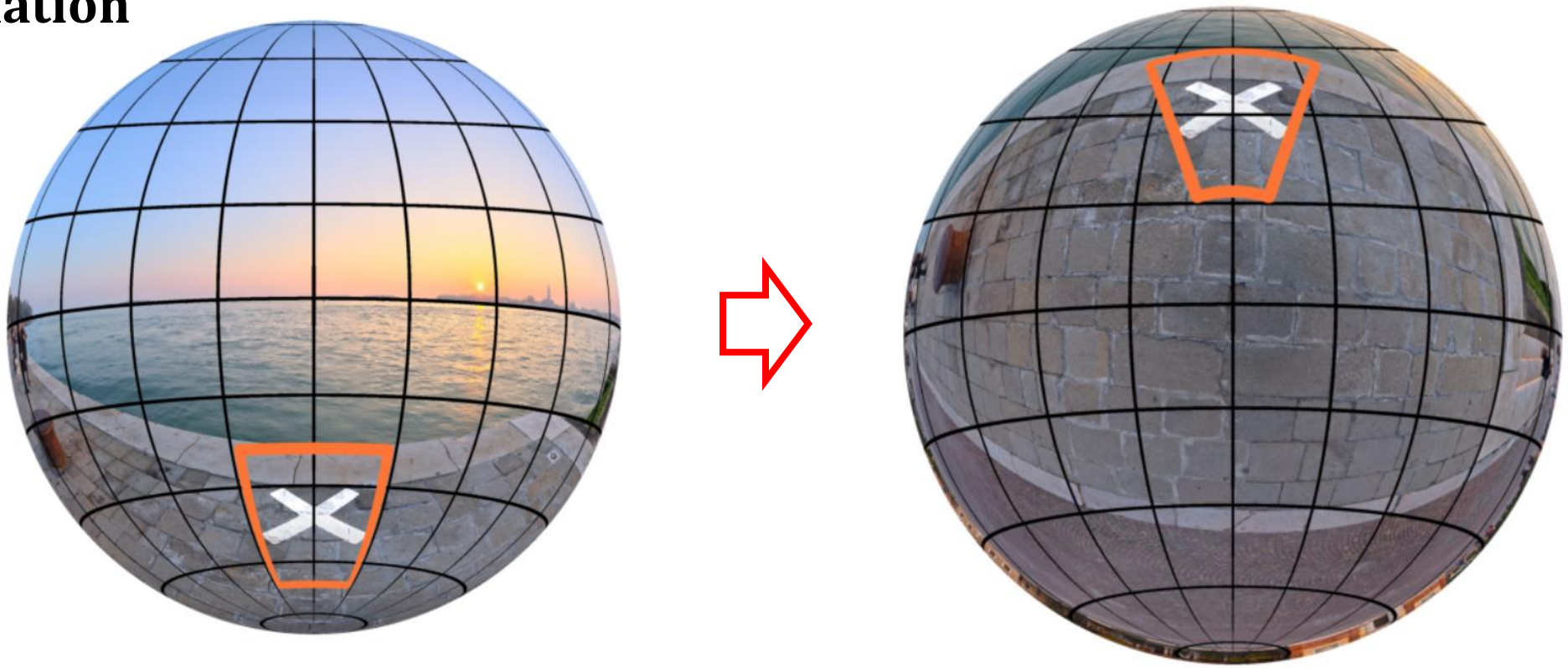
where $\boldsymbol{R}^{-1} = \boldsymbol{R}^{T}$.

Putting steps (25)-(27) together, the overall motion model $\boldsymbol{m}_{\mathrm{mpa}}$ is defined as

$$\boldsymbol{m}_{\mathrm{mpa}}(\boldsymbol{p}_{\mathrm{o}}, \boldsymbol{t}, \boldsymbol{R}) = \boldsymbol{\xi}_{\mathrm{o}} \left(\boldsymbol{R}^{-1} \boldsymbol{\xi}_{\mathrm{p}}^{-1} \left(\boldsymbol{\xi}_{\mathrm{p}} \left(\boldsymbol{R} \boldsymbol{\xi}_{\mathrm{o}}^{-1} (\boldsymbol{p}_{\mathrm{o}}) \right) + \boldsymbol{t} \right) \right). \tag{28}$$

A schematic representation of the described motion model is shown in Fig. 4. The figure also visualizes block motion for an exemplary block in an ERP-projected 360-degree image, where it is clearly visible that the proposed motion model is able to accurately replicate the distortions of the block in the ERP domain resulting from a translational motion on the street surface.

Motion Estimation

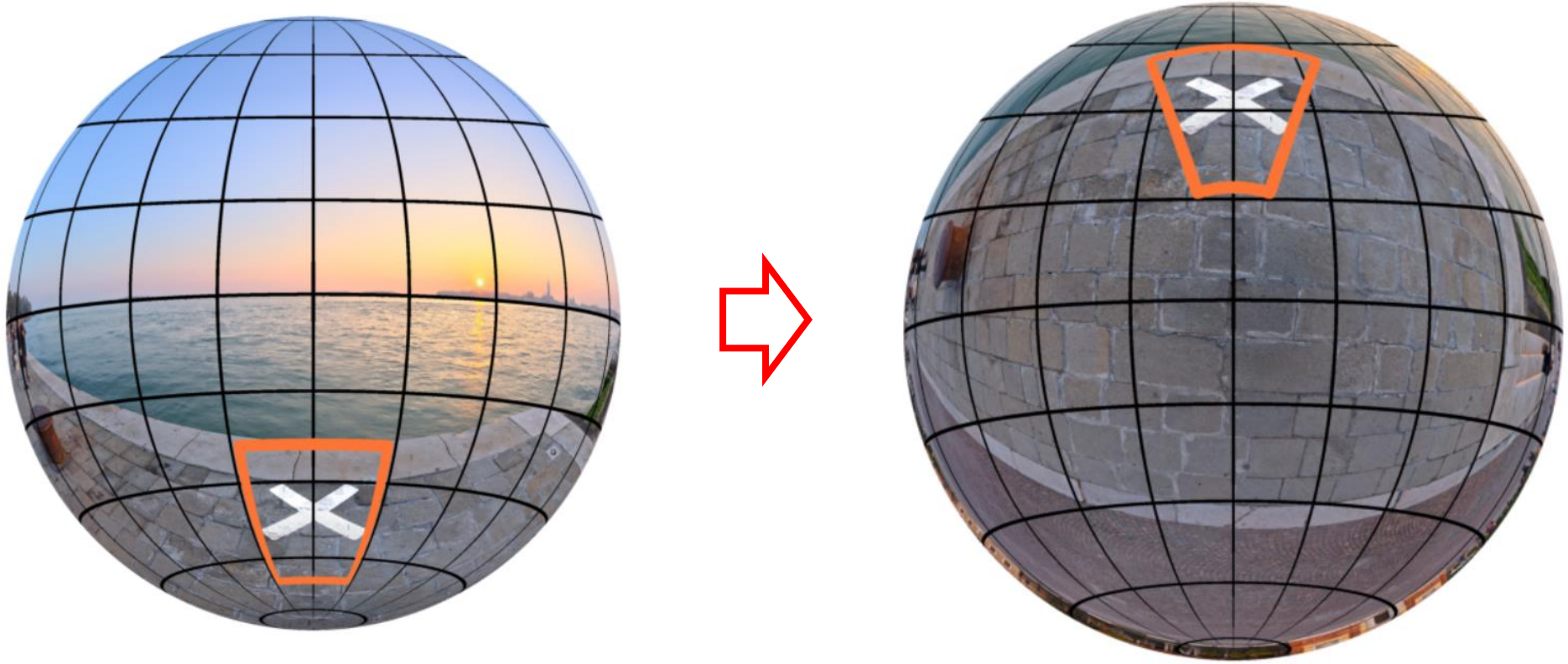


Rotations by multiples of 90 degrees around one or more axes can be formulated through simple transpositions of the 3D space coordinates, such that a suitably defined set of motion planes allows to considerably speed up the calculations for the inherent coordinate rotations. We thus formulate a limited set of three rotation matrices leading to the motion planes

- front/back: No rotation,
- left/right: $\pi/2$ around z-axis,
- top/bottom: $\pi/2$ around y-axis.

A potential encoder can then select the best matching motion plane through rate-distortion optimization. Please note, however, that in general, the motion-plane-adaptive motion model is not limited to these motion planes and could employ arbitrary motion plane rotation matrices $\boldsymbol{R}$.

Motion Estimation



However, in the context of MPA, the motion information at the different candidate positions could be represented on motion planes differing from the investigated motion plane for the currently regarded block. Hence, a method to efficiently translate motion information, i.e., motion vectors, between different motion planes is required.

Motion estimation in different planes

However, in the context of MPA, the motion information at the different candidate positions could be represented on motion planes differing from the investigated motion plane for the currently regarded block. Hence, a method to efficiently translate motion information, i.e., motion vectors, between different motion planes is required.

According to Fig. 5, let's assume a motion vector $\boldsymbol{t}_\mathrm{s}$ and a motion plane rotation matrix $\boldsymbol{R}_\mathrm{s}$ are obtained at a source candidate pixel coordinate $\boldsymbol{p}_\mathrm{s}$. To transform the motion vector to a different target motion plane described by the rotation matrix $\boldsymbol{R}_\mathrm{t}$, we need to find a motion vector $\boldsymbol{t}_\mathrm{t}$ on the target motion plane that results in an identical pixel shift in the 360-degree domain as the motion vector $\boldsymbol{t}_\mathrm{s}$ on the source motion plane, i.e.,

$$\boldsymbol{m}_\mathrm{mpa}(\boldsymbol{p}_\mathrm{s}, \boldsymbol{t}_\mathrm{s}, \boldsymbol{R}_\mathrm{s}) \stackrel{!}{=} \boldsymbol{m}_\mathrm{mpa}(\boldsymbol{p}_\mathrm{s}, \boldsymbol{t}_\mathrm{t}, \boldsymbol{R}_\mathrm{t}). \tag{29}$$

The source pixel coordinate $\boldsymbol{p}_\mathrm{s}$ is hereby used as an anchor for the motion plane translation. By inserting (28) into (29), we can solve for the required motion vector on the target motion plane as

$$\begin{aligned}\boldsymbol{t}_\mathrm{t} = \; & \boldsymbol{\xi}_\mathrm{p}\left(\boldsymbol{R}_\mathrm{t}\boldsymbol{R}_\mathrm{s}^{-1}\boldsymbol{\xi}_\mathrm{p}^{-1}\left(\boldsymbol{\xi}_\mathrm{p}\left(\boldsymbol{R}_\mathrm{s}\boldsymbol{\xi}_\mathrm{o}^{-1}(\boldsymbol{p}_\mathrm{s})\right) + \boldsymbol{t}_\mathrm{s}\right)\right) \\ & - \boldsymbol{\xi}_\mathrm{p}\left(\boldsymbol{R}_\mathrm{t}\boldsymbol{\xi}_\mathrm{o}^{-1}(\boldsymbol{p}_\mathrm{s})\right).\end{aligned}$$

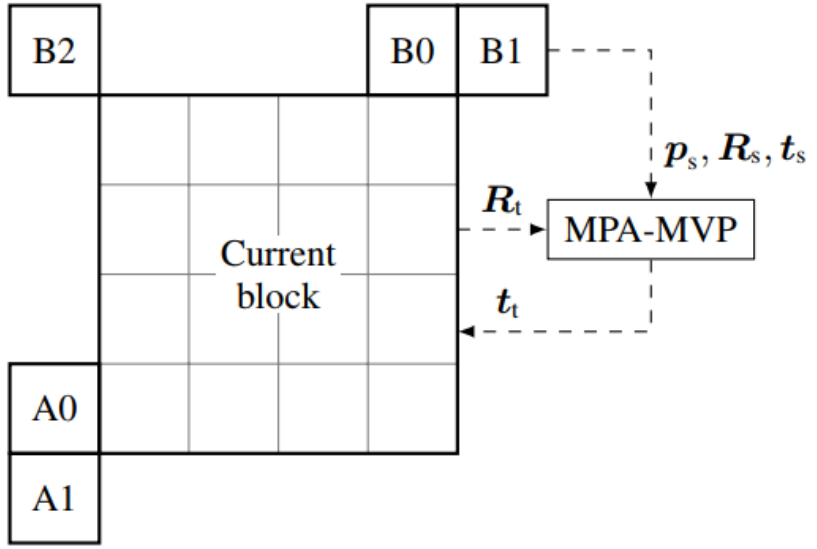


Fig. 5. Spatial motion vector predictor candidates in the H.266/VVC video coding standard [24] with a schmematic illustration of MPA-MVP for candidate position B1 at pixel coordinate $\boldsymbol{p}_\mathrm{s}$. With $\boldsymbol{R}_\mathrm{s}$ describing the motion plane of the candidate position and $\boldsymbol{R}_\mathrm{t}$ describing the motion plane of the current block, MPA-MVP translates the motion vector $\boldsymbol{t}_\mathrm{s}$ from B1 to the corresponding motion vector predictor $\boldsymbol{t}_\mathrm{t}$ for the current block.

$$\begin{aligned} \boldsymbol{t}_\mathrm{t} = \; & \boldsymbol{\xi}_\mathrm{p}\left(\boldsymbol{R}_\mathrm{t}\boldsymbol{R}_\mathrm{s}^{-1}\boldsymbol{\xi}_\mathrm{p}^{-1}\left(\boldsymbol{\xi}_\mathrm{p}\left(\boldsymbol{R}_\mathrm{s}\boldsymbol{\xi}_\mathrm{o}^{-1}(\boldsymbol{p}_\mathrm{s})\right)+\boldsymbol{t}_\mathrm{s}\right)\right) \\ & -\boldsymbol{\xi}_\mathrm{p}\left(\boldsymbol{R}_\mathrm{t}\boldsymbol{\xi}_\mathrm{o}^{-1}(\boldsymbol{p}_\mathrm{s})\right). \end{aligned}$$

Thereby, the first row explains, where the candidate pixel coordinate moved by $\boldsymbol{t}_\mathrm{s}$ on the source motion plane lies on the target motion plane, and the second row explains, where the *"unmoved"* candidate pixel coordinate lies on the target motion plane. Fig. 5 visualizes the general procedure of the described motion-plane-adaptive motion vector prediction (MPA-MVP) for an exemplary candidate position B1.

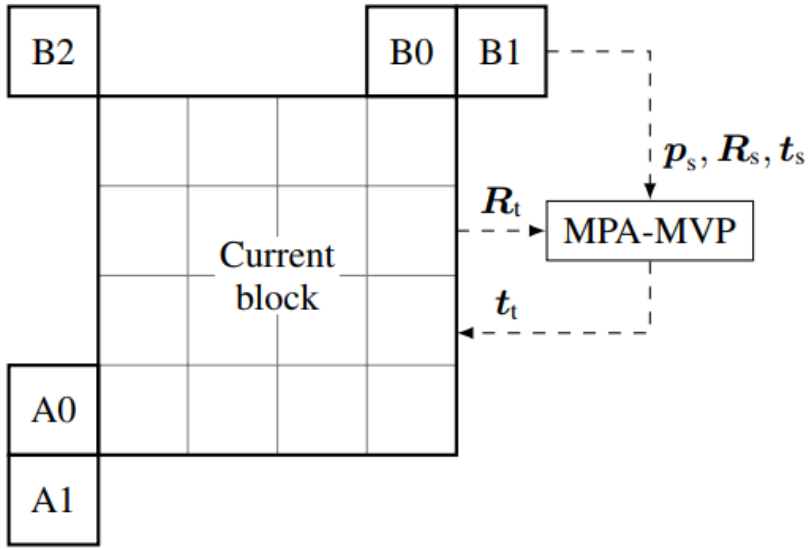
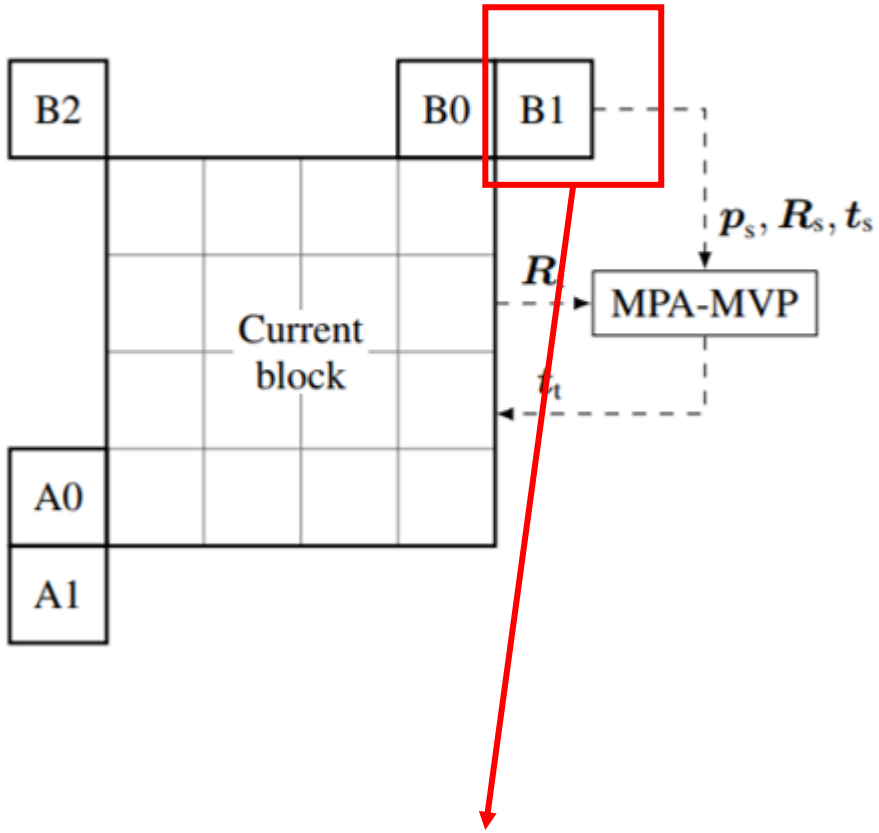


Fig. 5. Spatial motion vector predictor candidates in the H.266/VVC video coding standard [24] with a schmematic illustration of MPA-MVP for candidate position B1 at pixel coordinate $\boldsymbol{p}_\mathrm{s}$. With $\boldsymbol{R}_\mathrm{s}$ describing the motion plane of the candidate position and $\boldsymbol{R}_\mathrm{t}$ describing the motion plane of the current block, MPA-MVP translates the motion vector $\boldsymbol{t}_\mathrm{s}$ from B1 to the corresponding motion vector predictor $\boldsymbol{t}_\mathrm{t}$ for the current block.

Furthermore, to integrate MPA as an additional tool alongside the existing translational inter prediction procedure, MPA-MVP does not only need to be able to translate motion vectors between the different motion planes of MPA, but also between MPA and the classical translational motion model

$$\boldsymbol{m}_{\text{mpa}}(\boldsymbol{p}_{\text{s}}, \boldsymbol{t}_{\text{s/t}}, \boldsymbol{R}_{\text{s/t}}) \stackrel{!}{=} \boldsymbol{m}_{\text{t}}(\boldsymbol{p}_{\text{s}}, \boldsymbol{t}_{\text{t/s}}), \tag{31}$$

where both the translational or the motion-plane-adaptive motion model could be the source motion model of the prediction candidate (noted with the subscript s/t or vice versa, i.e., if one motion model is source, then the other motion model is target and the other way around).



Not in adaptive plane

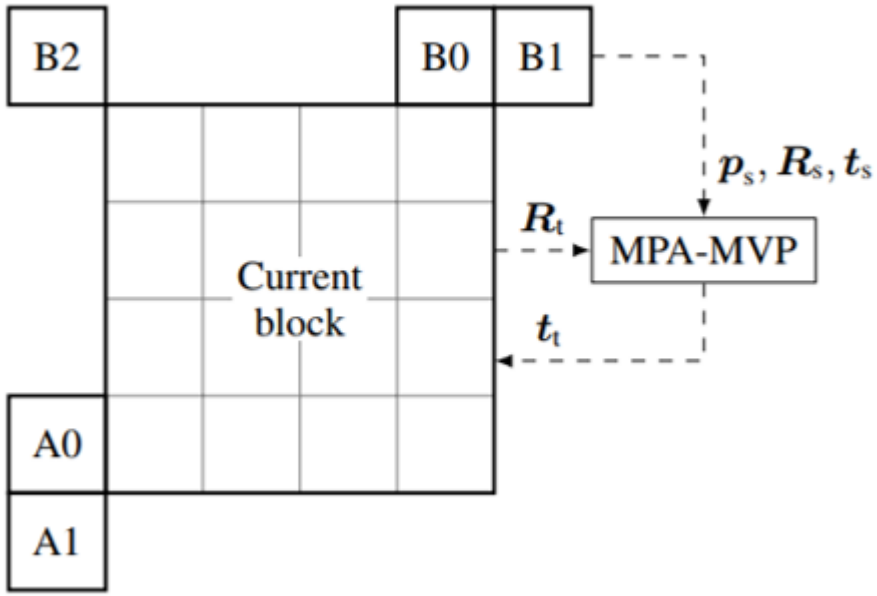
In case the motion-plane-adaptive motion model is the source model, an equivalent motion vector $\boldsymbol{t}_t$ for the classical translational motion model can be derived based on the motion vector $\boldsymbol{t}_s$ on the source motion plane as

$$\boldsymbol{t}_t = \boldsymbol{\xi}_o\left(\boldsymbol{R}_s^{-1}\boldsymbol{\xi}_p^{-1}\left(\boldsymbol{\xi}_p\left(\boldsymbol{R}_s\boldsymbol{\xi}_o^{-1}(\boldsymbol{p}_s)\right)+\boldsymbol{t}_s\right)\right)-\boldsymbol{p}_s. \quad (32)$$

On the other hand, in case the classical translational motion model is the source model, an equivalent motion vector $\boldsymbol{t}_t$ for the motion-plane-adaptive motion model can be derived based on the source motion vector $\boldsymbol{t}_s$ and the desired target motion plane as

$$\boldsymbol{t}_t = \boldsymbol{\xi}_p\left(\boldsymbol{R}_t\boldsymbol{\xi}_o^{-1}(\boldsymbol{p}_s+\boldsymbol{t}_s)\right)-\boldsymbol{\xi}_p\left(\boldsymbol{R}_t\boldsymbol{\xi}_o^{-1}(\boldsymbol{p}_s)\right). \quad (33)$$

The key differences between (32) and (33) are the domain in which the source motion vector is applied, and the domain in which the target motion vector is calculated. In (32), the source motion vector is applied on the source motion plane and the target motion vector is calculated in the 360-degree domain, whereas in (33), the source motion vector is applied in the 360-degree domain and the target motion vector is calculated on the target motion plane.



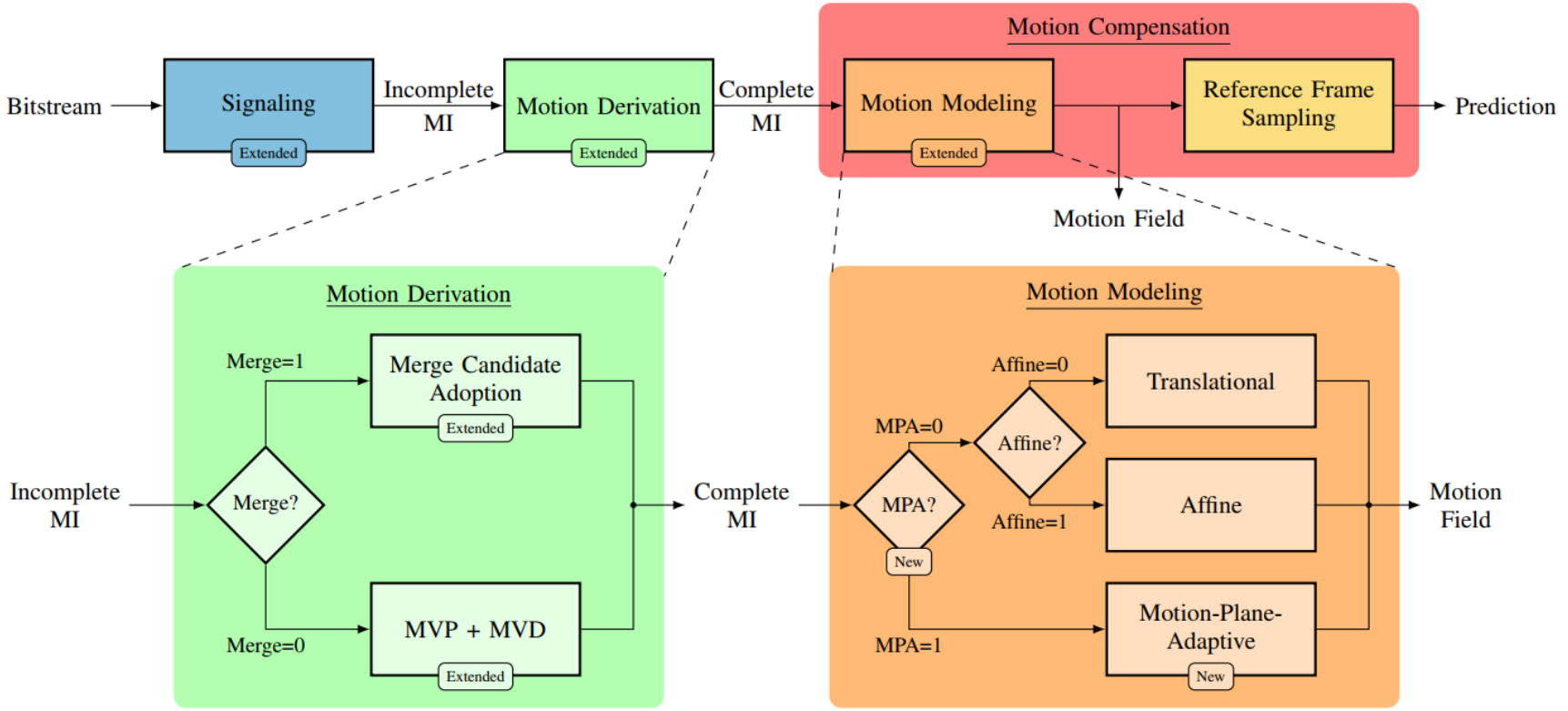


Fig. 6. Schematic representation of the decoder-side inter prediction pipeline in the H.266/VVC video coding standard including the proposed MPA tool. Components that need to be extended or added with respect to the original H.266/VVC inter prediction pipeline are labeled explicitly. For details on the performed extensions and additions, please see the text.

If a merge mode is signaled or the current CU is coded using an affine motion model, no adaptions to the signaling procedure are required for MPA. If no merge mode is signaled and the current CU is not coded using an affine motion model, the integration of MPA requires additional information to be signaled. The MPA-specific information (MPA flag, motion plane index) is signaled right after the affine flag and starts with a first bin encoding the MPA flag. If this is true, a second bin denotes whether the front/back motion plane is used, and, if this is not the case, a third bin denotes whether the left/right or the top/bottom motion plane is used. Similar to the existing motion information (MI) in the H.266/VVC video coding standard, the described information is signaled using the context-based binary arithmetic coder (CABAC) [27] with dedicated context models. In case of bi-prediction, i.e., two predictions with different MI are averaged to form an overall

Motion Compensation

To reduce the computational complexity of the motion-plane-adaptive motion modeling, the motion model is executed on 4×4 subblocks similar to the realization of the newly introduced 4-parameter and 6-parameter affine motion models [28], [29] as shown in Fig. 7. Hence, the resulting pixel shift needs to be calculated only once per 4×4 subblock and all pixels within a subblock share the same pixel shift. This greatly speeds up the motion modeling procedure.

The precise pixel position within each subblock, at which the motion-plane-adaptive motion model is evaluated, can in principle be chosen arbitrarily. While intuitively, one would choose the center position of the subblock, our experiments showed that better results are obtained using one of the actual pixel positions surrounding the floating point center position. For further evaluations, we use the pixel position in the second row and second column of each subblock as shown in blue in Fig. 7.

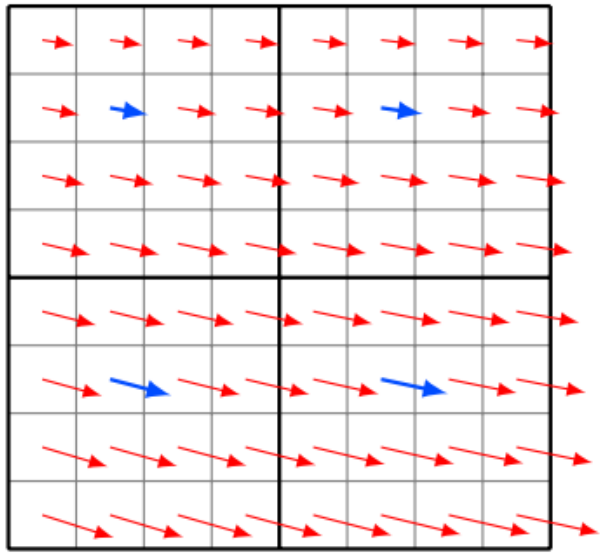
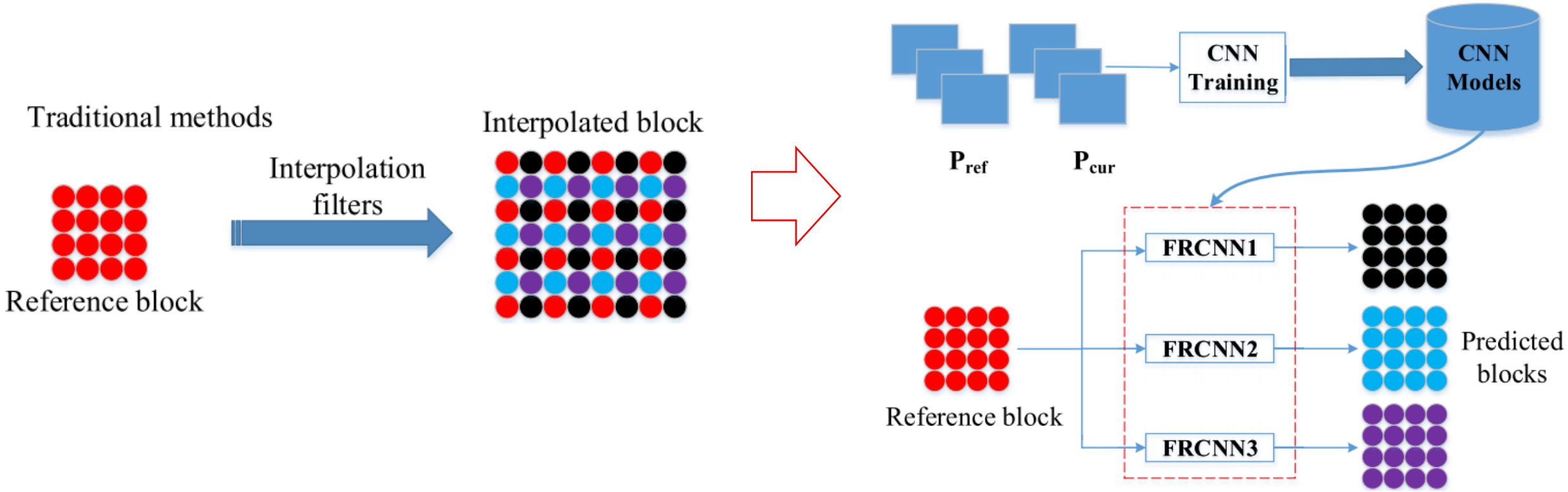


Fig. 7. Exemplary 4×4 subblock partitioning in MPA. Without subblock partitioning, the resulting pixel shift has to be calculated for each pixel individually (all arrows). With subblock partitioning, the pixel shift resulting from the motion-plane-adaptive motion model is only calculated for one pixel position within each 4×4 subblock (blue arrows).

Fractional enhancement

Convolutional Neural Network-Based Fractional-Pixel Motion Compensation

Ning Yan, Dong Liu, *Member, IEEE*, Houqiang Li, *Senior Member, IEEE*, Bin Li, *Member, IEEE*, Li Li, *Member, IEEE*, and Feng Wu, *Fellow, IEEE*



The traditional methods of fractional-pixel MC usually follow the approach of interpolation. They adopt different kinds of filters to interpolate fractional-pixel values from integer-pixel values in a reference picture. In earlier years, bilinear interpolation was usually adopted. Later on, more efficient filters have been studied, such as Wiener interpolation filter [4] and discrete cosine transform based interpolation filter (DCTIF) [5]. Such filters are often derived from the signal processing theory with assuming the signal to be interpolated is band-limited. Being computationally simple though, such filters may not deal with different kinds of content well enough as the content in natural videos is much more complex than ideal band-limited signal. Considering that different regions of natural videos have very different characteristics, content adaptive interpolation filter is also proposed in the literature [3], but has the drawback of transmitting the overhead of filter coefficients.

The interpolation methods work on the reference pictures with the expectation to generate imaginary fractional-pixel values. However, the goal of fractional-pixel MC is not to "enhance" the reference pictures with fractional-pixel values, but to "guess" the current to-be-coded picture from the reference pictures. In other words, we need to predict the pixel values of the current picture, rather than to predict the so-called fractional-pixel values of the reference picture, based on the integer-pixel values of a reference picture. It is a prediction between two pictures instead of (between the fractional and integer pixels) within one picture.

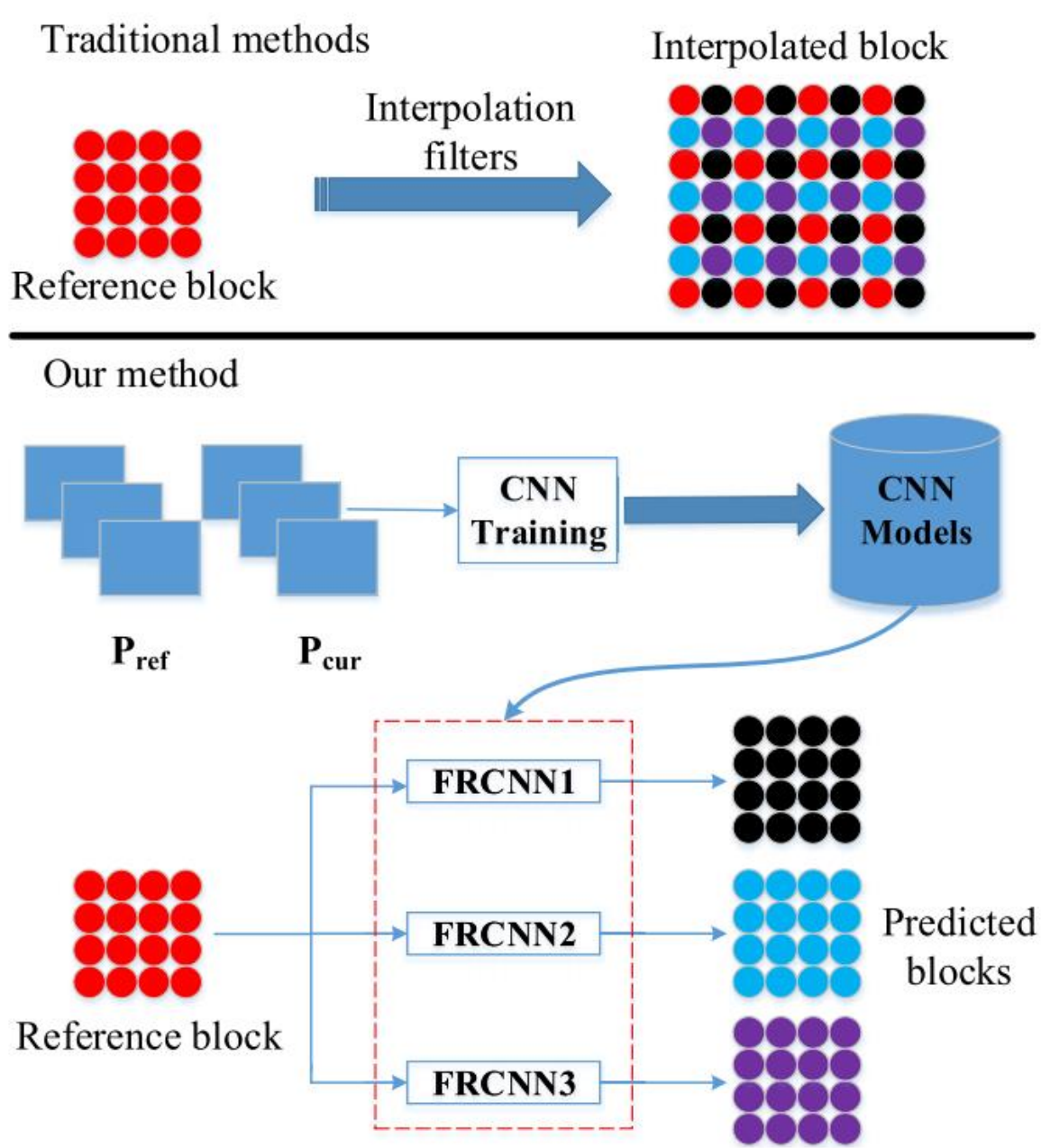


Fig. 2. Comparison between the traditional interpolation-based methods and the proposed CNN-based method for fractional-pixel reference generation.

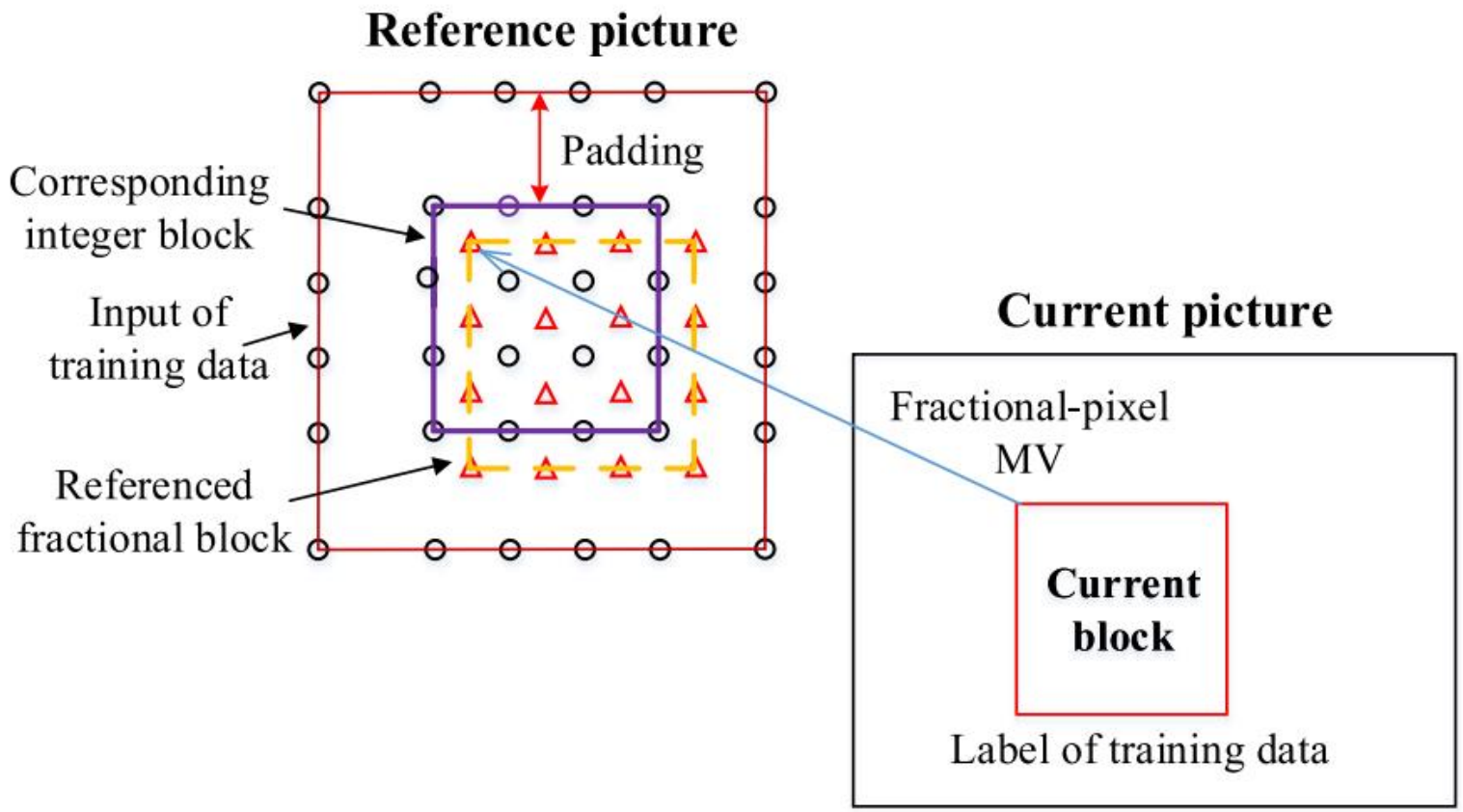
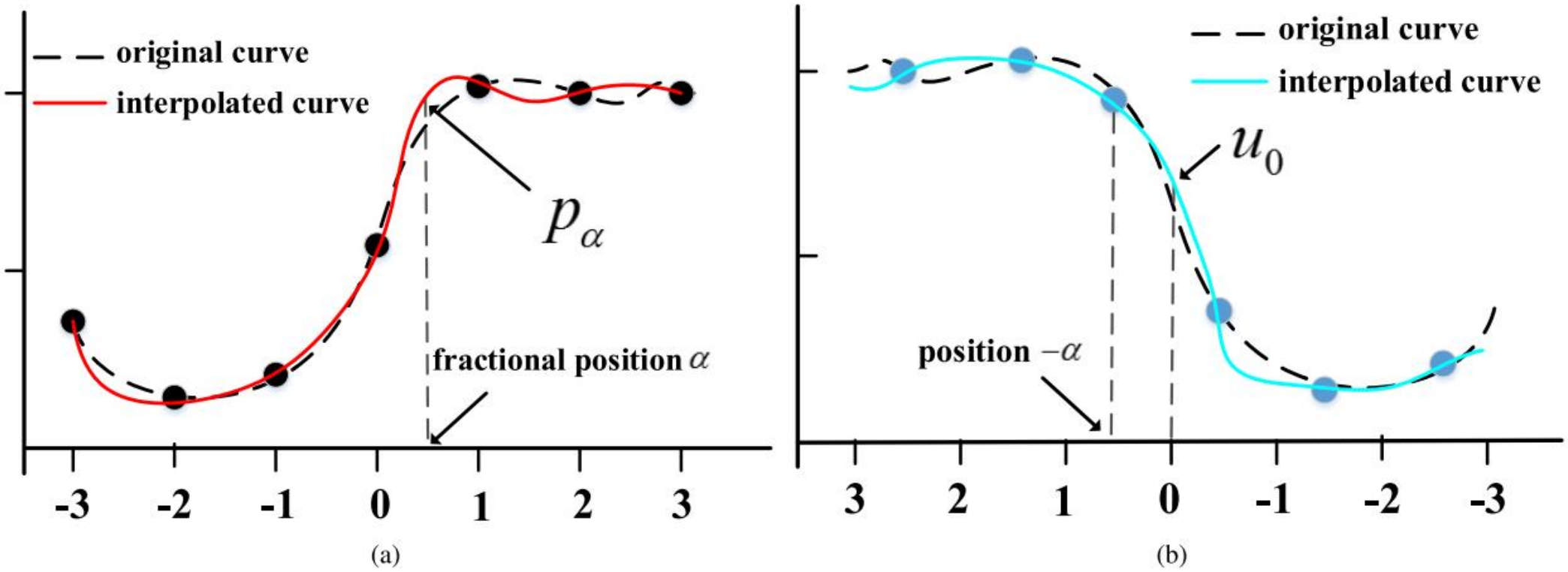


Fig. 4. Illustration of the process of generating training data for training uni-directional FRCNN.

Fractional enhancement

Invertibility-Driven Interpolation Filter for Video Coding

Ning Yan, Dong Liu, *Member, IEEE*, Houqiang Li, *Senior Member, IEEE*, Bin Li, *Member, IEEE*, Li Li, *Member, IEEE*, and Feng Wu, *Fellow, IEEE*



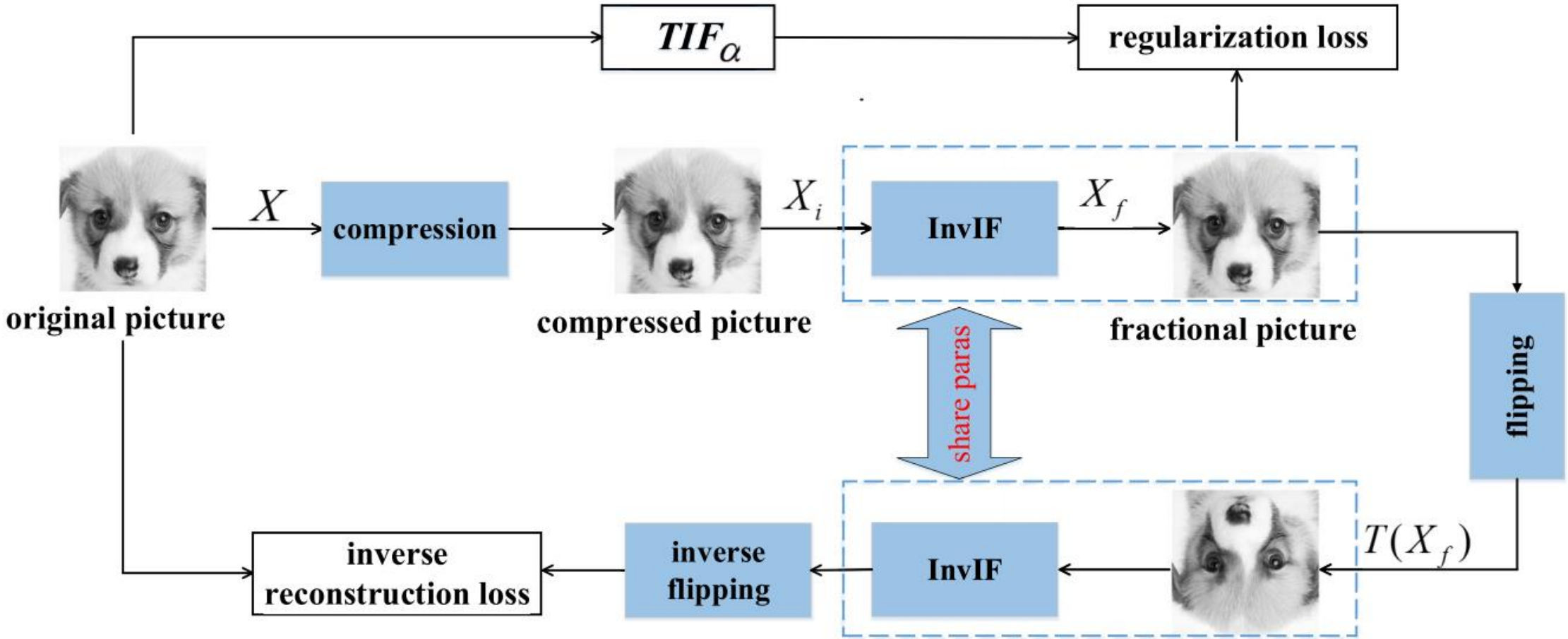
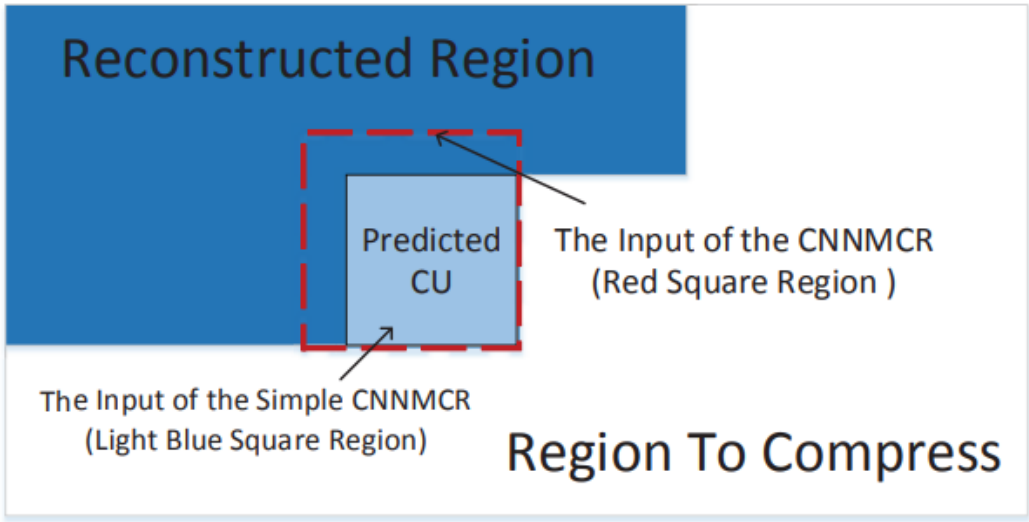


Fig. 3. The proposed end-to-end training scheme driven by invertibility.

NN-based enhancement

Deep Affine Motion Compensation Network for Inter Prediction in VVC

Dengchao Jin, Jianjun Lei, *Senior Member, IEEE,* Bo Peng, *Member, IEEE,* Wanqing Li, *Senior Member, IEEE,*
Nam Ling, *Fellow, IEEE* and Qingming Huang, *Fellow, IEEE*



Method 1: The Simple CNNMCR (CNN based Motion Compensation Refinement)

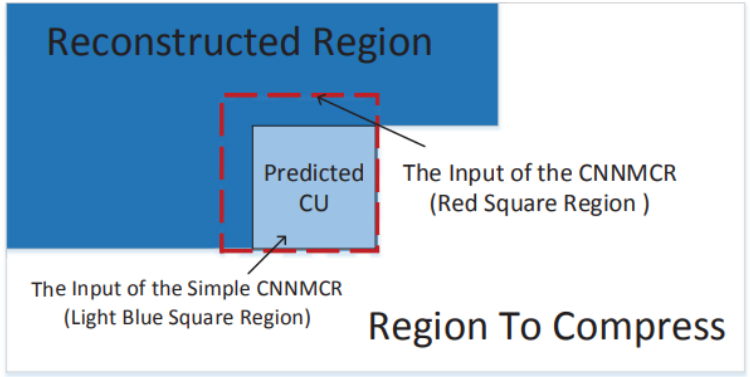
- Directly refine the inter-prediction without any other information

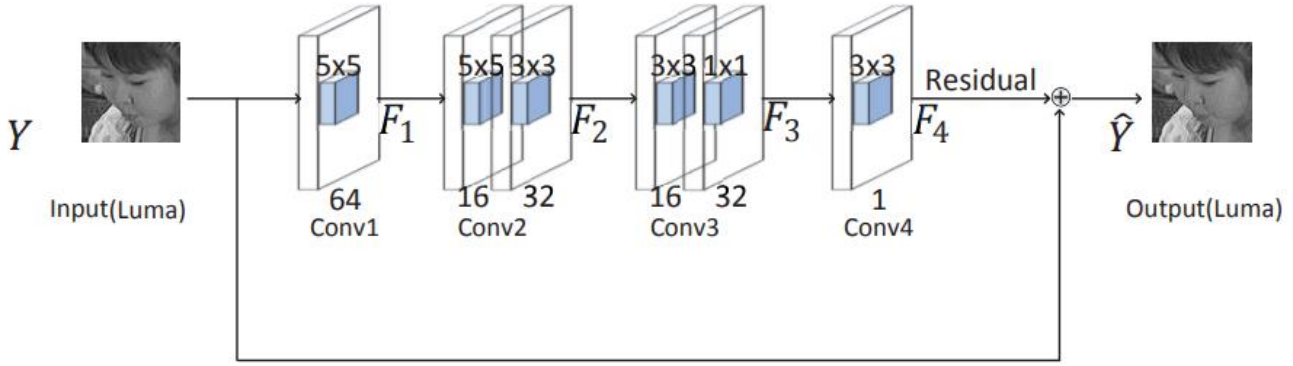
$$\min \mathcal{L} = \min_{\Theta} \sum_{i=1}^{N} (X_i - f(Y_i|\Theta))^2$$ Where *X* stands for the orig, *Y* stands for the reco

- To discover the capability of refining the motion compensation prediction of CNN

Method 2: The CNNMCR

- Further improve the prediction accuracy by exploiting *spatial* correlation
- The input of CNN expands from the current block to the red reg
  - Le the CNN to learn how to smooth the "seam", thus refines the prediction signal





Hight lights

- Description and performance comparison / analysis with related technologies

OBMC is to reduce the block artifact caused by block-level motion compensation. To that end, for the current block to be predicted, not only its own MV but also the MVs of its neighboring blocks (if available and not identical to the own MV) are all used to derive prediction signal for the current block. Then, multiple prediction blocks based on multiple MVs are combined to generate the final prediction signal. Although OBMC achieves noticeable compression gain, there are still several limitations. First, the combination of multiple prediction blocks is simple, usually achieved by fixed-weight averaging that cannot suit for different video content. Second, though the MVs of neighboring blocks are utilized, other information of neighboring blocks is ignored.

It can be observed that OBMC alone achieves nice coding gain than the HEVC anchor, leading to on average 3.2% BD-rate reduction. But the combination of OBMC and CNNMCR is much better than OBMC alone, gives out on average 5.2% BD-rate reduction than the HEVC anchor. Comparing the BD-rate reductions of CNNMCR alone (2.3%), OBMC alone (3.2%), and OBMC + CNNMCR (5.2%), it seems the coding gain of our proposed CNNMCR scheme and the OBMC technique can be accumulated largely. Therefore, although our CNNMCR and OBMC are both intended to refine motion compensation, their benefits are different in video coding.

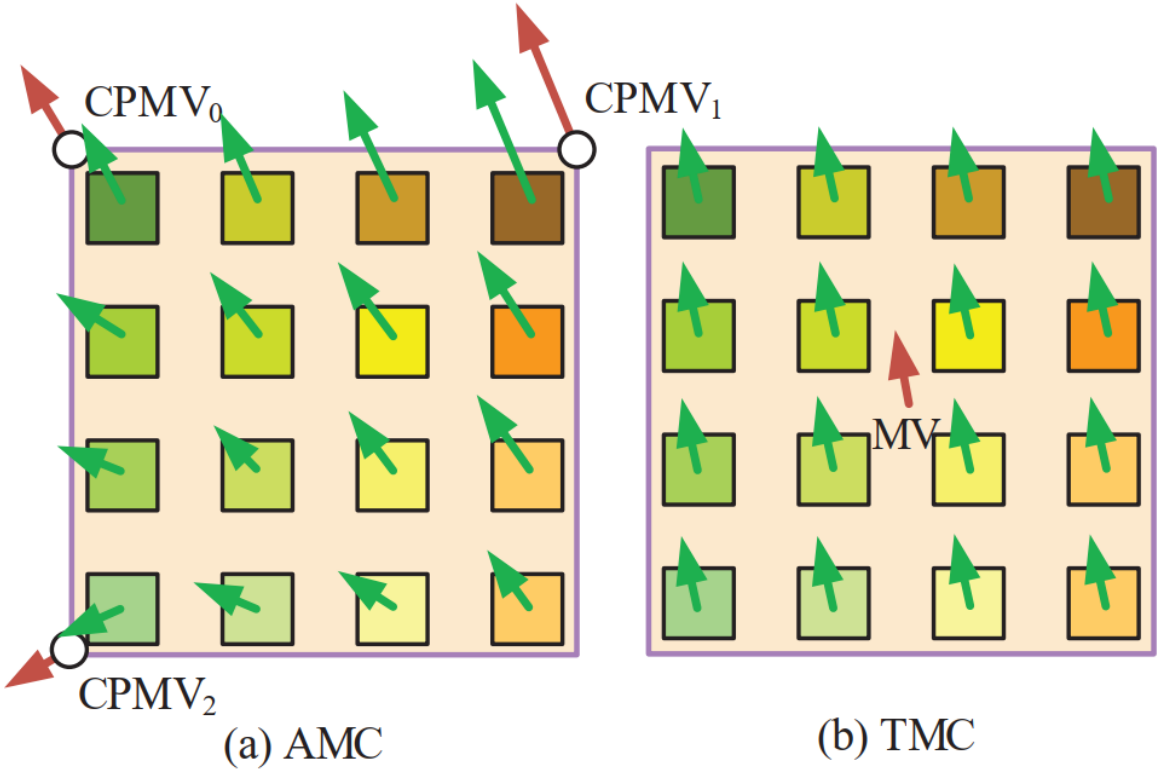
Performance

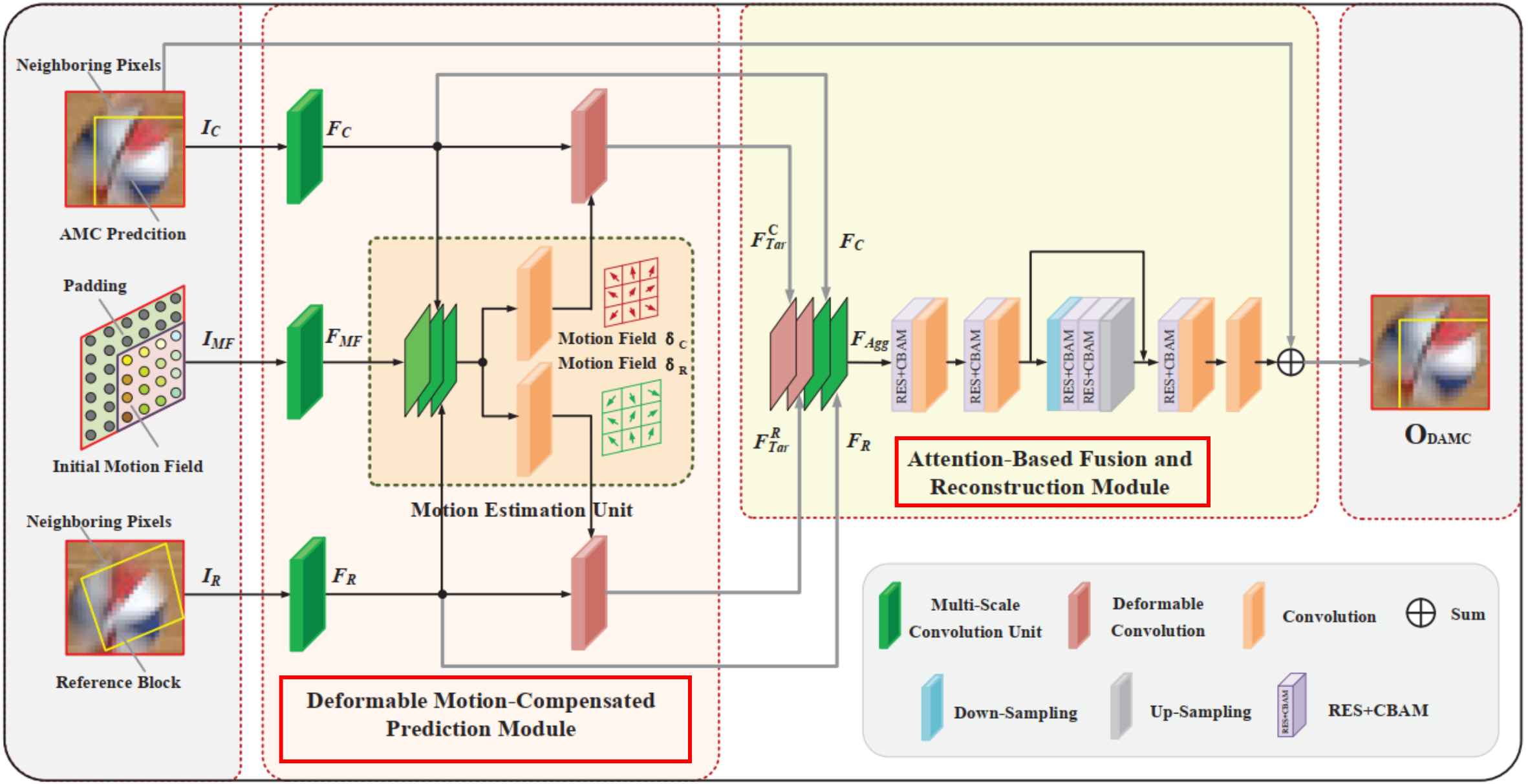
- HEVC, lowdelay-P
- 1.8% for The Simple CNNMCR
- 2.3% for The CNNMCR
- 5.2% for The CNNMCR together with OBMC

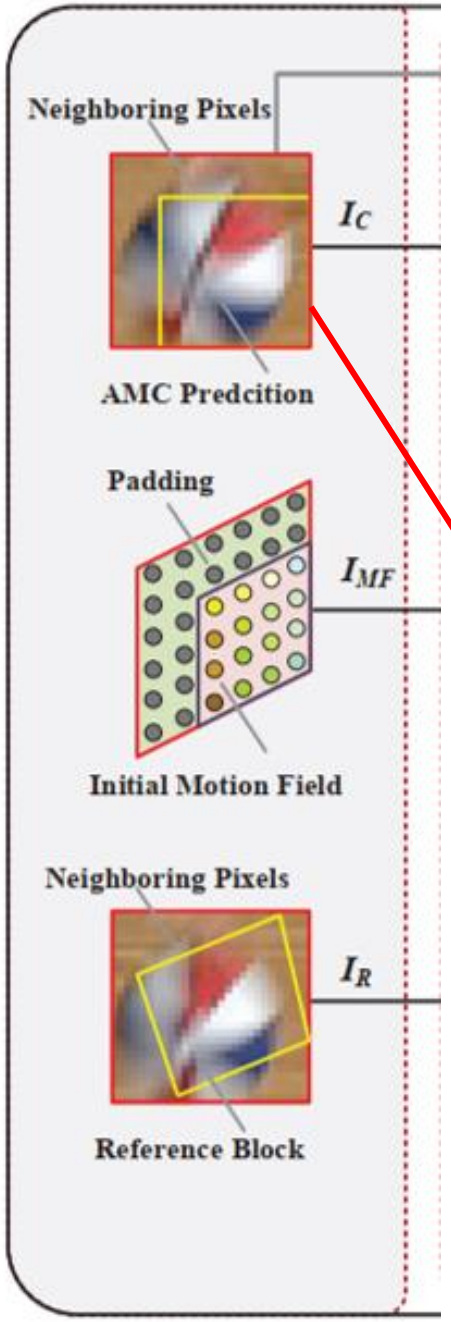
NN-based enhancement

Deep Affine Motion Compensation Network for Inter Prediction in VVC

Dengchao Jin, Jianjun Lei, *Senior Member, IEEE,* Bo Peng, *Member, IEEE,* Wanqing Li, *Senior Member, IEEE,* Nam Ling, *Fellow, IEEE* and Qingming Huang, *Fellow, IEEE*

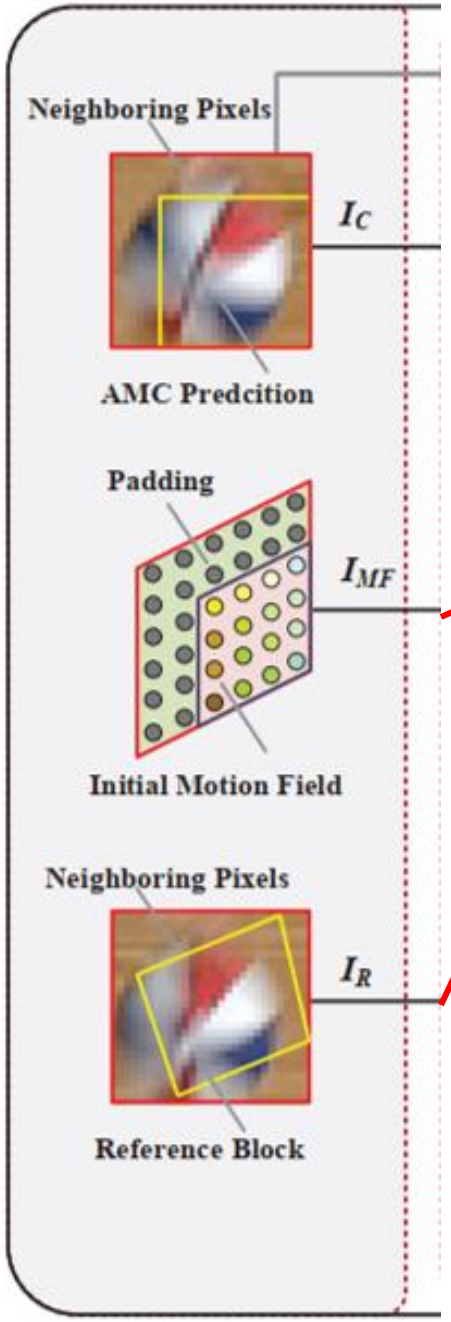



Neighboring Pixels
I_C
AMC Predcition
Padding
I_{MF}
Initial Motion Field
Neighboring Pixels
I_R
Reference Block
F_C
F_{MF}
F_R
Motion Field δ_C
Motion Field δ_R
Motion Estimation Unit
Deformable Motion-Compensated Prediction Module
F_{Tar}^C
F_C
F_{Agg}
F_{Tar}^R
F_R
RES+CBAM
RES+CBAM
RES+CBAM
RES+CBAM
RES+CBAM
Attention-Based Fusion and Reconstruction Module
O_{DAMC}
Multi-Scale Convolution Unit
Deformable Convolution
Convolution
Sum
Down-Sampling
Up-Sampling
RES+CBAM



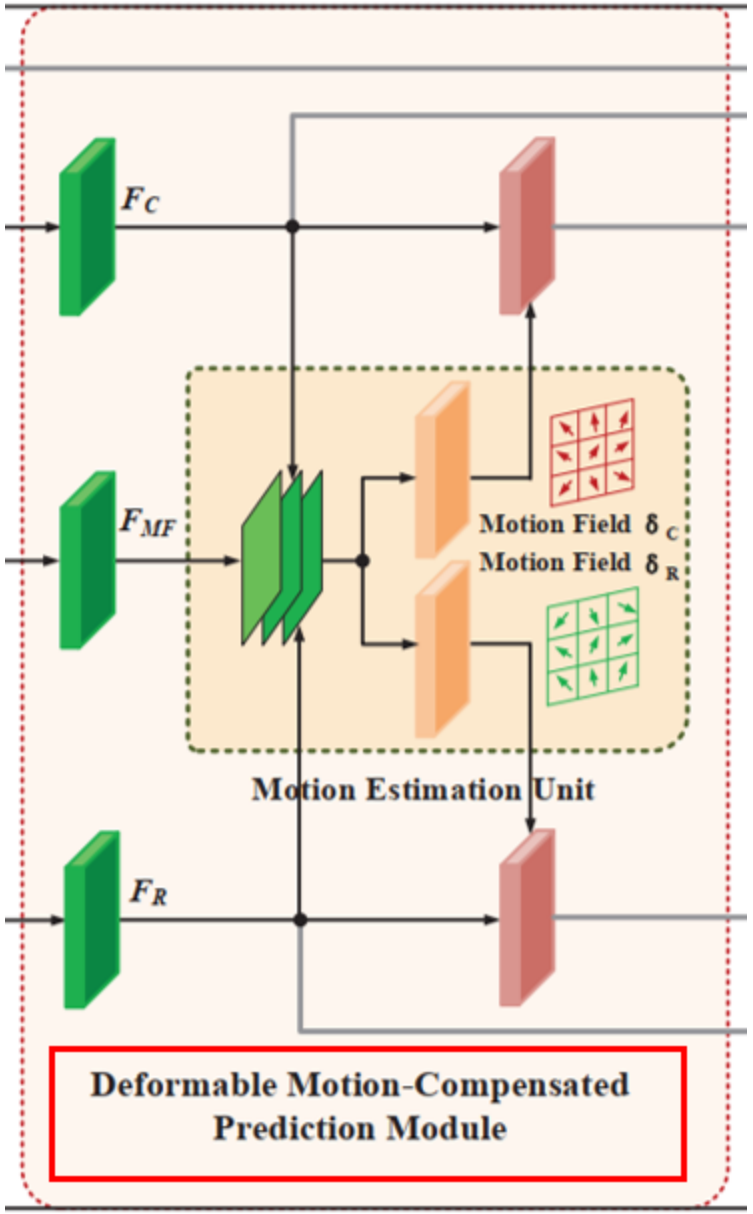
Traditional AMC compensates the current block merely by a parameterized affine model, which has limited capability to model complex motions. To solve this problem, a learning-based model, DAMC-Net, is designed to compensate the current block by explicitly estimating pixel-wise motion fields rather than implicitly deriving the subblock-wise motion field. Specifically, a spatial pixel-wise motion field is estimated to refine the prediction block for alleviating the spatial blocking artifacts, and a temporal pixel-wise motion field between frames is estimated to compensate the current block for alleviating temporal misalignment.

Fig. 2 shows the overall architecture of the proposed DAMC-Net. As shown in the figure, the multi-domain information is fully leveraged in the proposed DAMC-Net. In order to improve the prediction accuracy of AMC, the spatial neighboring pixels of the current block as well as its prediction of AMC are combined as the first input (I_C) to explore spatial correlations. Besides, to obtain as accurate as possible source



Besides, to obtain as accurate as possible source pixels in the temporal reference frame of the current block, the most similar block together with neighboring pixels in the reference frame is constructed based on CPMVs and used as the second input (I_R). More importantly, since the initial motion field (I_{MF}) constructed by CPMVs contains motion information, I_{MF} is utilized as the third input. Taking the I_C, I_{MF}, and I_R as the inputs, the proposed DAMC-Net is optimized with respect to jointly utilizing spatial neighboring and temporal correlative information to improve the prediction accuracy.

In the DMCP module, features F_C, F_{MF}, and F_R are first extracted from I_C, I_{MF}, and I_R respectively. Taking these features as inputs, the motion estimation unit (MEU) is designed to estimate motion fields. Based on estimated motion fields, deformable convolution is used to compensate F_C and F_R. Features of compensated output F^C_{Tar} and F^R_{Tar} are concatenated with F_C as well as F_R to construct the aggregated feature F_{Agg}. Finally, the output block, O_{DAMC}, is reconstructed from F_{Agg} by an AFR module.



Due to the limitation of deriving the subblock-wised motion field, the prediction of AMC suffers from misalignment at pixel-level. In order to improve the granularity of AMC, the DMCP module is designed to estimate pixel-wise motion fields for compensating the current block.

In DMCP, to extract deep features with abundant information, features F_C, F_{MF}, and F_R are extracted from I_C, I_{MF}, and I_R by multi-scale convolution unit [25] respectively. Then, the MEU is designed to estimate accurate motion fields. As shown in Fig. 2, F_C, F_{MF}, and F_R are first concatenated, then separate convolution operations are followed to generate offsets for each texture branch in MEU. It should be noticed that not only the texture information I_C and I_R are exploited in MEU, but also the initial motion field I_{MF} are jointly utilized to estimate accurate motion fields. Compared to a network which learns motion fields from scratch, the DMCP-Net estimates accurate motion fields with coarse input, which helps to reduce the network training difficulty and ensure the quality of learned motion fields.

Performance

TABLE I
BD-RATE RESULTS OF THE PROPOSED DAMC-NET FOR AFFINE INTER-MODE

Class	Sequence	Inter AMC			Inter DAMC-Net		
		Y	Cb	Cr	Y	Cb	Cr
Class B	MarketPlace	-5.27%	-3.77%	-7.24%	-5.36%	-3.76%	-6.74%
	RitualDance	-1.69%	-1.13%	-1.20%	-1.89%	-1.34%	-1.31%
	Cactus	-8.43%	-6.06%	-6.24%	-8.88%	-6.30%	-6.27%
	BasketballDrive	-2.45%	-2.11%	-2.18%	-2.68%	-2.36%	-2.01%
	BQTerrace	-0.77%	-0.24%	-0.64%	-1.74%	-0.72%	-0.96%
	Average Class B	-3.72%	-2.66%	-3.50 %	-4.11%	-2.90%	-3.46%
Class C	RaceHorses	-1.39%	-1.04%	-1.02%	-1.63%	-1.33%	-0.23%
	BQMall	-0.88%	-0.38%	0.17%	-1.20%	-0.48%	-0.03%
	PartyScene	-1.52%	-1.21%	-1.12%	-2.37%	-1.83%	-1.73%
	BasketballDrill	-1.04%	-0.20%	-0.46%	-1.16%	-0.23%	-0.07%
	Average Class C	-1.21%	-0.71%	-0.61 %	-1.59%	-0.97%	-0.52%
Class D	RaceHorses	-1.79%	-0.83%	-0.40%	-2.14%	-1.03%	-0.79%
	BQSquare	-3.43%	-1.21%	-2.66%	-6.13%	-2.95%	-4.27%
	BlowingBubbles	-1.88%	-0.68%	-1.66%	-2.62%	-1.18%	-1.62%
	BasketballPass	-1.74%	-1.77%	-0.32%	-2.81%	-2.47%	0.86%
	Average Class D	-2.21%	-1.12%	-1.10 %	-3.43%	-1.91%	-1.46%
Class E	FourPeople	-0.75%	-0.22%	-0.57%	-1.27%	-0.65%	-0.66%
	Johnny	-2.76%	-1.58%	-1.72%	-4.39%	-2.61%	-2.36%
	KristenAndSara	-3.18%	-2.27%	-2.50%	-3.41%	-2.31%	-2.99%
	Average Class E	-2.23%	-1.36%	-1.60 %	-3.02%	-1.86%	-2.00%
Overall		**-2.44%**	-1.54%	-1.82%	**-3.11%**	-1.97%	-1.95%

2) Encoding Configurations: DAMC-Net is integrated into VVC reference software VTM (version 6.2). Experiments are performed under the JVET common test conditions (CTC) [40]. Since single reference frame is utilized in the proposed DAMC-Net, LDP configuration and Classes B∼E are tested. In the experiments, the testing QPs are set as $\{22, 27, 32, 37\}$, and the widely employed BD-rate [41], [42] is used as the objective metric to evaluate the coding performance. A CPU + GPU cluster is used as the test environment, where VVC coding is tested in CPU and the DAMC-Net is running in GPU. The CPU is Inter(R) Core(TM) i9-9900K CPU @ 3.60GHz, and the GPU is NVIDIA GeForce GTX 1080Ti.

To optimize the proposed network, $\mathcal{L}_2$ loss is utilized as the loss function:

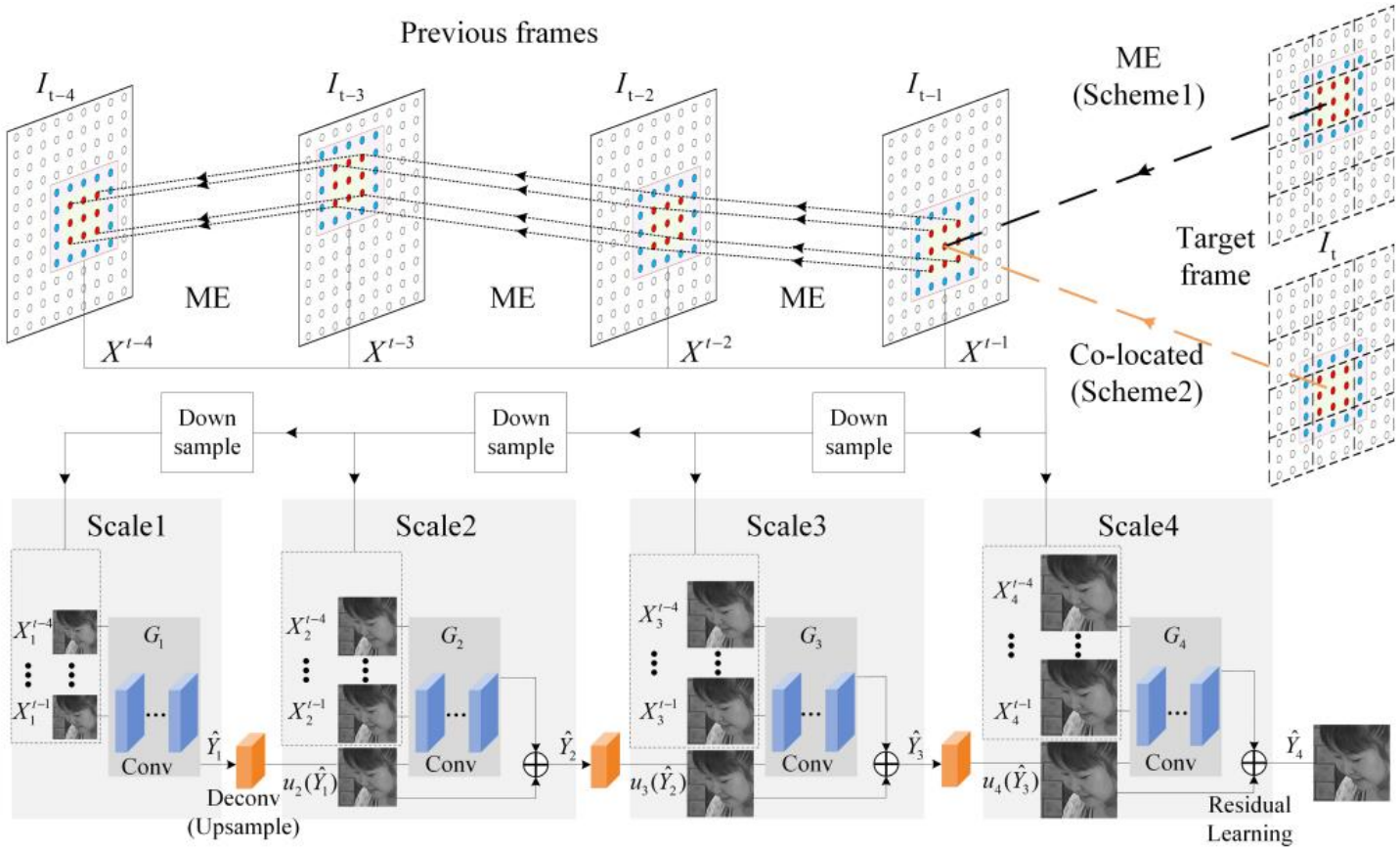
$$\mathcal{L} = || \, (O_{GT} - O_{DAMC}) \, ||_2^2 \tag{3}$$

where O_{GT} is the corresponding block in the raw videos and O_{DAMC} is the output of AFR.

Reference Generation

Deep Network-Based Frame Extrapolation With Reference Frame Alignment

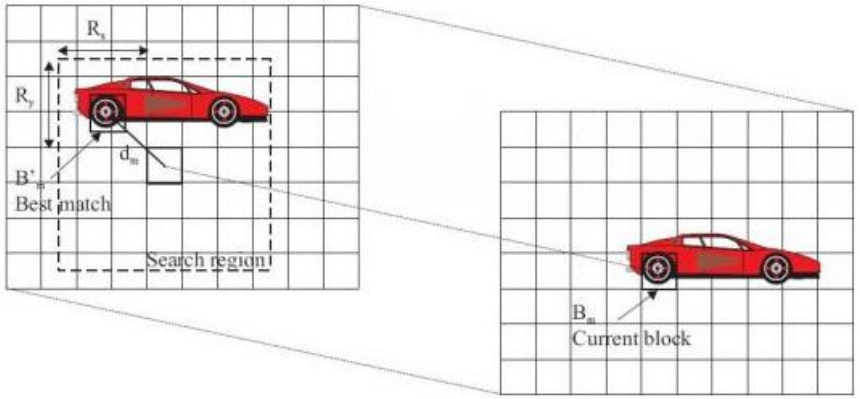
Shuai Huo, Dong Liu, *Senior Member, IEEE*, Bin Li, *Member, IEEE*, Siwei Ma, *Member, IEEE*, Feng Wu, *Fellow, IEEE*, and Wen Gao, *Fellow, IEEE*



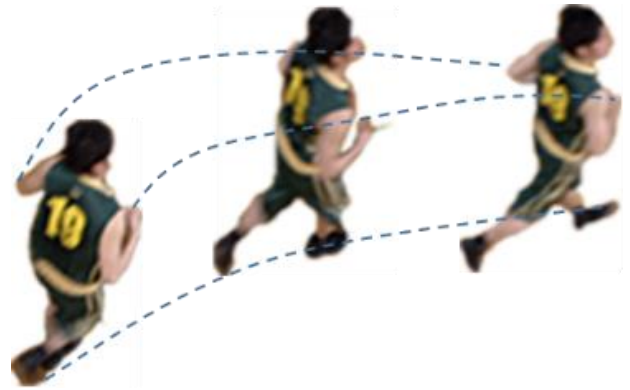
Frame extrapolation is to predict future frames from the past (reference) frames, which has been studied intensively in the computer vision research and has great potential in video coding. Recently, a number of studies have been devoted to the use of deep networks for frame extrapolation, which achieves certain success. However, due to the complex and diverse motion patterns in natural video, it is still difficult to extrapolate frames with high fidelity directly from reference frames. To address this problem, we introduce reference frame alignment as a key technique for deep network-based frame extrapolation. We propose to align the reference frames, e.g. using block-based motion estimation and motion compensation, and then to extrapolate from the aligned frames by a trained deep network. Since the alignment, a preprocessing step, effectively reduces the diversity of network input, we observe that the network is easier to train and the extrapolated frames are of higher quality.

线性模型

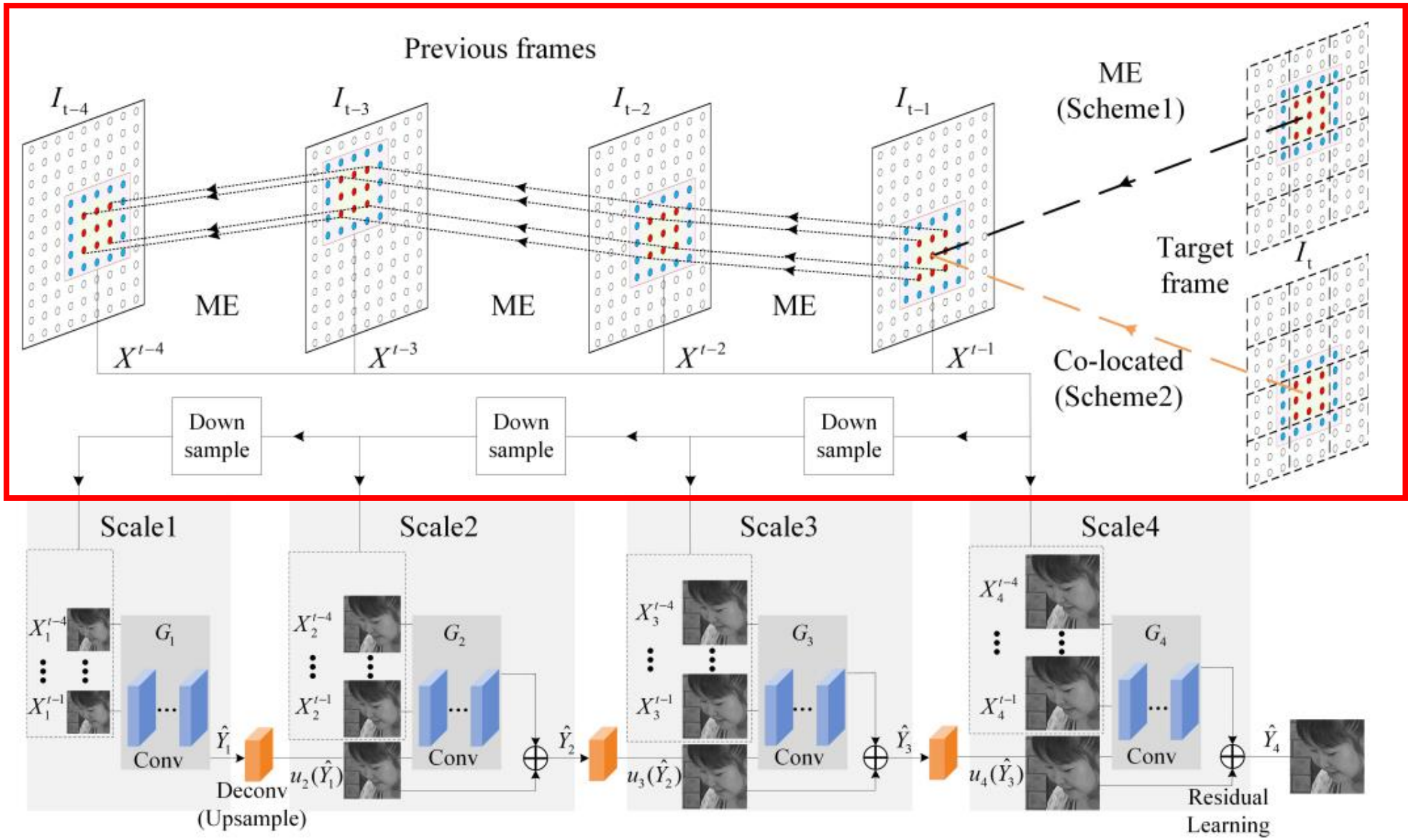
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$



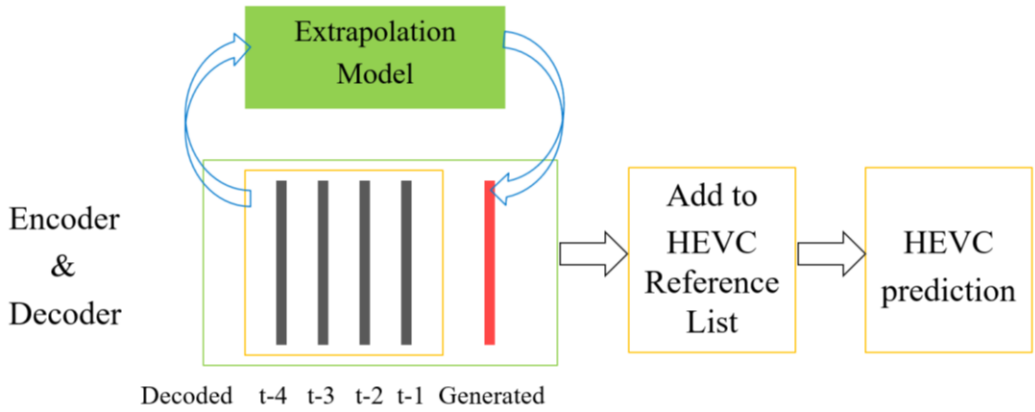
复杂形变



ME


Previous frames
I_{t-4}
I_{t-3}
I_{t-2}
I_{t-1}
ME
(Scheme1)
Target
frame
I_t
Co-located
(Scheme2)
ME
ME
ME
X^{t-4}
X^{t-3}
X^{t-2}
X^{t-1}
Down
sample
Down
sample
Down
sample
Scale1
Scale2
Scale3
Scale4
X_1^{t-4}
X_1^{t-1}
G_1
Conv
$\hat{Y}_1$
Deconv
(Upsample)
$u_2(\hat{Y}_1)$
X_2^{t-4}
X_2^{t-1}
G_2
Conv
$\hat{Y}_2$
$u_3(\hat{Y}_2)$
X_3^{t-4}
X_3^{t-1}
G_3
Conv
$\hat{Y}_3$
$u_4(\hat{Y}_3)$
X_4^{t-4}
X_4^{t-1}
G_4
Conv
$\hat{Y}_4$
Residual
Learning

In this paper, we propose a key idea to address the problem. Our idea is to utilize block-based ME/MC to align the reference frames and then to extrapolate from the already aligned frames by a trained deep network. By means of alignment, we effectively reduce the diversity of network input as the block-level translation between reference frames is effectively removed, but there is still subtle motion between these frames, which corresponds to high-order motion in the original frames. The diversity reduction will ease the network training as the network is expected to deal with the subtle motion only. As we observed, this design indeed improves the network performance. Actually, a lot of deep learning techniques have tried to reduce the diversity before network input or within network, for example many different normalization methods [6]–[9] and residual learning [10], [11]. To the best of our knowledge, we are the first to propose reference frame alignment as a diversity reduction technique for deep network-based frame extrapolation.



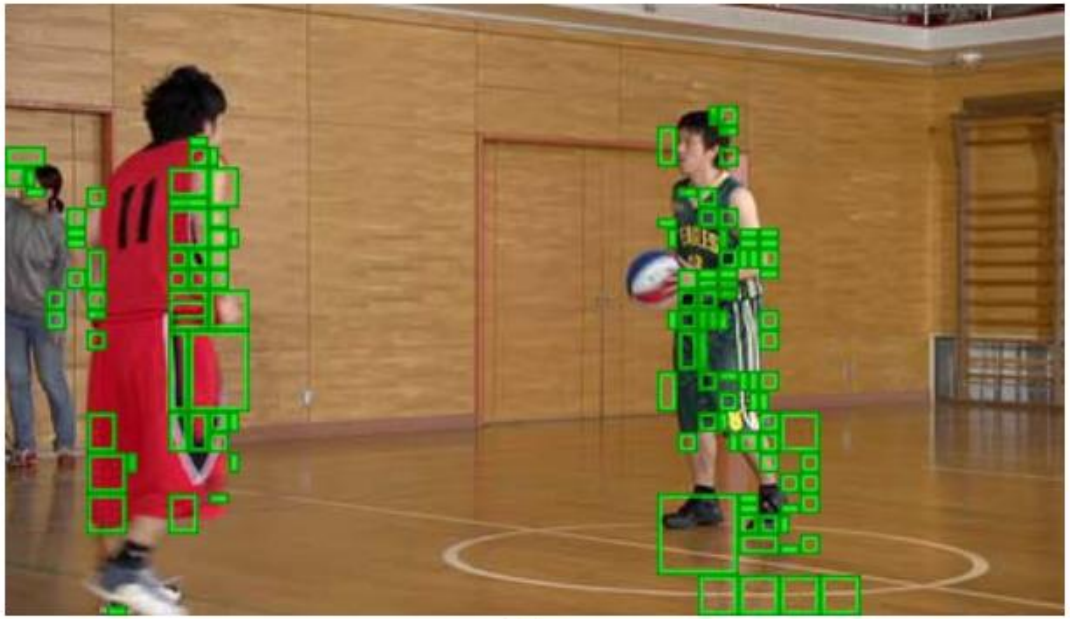
将外插帧作为新的参考帧，加入现有参考帧列表

参考帧机制允许在外插帧上自适应选择，选择复杂运动预测准的区域

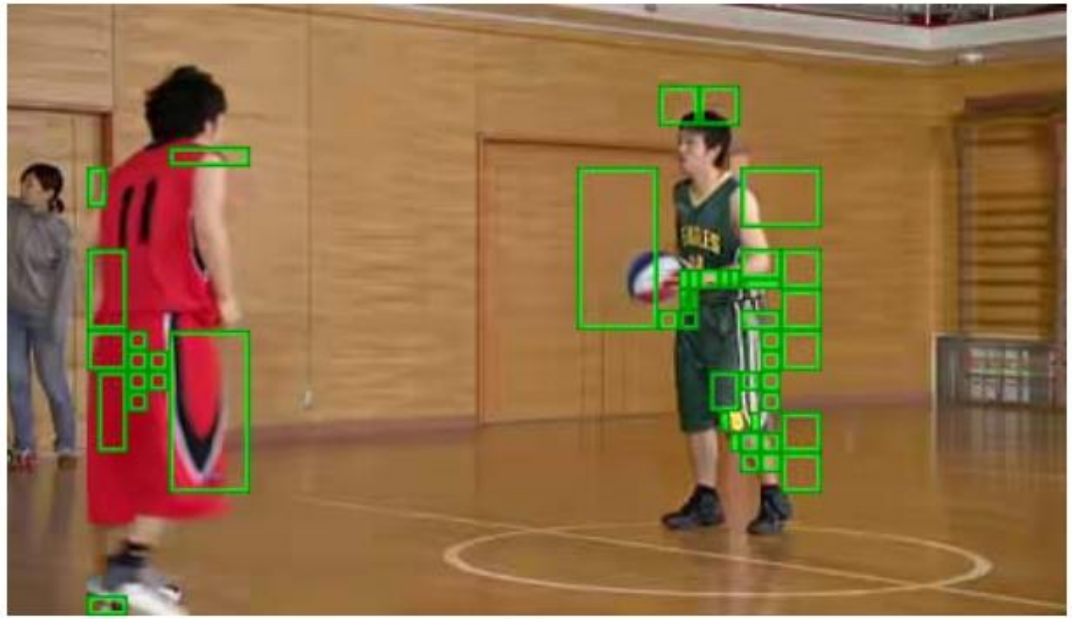
Performance

Class	Sequence	LDP			LDB		
		Y (%)	U (%)	V (%)	Y (%)	U (%)	V (%)
Class B	Kimono	−6.4	−5.0	−2.4	−2.6	−2.7	−1.4
	ParkScene	−3.7	−2.3	−1.4	−2.4	−1.5	−0.7
	Cactus	−7.6	−8.5	−5.4	−4.0	−5.5	−3.5
	BasketballDrive	−3.9	−2.3	−2.1	−1.3	−0.1	−0.3
	BQTerrace	−7.4	−3.1	−1.0	−2.0	0.6	1.2
Class C	BasketballDrill	−3.6	−2.3	−1.9	−1.5	−0.6	−0.4
	BQMall	−5.2	−3.9	−4.2	−3.5	−3.1	−3.4
	PartyScene	−3.0	−3.0	−2.3	−1.8	−1.9	−1.1
	RaceHorses	−1.9	−0.6	−0.9	−0.7	0.1	−0.0
Class D	BasketballPass	−4.3	−1.5	−1.8	−2.6	−0.7	−0.8
	BQSquare	−2.8	2.3	2.0	−1.5	3.5	3.0
	BlowingBubbles	−4.1	−4.2	−3.0	−3.1	−3.7	−1.9
	RaceHorses	−1.9	−0.9	−0.8	−1.3	−0.1	−0.1
Class E	FourPeople	−10.2	−9.3	−10.0	−6.7	−7.1	−8.0
	Johnny	−9.9	−10.1	−5.2	−4.4	−4.1	−1.9
	KristenAndSara	−8.6	−9.8	−7.2	−4.8	−6.0	−4.7
Class F	BasketballDrillText	−2.9	−1.8	−1.1	−1.1	−0.7	0.3
	ChinaSpeed	−0.7	−1.9	−1.6	−0.2	−0.7	0.1
	SlideEditing	0.1	0.2	0.3	−0.1	−0.1	−0.2
	SlideShow	−0.6	−1.2	−3.4	−0.2	0.1	0.9
Class Summary	Class B	−5.8	−4.2	−2.5	−2.5	−1.8	−0.9
	Class C	−3.4	−2.5	−2.3	−1.9	−1.4	−1.2
	Class D	−3.3	−1.1	−0.9	−2.1	−0.3	−0.0
	Class E	−9.6	−9.7	−7.5	−5.3	−5.8	−4.9
	Class F	−1.0	−1.2	−1.5	−0.4	−0.4	0.3
Overall	**Classes B–E**	**−5.3**	**−4.0**	**−3.0**	**−2.8**	**−2.1**	**−1.5**

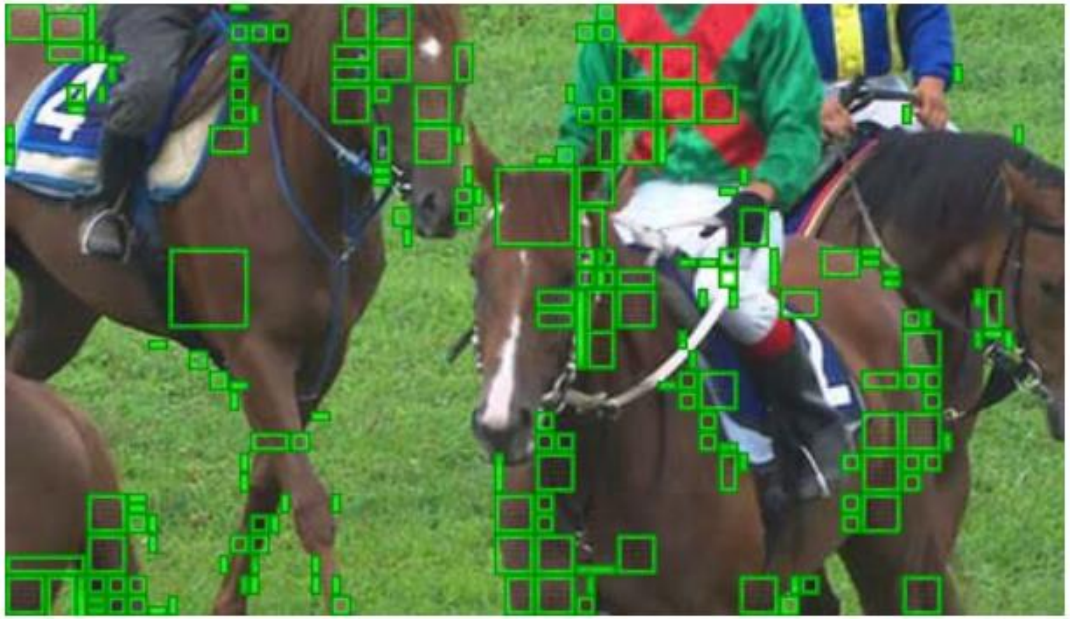
Selection



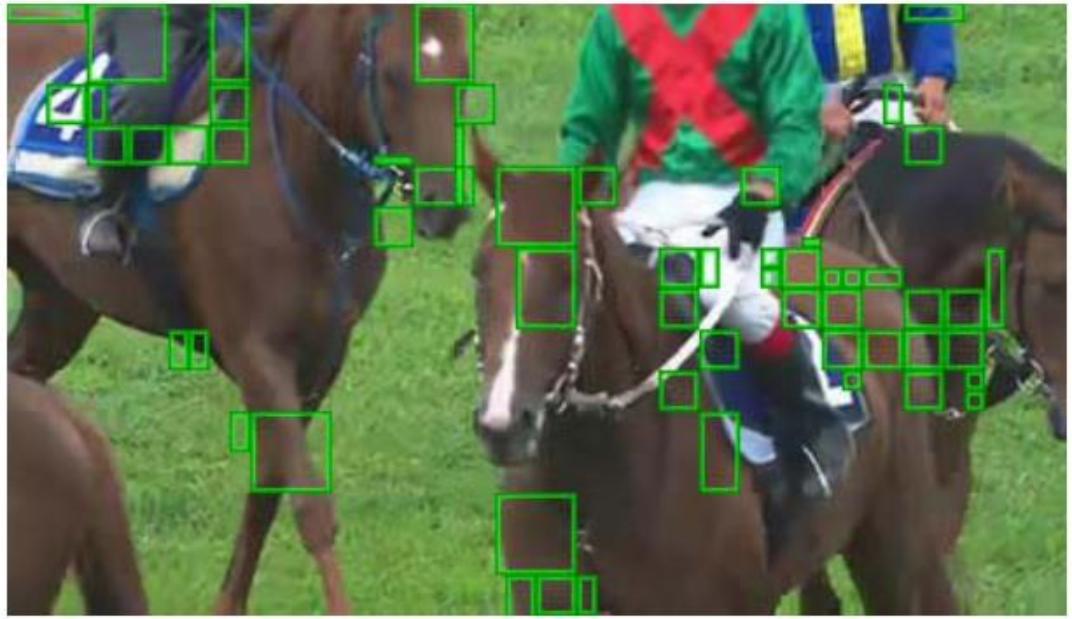
(a)



(b)



(c)

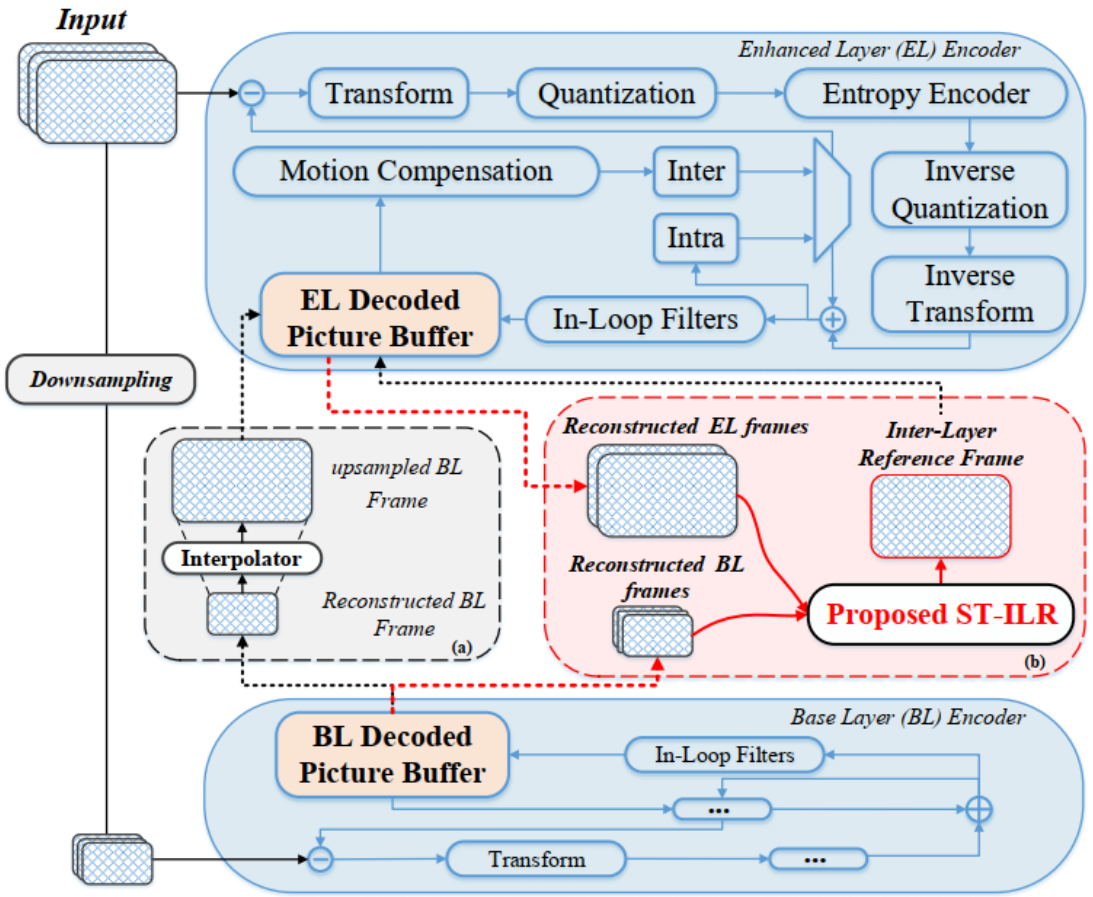


(d)

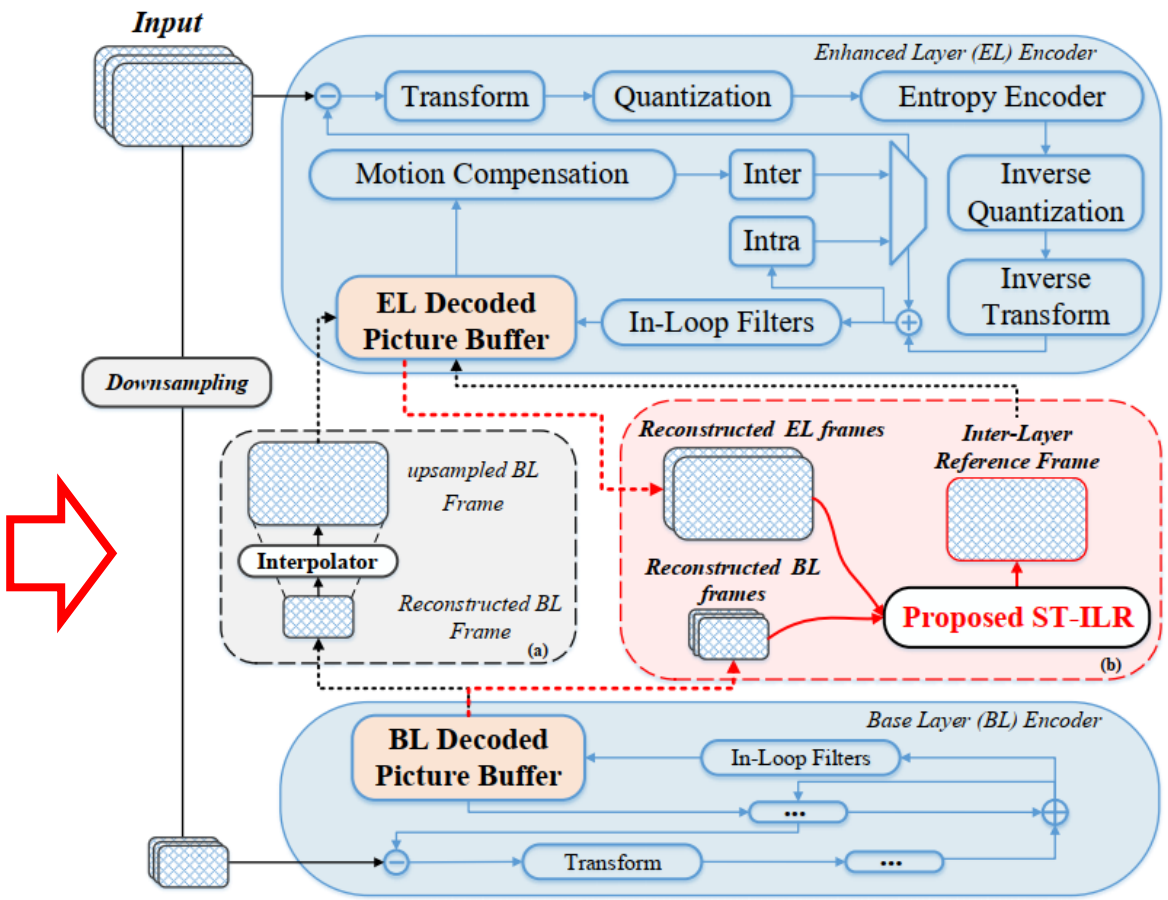
Reference Generation

Spatial-Temporal Inter-Layer Reference Frame Generation Network for Spatial SHVC

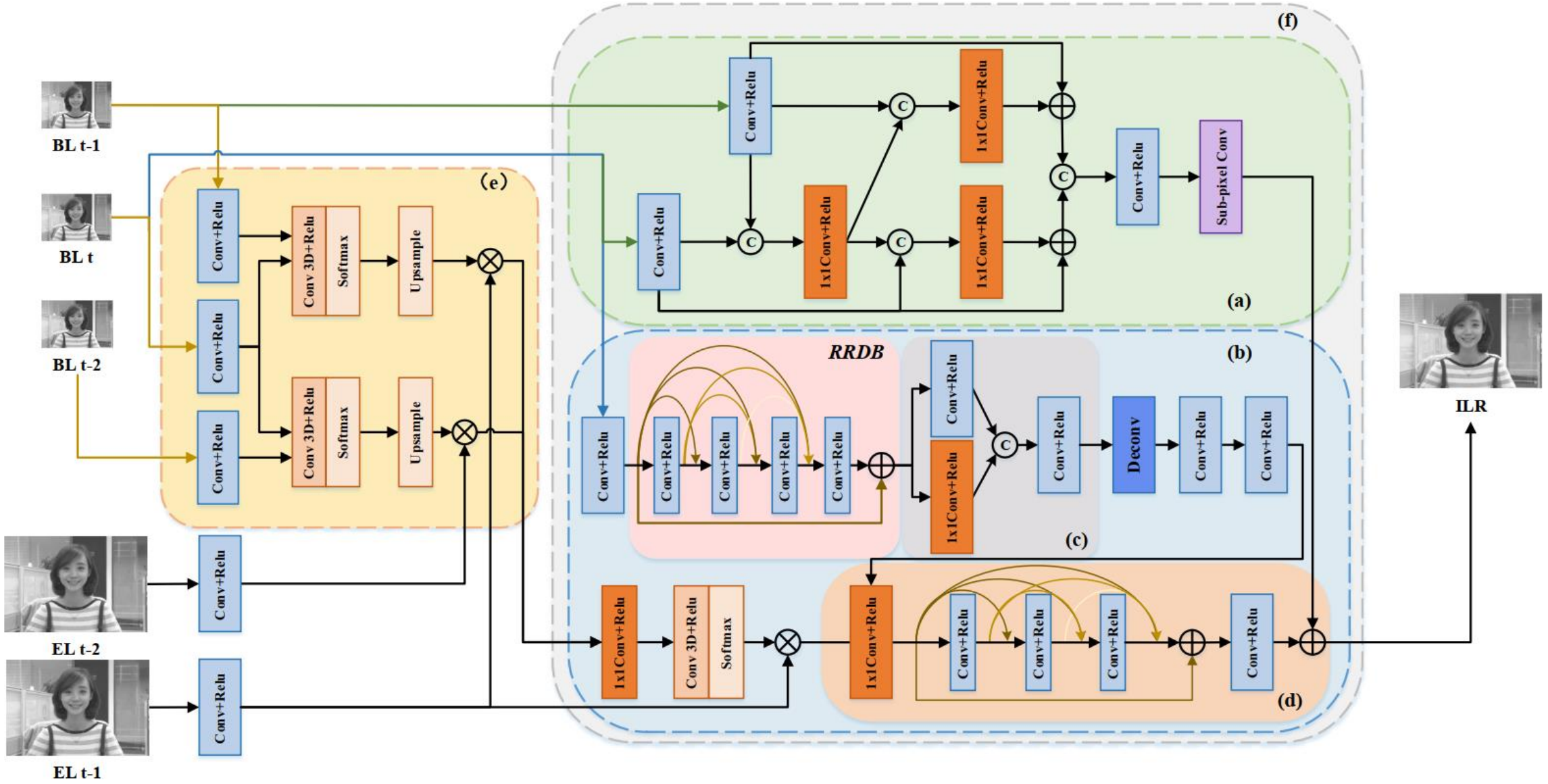
Shiwei Wang, Liquan Shen, Jingyue Liu



In spatial SHVC, the video is encoded into multiple bit-streams of different resolutions with the same quantization parameters. As shown in Fig. 1(a), the original video is downsampled to obtain low-resolution BL input, and the BL reconstructed images in DPB are later interpolated so that the resolution of the interpolated BL frames is same as the resolution of the EL frame with high resolution. Thus, in addition to the previously reconstructed frame, the upsampled BL can also be used as the inter-layer reference frame reference range for EL layer coding. By introducing interlayer reference, the efficiency of EL coding will be much higher than that without interlayer reference. After that, the SHVC encoder outputs corresponding two-layer code streams, which are transmitted according to the needs of the decoder. In the SHVC standard, this interpolation tool is defined as an 8-tap filter with fixed parameters, and each subpixel is estimated using 16 pixels in the neighborhood.



Framework


BL t-1
BL t
BL t-2
EL t-2
EL t-1
ILR
(a)
(b)
(c)
(d)
(e)
(f)
RRDB
Conv+Relu
1x1Conv+Relu
Conv 3D+Relu
Softmax
Upsample
Sub-pixel Conv
Deconv
C

Performance

TABLE I
OVERALL ΔBITRATE AND ΔPSNR OF THE PROPOSED ST-ILR COMPARED WITH SPATIAL SHVC UNDER RA CONFIGURATION

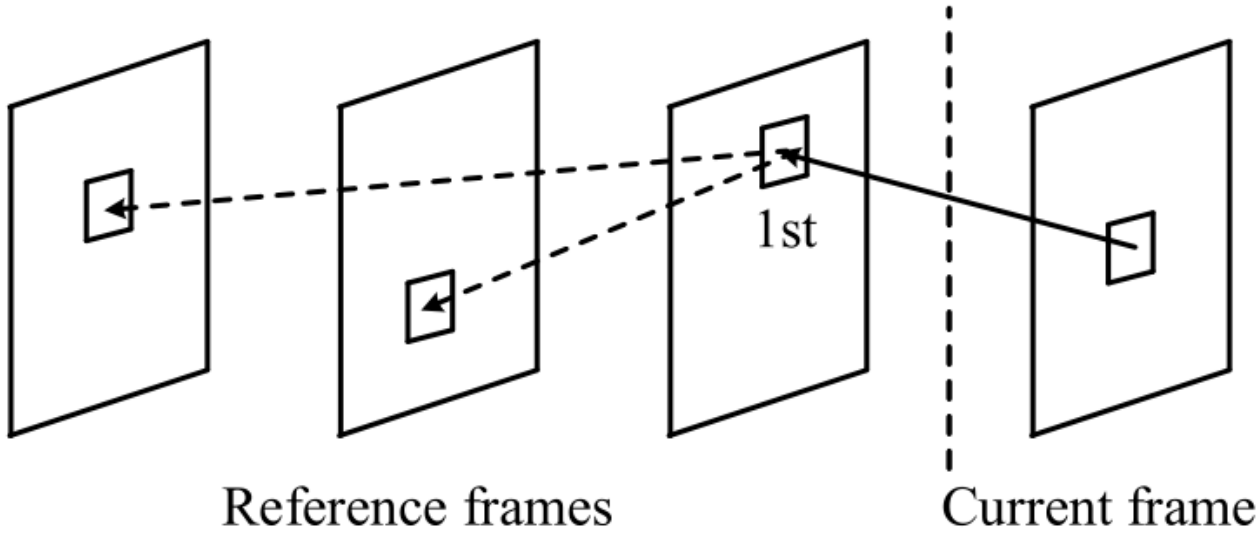
Classes	Sequences	ΔT_{enc}	ΔBitrate(%) / ΔPSNR (dB)				BD-BR(%)		BD-PSNR(dB)	
			BL QP = 22 BL QP = 22	BL QP = 26 BL QP = 26	BL QP = 30 BL QP = 30	BL QP = 34 BL QP = 34	BL + EL	Only EL	BL + EL	Only EL
A	Traffic	2.38	-9.008/-0.009	-13.650/-0.004	-15.441/0.013	-15.307/0.070	-10.401	-14.270	0.314	0.371
	PeopleOnStreet	1.82	-27.513/-0.116	-35.171/-0.176	-39.360/-0.181	-39.891/-0.183	-23.060	-33.513	1.138	1.583
B	BasketballDrive	1.95	-1.762/-0.008	-5.303/-0.0170	-6.787/-0.013	-6.984/-0.011	-3.578	-5.079	0.100	0.100
	BQTerrace	1.81	-1.962/0.006	-2.679/0.013	-2.812/0.023	-3.176/0.036	-2.778	-4.168	0.042	0.045
	Cactus	1.94	-4.058/0.012	-6.630/0.028	-7.245/0.056	-8.100/0.081	-5.424	-8.852	0.141	0.160
	Kimono	1.89	-1.928/-0.008	-4.300/0.0003	-4.353/0.005	-3.109/0.017	-2.487	-3.962	0.080	0.108
	ParkScene	2.07	-1.938/-0.008	-3.694/-0.011	-4.015/-0.001	-4.014/0.018	-2.566	-3.503	0.079	0.096
Overall	**All**	**1.98**	**-6.881/-0.019**	**-10.204/-0.024**	**-11.430/-0.014**	**-11.512/0.004**	**-7.185**	**-10.478**	**0.270**	**0.352**

Multi-hypothesis Prediction

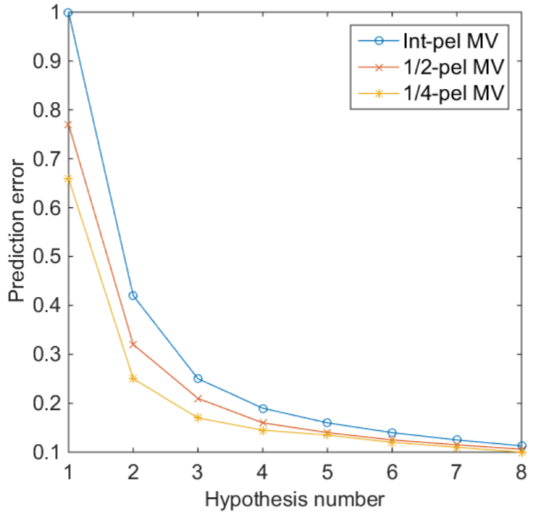
MULTI-HYPOTHESIS PREDICTION BASED ON IMPLICIT MOTION VECTOR DERIVATION FOR VIDEO CODING

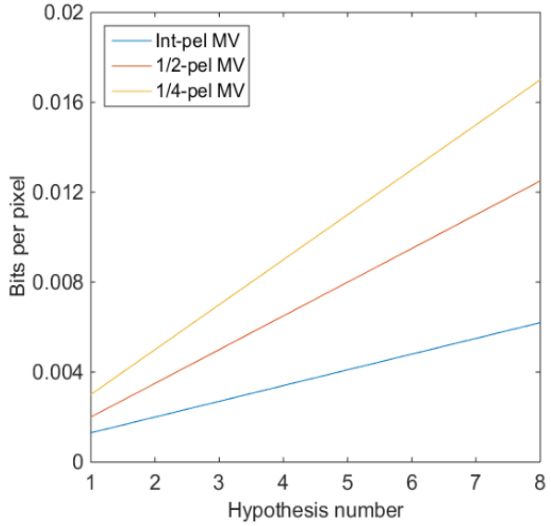
Zhao Wang[*], *Shiqi Wang*[+], *Xinfeng Zhang*[#], *Shanshe Wang*[*], *Siwei Ma*[*]

*Institute of Digital Media, Peking University, Beijing 100871, China
+Department of Computer Science, City University of Hong Kong, Hong Kong, China
#University of Southern California, Los Angeles, California, USA
{zhaowang, sswang, swma}@pku.edu.cn; {sqwang1986,zhangxinf07}@gmail.com



➢ More hypothesis (reference) leads to better prediction accuracy, along with more header bits





➢ Propose an implicit motion vector derivation scheme

- Use the first hypothesis (reference) to search other hypothesis
- More accurate prediction without additional motion info

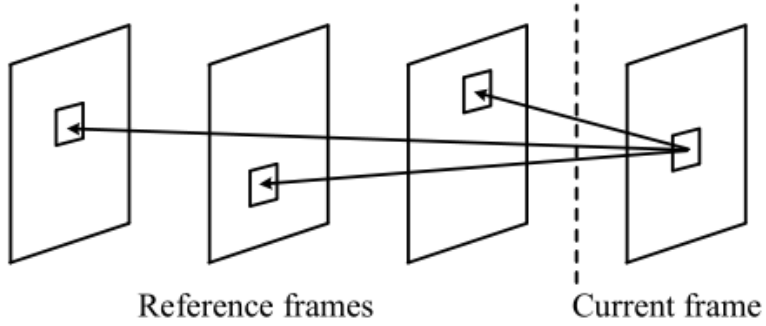


Fig. 4. Illustration of the traditional MHP scheme.

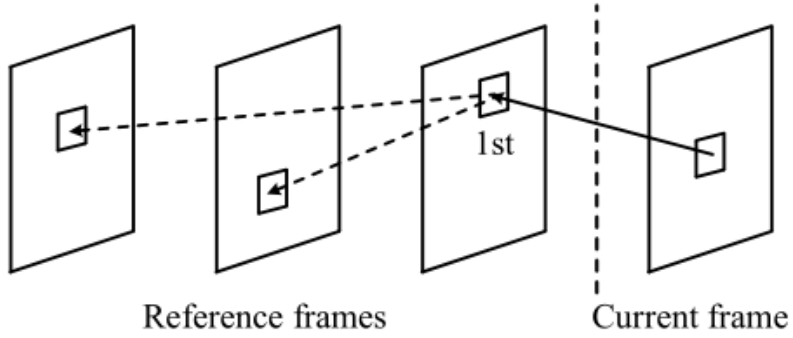


Fig. 5. Illustration of the proposed MHP scheme.

➢ Performance

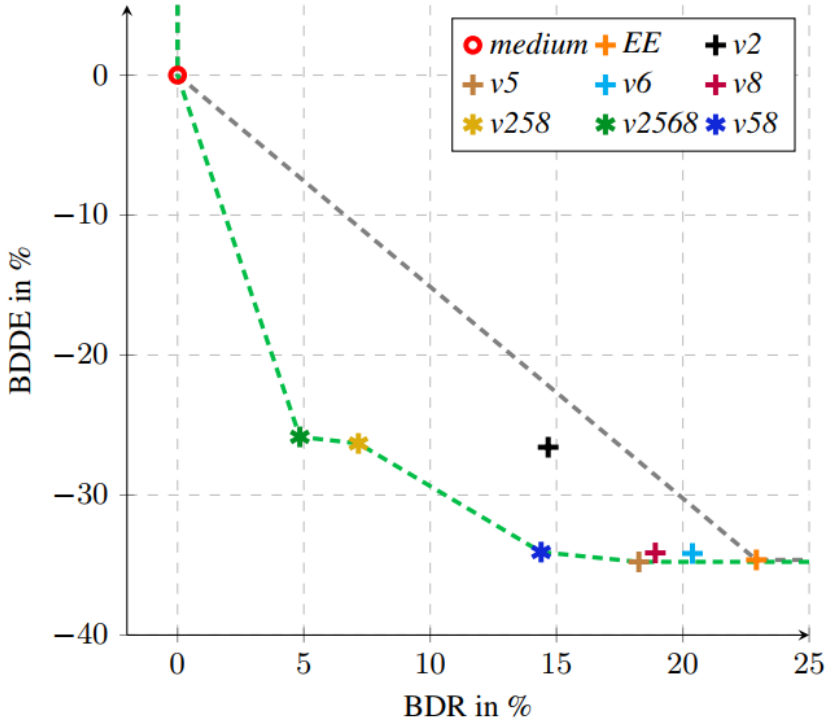
- HEVC, Random Access
- 0.24% for 1 additional hypothesis
- 1.14% for 4 additional hypothesis

Energy-Aware Quality Optimization （Green Coding）

OPTIMIZED DECODING-ENERGY-AWARE ENCODING IN PRACTICAL VVC IMPLEMENTATIONS

Matthias Kränzler⋆ *Adam Wieckowski*† *Geetha Ramasubbu*⋆ *Benjamin Bross*†
André Kaup⋆ *Detlev Marpe* † *Christian Herglotz*⋆

⋆Multimedia Communications and Signal Processing,
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany
†Video Communication and Applications Department,
Fraunhofer Heinrich-Hertz Institute (HHI), Berlin, Germany



The optimization of the energy demand is crucial for modern video codecs. Previous studies show that the energy demand of VVC decoders can be improved by more than 50% if specific coding tools are disabled in the encoder. However, those approaches increase the bit rate by over 20% if the concept is applied to practical encoder implementations such as VVenC. Therefore, in this work, we investigate VVenC and study possibilities to reduce the additional bit rate, while still achieving low-energy decoding at reasonable encoding times. We show that encoding using our proposed coding tool profiles, the decoding energy efficiency is improved by over 25% with a bit rate increase of less than 5% with respect to standard encoding. Furthermore, we propose a second coding tool profile targeting maximum energy savings, which achieves 34% of energy savings at bitrate increases below 15%.

	Intra			
1	Cross-component linear model (CCLM)	✓	✓	✓
2	Intra sub-partition (ISP)	✓	✓	✓
3	Matrix-based intra-picture prediction (MIP)	✓	✗	✓
4	Multiple reference line (MRL)	✓	✓	✓
	Inter			
5	Affine motion (AFFINE)	-	✓	✓
6	Adaptive MV resolution (AMVR)	-	✓	✓
7	Biprediction with CU-level weights (BCW)	-	✓	✓
8	Bidirectional optical flow (BDOF)	-	-	✓
9	Combined inter-/intra-picture prediction (CIIP)	-	✓	✓
10	Decoder-side MV refinement (DMVR)	-	-	✓
11	Geometric partitioning mode (GPM)	-	✓	✓
12	Merge with MVD (MMVD)	-	✓	✓
13	Prediction refinement with optical flow (PROF)	-	✓	✓
14	Subblock-based temporal MVP (SBTMVP)	-	✓	✓
15	Symmetric MVD (SMVD)	-	-	✓
	Transformation and Quantization			
16	Dependent quantization (DQ)	✓	✓	✓
17	Joint coding of chroma residual (JCCR)	✓	✓	✓
18	Low-frequency non-separable transform (LFNST)	✓	✗	✓
19	Multiple transform selection (MTS)	✓	✓	✓
20	Subblock transform (SBT)	-	✓	✓
	In-Loop Filter			
21	Adaptive loop filter (ALF)	✓	✓	✓
22	Cross-component adaptive loop filter (CCALF)	✓	✓	✓
23	Deblocking filter (DBF)	✓	✓	✓
24	Luma mapping with chroma scaling (LMCS)	✓	✓	✓
25	Sample adaptive offset (SAO)	✓	✓	✓
	Others			
26	Block-level differential PCM (BDPCM)	✗	✗	✗
27	Intra-picture block copy (IBC)	✗	✗	✗
28	Chroma separate tree (CST)	✓	✓	✓

In the following, we will describe the basic concept to achieve increased energy efficiency with a design space exploration (DSE) [29]. In [29], the encoder's complexity of the HEVC mode decision process is optimized by parallelization and skip decisions. In contrast to [29], we will focus on coding tools and the energy demand reduction of the decoder. Based on the findings in [5], we disable and enable coding tools of VVC in the encoder and thereby, reduce the decoding energy demand.

We define the design space by the tradeoff between energy and compression efficiency. To achieve a higher energy efficiency for VVC, we introduce the coding tool profile $\boldsymbol{u}$, which is defined by

$$\boldsymbol{u} = \begin{pmatrix} u(1) \\ \vdots \\ u(\eta) \\ \vdots \\ u(N) \end{pmatrix}, \tag{1}$$

where $u(\eta)$ indicates the usage of a coding tool, η corresponds to the index of a specific coding tool, and N to the number of coding tools from Table I. Each entry represents a binary value $u(\eta) \in \{0, 1\}$ indicating whether the tool is disabled or enabled. For the initialization of $\boldsymbol{u}$, we consider 28 coding tools according to Table I. For each coding tool η that is marked with (✓), $u(\eta)$ is 1. Otherwise, for a tools that is marked with (✗), $u(\eta)$ is 0, and the remaining tools marked with (–) are not considered.

To evaluate the influence of a changed coding tool profile on energy and compression efficiency, we use the Bjøntegaard-Delta (BD) metric. With a BD-metric, we compare the efficiency of an arbitrary video codec to another codec. Each BD metric that we use in this paper is based on the Bjøntegaard-Delta bit rate (BDR-PSNR) [30], which describes the bit rate savings in % for the same objective quality measured in PSNR. To evaluate the energy efficiency of a decoder, we substitute the bit rate by the decoder's energy demand. We call the resulting BD metric Bjøntegaard-Delta decoding energy (BDDE-PSNR), which describes the energy savings in % for the same PSNR. The measurement of the decoding energy and of PSNR will be explained in detail in Section IV.

Question