SOURCE : DSRC AVC - 577

Data Partititoning Additions to Working Draft

D.3.1.3.1 Use of Data Partitioning

A two layer data partitioning scheme and the use of two channels with different error performance are used here to illustrate the error concealment capability of layering.

At the coder the value of the PBP pointer may be different for each slice such that the distribution of bits between the two channels may be controlled or maintained constant. The distribution may be different for I, P, and B frames.

Data partition 0 contains the address and control information and the low frequency DCT coefficients, while data partition 1 contains the high frequency DCT coefficients.

Two independent channels are required. Channel errors are distributed so that data partition 1 receives most errors.

It is assumed that at the decoder channel errors can be detected so that actions can be taken to prevent errored data from being displayed. For data partition 1 errored data is simply not displayed. For data partition 0 a suitably high quality channel, or decoder concealment actions are required.

D.4.1 Bitrate allocation in Data Partitioning

Data partitioning allows splitting a bitstream for increased error resilience when two channels with different error performance are available. It is often required to constrain the bitrate of each partition. This can be achieved at the encoder by adaptively changing priority breakpoint at each slice.

The encoder can use two virtual buffers for HP and LP channels, and implement feedback rate control by picking a priority breakpoint that approximately meets the target rate for each channel. Difference between target and actual rates is used to revise the target for the next frame in a feedback loop.

It is desirable to vary the bitrate split from frame to frame for higher error resilience. Typically, I frames benefit from a higher HP/LP ratio, while B frames could be placed entirely in the LP partition.

1.4.2.4 Data partitioning extension

Data Partitoning is a tool intended for use when multiple channels are available for transmission and or storageof a video bitstream, as may be the case in ATM networks, terrestrial broadcast, magnetic media etc. The bitstream is multiplexed onto these channels such that more critical parts of the bitstream (such as headers, motion vectors, DC coefficients) are transmitted in the channel with the better error performance, and less critical data (such as higher DCT coefficients) are transmitted in the channel with poor error performance. Thus, degradation to channel errors are minimized since the critical parts of a bitstream are better protected.

6.?.? Sequence_data_partitioning extension

This extension will be present between sequence and picture headers following the sequence_scalable_extension() when the scalable_mode field in this extension indicates the presence of data partitioning.

partition_identifier — This is a 4 bit integer that identifies the partition number of the current bitstream. Data partitions are multiplexed at the systems layer, and partitition 0 is the base, or high priority layer, while partition 1 is the low priority layer. All other values are reserved.

6.?.? Slice layer in data partitioned bitstreams

priority_breakpoint — This is a 7 bit integer that indicates the point in the syntax where the bitstream is supposed to be partitioned. The allowed values and their semantic interpretation is given in Table ?-?. Note that when MSB of priority_breakpoint is set, the remaining 6 bits indicate the number of (run,level) DCT codewords present in the base partition.

sequence_data_partitioning_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
partition_identifier	4	uimsbf
next_start_code()		
}		

slice() {	No. of bits	Mnemonic
slice_start_code	32	bslbf
if (data partitioning extension was present)		
priority_breakpoint	7	uimsbf
if ((vertical_size_value > 2800) && (!ISO/IEC 11172-2))		
slice_vertical_position_extension	3	uimsbf
quantizer_scale_code	5	uimsbf
while (nextbits() == '1') {		
extra_bit_slice	1	"1"
extra_information_slice	8	
}		
extra_bit_slice	1	"0"
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

Table ?-? Priority breakpoint values and associated semantics

priority_breakpoint	Syntax elements included in the high priority stream
0	All data at the sequence, GOP, picture and slice layers down to **extra_bit_slice**.
1	All data included above, plus macroblock syntax elements up to and including **macroblock_address_increment**.
2	All data included above, plus macroblock syntax elements up to but not including coded_block_pattern()
3 .. 63	Reserved.
64	All syntax elements up to and including coded_block_pattern() or DC coefficient (**dct_dc_differential**).
65	All syntax elements above, plus 1 (run,level) DCT coefficient pair.
...	
64+*j*	All syntax elements above, plus *j* (run,level) DCT coefficient pairs.

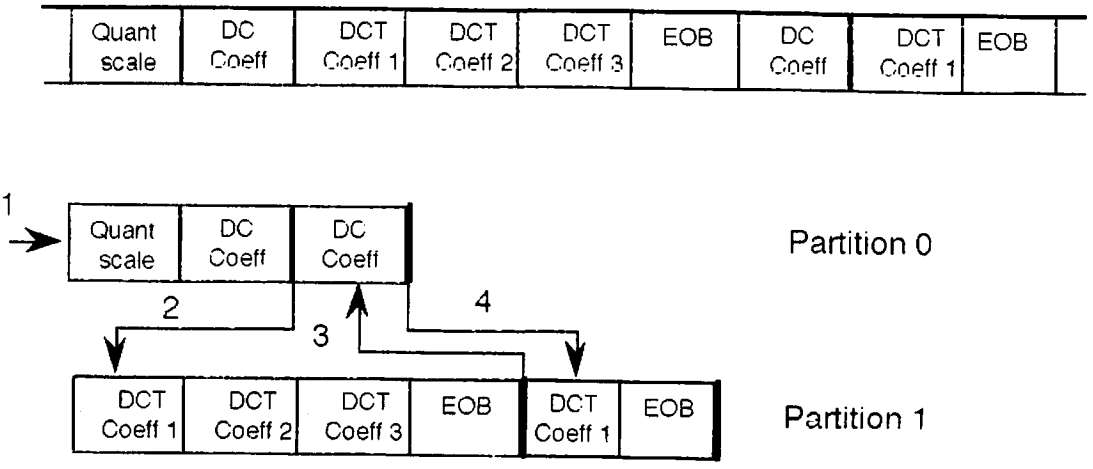

Figure ?-? A segment from a bitstream with two partitions, with priority_breakpoint set to 64 (up to DC coefficient). The two partitions are shown, with arrows indicating how the decoder needs to switch between partitions.

7.? Data Partitioning

Data partitioning is a technique that splits a video bitstream into two layers. A priority breakpoint indicates which syntax elements are placed in the base partition (which is also called HP, or high priority partition). The remainder of the bitstream is placed in the second partition (which is also called LP, or low priority partition). Sequence, GOP, and picture headers and extensions are redundantly copied in the LP partition to facilitate error recovery. Semantic interpretation of priority_breakpoint is given in Table ?-?. The decoding process is modified in the following manner:

Set *current_partition* to 0, and start decoding from bitstream with the identifier *current_partition*.

If *current_partition*=0, check to see if the current point in the bitstream is a priority breakpoint.

If yes, set *current_partition* to 1. Next item will be decoded from LP partition.

Otherwise, continue decoding. Remove sequence, GOP, and picture headers from both partitions.

If *current_partition*=1, check the priority breakpoint to see if the next item to be decoded is expected in the base partition.

If yes, set *current_partition* to 0. Next item will be decoded from HP partition.

Otherwise, continue decoding.

An example is shown in Figure ?-?, where the priority breakpoint is set at 64 (up to DC coefficient).