

Description of SDR video coding technology proposal by IEEE 1857.10 Study Group

University of Science and Technology of China

Peking University

Harbin Institute of Technology

Wuhan University

DLVC-Deep Learning-based Video Coding

- The proposal is referred to as Deep Learning-Based Video Coding
 - It contains two coding tools, convolutional neural network-based loop filter (CNNLF) and convolutional neural network-based block adaptive resolution coding (CNN-BARC), which are based on deep convolutional neural network (CNN)
- In addition to the two CNN-based coding tools, a set of regular coding tools are proposed, focusing on block partition, inter coding, loop filtering and background modeling
- Compared to HM, the performance is -39.63% and -33.0% under CS1 and CS2, respectively

Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

Overview of the Added Techniques

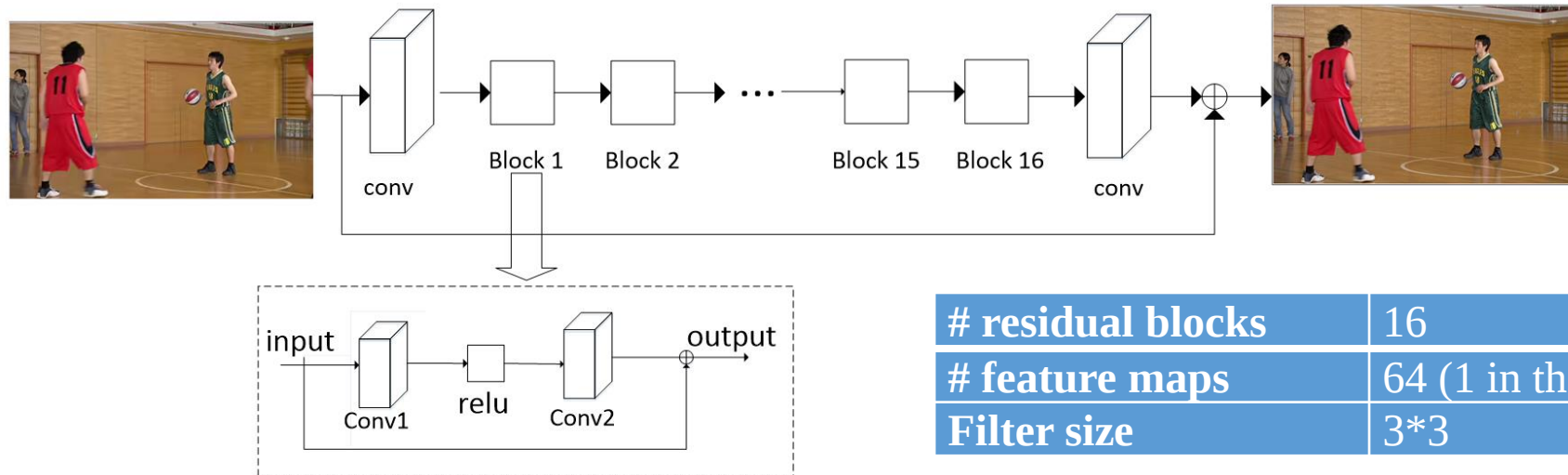
- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

Network Architecture of CNNLF

- CNNLF is an in-loop filter, which is applied on both intra frame and inter frame



- The network architecture



# residual blocks	16
# feature maps	64 (1 in the last layer)
Filter size	3*3

Training of CNNLF

- Dataset
 - DIV2K dataset, which contains 800 2k images, is used for training
- Compression
 - To train the model for a particular QP, images are compressed by intra coding at the corresponding QP
- Training patches
 - The original image and compressed image are both divided into nonoverlapping 70*70 sub-images

Other Issues of CNNLF

- RDO
 - For each CTU, two flags are signaled to indicate whether CNNLF will be applied on luma and chroma, respectively
- Number of models
 - We train 4 different models according to QPs {28, 34, 40, 46}, each frame chooses the model with the most similar QP

Performance of CNNLF

	Class SDR-A	Class SDR-B
CS1	-5.1%	-5.9%
CS2	-	-5.2%
AI	-5.7%	-7.1%

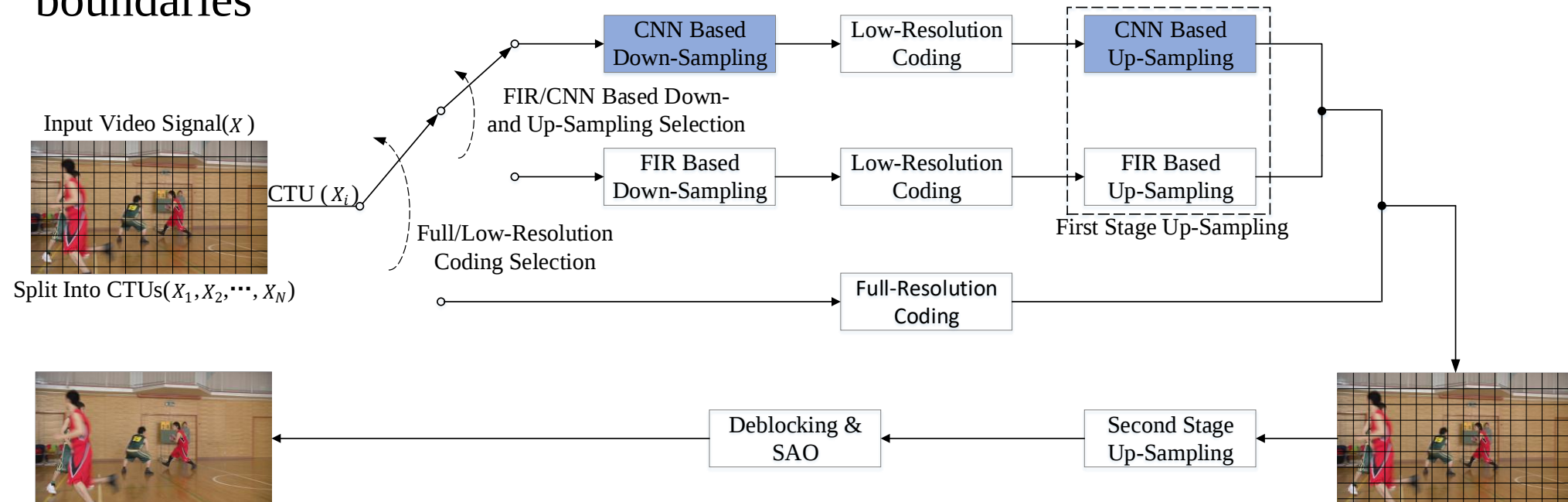
Tool-On Performance (Anchor: HM16.16+QTBT, Test: HM16.16+QTBT+CNNLF)

Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

Framework of CNN-BARC

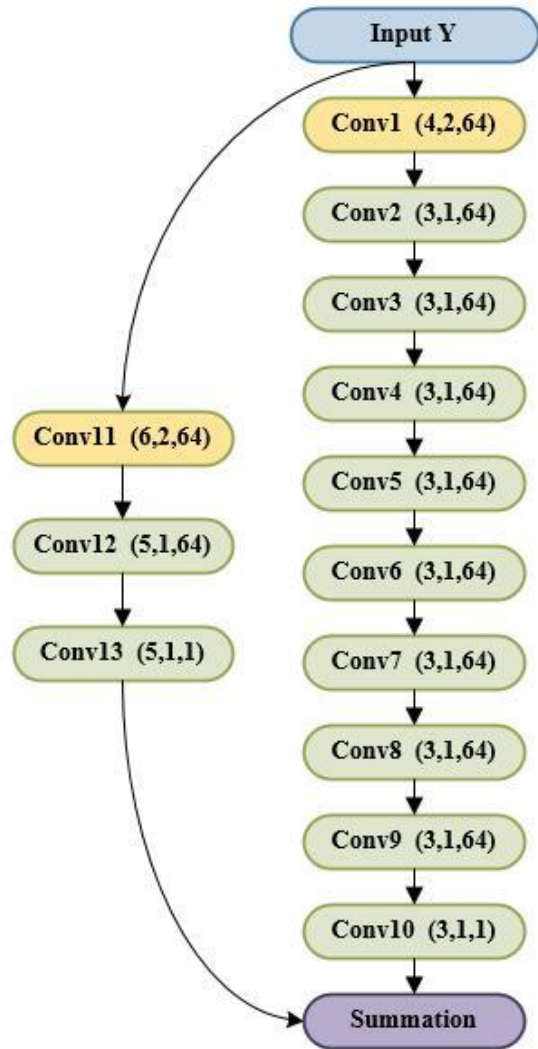
- Each CTU can be coded at its full-resolution version or the low-resolution version
- For the low-resolution coding scheme, there are two down/up-sampling methods
 - The FIR filters
 - The CNN-based filters
- The second stage up-sampling is used to refine each up-sampled CTU around its boundaries



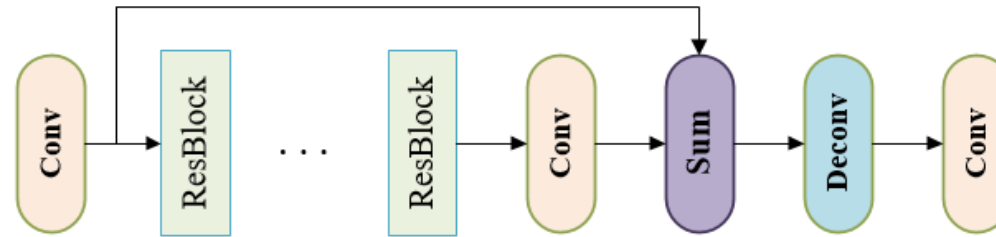
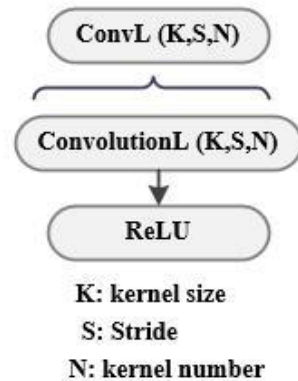
Down/Up-sampling Filters in CNN-BARC

- FIR based
 - The down-sampling filter coefficients are defined as follows:
$$\{4, 0, -12, 0, 40, 64, 40, 0, -12, 0, 4\}$$
 - The up-sampling filter coefficients are defined as follows:
$$\{-1, 4, -11, 40, 40, -11, 4, -1\}$$
- CNN based
 - Two convolutional neural networks are trained for the CNN-based down and up-sampling, respectively
 - The CNN for down-sampling is denoted as CRCNN (CNN for Compact-Resolution)
 - The CNN for up-sampling is denoted as CNN-SR (CNN for Super-Resolution)

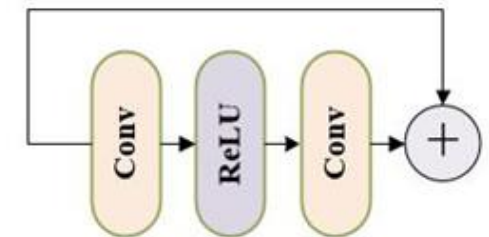
CRCNN and CNN-SR



(a): CRCNN



(b): CNN-SR

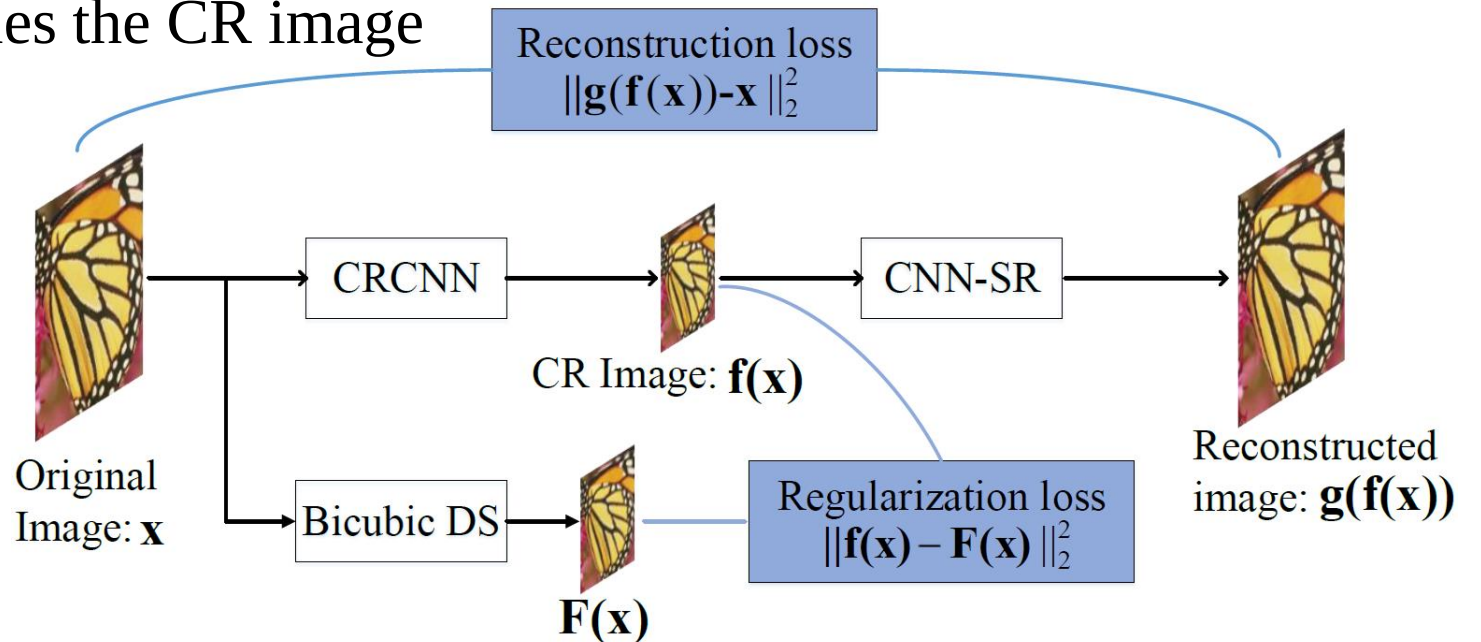


(c): Structure of ResBlock

- The architecture of CRCNN is shown in (a)
 - Stride of 2 represents reducing the size by a factor of 2
- The architecture of CNN-SR is shown in (b)
 - CNN-SR is composed of 16 residual blocks (Their structure are shown in (c))
 - Kernel size is 3×3 in convolutional layers
 - Kernel size is 6×6 in deconvolutional layers
 - Number of feature maps is 64 (1 for the last layer)

Training of CRCNN

- Dataset
 - DIV2K dataset, which contains 800 2k images, is used for training
- End-to-end training with two loss functions
 - Reconstruction loss is to keep the CR image as informative as possible compared to the original image
 - Regularization loss enables the CR image friendly to compression



Training of CNN-SR

- The original image is down-sampled by the CRCNN, then compressed by intra coding.
- CNN-SR is trained using the reconstructed low-resolution image and the corresponding high-resolution original image

CTU Boundary Pixels Refinement

- Whenever the bottom and right boundaries are available, the up-sampling operation is performed again for pixels near the bottom and right boundary of CTUs that have chosen the low-resolution coding mode.
- This up-sampling result replaces the previous result to refine the pixels near CTU boundaries
- The same process is performed at both encoder and decoder, then no overhead bit is required.



Other Issues of CNN-BARC

- RDO
 - Currently, CNN-BARC is only applied on the luma component of Intra frame (It is intentionally turned off in P and B frames)
 - For each CTU, two flags are signaled to indicate whether CNNBARC will be used, and which kind of down/up-sampling (FIR-based or CNN-based) will be used
- Number of models
 - We train 1 CRCNN model, and 4 different CNN-SR models according to 4 QPs

Benefit of CRCNN

- CRCNN outperforms bicubic down-sampling significantly, and achieves on average 2.25 dB improvement in terms of reconstruction quality over all the four testing datasets.
- Table also includes the results of bicubic down-sampling plus EDSR, which represent the state-of-the-art results of image SR. It is obvious that our CRCNN plus CNN-SR can outperform the results of EDSR by a considerable margin.
- Such results demonstrate that the proposed CRCNN indeed preserves more information during decreasing image resolution compared to bicubic down-sampling. And the preserved information can boost the performance of image SR significantly

	Bicubic ↓ Bicubic ↑	CRCNN ↓ Bicubic ↑	Bicubic ↓ EDSR ↑ [4]	Bicubic ↓ CNN-SR ↑	CRCNN ↓ CNN-SR ↑	Gain
Set5	33.67	33.41	38.11	37.69	38.88 (47.20)	1.19
Set14	30.01	29.81	33.92	33.13	35.40 (43.60)	2.27
BSD100	29.57	29.36	32.32	32.05	33.92 (42.49)	1.87
Urban100	26.56	26.42	32.93	31.01	33.68 (40.83)	2.67
Average	28.32	28.14	32.83	31.77	34.02 (41.91)	2.25

Performance of CNN-BARC+TT

	Class SDR-A	Class SDR_B
CS1	-1.8%	-1.0%
CS2	-	-0.3%
AI	-7.2%	-3.6%

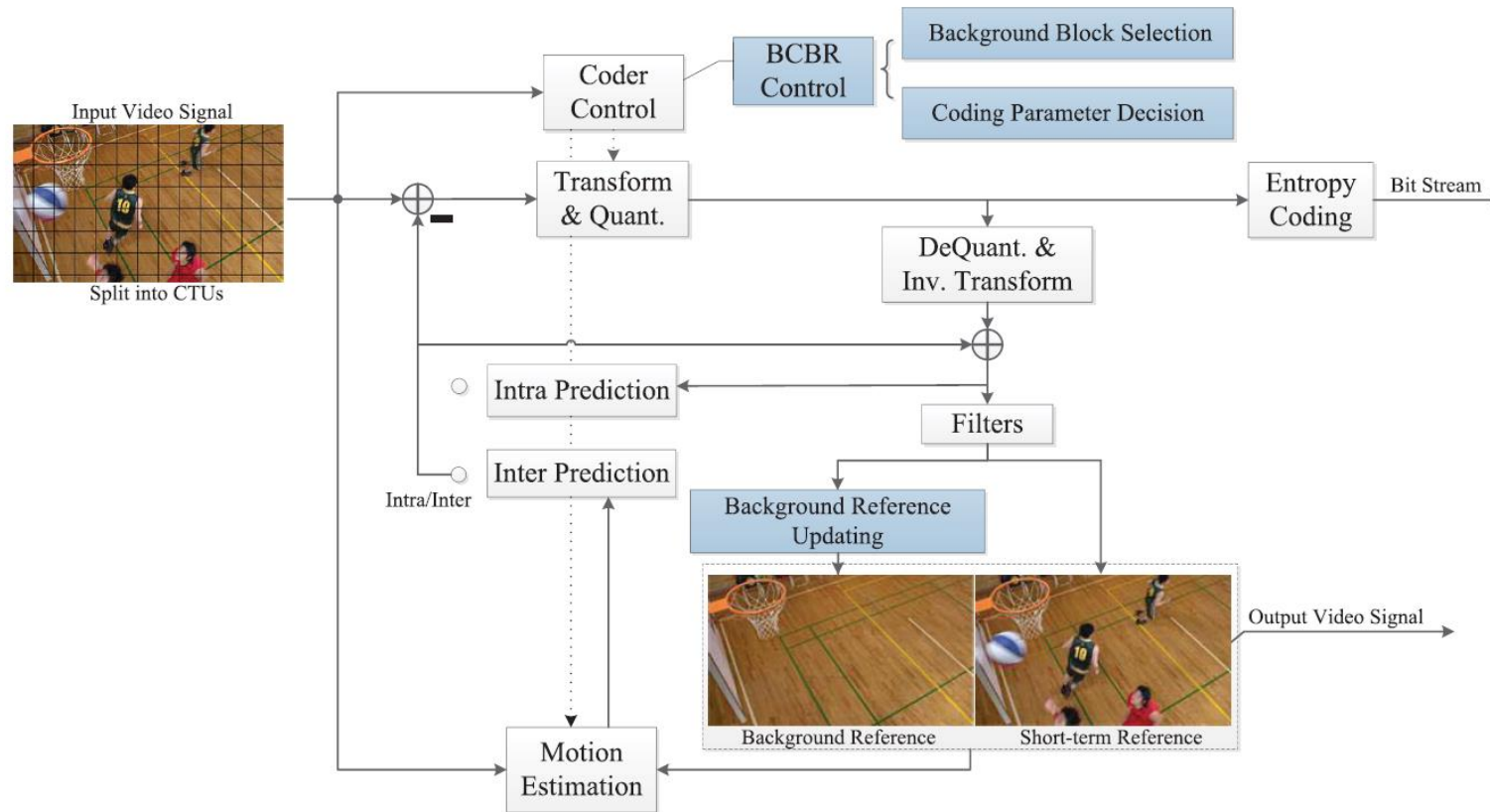
Tool-On Performance (Anchor: HM16.16+QTBT+TT, Test: HM16.16+QTBT+TT+CNN-BARC)

Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- **block-composed background reference (BCBR)**
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

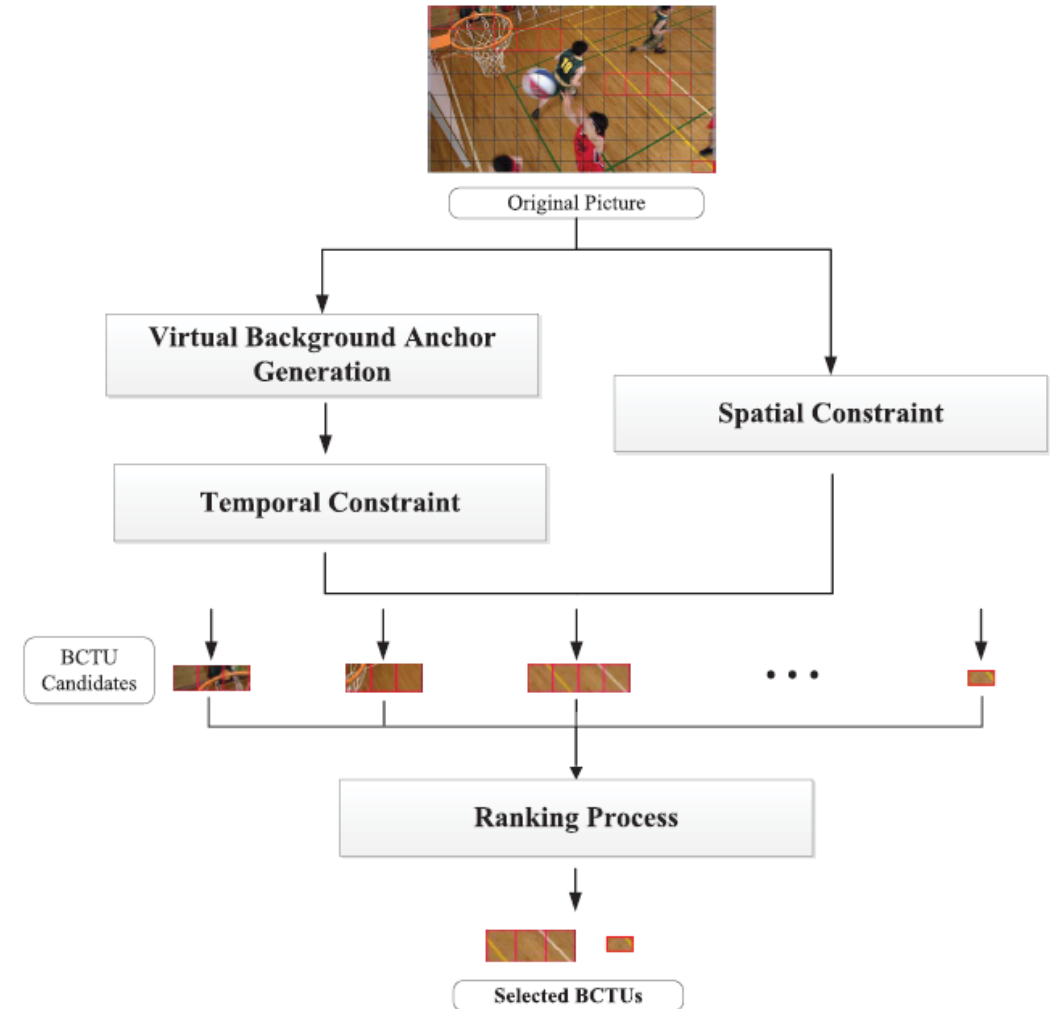
Framework of BCBR

- A high-quality background reference is generated to better exploit the long-term temporal correlation in the video.
- The flowchart of BCBR is shown in the figure below



Background Block Selection

- Step 1: Virtual Background Anchor Generation
- Step 2: two judgments, which contain temporal and spatial correlation constraints, are separately carried out to obtain BCTU candidates
- Step 3: To select the optimal CTUs among these candidates, a ranking process is conducted



Background Reference Updating

- When completing the coding of the current picture, the background reference buffer in the location of selected CTUs will be updated with the reconstructed ones. And the new background picture will be indicated as the long-term reference and utilized for the coding of the next picture.
- When a new IDR frame is coming, the background reference will be refreshed and initialized with the reconstructed IDR frame again. Then, a new BCBR progress starts.
- Since not enough information can be acquired to judge the selected BCTUs at the decoder, a flag signaling the BCTU is transmitted to the decoder.

Performance of BCBR

Sequence	Y
BQTerrace	0.1%
RitualDance	0.0%
MarketPlace	0.1%
BasketballDrive	0.1%
Cactus	-4.1%
Average	-0.8%

Tool-On Performance (Anchor: HM16.16+QTBT, HM16.16+QTBT+BCBR)

Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

QT-BT/TT tree

- In this contribution, a QT plus BT/TT tree (QT-BT/TT) structure is proposed
- In addition to quad-tree and binary-tree, triple-tree partition is employed to capture objects in the center of blocks while the quad-tree and binary-tree are splitting along block center
- There are horizontal and vertical triple-tree splits, as reported in the figure (d) and (e)

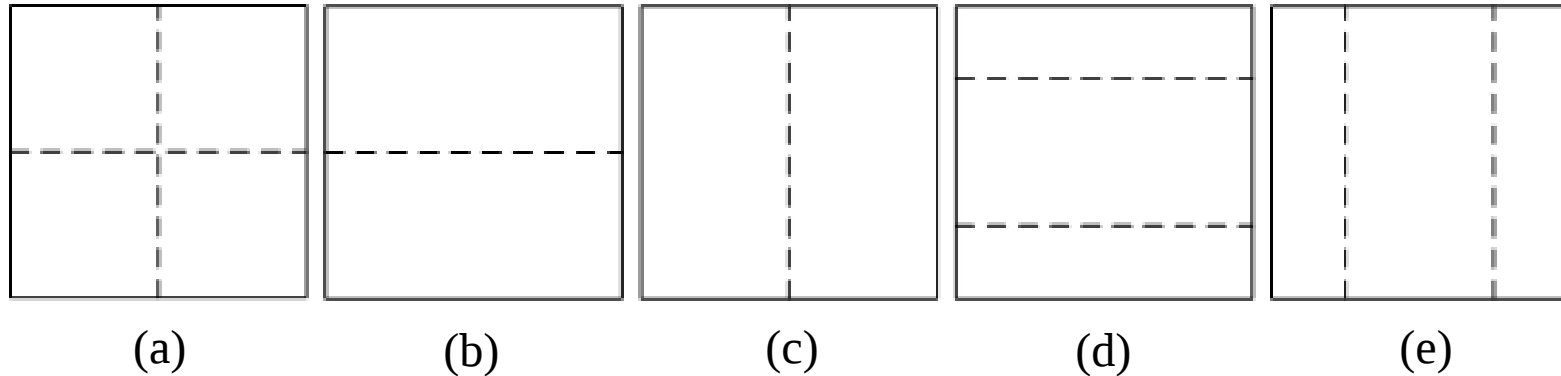


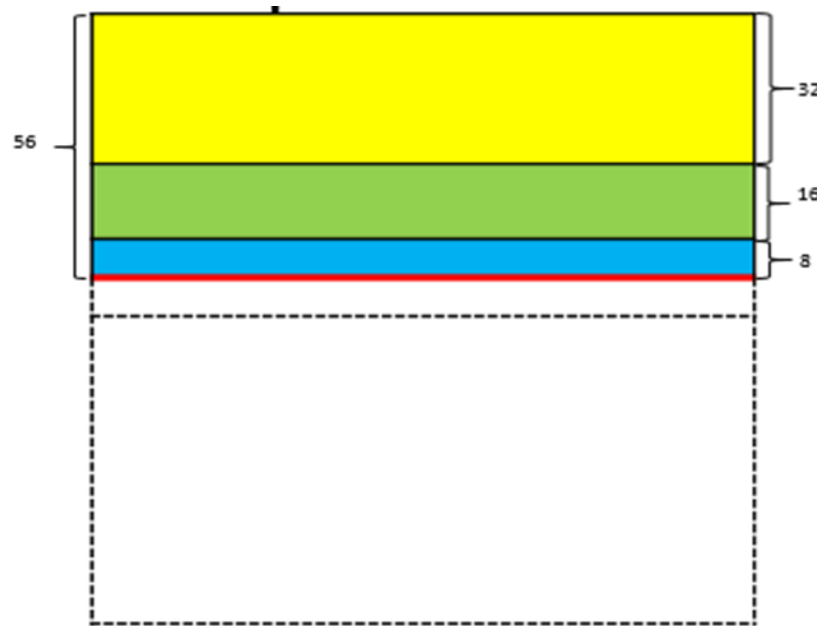
Figure. (a) quad-tree split; (b) horizontal binary-tree split; (c) vertical binary-tree split; (d) horizontal triple-tree split; (e) vertical triple-tree split.

Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

Forced binary-tree partitioning (FBP) for boundary CTUs

- In JEM, the forced quad-tree partitioning is used for boundary CTUs, which is not efficient
- In this proposal, binary-tree partitioning is used for boundary CTUs



Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- **non local filter (NF)**
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

Non-local Filter

- NF filter is used as an in-loop filter after SAO and before ALF
- Three filtering modes
 - The whole reconstructed frame is not filtered
 - The whole reconstructed frame is filtered
 - Adaptive filtering decisions for different blocks with a predefined block size
- For each frame, the best filtering mode is decided by RDO
- Filtering process of one block
 - Step 1 : K nearest neighbors of the block are found according to the Euclidean distance, those patch groups are then reshape into a matrix
 - Step 2: Perform Singular value decomposition(SVD) on the matrix to get the sparse representation
 - Step 3: Threshold the singular values of the SVD transform
 - Step 4: Reconstruct blocks using the coefficients after thresholding

Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced boundary partition (FBT)
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

Process of FRUC Improvement

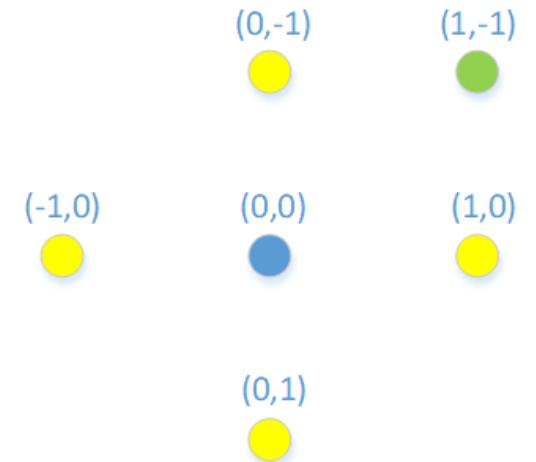
- Three FRUC-TM modes:
 - Forward uni-directional prediction
 - backward uni-directional prediction
 - Bi-directional prediction
- Mode derivation process:
 - Uni-directional or Bi-directional prediction?
 - Using a flag
 - Forward or Backward Uni-directional prediction?
 - Comparing template costs of two uni-directional predictions

Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

DMVR Improvement

- For the decoder-side motion vector refinement (DMVR) technique in JEM, the two motion vectors are further refined by a template matching process.
- This contribution proposes an improvement to the current DMVR method
 - Two-stage refinement search
 - Stage 1 : max 11 integer pixel search iteration, go to stage 2 if early termination criteria met
 - Stage 2 : 1 half pixel search iteration
 - Same search pattern is employed in each search iteration
 - Bilateral matching cost instead of template matching cost

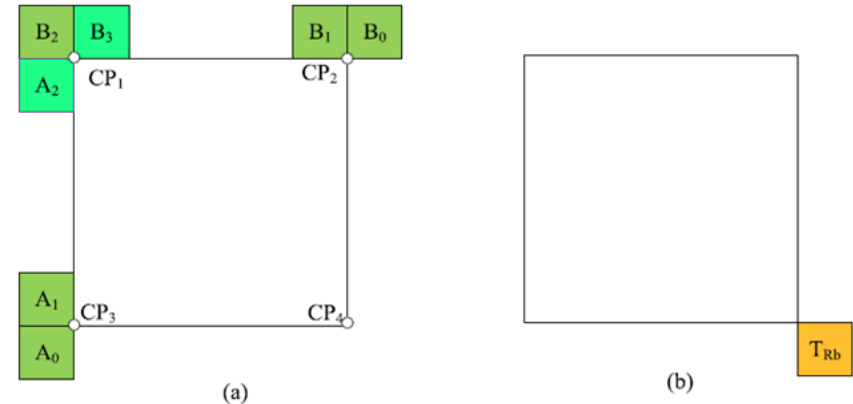


Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

MERGE Improvement

- The complex merge mode combines several blocks with similar motions into a single region
 - As a result, all blocks within this region share the same motion information, and thus the motion parameters need to be transmitted only once
 - This method considers bilinear interpolation model, six-parameter affine model and four-parameter affine model simultaneously
- The control points are needed to determine the complex motion models. The first step is to determine the candidates of the control points
 - The candidates of the control points are shown in the figure below. CP_k represents the k th control point. The spatial candidates for predicting the motion information of CP_1 , CP_2 , and CP_3 are shown in Fig (a). The temporal candidate position for predicting the motion information of CP_4 is shown in Fig (b)
 - The motion information of each control point is determined in the following priority order
 - 1) For CP_1 , the checking priority is B_2 , A_2 , and B_3
 - 2) For CP_2 , the checking priority is B_0 and B_1
 - 3) For CP_3 , the checking priority is A_0 and A_1
 - 4) For CP_4 , T_{RB} is used



MERGE Improvement

- The control points are then used to construct the candidate list. The candidate model list is constructed in the following order.
 - 1) Affine (CP2, CP3)
 - 2) Affine (CP1, CP3)
 - 3) Affine (CP1, CP2, CP3)
 - 4) Affine (CP1, CP2)
 - 5) Affine (CP2, CP4);
 - 6) Affine (CP3, CP4);
 - 7) Affine (CP1, CP4);
 - 8) Bilinear
 - 9) Affine (CP1, CP2, CP4);
 - 10) Affine (CP2, CP3, CP4);
 - 11) Affine (CP1, CP3, CP4);
- The motion information of the control points and the high-order motion model are then used to interpolate the MV of each sub-block in the current coding unit
 - The reference index with the highest utilization rate among all the reference indices is selected as the final reference index. The control points with different reference indices are scaled to the final reference index
 - Motion vectors of all control points are scaled to the same reference picture for constructing a motion model

Overview of the Added Techniques

- convolutional neural network-based loop filter (CNNLF)
- convolutional neural network-based block adaptive resolution coding (CNN-BARC)
- block-composed background reference (BCBR)
- triple-tree partition (TT)
- forced binary-tree partitioning (FBP) for boundary CTUs
- non local filter (NF)
- frame-rate up-conversion improvement (FRUCI)
- decoder side motion vector refinement improvement (DMVRI)
- merge improvement (MERGEI)
- affine improvement (AFFINEI)

AFFINE Improvement

- AFFINE_MERGE_IMPROVE
 - Instead of finding the first neighboring block with affine mode, the improvement tries to find the neighboring block with the largest coding unit size ($W \times H$) as the affine merge candidate
- AFFINE_AMVP_IMPROVE
 - Add the neighboring blocks with affine mode to the affine AMVP candidate list, use the neighboring affine parameters to generate the motion vectors predictors
- AFFINE_SIX_PARAM
 - In affine inter mode, four parameters or six-parameter affine model is decided by a flag

Performance of DLVC

	Constraint Set 1					
	Over HM16.16					
	BD-rate(Y)	BD-rate(U)	BD-rate(V)	BD-PSNR(Y)	BD-PSNR(U)	BD-PSNR(V)
Average SDR-A	-42.45%	-46.64%	-47.24%	1.55	1.16	1.26
Average SDR-B	-36.81%	-49.55%	-50.14%	1.48	1.04	1.30
Average all	-39.63%	-48.10%	-48.69%	1.52	1.10	1.28

	Constraint Set 1					
	Over JEM7.0					
	BD-rate(Y)	BD-rate(U)	BD-rate(V)	BD-PSNR(Y)	BD-PSNR(U)	BD-PSNR(V)
	-10.95%	-8.27%	-9.72%	0.29	0.13	0.18
	-9.27%	-10.91%	-10.23%	0.29	0.15	0.19
	-10.11%	-9.59%	-9.97%	0.29	0.14	0.18

	Constraint Set 2					
	Over HM16.16					
	BD-rate(Y)	BD-rate(U)	BD-rate(V)	BD-PSNR(Y)	BD-PSNR(U)	BD-PSNR(V)
Average SDR-B	-33.00%	-49.64%	-51.50%	1.32	0.87	1.14

	Constraint Set 2					
	Over JEM7.0					
	BD-rate(Y)	BD-rate(U)	BD-rate(V)	BD-PSNR(Y)	BD-PSNR(U)	BD-PSNR(V)
	-11.83%	-13.22%	-16.49%	0.39	0.17	0.27

Software implementation description

- DLVC was developed on top of the JEM reference software
- In addition, DLVC uses a well-known deep learning framework—Caffe to fulfill the CNN-BARC and CNNLF tools
- Currently, DLVC was developed under Windows OS with Visual Studio 2015 and x64 configuration
- The Caffe for Windows (<https://github.com/BVLC/caffe/tree/windows>) is adopted for DLVC
 - More specifically, the Caffe for Windows is compiled and built as a DLL, this DLL as well as Caffe's dependencies DLL's are necessary when running DLVC executables. In addition, the related header files and library files are included when compiling and building DLVC executables
- It is worth noting that Caffe for Windows provides the flexibility to use CPU or GPU. The current DLVC is however stick to CPU only