

AHG11: Deep omnidirectional video compression in YUV domain

Qipu Qin¹, Cheolkon Jung¹, Dan Zou², and Ming Li²

¹Xidian University, China

²OPPO, China

Introduction

2

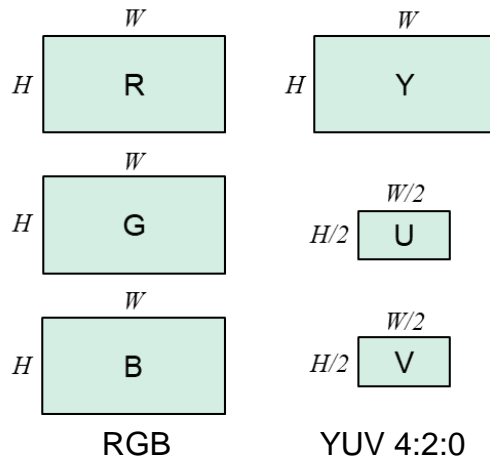


■ Background

- Deep learning-based video compression (DLVC) can be summarized into the following two aspects:
 - (1) Combine deep learning with traditional hybrid video compression (Module-level);
 - (2) Establish a novel deep video compression framework (Codec-level).
- In the previous contribution JVET-X0043, deep omnidirectional video compression in RGB domain (DOVC-RGB) has been proposed.

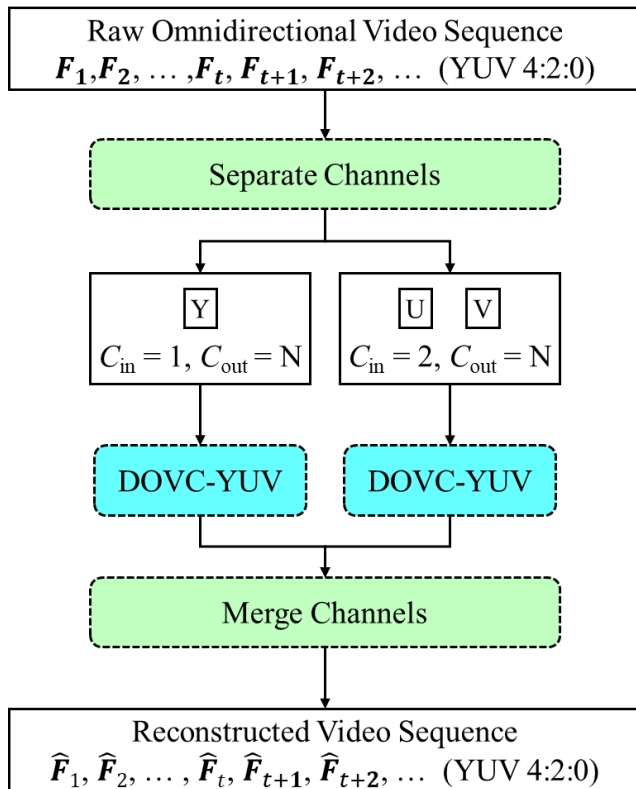
■ Motivation Encoding time of VTM (8K): 3~4 days/seq

- Omnidirectional videos with high resolution (6K and 8K)
- RGB domain (4:4:4 format) \Rightarrow YUV domain (4:2:0 format)

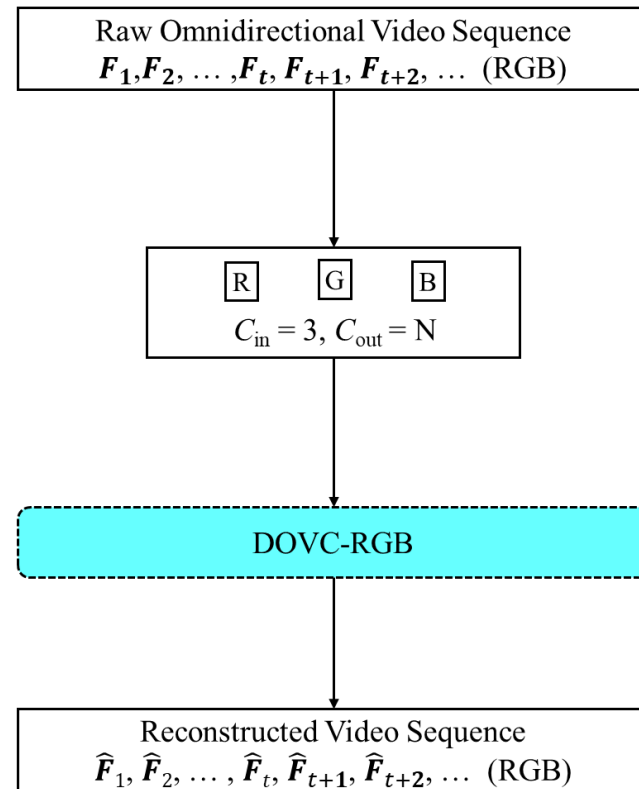


DOVC-YUV vs DOVC-RGB

3



JVET-Y0051



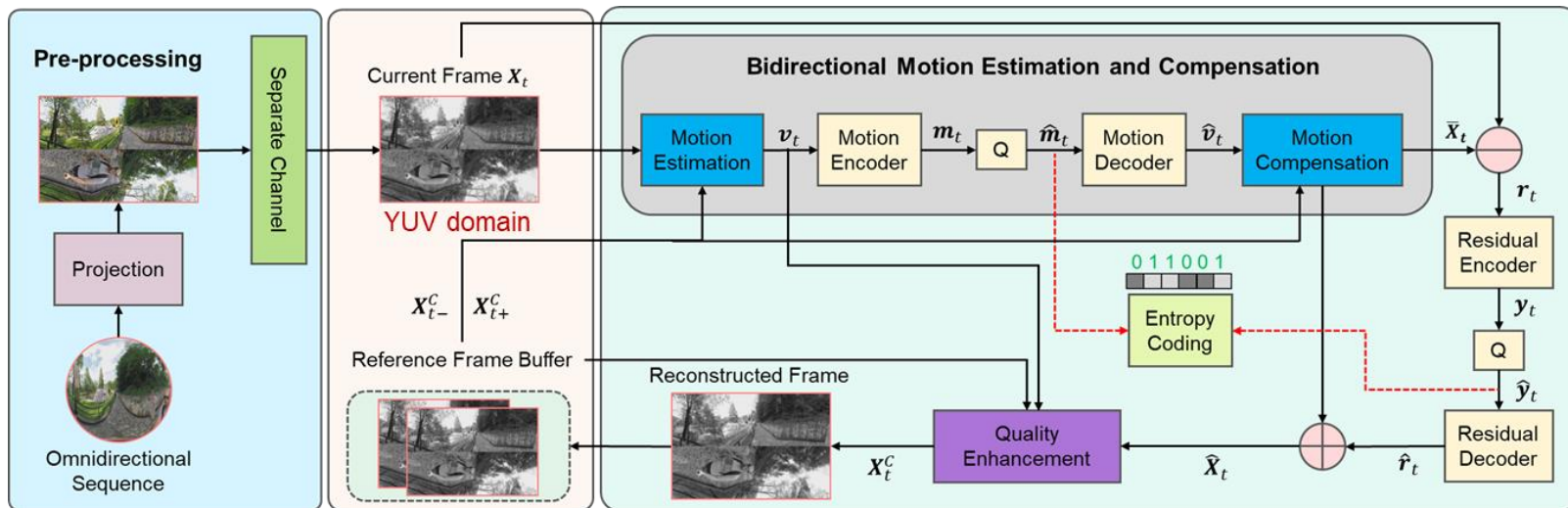
JVET-X0043

■ Network Architecture (Encoder)

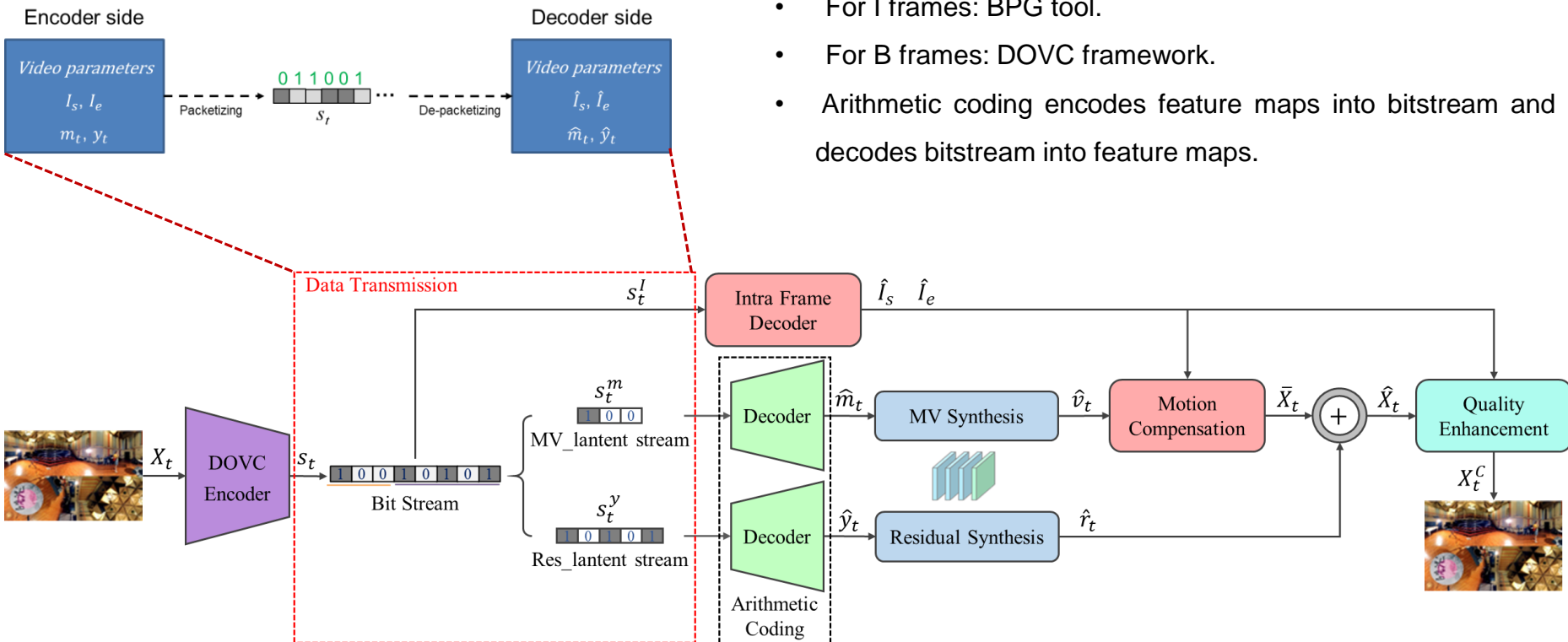
End-to-end deep omnidirectional video compression framework in YUV 4:2:0 domain.

DOVC mainly contains:

- Projection and Separate channels
- Bidirectional motion estimation
- Motion encoder/decoder
- Bidirectional motion compensation
- Residual encoder/decoder
- Quality enhancement
- Entropy coding



Network Architecture (Bitstream and Decoder)



| Network Information in Inference Stage | | |
|--|--|---|
| Mandatory | GPU Type | GeForce GTX1080 Ti |
| | Framework: | Python3.6, PyTorch1.6, CUDA10.1 |
| | Number of GPUs per Task | 1 |
| | Total Parameter Number | Including five model, i.e. $\lambda = 256, 512, 1024, 2048$ and 4096 for luma and chroma, respectively. Each luma's model is 12.3M and each chroma's model is 6.7M. |
| | Parameter Precision (Bits) | 32 (F) |
| | Memory Parameter (MB) | 89.54 MB for each model |
| | Test data information: | 360-degree sequences [3] provided by JVET (CTC) |
| | MAC (Giga) | $\text{kernel}^2 \times \sum (\text{in_channels} \times \text{out_channels})$ |
| Optional | Total Conv. Layers | DOVC is composed of a series of CNNs, including a total of 7 modules. |
| | Total FC Layers | 0 |
| | Total Memory (MB) | / |
| | Batch size: | 4 |
| | Patch size | Whole CMP frame (3840×2560) |
| | Changes to network configuration or weights required to generate rate points | / |
| | Peak Memory Usage | / |
| | Other information: | / |

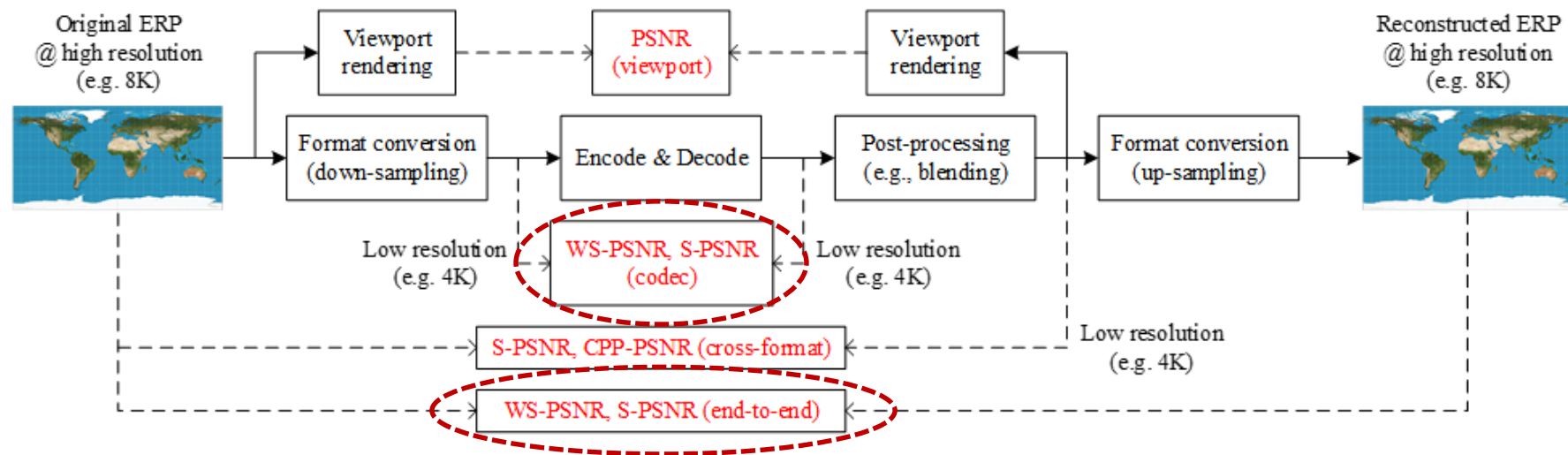
| Network Information in Training Stage | | |
|---------------------------------------|--|--|
| Mandatory | GPU Type | Nvidia Tesla V100, 32G |
| | Number of GPUs per Task | 2 |
| | Framework: | Python3.6, PyTorch1.6, CUDA10.0 |
| | Epoch: | 100 |
| | Batch size: | 4 |
| | Loss function: | RDO: $R + \lambda \cdot WMSE$ Note that R represents the total number of bits for encoding, λ is Lagrangian multiplier, and $WMSE$ is a weighted MSE based on different sphere-to-plane projection formats. |
| | Training time: | About 64 hours / luma model and 29 hours / chroma model |
| | Training data information: | VQA-ODV dataset [9] |
| Optional | Training configurations for generating compressed training data (if different to VTM CTC): | $\lambda = 256, 512, 1024, 2048, 4096$ |
| | Number of iterations: | 2000000 |
| | Patch size: | 1280×1280 |
| | Learning rate: | 1e-4 |
| | Optimizer: | ADAM |
| | Preprocessing: | See description in main text |
| | Mini-batch selection process: | / |
| | Other information: | / |
| | Preprocessing: | Projection: CMP (3×2) |

■ Experimental Setup

- **Training environment:** Nvidia Tesla V100 (32G), Python3.6, PyTorch1.6, CUDA10.0.
- **Test environment:** GeForce GTX1080Ti (12G), Python3.6, PyTorch1.6, CUDA10.1.
- **Anchors:** Intel Xeon CPU (Dual processor, RAM 32.G), HM-16.16 (with 360Lib-5.0), VTM-12.0 (with 360Lib-12.0), Visual Studio 2013.
- **Training dataset:** VQA-ODV^[1] .
- **Test dataset:** 360-degree sequences provided by JVET (CTC).
- **Other details:** Epoch = 100, Batch Size = 4, Learning Rate = 1e-4, Patch Size = 1280*1280, Test Size = 3840*2560 (whole frame with CMP format), Optimizer = ADAM, lambda = 256, 512, 1024, 2048 and 4096 for luma and chroma.

^[1] Proc. ACM Multimedia 2018.

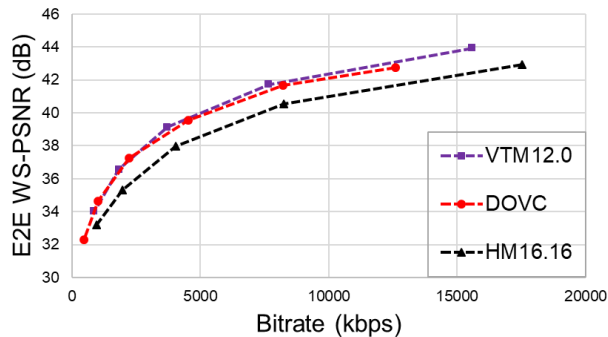
■ Test Procedure



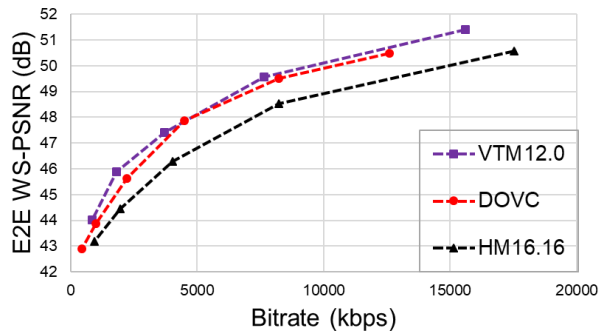
Test procedure of 360-degree video dataset provided by JVET.

Rate-distortion Curves

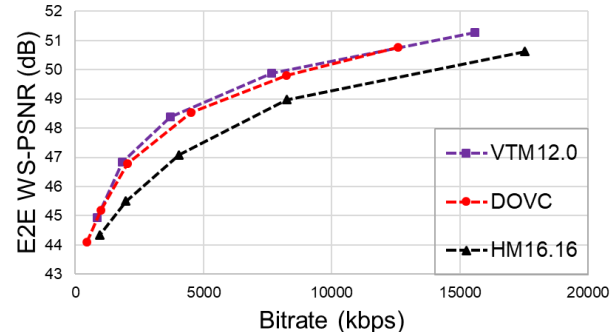
Class S1 (Y)



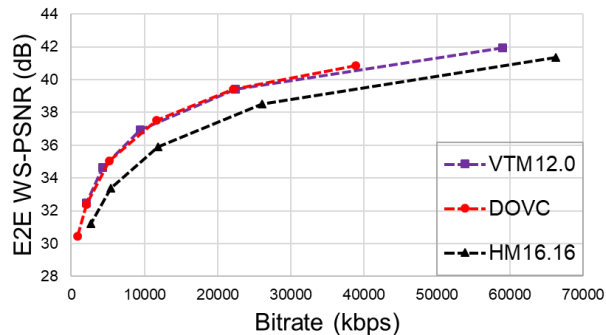
Class S1 (U)



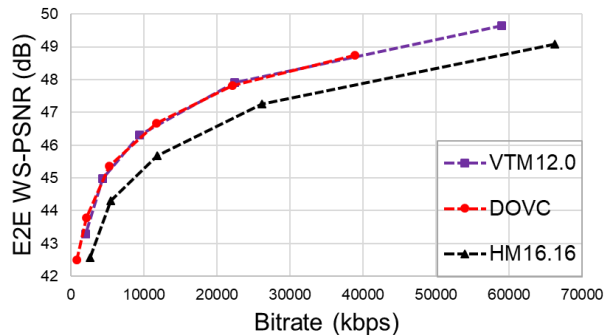
Class S1 (V)



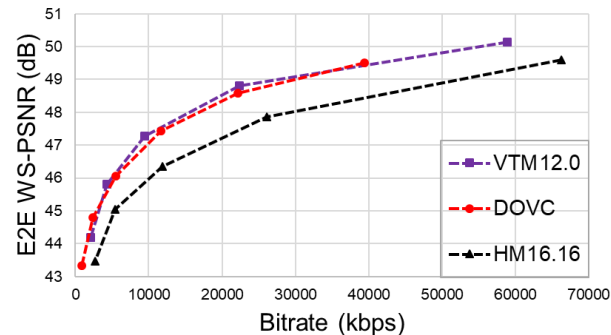
Class S2 (Y)



Class S2 (U)

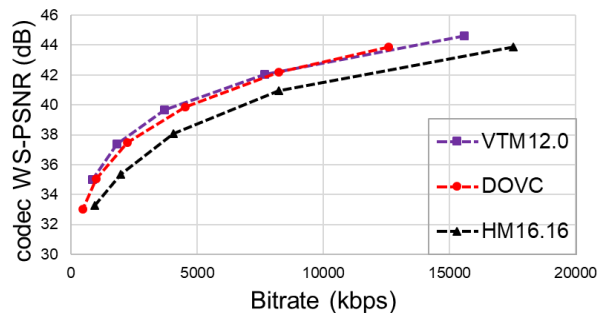


Class S2 (V)

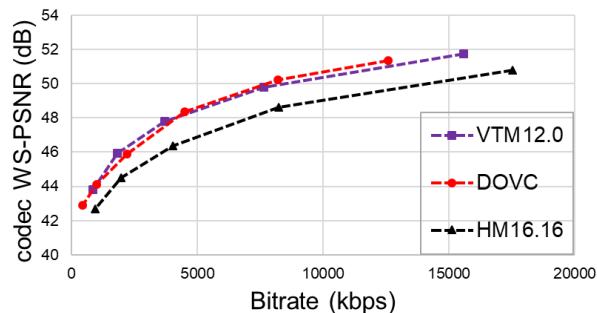


Rate-distortion curves in term of the end-to-end WS-PSNR on 360-degree video dataset.

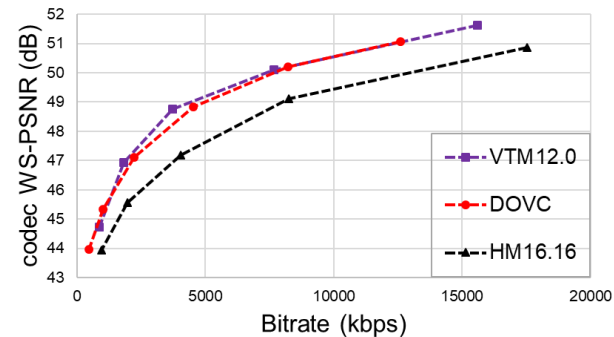
Class S1 (Y)



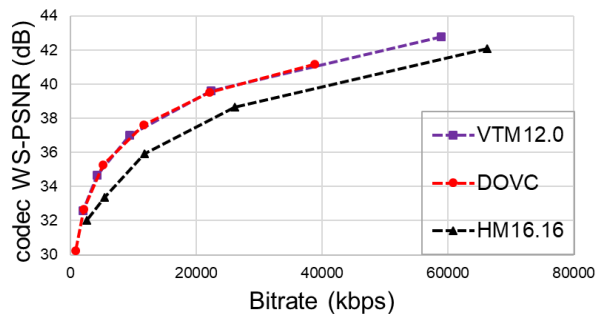
Class S1 (U)



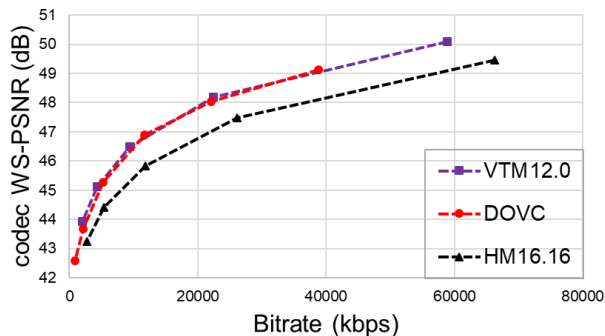
Class S1 (V)



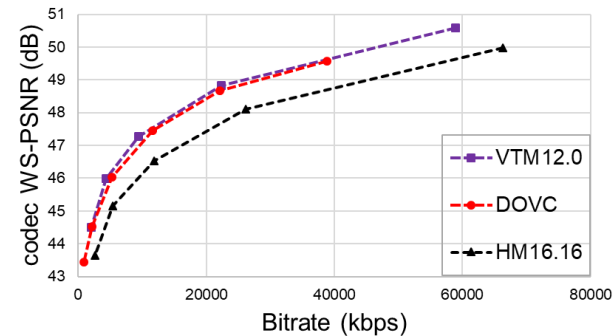
Class S2 (Y)



Class S2 (U)



Class S2 (V)



Rate-distortion curves in term of the codec WS-PSNR on 360-degree video dataset.

■ BDBR and BD-WSPSNR
















Performance comparison with HM16.16 on the 360-degree video sequences in terms of BDBR (%) and BD-WSPSNR (dB) (end-to-end).

| Sequences | Y | | U | | V | |
|---------------|----------|----------------|----------|----------------|----------|----------------|
| | BDBR (%) | BD-WSPSNR (dB) | BDBR (%) | BD-WSPSNR (dB) | BDBR (%) | BD-WSPSNR (dB) |
| Class S1 (6K) | -30.82 | 1.03 | -35.64 | 0.83 | -35.84 | 0.78 |
| Class S2 (8K) | -39.02 | 1.15 | -36.21 | 0.79 | -41.47 | 0.83 |
| Average | -34.10 | 1.09 | -35.87 | 0.81 | -38.09 | 0.81 |

Performance comparison with VTM12.0 on the 360-degree video sequences in terms of BDBR (%) and BD-WSPSNR (dB) (end-to-end).

| Sequences | Y | | U | | V | |
|---------------|----------|----------------|----------|----------------|----------|----------------|
| | BDBR (%) | BD-WSPSNR (dB) | BDBR (%) | BD-WSPSNR (dB) | BDBR (%) | BD-WSPSNR (dB) |
| Class S1 (6K) | 4.92 | -0.16 | 6.99 | -0.27 | 7.48 | -0.31 |
| Class S2 (8K) | 2.20 | -0.06 | 5.87 | -0.16 | 2.97 | -0.19 |
| Average | 3.83 | -0.11 | 6.54 | -0.22 | 5.67 | -0.25 |

Visual Comparison

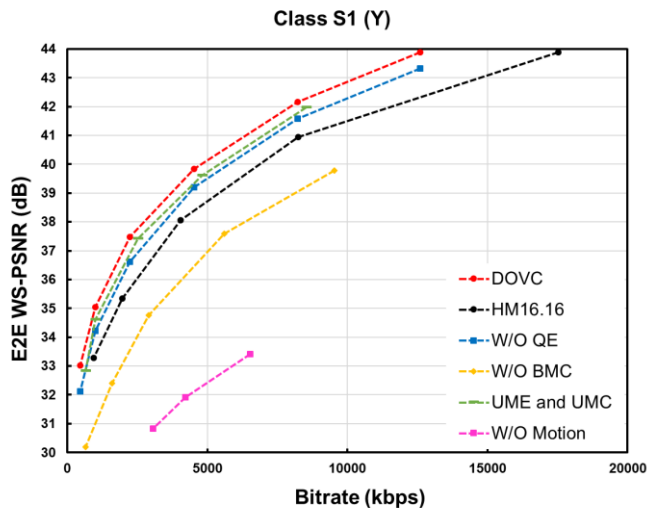
| Raw Frame | HEVC/H.265 | DOVC (W/O QE) | DOVC | VVC/H.266 |
|---|---|--|---|---|
|  |  |  |  |  |
| bpp / E2E WS-PSNR | 0.13bpp / 33.48dB | 0.11bpp / 33.82dB | 0.11bpp / 34.07dB | 0.12bpp / 34.13dB |
|  |  |  |  |  |
| bpp / E2E WS-PSNR | 0.10bpp / 35.18dB | 0.09bpp / 35.45dB | 0.09bpp / 35.68dB | 0.08bpp / 35.72dB |
|  |  |  |  |  |
| bpp / E2E WS-PSNR | 0.06bpp / 36.12dB | 0.08bpp / 36.62dB | 0.07bpp / 36.68dB | 0.09bpp / 36.90dB |

Visual comparison. The reconstructed frames are from HM16.16, DOVC-YUV (W/O QE), DOVC-YUV and VTM12.0.

■ Runtime and Ablation Study

Comparison of encoding and decoding time complexity of HM16.16, VTM12.0 and DOVC-YUV.

| Sequences | HEVC/H.265 | | VVC/H.266 | | DOVC | |
|---------------|------------|-----------|-----------|-----------|-----------|-----------|
| | Enc T (s) | Dec T (s) | Enc T (s) | Dec T (s) | Enc T (s) | Dec T (s) |
| Class S1 (8K) | 58326.28 | 117.21 | 167753.06 | 158.72 | 1787.93 | 216.42 |
| Class S2 (6K) | 125624.71 | 217.13 | 496459.81 | 367.95 | 6029.99 | 381.77 |
| Average | 91975.50 | 167.17 | 332106.44 | 263.34 | 3908.96 | 299.10 |

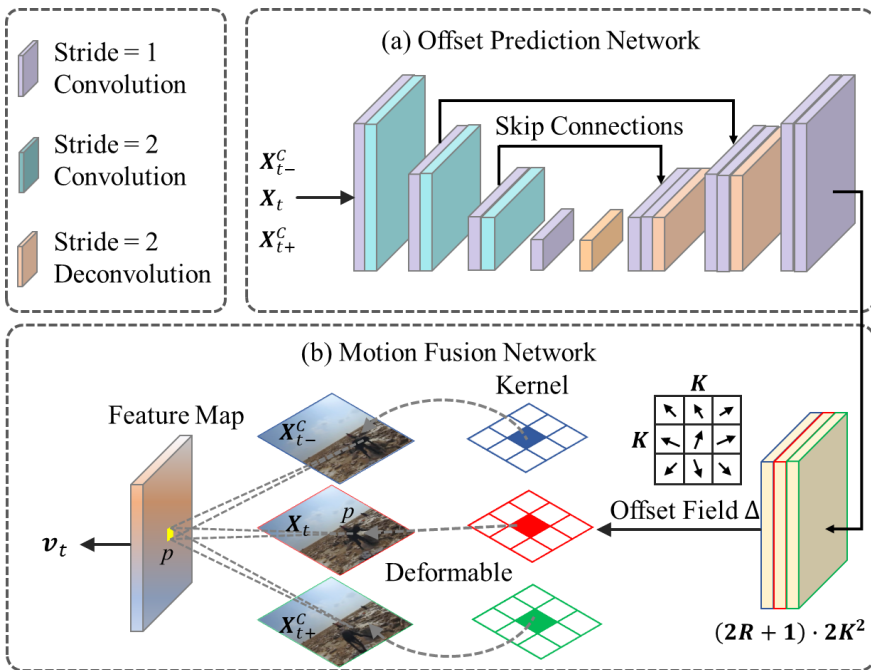
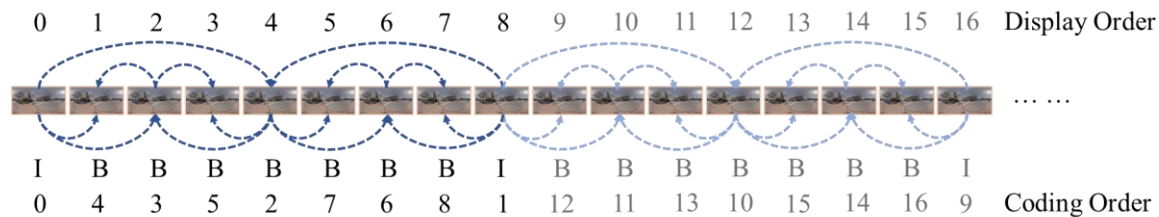


Rate-distortion curves of ablation study.

- **DOVC-YUV:** An end-to-end deep omnidirectional video compression framework in YUV domain based on CNNs.
- **DOVC-YUV vs HM16.16:** Compared with HM-16.16 under RA configuration, DOVC shows on average {34.10% (Y), 35.87% (U), and 38.09% (V)} and {1.09dB (Y), 0.81dB (U), and 0.81dB (V)} BD-rate reductions and BD-WSPSNR (end-to-end) gain on 360-degree video dataset (CTC) .
- **DOVC-YUV vs VTM12.0:** Compared with VTM-12.0 under RA configuration, DOVC shows on average {-3.83% (Y), -6.54% (U), and -5.67% (V)} and {-0.11dB (Y), -0.22dB (U), and -0.25dB (V)} BD-rate reductions and BD-WSPSNR (end-to-end) gain on the same dataset.
- **Encoding time:** The average encoding time of DOVC is only 0.0425 times that of HM-16.16 and 0.0118 times that of VTM-12.0 thanks to GPU parallel processing.

Appendix

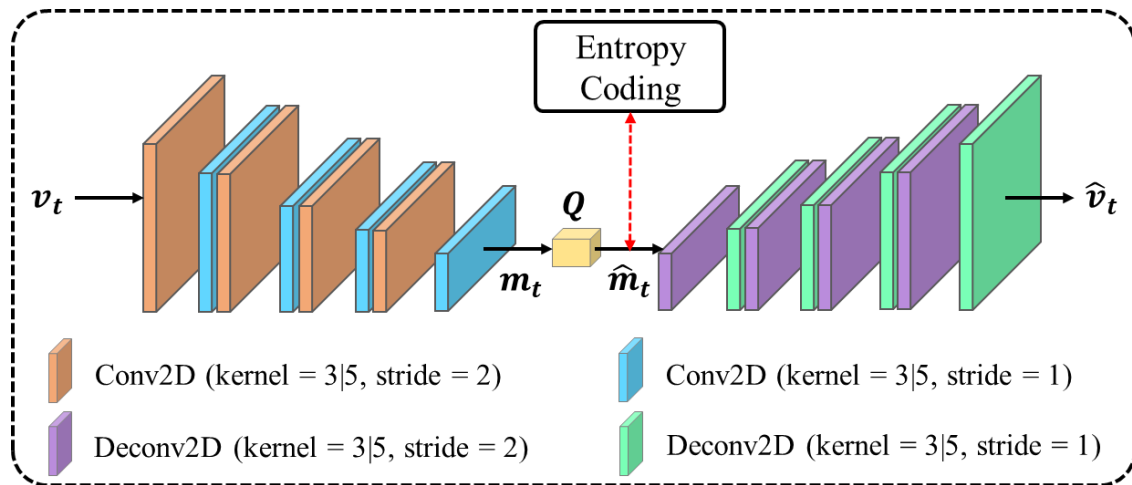
Bidirectional Motion Estimation



- Input: Current frame and reference frames.
- Convolutional and layers with stride = 2 for downsampling.
- Deconvolutional layers with stride = 2 for upsampling.
- Convolutional layers with stride = 1 and padding = 1 for keeping the shape of features.
- The kernel sizes for convolution and deconvolution are 3×3 and 4×4 .
- ReLU activation function for all layers.
- Output: Motion vector ($H \times W \times 2$).

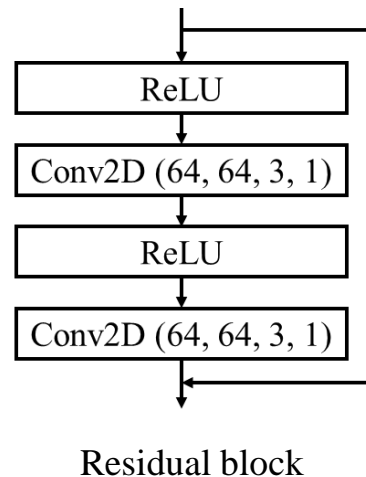
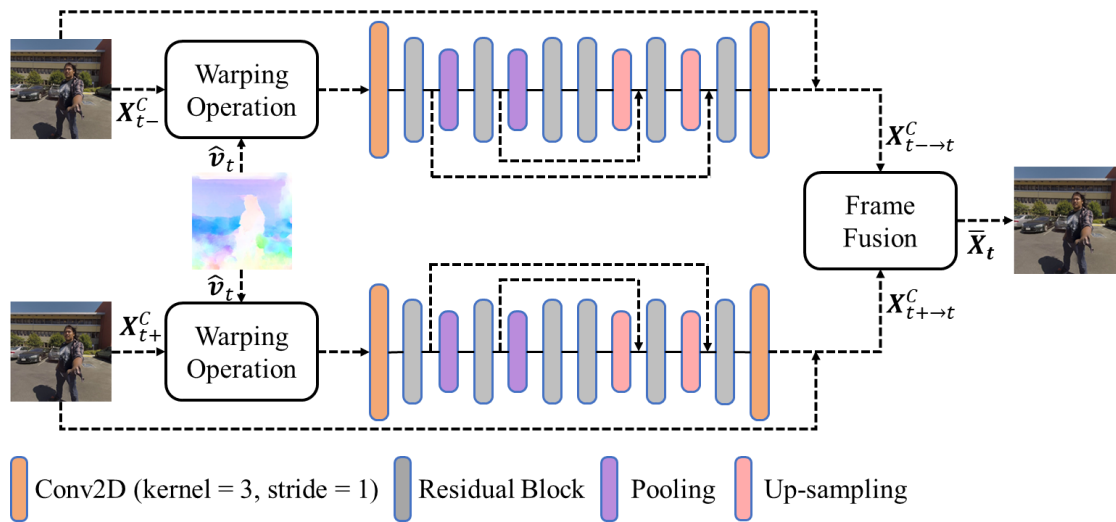
■ Motion Encoder/Decoder

- Input: Motion vector ($H \times W \times 2$).
- Quantization: Adding the uniform noise for training and $\text{round}(\cdot)$ operation for testing.
- Kernel: kernel size = 3×3 for chroma model and kernel size = 5×5 for luma model.
- Activation functions: LeakyReLU (i.e. PReLU)
- Output: Reconstructed motion vector ($H \times W \times 2$).
- \hat{m}_t are sent to the entropy coding module for generating bitstream.



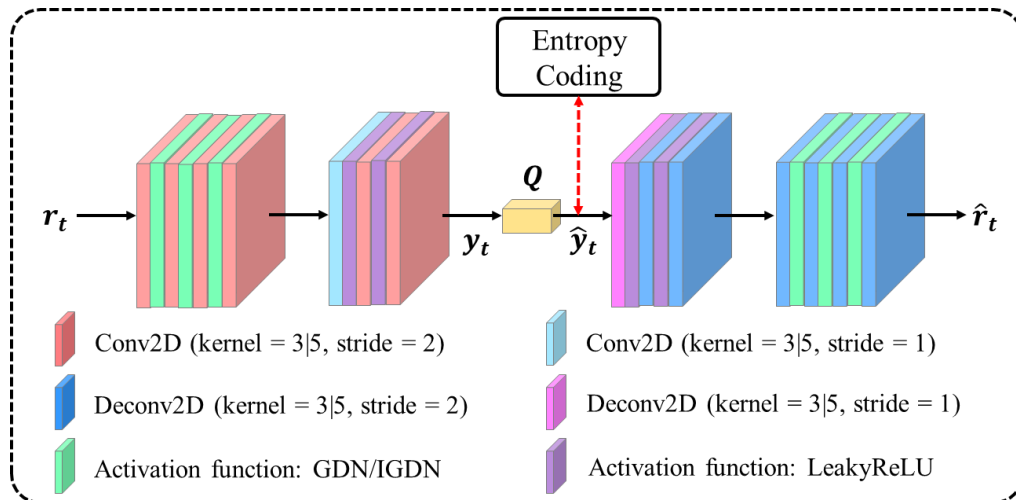
■ Bidirectional Motion Compensation

- Input: Reconstructed motion vector and reference frames.
- Convolutional layers (2) + Residual blocks (6).
- Convolutional layers with stride = 1 and kernel size = 3×3 .
- Average pooling for pooling operation and bilinear interpolation for to up-sampling.
- Output: Predicted frame.



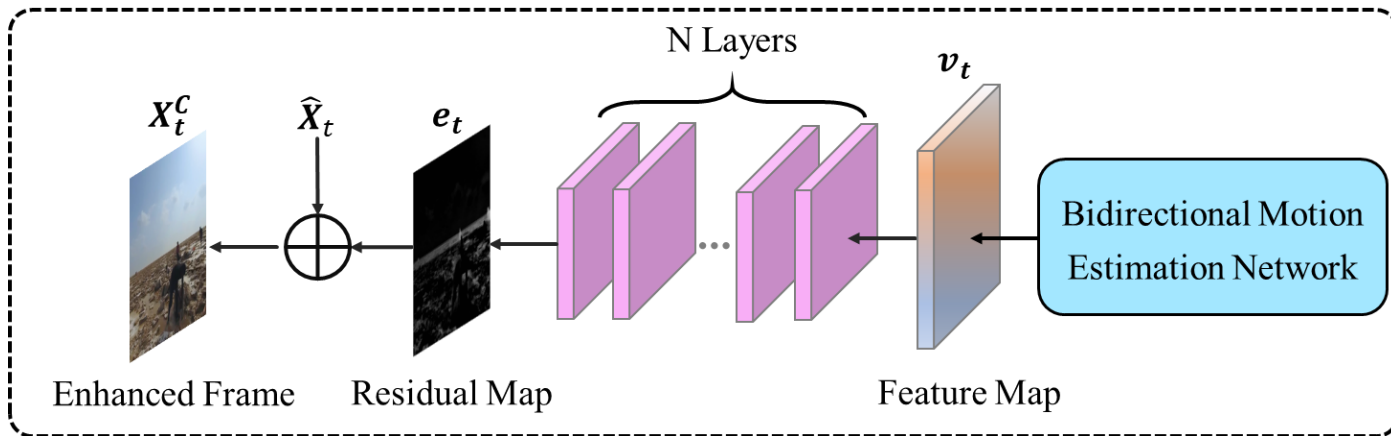
■ Residual Encoder/Decoder

- Input: Residual information.
- Quantization: Adding the uniform noise for training and $round(\cdot)$ operation for testing.
- Kernel: kernel size = 3×3 for chroma model and kernel size = 5×5 for luma model.
- Activation functions: PReLU for the six middle of hyper-prior network and GDN (iGDN) for other.
- Output: Reconstructed residual information.
- \hat{y}_t are sent to the entropy coding module for generating bitstream.



■ Quality Enhancement

- Input: Reuse temporal information, i.e. motion vector ($H \times W \times 2$).
- L convolutional layers with stride = 1.
- ReLU activation.
- Output: Enhanced residual map.



■ Projection and Loss Function

- Sphere-to-plane projection: CMP format.
- Weighted factors of CMP.
- Loss Function: WMSE.

$$w_{cmp}(i, j) = \left(3 + \frac{(i+1)^2 + (j+1)^2 - (i+j) \cdot a}{a^2/4} \right)^{-3/2}$$

$$W(i, j) = \frac{w(i, j)}{\sum_{i=0}^{Width-1} \sum_{j=0}^{Height-1} w(i, j)}$$

$$WMSE = \sum_{i=0}^{Width-1} \sum_{j=0}^{Height-1} \left(\mathbf{X}(i, j) - \hat{\mathbf{X}}'(i, j) \right)^2 \cdot W(i, j)$$

Loss $\min \left\{ J = \mathbf{R} + \lambda \cdot WMSE \left(\mathbf{X}_t, \hat{\mathbf{X}}_t \right) \right\}$



Projection

