

Graceful Power Degradation

Z. Ma and A. Segall
JCTVC-B114

- Motivation
 - Power is a raising issue for future applications
 - Fixed applications: higher power results in increased cost
 - Mobile applications: higher power results in decreased battery life, increased heat and increased cost
- Major sources in the HEVC system
 - Memory bandwidth – increasing with higher resolutions
 - Pixel processing – also increasing with higher resolutions. Moreover, higher coding efficiency typically comes at the expense of higher complexity

Memory Bandwidth

Pixel Operations

- Our solution
 - Graceful power degradation
 - At the high level: enable a low-resolution and low memory bandwidth decoding mode within the HEVC bit-stream
 - Specifically, with the same bit-stream
 - Support decoding a low-resolution version of the sequence (without completely decoding the high resolution)
 - Support decoding a full-resolution version of the sequence
 - Support switching between these versions on a frame-by-frame basis and without drift

Bit-stream



High Resolution Mode



Low Power Mode

- Advantages

- Decoders may choose to operate at either:
 - low resolution – low power*or*
 - Full resolution – high power
- Additional benefit: Devices that are not capable of displaying the full resolution may decode a low resolution version
 - Without decoding the full resolution and down-sampling
 - Example1: HD decode on a mobile terminal with non-HD display
 - Example2: 8K/4K decode for 1080p display

Bit-stream



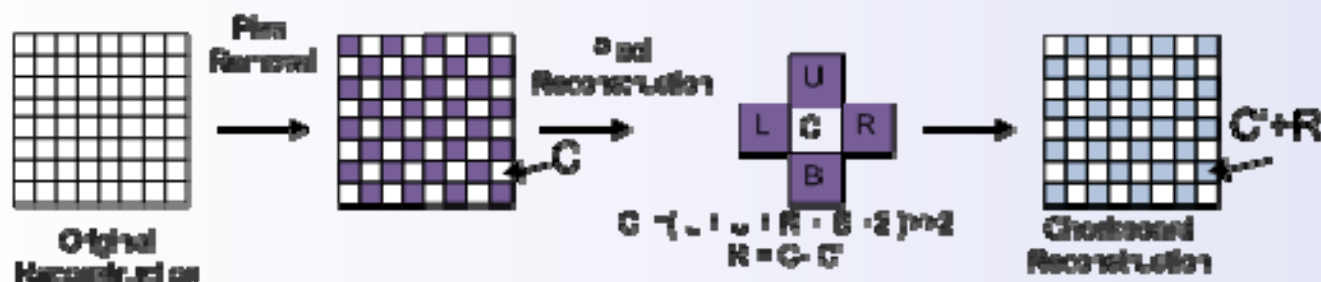
4k Resolution Mode



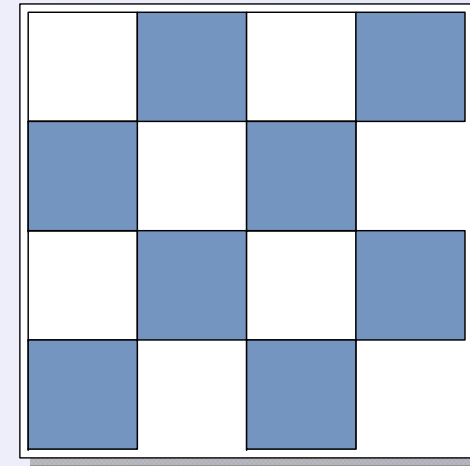
HD Resolution Mode

■ System Design

- Step #1: Divide reference picture buffer into a low-resolution and high-resolution component
 - Low-resolution component is compressed independently
 - High-resolution component predicted from low-resolution data. Residual then coded.
 - We use a checker-board/quincunx sampling



- Frame buffer compression
 - Low-resolution:
 - We use a modified AMBTC (adaptive moment BTC)
 - Compress 12bpp to 8bpp
 - High resolution
 - We predict pixels using bi-linear interpolation (4-point)
 - Residual coded with modified AMBTC
 - Compress 13bpp to 8bpp



Note: ~ 1/3 reduction in memory bandwidth

■ Frame Buffer Compression Performance

Size 1920x1080 **TMuC v0.3**
Coding structure HierB 10 s frames

		Kimono_1080p	ParkScene_1080p	Cactus_1080p	BasketballDrive_1080p	BQTerrace_1080p	Average
HEVC-v0.3 SLA 8/8	BD rate	0.00	0.24	0.63	0.01	1.01	0.378
	BD PSNR	0.00	-0.01	-0.01	0.00	-0.01	-0.006

Size wvga 832x480 **TMuC v0.3**
Coding structure HierB 10 s frames

		BasketballDrill_wvga	BQMail_wvga	PartyScene_wvga	RaceHorse_wvga		Average
HEVC-v0.3 SLA 8/8	BD rate	0.44	0.48	0.25	0.02		0.298
	BD PSNR	-0.02	-0.02	-0.01	0.00		-0.012

Size wqvga 416x240 **TMuC v0.3**
Coding structure HierB 10 s frames

		Basketballpass_wqvga	BQSquare_wqvga	BlowingBubble_wd	RaceHorse_wqvga		Average
HEVC-v0.3 SLA 8/8	BD rate	0.22	1.15	0.34	0.04		0.437
	BD PSNR	-0.01	-0.04	-0.01	0.00		-0.017

■ System Design

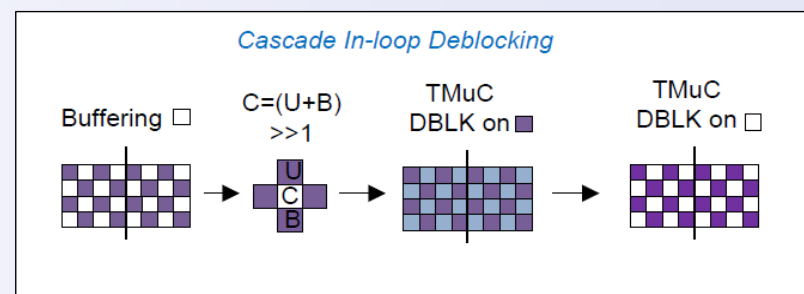
■ Step #2:

- Transmit a flag in the bit-stream to signal low-resolution capability
- When flag is signaled, do not read or store residual information in the buffer
 - Additional 2x reduction in memory bandwidth (2/3 reduction)

1.1.1 Picture parameter set RBSP syntax

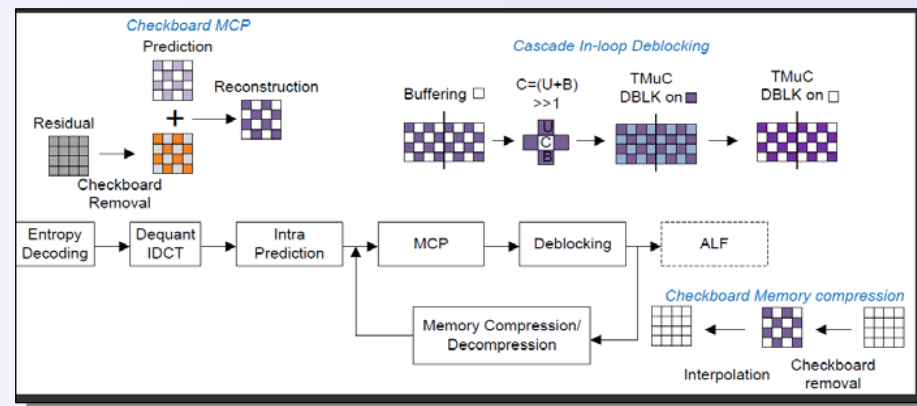
	C	Descriptor
pic_parameter_set_rbsp() {		
pic_parameter_set_id	1	ue(v)
seq_parameter_set_id	1	ue(v)
low_power_decode_id	1	u(1)
num_ref_idx_l0_default_active_minus1	1	ue(v)
num_ref_idx_l1_default_active_minus1	1	ue(v)
pic_init_qp_minus26 /* relative to 26 */	1	se(v)
constrained_intra_pred_flag	1	u(1)
for(i=0; i<15; i++){		
numAllowedFilters[i]	1	ue(v)
for(j=0; j<numAllowedFilters; j++){		
filtIdx[i][j]	1	ue(v)
}		
}		
rbsp_trailing_bits()	1	
}		

- Step #3
 - When flag is signaled, modify de-blocking operation
 - De-blocking decisions made on low-resolution pixels
 - Same decision process as TMuC – use interpolation to predict high-resolution pixels
 - Weak filter unchanged
 - Strong filter
 - Filter low and high-resolution pixels in cascade
 - Store frame after de-blocking (prior to ALF)



Results

- With Steps #1-#3, we can operate the decoder completely at the low-resolution point when the bit-stream contains the low-resolution flag
- Steps
 - Load low-resolution pixels from frame buffer
 - Perform motion compensation at low-resolution using modified interpolation filters
 - Possible because of bi-linear interpolation of high resolution data
 - Compute inverse transform at the low-resolution pixel locations
 - Perform de-blocking as in Step #3

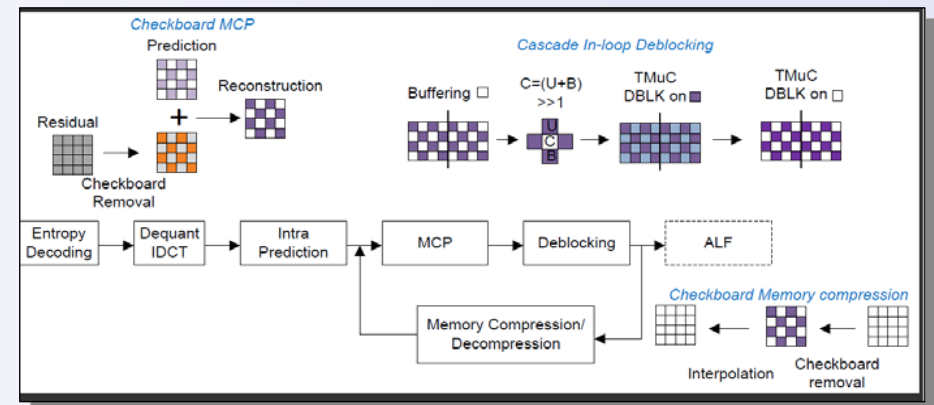


Results

- With Step #1-#3, we can also operate the decoder completely at the full-resolution point when the bit-stream contains the low-resolution flag

Steps

- Load low-resolution pixels from frame buffer; interpolate high-resolution pixels
- Perform motion compensation as normal
- Compute inverse transform at the low-resolution and high-resolution pixel locations
- Perform de-blocking as in Step #3

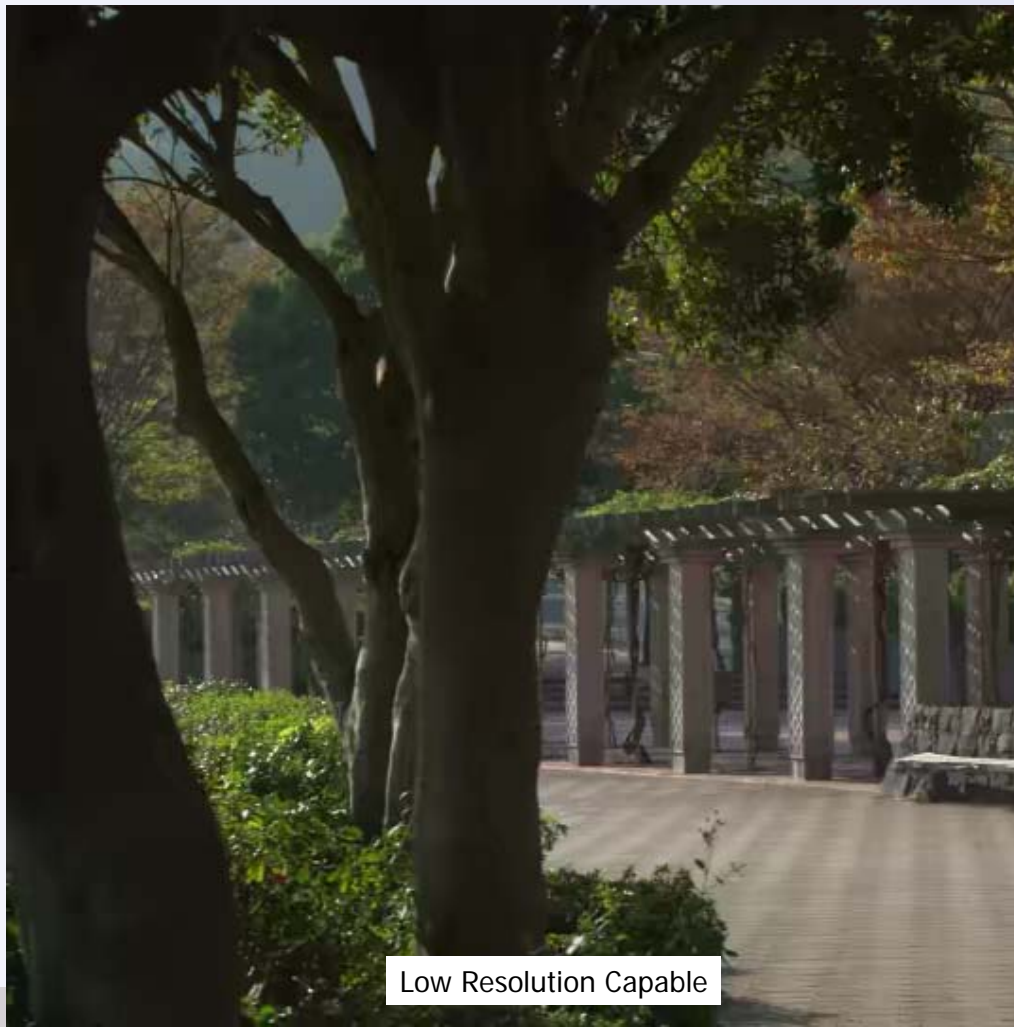
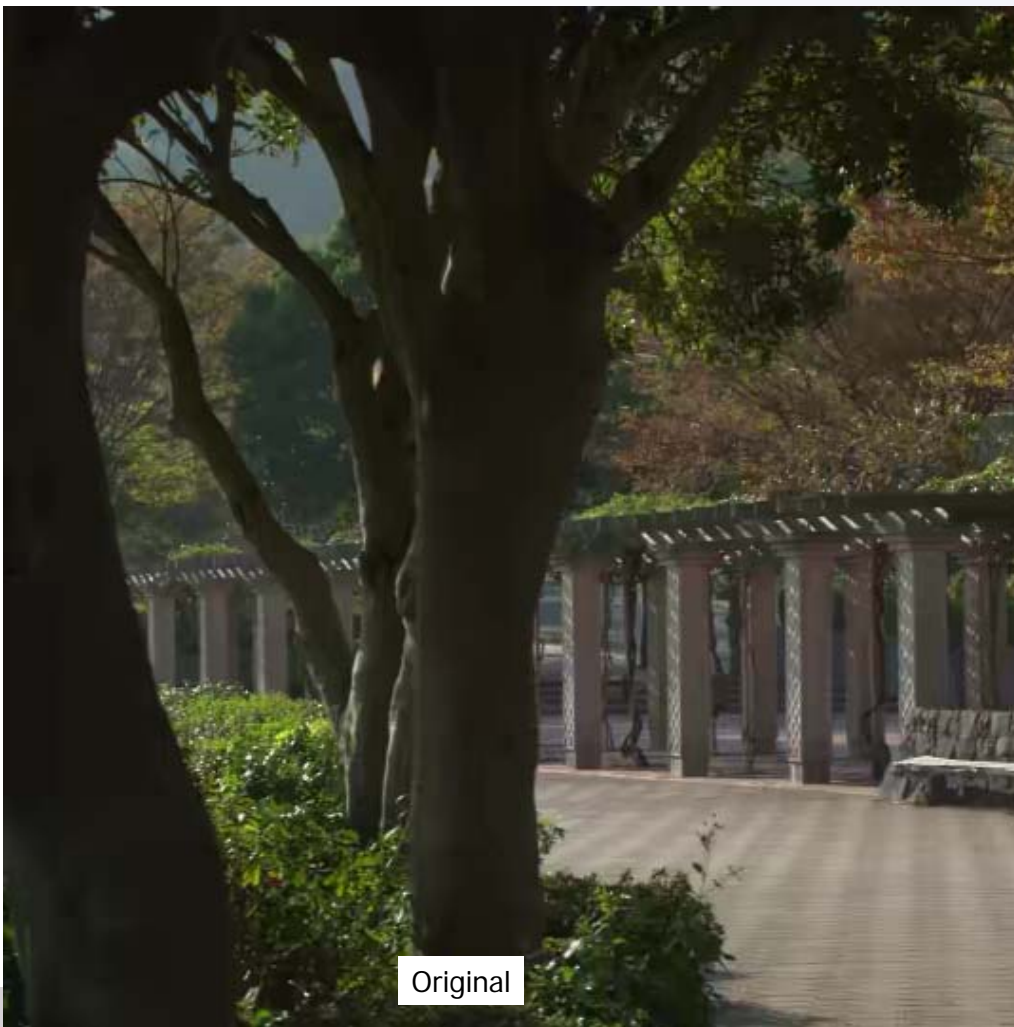


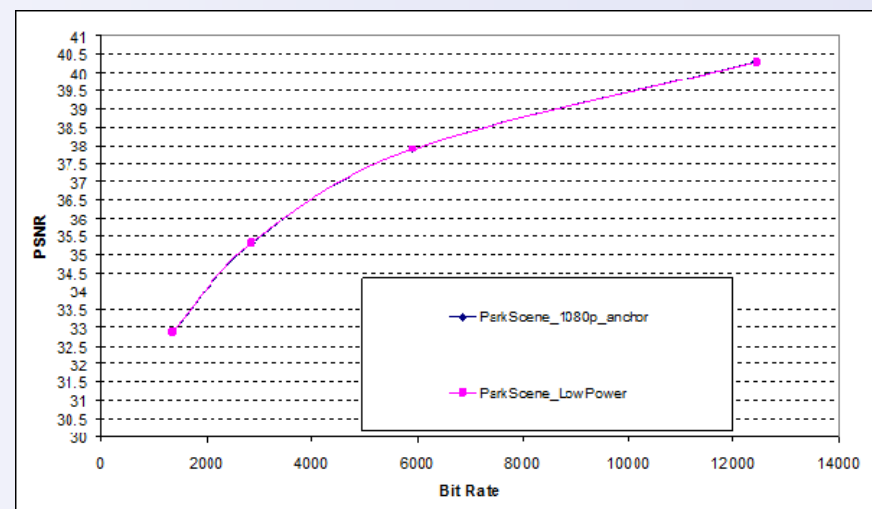
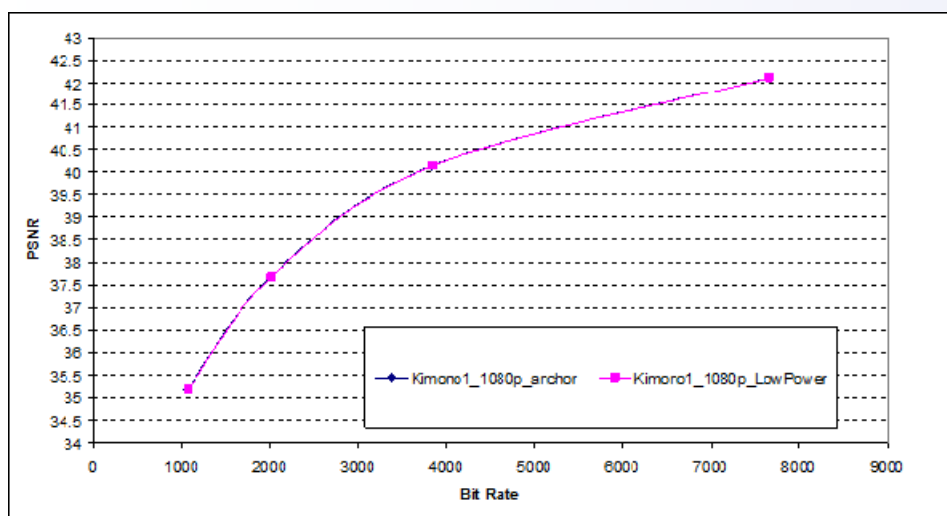


Original



Low Resolution Capable





- Power reduction

Configuration	Memory Bandwidth Savings	Pixel Operation Reduction
Full-resolution Decode No low-resolution flag	33%	X
Full-resolution Decode Low-resolution flag	66%	X
Low-resolution Decode Low-resolution flag	66%	50%

- Conclusions
 - Presented a system for graceful power degradation
 - System consists of:
 - Frame buffer compression that organizes data into a low-resolution and high-resolution component
 - Buffer compression algorithm for the low-/high-resolution structure
 - Flag in bit-stream to signal low resolution capability
 - Modified de-blocking operator to operate on low-resolution component independently
 - Propose further evaluation of the technique