computers

MDPI

Article

Data Partitioning Technique for Improved Video Prioritization †

Ismail Amin Ali [1], Sandro Moiron [2], Martin Fleury [2],* and Mohammed Ghanbari [2]

[1] Department of Electrical and Computer Engineering, University of Duhok, Duhok P.O. Box 78, Iraq; ismail@uod.ac
[2] School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK; sandromoiron@gmail.com (S.M.); ghan@essex.ac.uk (M.G.)
* Correspondence: fleum@essex.ac.uk; Tel.: +44-1206-872-678

† This paper is an extended version of our paper published in 8th Conference on Computer Science and Electronic Engineering, Colchester, UK, 2016.

Received: 4 April 2017; Accepted: 4 July 2017; Published: 6 July 2017

Abstract: A compressed video bitstream can be partitioned according to the coding priority of the data, allowing prioritized wireless communication or selective dropping in a congested channel. Known as data partitioning in the H.264/Advanced Video Coding (AVC) codec, this paper introduces a further sub-partition of one of the H.264/AVC codec's three data-partitions. Results show a 5 dB improvement in Peak Signal-to-Noise Ratio (PSNR) through this innovation. In particular, the data partition containing intra-coded residuals is sub-divided into data from: those macroblocks (MBs) naturally intra-coded, and those MBs forcibly inserted for non-periodic intra-refresh. Interactive user-to-user video streaming can benefit, as then HTTP adaptive streaming is inappropriate and the High Efficiency Video Coding (HEVC) codec is too energy demanding.

Keywords: data partitioning; error resiliency; video coding; wireless channels

1. Introduction

Data partitioning is a method of error resilience present in standard codecs up to the H.264/Advanced Video Coding (codec) [1], though its emphasis has changed over time. Immediately before H.264/Advanced Video Coding (AVC) (also known as Moving Picture Experts Group (MPEG)-4 part 10), MPEG-4 part 2 [2] partitioned data within a single packet. The data were separated out into less-important transform coefficient residuals, and more important headers along with shape and motion data. When the entropy decoder receives a data-partitioned packet, it normally begins to decode the more important data at the beginning of the packet before coming to a mid-packet re-synchronization marker. Reversible Variable-Length Coding (RVLC) [3] supports recovery from that marker. As evidenced by RVLC, the creators of MPEG-4 part 2 aimed at bit-level protection within a packet. However, for H.264/AVC, the subject of this paper, data-partitioning is at the packet level. Particularly for mobile video streaming [4] over error-prone channels, data-partitioning at the packet level allows higher priority packets to be unequally protected. Additionally, in congested channels, including congested wireless channels, lower priority packets can be selectively dropped. As a result, in H.264/AVC, RVLC is no longer present, and bit-level protection is delegated in channel coding at the physical layer of the protocol stack. When data partitioning is configured, H.264/AVC creates up to three packets of increasing importance [5] within any one video slice. However, though standard H.264/AVC now has three data partitions, at standardization time, there were two other proposals. Those proposals respectively focussed on the highest priority partition, Data-Partition-A (DP-A) [6], and the lowest priority partition, Data-Partition-C (DP-C) [7]. These proposals were judged unsuitable for standardization [8] either because they were only suitable in special situations or because they introduced additional overheads (refer to Section 2.3).

However, in this paper, in general, it is shown that additional partitioning can be implemented while keeping to the standard. Specifically, H.264/AVC explicit mode Flexible Macroblock Ordering (FMO) [9] offers a smooth means of implementation. Section 4 further discusses the implications of an FMO implementation. This paper proposes to sub-partition Data-Partition-B (DP-B), the second of H.264/AVC's partitions in order of coding data priority. Data within DP-B (mainly intra-coded transform coefficient residuals) are separated into two sub-partitions according to their coding origin. These two sub-partitions of DP-B allow more appropriate prioritization of compressed video data, which will benefit unequal error protection or during access control to a congested network. This sub-partitioning technique is appropriate when gradual decoder intra refresh is employed, rather than periodic intra refresh. Periodic intra-refresh, the insertion of exclusively spatially-coded I (intra)- or Instantaneous Decoding Refresh (IDR) frames [1], can result in sudden increases in the data-rate and/or latency [10]. The cause is the relative inefficiency of intra-coding compared to inter coding. The restricted bandwidth of wireless channels makes it awkward to adjust to such data-rate fluctuations. Interactive video applications, such as mobile video phone or video conferencing, are also jitter intolerant. Therefore, the principal contribution of this paper is the sub-division of DP-B into two sub-partitions, originating from intra-coded macroblocks (MBs) naturally inserted by an H.264/AVC encoder itself or from intra-coded MBs forcibly inserted in support of non-periodic intra refresh (or some other cause of intra-coded MB insertion). For whatever reason intra-coded MBs are inserted, they are inserted into predominantly inter-coded frames, which is possible under H.264/AVC whether those frames are predictively coded (P-frames) or possibly bi-predictively coded (B-frames).

Thus, forcible insertion of intra-coded MBs is a means of providing non-periodic intra-refresh. Just as in the more common periodic intra-refresh, the insertion of intra-coded MBs helps reduce error propagation across largely temporally predicted pictures. Other forms of non-periodic intra-refresh are considered within Section 5.2. Additionally, an encoder may well also "naturally" choose some MBs to be intra-coded. Naturally intra-coded MBs are commonly inserted when the encoder cannot locate a suitable reference MB during motion estimation. For instance, this could occur when a new object appears during camera motion or zoom or alternatively after a scene change. To reiterate, this paper proposes dividing DP-B, which mainly contains intra-coded transform coefficient residuals, into two sub-partitions to contain separately the residuals of each type of intra-coded MBs, namely naturally and forcibly inserted. Consider if data loss only takes place from the sub-partition that holds the residuals of forced intra-coded MBs. This might happen if that sub-partition was afforded less application-layer channel coding, i.e., Unequal Error Protection (UEP) (refer to Section 2.2) or if that sub-partition was selectively dropped from a congested buffer during prioritised access queueing [11]. Our results indicate that there is a potential gain in objective video quality of up to 5 dB compared to dropping DP-B bearing network packets without any sub-partitioning.

Our demonstration of sub-partitioning of DP-B uses a cyclic intra-refresh line (IRL) of MBs. Though an IRL is conveniently implemented for the purposes of evaluation, there are variants of non-periodic intra refresh that position their intra-coded MBs in non-contiguous locations (unlike the IRL technique, for which, of course, the MBs form a single row within a picture). For instance, in [12], three techniques were evaluated for their effect on four Constant Bit Rate (CBR) Common Intermediate Format (CIF) reference video clips, i.e., the same configuration employed in our evaluations. The first technique [13] randomly places intra-coded MBs within each successive video picture. The second technique considered utilized motion following [14], when selecting which MBs to intra code. Finally, the third technique is called frame filling [15]. In the comparative evaluation [12], technique three, frame filling, was better in regard to the rate that errors decayed over time and in regard to the sequence's average Peak Signal-to-Noise Ratio (PSNR) as the MB loss rate increased. Frame filling randomly places intra-coded MBs as in [13]. However, in addition all of the MBs of a frame are intra-coded at a specified refresh rate. This is accomplished by means of "intra" counters for every MB. The net effect is that a complete frame is cyclically refreshed just as when using a cyclic IRL. Just as for cyclic IRL, in frame filling intra-coded MBs are principally inserted to improve error resilience by reducing temporal error propagation. The main implementation requirement for our proposed sub-partitioning of DP-B is that the location of forced intra-coded MBs is available. For frame filling, forced intra-coded MBs are readily found, as the intra counters for each MB pass a threshold whenever their coding mode changes from inter to intra. Consequently, the proposed sub-partitioning is easily used with the frame-filling intra-refresh technique. In fact, the frame filling technique is built into H.264/AVC's Joint Model (JM) reference codec software. Within the the JM codec, all forced intra-coding is controlled by software and, thus, for this codec, it is possible anyway to record the position of intra-coded MBs when they are first inserted.

Non-periodic intra-refresh is not the only error resilience technique that can benefit from our proposal. The normal method of rate-distortion (R-D) optimization in H.264/AVC codecs is to optimize the selection of the quantization parameter (QP). For that purpose, the Lagrangian multiplier method of optimization is employed, with distortion matched against the resulting PSNR and the rate being the bit-rate. Alternatively, the number of intra-coded MBs together with a measure of channel conditions can be input into the R-D optimization process. For example, in [16], inter-coded MBs were changed to intra-coded MBs through a probabilistic estimate of their reliability according to channel statistics. Examples of forced insertion of intra-coded MBs can also be found in [17,18].

The High Efficiency Video Coding (HEVC) codec limits its support for error resiliency [19], in part to reduce codec complexity, in part because its orientation is away from the network support of H.264/AVC, and in part because MBs have been replaced in HEVC by quad-tree structures. Until error resiliency is fully supported in HEVC, there will be a continuing requirement for H.264/AVC in interactive and real-time video communication, especially over error prone channels. HEVC computational complexity remains an issue for the consumer electronics industry. Thus, Ambarella's H2 System-on-Chip (SoC) (Santa Clara, CA, USA) with HEVC hardware, which might meet that concern, is apparently not suited to cameras such as the prominent GoPro series because of its energy consumption. For example, the recent GoPro 4 camera (GoPro Inc., San Mateo, CA, USA) opted for the H.264/AVC codec. Thus, data-partitioning techniques such as the proposed sub-partitions of DP-B remain of interest to the video network community.

This paper extends the six-page conference version of the paper presented as [20] by about 50%. In doing so, it includes new material about how the proposal fits into the structure of H.264/AVC and how FMO was used by us to implement the proposal without breaching the standard. Additional results for data partitioning are also now provided. The application of the technique is to the video network for which commercial activity is likely to be intense because, according to the Cisco Visual Networking Index [21], video traffic to mobile and portable devices will now have overtaken wired communication. The remainder of this paper is organized as follows. Section 2 supplies the source coding background necessary for an understanding of the proposal. Our implementation method is described in Section 3, while Section 4 is an analysis of the potential performance of the proposal. Section 5 is a performance evaluation of the proposal. Finally, Section 6 offers some conclusions.

2. Source Coding Context

This section provides the H.264/AVC source coding background that is required to properly comprehend the proposal to sub-partition DP-B and the performance results. To recap, refined DP-B refers to splitting DP-B into two sub-partitions, one of which contains intra-coded MBs "naturally inserted" by the encoder and the other of which contains intra-coded MBs as part of a non-periodic intra-refresh process. The latter are normally called "forced" intra-coded MBs.

2.1. H.264/AVC Coding

The H.264/AVC standard codec conceptually separates the Video Coding Layer (VCL) from the Network Abstraction Layer (NAL). The VCL specifies the core compression features, while the NAL supports delivery over various types of network. In a communication channel, the quality of service is affected by the two parameters of bandwidth and the probability of error. Therefore, as well as video compression efficiency, which is provided for through the VCL layer, adaptation to communication channels should be carefully considered. The NAL facilitates the delivery of the H.264/AVC VCL data to the underlying transport layers such as RTP/IP, H.32X and MPEG-2 transport stream. The concept of the NAL, together with the error resilience features in H.264/AVC [22], allows communication over a variety of different channels. Each NAL unit (NALU) could be considered as a virtual packet [23] that contains a header and a payload. The 8-bit header, Figure 1, specifies the NALU payload type (nal-unit-type) and the relative importance of the NALU (nal-ref-idc) while the payload contains the related data. The payload padding bits are added to make the payload a multiple of bytes. Table 1 is a summarized list of different NALU types. NALUs types 6 to 12 are non-VCL units containing additional information such as parameter sets and supplemental information. NALUs types 1 to 5 contain different VCL data, as described after the introduction of the slice concept in the next paragraph.

In the H.264/AVC codec, each frame can be divided into one or more slices, each of which contains a flexible number of MBs. Variable Length Coding (VLC) that is entropy coding of the compressed data normally takes place as the final stage of a hybrid codec. In H.264/AVC, in the main profile, the more efficient Context Adaptive Binary Arithmetic Coding (CABAC) [24] can replace VLC. In each slice, the arithmetic coder is aligned and its predictions are reset. Hence, every slice in the frame is independently decodable. Therefore, they can be considered as re-synchronization points that prevent error propagation to the entire picture. Each slice is placed within a separate NALU. The slices of an Instantaneous Decoder Refresh-(IDR-) or I-picture (i.e., a picture with all intra slices) are located in type 5 NALUs, while those belonging to a non-IDR or I-picture (P-or B-pictures) are placed in NALUs of type 1, and in types 2 to 4 when data partitioning mode is active. (An IDR picture is confusedly equivalent to an I-picture in previous standards and an I-picture in H.264/AVC allows predictive references beyond the boundary of a Group of Pictures (GOP)).

In type 1 and type 5 NALUs, MB addresses, motion vectors (MVs) and the transform coefficients of the blocks, are packed into the packet, in the order they are generated by the encoder. In Type 5, all parts of the compressed bit-stream are equally important, while in type 1, the MB addresses and MVs are much more important than the transform coefficient residuals. In the event of errors in this type of packet, the fact that symbols appearing earlier in the bit-stream suffer less from errors than those which come later means that bringing the more important parts of the video data (such as headers and MVs) ahead of the less important data or separating the more important data altogether for better protection against errors can significantly reduce channel errors. In prior standard video codecs, that arrangement was known as data partitioning.

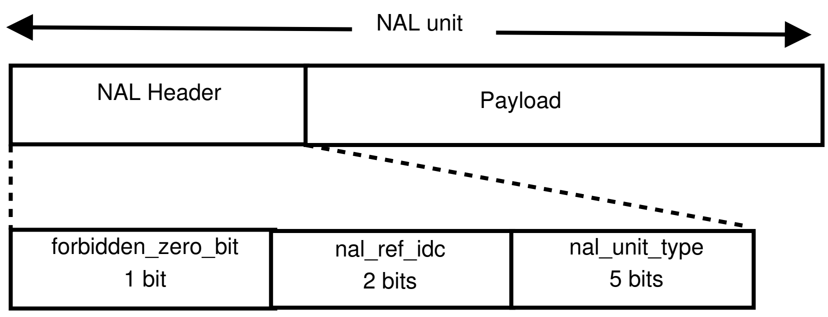


Figure 1. Network Abstraction Layer unit (NALU) format.

Table 1. Network Abstraction Layer unit (NALU) types.

NALU Type	Class	Content of NALU
0	-	Unspecified
1	VCL	Coded slice
2	VCL	Coded slice partition A
3	VCL	Coded slice partition B
4	VCL	Coded slice partition C
5	VCL	Coded slice of an IDR picture
6–12	Non-VCL	Supplementary information, Parameter sets, etc.
13–23	-	Reserved
24–31	-	Unspecified

VCL = Video Coding Layer.

2.2. Data Partitioning

In H.264/AVC when data partitioning is enabled, every slice is divided into three separate partitions and each partition is located in either of type 2 to type 4 NALUs, as listed in Table 1. NALU of type 2, also known as Data-Partition-A (DP-A), comprises the most important information of the compressed video bit stream of P- and B-pictures, including the MB addresses, types (intra or inter), QPs, MVs and essential headers. If any MBs in these pictures are intra-coded (as signalled by their MB type), their transform coefficients are packed into the type 3 NALU, also known as Data-Partition-B (DP-B). Type 4 NALU, also known as Data-Partition-C (DP-C), carries the transform coefficients of the motion-compensated, inter-picture coded MBs. DP-B and DP-C also contain Coded Block Patterns (CBPs). A CBP is a bit-map indicating which sub-blocks within each MB have non-zero coefficients. The overall arrangement is illustrated in Figure 2. There is a small one-byte per NALU header overhead for the type 1 NALU in Figure 2. When the MB data are split between the data-partitioned NALUs of Figure 2, there is a net gain of two bytes due to the additional two NALU headers. An additional six synchronization bits are also required for each NALU, leading to a net gain of 12 bits. This overhead (the two NALU header bytes and the 12 synchronization bits) increases for each extra slice within a frame (that is about 3 B extra per slice). The procedure of forming data partitions within video slices is shown in Figure 3, with a further illustration in Figure 4 for one of the video sequences used in the tests of Section 5.

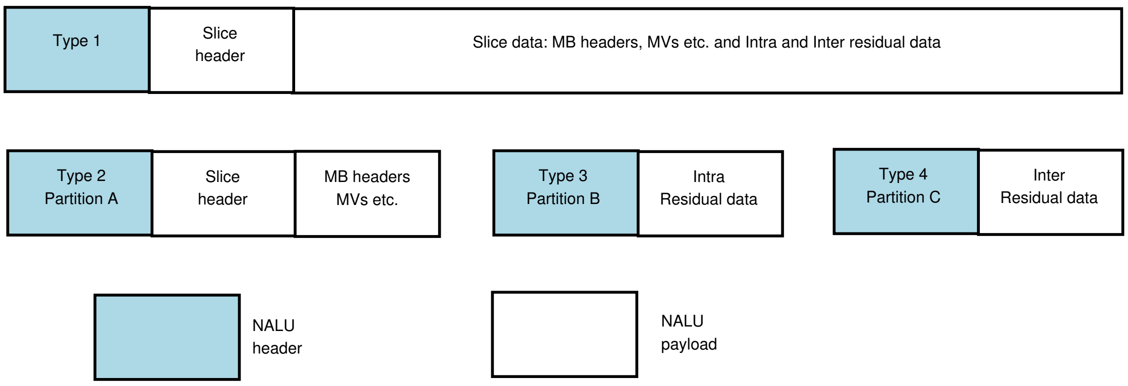


Figure 2. H.264/Advanced Video Coding (AVC) Network Abstraction Layer unit (NALU) structure, without (**top**) and with (**bottom**) data partitioning.

In order to decode DP-B and DP-C, the decoder must know the location from which each MB was predicted, which implies that DP-B and DP-C cannot be reconstructed if DP-A is lost. Though DP-A is independent of DP-B and DP-C, Constrained Intra Prediction (CIP) should be set [25] to make

DP-B independent of DP-C. By setting this option, intra-coded MBs can no longer be predicted from neighboring inter-coded MBs, the prediction residuals of which reside in DP-C. For coding reasons detailed in [25], even with CIP enabled, DP-C cannot be made independent of DP-B within the profile structure of H.264/AVC.

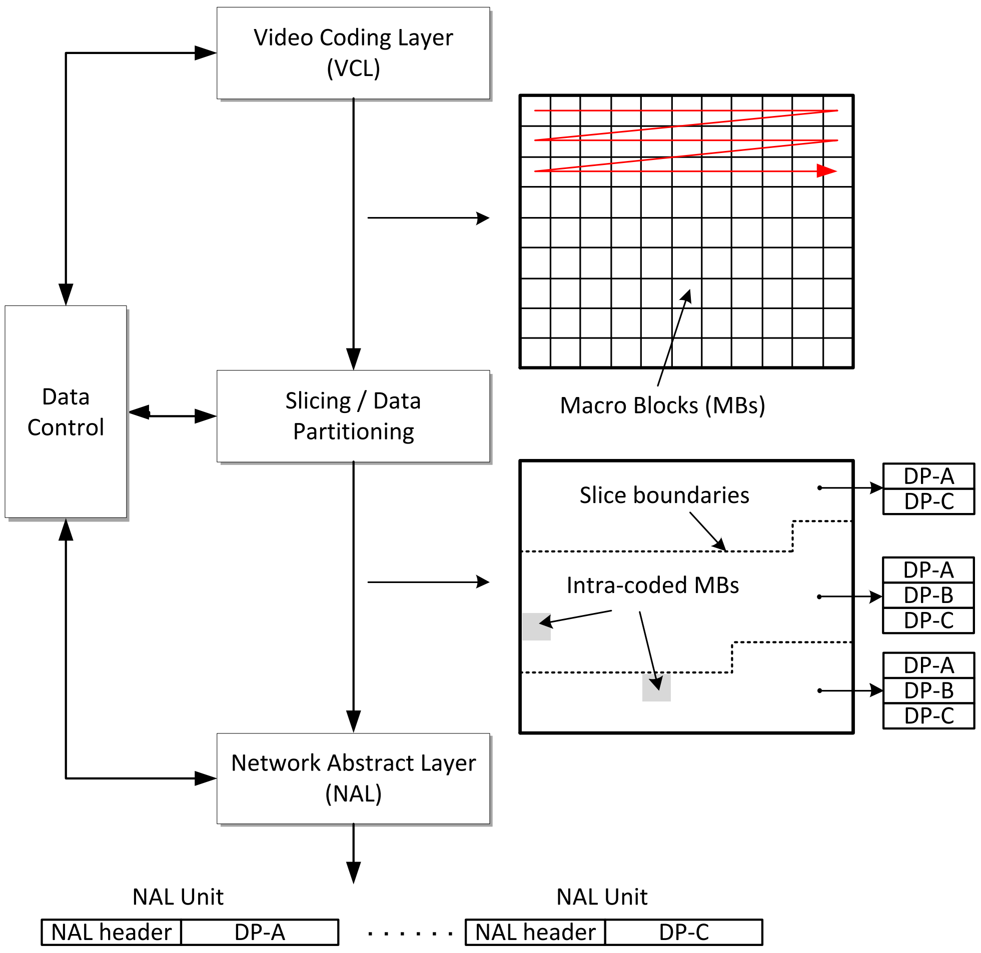


Figure 3. H.264/AVC data flow for combined slicing with data-partitioning.

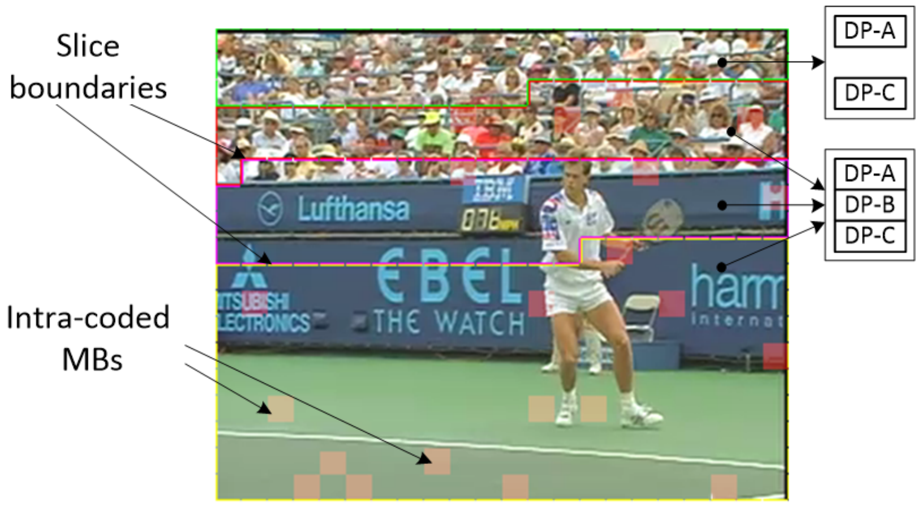


Figure 4. Example video frame from Stefan split into equal bit length slices, showing a slice partitioned into three partitions (A, B, C). Notice that it is possible for partition B (or C) to be missing.

Because of the increasing importance of the data in DP-C, DP-B, and DP-A, respectively, data-partitioning lends itself to Unequal Error Protection (UEP). In [26], MPEG-4 part 2, internal partitions were split between General Packet-Radio Service (GPRS) channels, with different FEC rates for each channel. With the standardization of H.264/AVC, data partitions [23] were no longer built-in, facilitating UEP. In [27], hierarchical Quadrature Amplitude Coding (QAM) allowed physical-layer prioritized transmission of data partitions. Another physical-layer UEP technique was demonstrated in [28]. Data partitions were mapped onto different antennas [28] in a form of space-time block coding for Multiple Input Multiple Output (MIMO) wireless transmission. Returning to software UEP, in [4], differential protection was achieved by selecting from a set of discrete channel coding rates, through punctured convolutional codes. In establishing the degree of protection for each partition type, joint optimization of the QP and the channel coding rate took place.

2.3. Concealing Lost Packets

A detailed guide to reconstruction of data-partitioned video by an H.264/AVC decoder is given in [29]. Lost partition-C slice packets are compensated for by error concealment at the decoder using the motion vectors in partition-A at the decoder to identify candidate replacement MBs in the last previously correctly-received frame.

In general, in the H.264/AVC codec standard, error concealment is a non-normative feature, i.e., a feature which is not needed for compliance with the standard. Nevertheless, in [30], a number of non-normative error concealment algorithms for H.264/AVC were recommended, as, though error concealment is outside the scope of the standard, it is nevertheless needed and its implementation can serve to distinguish one decoder from another. An attempt is made to conceal any lost slices. Error concealment within a lost slice is on an MB basis. Previously concealed MBs can be used to conceal missing MBs. Concealment proceeds from the edges of a lost slice inwards. For intra-coded concealment of a missing MB, spatially adjacent pixels to a missing MB, if available, are interpolated to form the pixels of a missing MB. For inter-coded MBs within a lost slice, if very little motion has occurred, replacement by the matching MB in the previous frame occurs (known as error concealment by previous frame replacement). Otherwise, it is recommended [30] to use one of the motion vectors of the surrounding MBs to identify a replacement MB. An algorithm to choose that motion vector is detailed in [30]. In the case of an MB split into sub-blocks, an average of the motion vectors of the sub-blocks within the MB is taken to form a candidate motion vector. The H.264/AVC algorithms will work even if only one correctly received slice is available within a frame.

If DP-B is lost, missing MBs can be concealed by employing motion vectors from DP-A and intra error concealment is optionally employed. In this sense, optional has a similar meaning to non-normative, and, in fact, in the JM implementation of H.264/AVC used herein, intra error concealment is included in the decoder implementation. If both DP-B and DP-C go missing, then they are replaced by the MBs pointed to by the motion vectors in DP-A. If DP-A is lost, it is recommended to use the motion vectors of adjacent MB rows, i.e., MBs from adjacent slices if these are available. Table 2 summarizes these possibilities.

As mentioned in Section 1, other ways of partitioning H.264/AVC coding data were also considered at the time of standardization such as splitting low and high frequency transform coefficients normally present in DP-C and placing the low frequency ones in DP-A [7] or duplicating the slice header and MB type information present in DP-A and placing it in DP-B [6]. The former [7] was recommended when zig-zag scanning of the transform coefficients is replaced by double-scanning. However, the proposal was not adopted by the standardization committee [8] in order to avoid over complicating the standard. The proposal in [7] introduces additional error resilience, as in double scanning losing low frequency coefficients can significantly affect video quality. However, double scanning only occurs in H.264/AVC when coding the luma coefficients in intra-mode 4×4 blocks when QP < 24. The proposal in [6] is also not adopted as part of the standard. If it had been accepted, the default case would have been to introduce extra overhead (the duplication of coded information), whereas, in many circumstances, the overhead would be unnecessary.

However, our proposal need not involve breaking the H.264/AVC standard as a mechanism exists to split DP-B into two sub-partitions, one of which contains naturally inserted intra-coded MBs and the other of which contains forced intra-coded MBs. As with all standard codecs, it is the format of the bitstream received by the decoder that is standardized. This allows an encoder to be optimized for a particular purpose. For example, it is possible that an encoder might insert intra-coded MBs if the CBR is not reached. Inserting these MBs may increase the bitrate to the desired level while increasing the overall quality. A method of implementing our proposal without breaching the H.264/AVC standard is now returned to in the following Section.

Table 2. Concealment options for missing data-partitions.

Available Partitions	Error Concealment Method
A and B	Conceal missing MBs with MVs from partition-A and texture from partition-B, optionally where appropriate perform intra-concealment
A and C	Conceal missing MBs with MVs from partition-A and texture from partition-C, optionally where appropriate perform inter-concealment
A	Conceal missing MBs using MVs from partition-A
B and/or C	Drop the partitions and either employ whole frame concealment or use the MVs from the MB row spatially above the missing MBs.

MB = Macroblock, MV = Motion Vector.

3. Implementation with FMO

One way (our way) to quickly implement refined data partitioning in a standardized manner is through H.264/AVC's Flexible Macroblock Ordering (FMO). As mentioned in the previous section, H.264/AVC can divide a picture into slices, whose size can be as small as an MB and as large as one complete picture. MBs are assigned to slices in raster scan order, unless FMO is enabled [23]. FMO [9] allows different arrangements of MBs in a slice by utilizing the concept of independent slice groups. The MBs are arranged in a slice in different order compared to the scan order, enhancing error resilience. In each slice group, i.e., within a set of slices, the MBs are arranged according to an MB to slice group map.

As first mentioned in Section 1, whenever no suitable closely matching MB can be found within reference frames during motion estimation, an encoder may decide to naturally insert intra-coded MBs amongst a P- (or B-) frame's inter-coded MBs. Thus, two varieties of intra-coded MBs can exist within the same P- (or B-) frame. To divide the compressed data of forced intra-coded MBs from that of naturally intra-coded MBs, we used FMO's explicit mode to emulate the changes to the codec software that otherwise would be needed. Altering software may have unexpected side-products, whereas, using existing FMO software for that purpose eases the implementation and does not lead to unexpected effects or even software errors. Forced intra-coded MBs are placed in one slice group and the NALUs of this slice group are indicated by altering their nal-ref-idc (NRI) bits. In this way, it is possible to identify DP-B packets containing forced intra-coded MBs (DP-B_F) and separate these from DP-B packets bearing naturally intra-coded MBs (DP-B_N). This arrangement is illustrated in Figure 5. Within the first FMO slice group, there is no DP-C. In the second FMO slice group, all three partition types may be present. However, only the encoded data from naturally intra-coded MBs will be present in the second FMO's DP-B.

As a number of commercial implementations of H.264/AVC do not implement FMO or allow the user to control the slice structure, it is possible [31] to introduce these features into the compressed output of an encoder. Moreover, adding FMO to the compressed bitstream after encoding results in exactly the same output at the decoder as when FMO takes place within the encoder. The same

FMO can even be removed in the compressed domain [32], after accounting for any slice losses but before decoding by a non-FMO compliant decoder. The two transcoding operations effectively allow retrofitting of FMO error resiliency to non-compliant codecs.

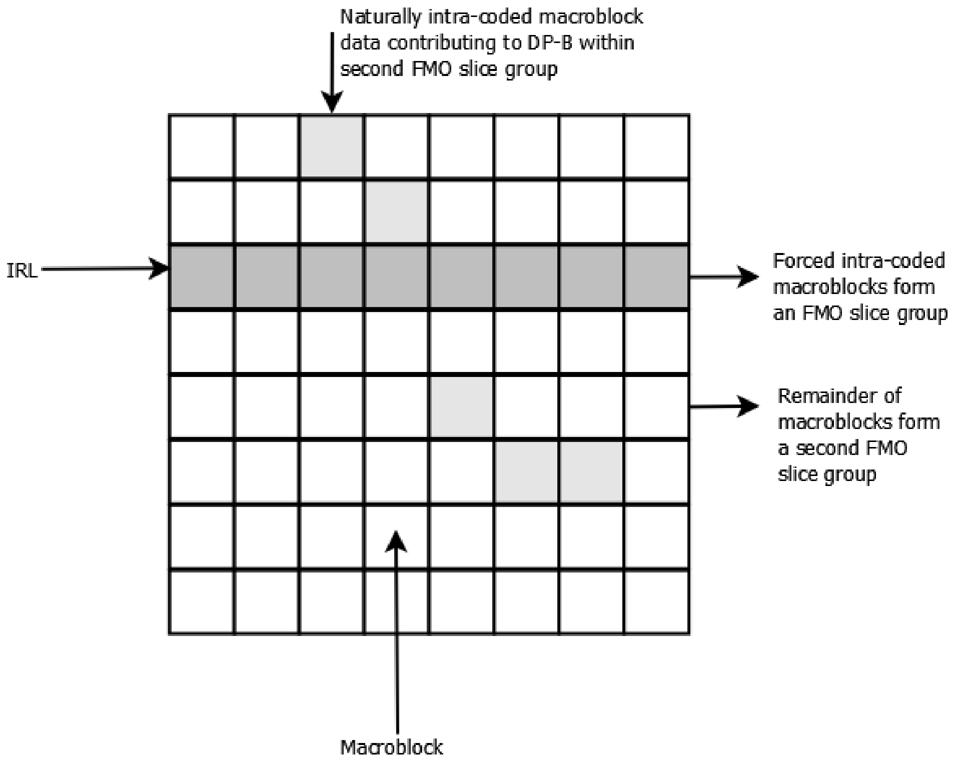

Naturally intra-coded macroblock data contributing to DP-B within second FMO slice group
IRL
Forced intra-coded macroblocks form an FMO slice group
Remainder of macroblocks form a second FMO slice group
Macroblock

Figure 5. Separating forced intra-coded macroblocks (MBs) from naturally intra-coded MBs by means of Flexible Macroblock Ordering (FMO) explicit mode slice groups.

In general, the explicit mode of FMO has been widely adopted for a variety of coding purposes, e.g., [33–35]. Research in [36] employed FMO's explicit mode for adaptive intra-coded MB grouping. The effect on distortion if an MB was lost was assessed so as to determine which MBs to apply forced intra coding to. Then, according to channel conditions, a given number of MBs were selected to be intra-coded. Xu et al. [36] highlighted the advantage of being able to dynamically group intra MBs within a slice: the more important intra-coded MBs could be preferentially protected by a higher rate of channel coding.

4. Performance Analysis

The potential improvement in relative video quality resulting from FMO explicit mode with our refined DP-B partitioning does bring some overheads. As previously mentioned, FMO requires [9] the presence of a slice-group map, a MacroBlock Allocation map (MBAmap), at the decoder, which, in the case of FMO explicit mode, is transmitted for each picture in which the MBAmap changes. An MBAmap forms (an optional) part of a Picture Parameter Set (PPS) NALU (type 8), indexed by a slice header. The PPS in turn indexes a Sequence Parameter Set (SPS) (NALU type 7), which contains decoding parameters that remain constant within a Group of Pictures (GoP). The MBAmap itself [37] is by default an ordered string of ASCII integers, the MB addresses, that map onto the MB locations in raster scan order within a picture. As pointed out in [37], the coding of the PPS is non-normative, implying that the MBAmap can be compressed by means of a binary representation (requiring 3-bits per MB for eight slice groups) and possibly a Zipf code. There are also [38] 20 bits within a PPS for FMO purposes, coding FMO type and other parameters. PPSs are anyway likely to be frequently transmitted for some applications [37], including broadcast TV, whether MBAs are included or not. Parameter sets should be transmitted [23] by a reliable session control protocol in advance of the arrival of the matching coding data at the decoder. As SPSs are obligatory, no protocol overhead results from their use with FMO.

In [9], the bitrate overhead from type 6 FMO explicit mode was compared to type 2 FMO, when either is used to code rectangular regions of interest. For this application, the only extra FMO explicit mode overhead was the constant overhead of PPSs containing an MBAmap. For the Stefan and Table CIF sequences tested at 30 frame/s, the fixed overhead was 529 B and 439 B, respectively, "a minuscule overhead when considered against the total number of bytes contained in a coded sequence" [9]. For both FMO modes, compared to no FMO, the percentage overhead varied according to the number of MBs in the slices (50, 100 or 400) and the QP. The GoP frame structure within an 18 picture GoP had a limited impact. The median (mean) overhead was found to be 0.9% (1.7%). The maximum overhead was 8.6% resulting from 400 MBs per slice, with an IBBP GoP structure at a high QP (40/40/42 for I (Intra), P (Predictive),B (Bi-predictive) pictures, respectively). There is also a decrease in intra coding efficiency whenever MBs are isolated from each other, which in error prone environments is counter-balanced by the increase in error resiliency. Clearly when no errors occur, there is an additional overhead from using FMO. Further evidence of the high variability in FMO bitrate overhead according to coding configuration is tabulated in [38] for choice of H.264/AVC entropy coder and Variable Bit Rate (VBR) or CBR. If in the extreme case, each slice contained a single MB, the bitrate overhead arising from the coding inefficiency was found in [39] to vary from 0.6% to 22%.

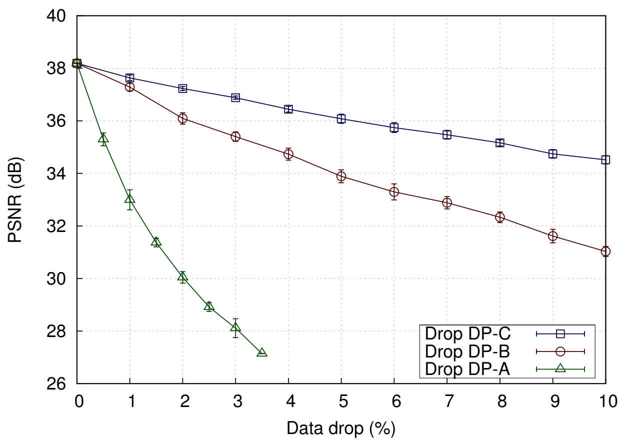
The increased computational complexity of FMO is less than 5% for CIF video at 30 frame/s according to [37], where it is characterized as "very small". It mainly consists of looking up MBA addresses in the MBAmap, which may incur a memory access overhead. Therefore, the increased computational complexity of the proposed method is also likely to be at most 5%.

5. Evaluation

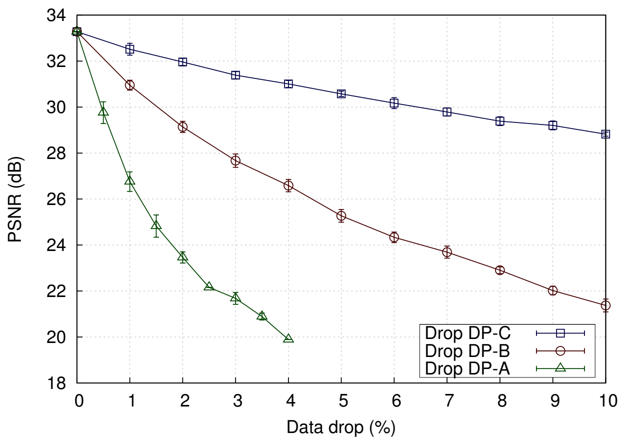
5.1. Effect of Packet Drops

To assess the effect of dropping of different data partitions on video quality, the Paris and Stefan sequences were coded with data partitioning enabled at a CBR target bitrate of 1 Mbps (CIF coded at 30 frame/s). Notice that this format was chosen for ease of testing, as the results scale up to higher spatial resolutions. Different percentages of data were from each partition were individually dropped on a random basis and the sequence was reconstructed from the remaining parts. The resulting objective video quality (PSNR) is shown in Figure 6. Notice also that the increased coding complexity of Stefan, due to its rapid motion activity, results in a reduction in video quality in Figure 6b. For the same percentage of video data loss rate, it is easy to observe a severe adverse effect on video quality when DP-A is lost. Therefore, this partition is normally given the highest protection level. Intra-coded MBs help to limit error propagation; therefore, the loss of DP-B will impair the recovery of successive frames. As compared to DP-A and DP-B, DP-C packets are of less importance. Normally, a large number of MBs are inter-coded, unless a very high bitrate is specified and Rate-Distortion Optimization is turned off, when an encoder may even insert extra intra-coded MBs to meet a bitrate target.

To better understand the other results with CBR streaming, VBR streaming according to a target video quality was also tested. As shown in Figure 7, DP-C is the biggest partition of a coded stream in the acceptable range of QP settings. (The value of QP governs the coarseness of quantization. In H.264/AVC, it ranges from 0 (no quantization) to 51 (very coarse quantization). In practice, very low and very high values of QP are not used.) It should be noted that data partitioning adds a small overhead from the extra NALU headers, as earlier analyzed in Section 2.2.



(**a**)



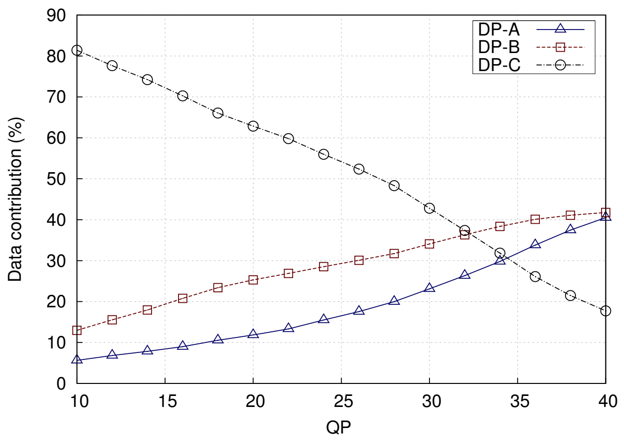
(**b**)

Figure 6. Video quality when dropping different data partitions. (Notice the different vertical axis in (**b**)). (**a**) for Paris test sequence; (**b**) for Stefan test sequence.

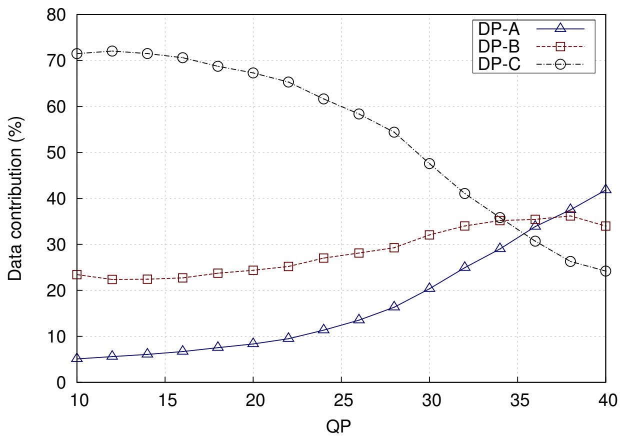
5.2. Intra-Refresh

As mentioned in Section 1, when mobile video streaming with Internet Protocol (IP)/User Datagram Protocol (UDP)/Real-time Transport Protocol (RTP) packetization, there is a need to avoid periodic increased delay due to less efficient intra-coded I-pictures [40] at the start of each Group of Pictures (GoP). This can be achieved through an IPPP picture structure (for Intra- (I-) and Predictive (P-) pictures) . This means that there is just one I-picture at the start of a stream, followed by a continuous stream of predictively-coded P-pictures. Notice that bi-predictive B-pictures are not available in H.264/AVC's Baseline and Constrained Baseline profiles.

Using a continuous sequence of predominantly inter-coded P-pictures runs the risk upon packet loss of spatio-temporal propagation of errors. To counteract this problem, an H.264/AVC codec permits the inclusion of intra-coded MBs within the P-slices making up a compressed video frame. These MBs can be forcibly inserted as a form of non-periodic intra-refresh. Notice that non-periodic intra-refresh may still allow random access to take place (if needed). For example, a cyclic IRL will refresh all content over $2n - 1$ frames [14], where n is the number of MB rows in a frame, thus providing a random access point every $2n - 1$ frames. The IRL can also consist of more than one row of MBs so that the random access points are doubled in frequency. There are various forms of non-periodic intra-refresh including: random placement of intra-coded MBs up to a given number within each picture [13]; as part of an evolving isolated region [41]; and as a line of intra-coded MBs that cycles in position over a sequence of pictures [14]. The issue of which of these to choose is an interesting debate, but, as insertion of a cyclic intra-coded line in general does result in a complete refresh despite corrupted data [14], this paper assumes that this simply implemented mechanism is used. Further investigation of the value of inserting an IRL can be found in the authors' earlier research paper [42]. As an illustration, the insertion of an IRL is shown in Figure 8 for the Paris test sequence.



(**a**)



(**b**)

Figure 7. Percentage contributions of data partitions A, B and C over a range of quantization parameters (QPs). The sequences are IPPP... coded (for Intra (I-) and Predictive (P-) pictures) with a cyclic Intra Refresh Line (IRL) (18 rows of macroblocks (MBs) per frame). (**a**) for Paris test sequence; (**b**) for Stefan test sequence.

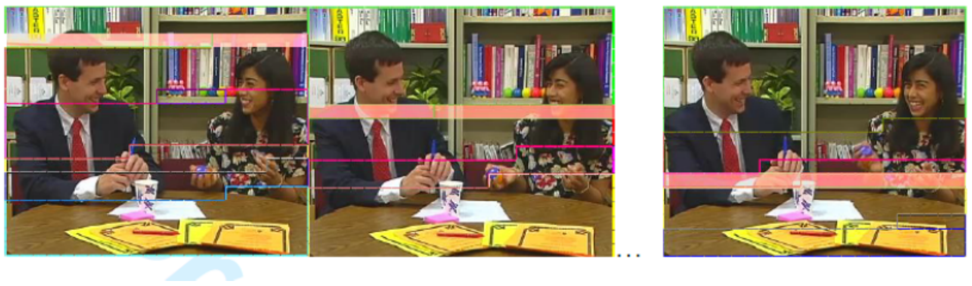


Figure 8. Example video frames from Paris, showing a cyclic Intra Refresh Line (IRL) and (optional) slices.

When extra MBs are forced to be intra-coded when using a cyclic IRL, it is logical that the size of DP-B will increase, as is necessary to compensate for the absence of periodic I-frames. Figure 9 shows the percentage of data occupied by each type of partition for well-known test videos in CIF, with a CBR target bitrate of 1 Mbps. The data from DP-C dominates, while in some sequences partition B is hardly used. When intra-coded MBs are included in partition-B, it is because they have been naturally encoded, that is, if the encoder cannot form a suitable prediction that reduces the difference signal sufficiently. This is an implementation dependent decision, as only the decoder is standardized in the standard codecs. Then, in Figure 10, an IRL has been included. The growth in size of DP-B data is the main but expected feature of this Figure.

The relative fraction of DP-B data between video sequences can change when going between Figures 9 and 10. This is because, when an IRL is added, the coding efficiency declines due to the use of intra coding rather than inter coding. A compiler may decide to drop some naturally encoded intra MBs to meet a target CBR. As more active Stephan is likely to have more such intra-coded MBs, its DP-B size is likely to reduce relatively more as a result compared to Paris. To meet the target CBR bitrate, an encoder can also reduce the bitrate by changing the QP. For example, in Figure 7, the relative size of DP-B between the Paris and Stephan sequences changes around QP = 30. For similar reasons, the percentage size of DP-A for the Paris and Stephan sequences increases as the QP is increased to compensate for a drop in coding efficiency due to the use of an IRL. With an IRL, the coding efficiency loss is amortized across the frame sequence, whereas, for periodic intra-refresh, the efficiency drop is more severe and occurs whenever a periodic intra-refresh frame occurs [10]. As observed in [10], periodic intra-refresh frames cause longer transmission delays due to their greater size resulting in jitter, which is harmful to real-time video applications.

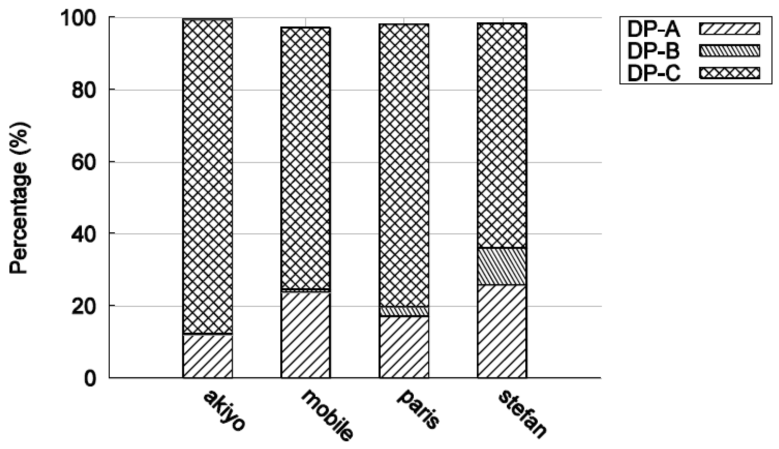


Figure 9. Percentage of data for the three data partitions when no IRL is present.

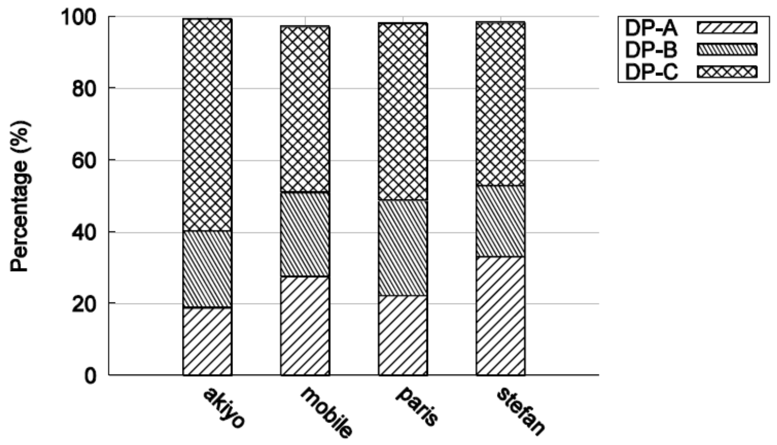
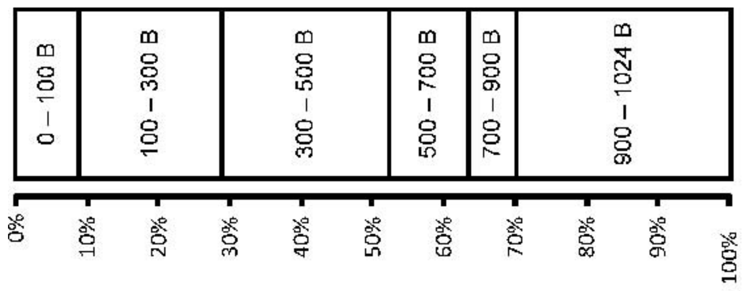


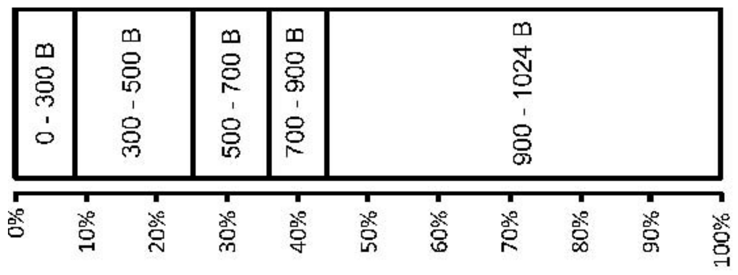
Figure 10. Percentage of data for the three data partitions when there is an IRL present.

5.3. Impact of Data Loss

To evaluate the effectiveness of the proposed classification scheme, up to 300 frames of the test sequences Akiyo, Bus, Foreman, Mobile, Paris, Soccer, Stefan and Tennis were coded at 30 frames/s and, of course, data-partitioning enabled with an IPPP. . . coding structure. CIF was selected for ease of repeated testing and comparison with the work of others. The sequences were coded with a cyclic IRL scanning of successive pictures from top-to-bottom. The sequences were CBR coded with a target rate of 1 Mbps. Packet sizes were limited at the encoder to 1 kB. The packet size distribution as a result of this restriction for data-partitioned Paris and Stefan is shown in Figure 11. The smaller packet sizes tend to arise from DP-A packets, which mainly contain motion vectors rather than mainly the transform coefficients of DP-B and DP-C. Because of the greater activity within Stephan relative to Paris, the changing texture information cannot be coded as efficiently, resulting in larger DP-C packets taking up a greater percentage of the packet data.



(**a**)



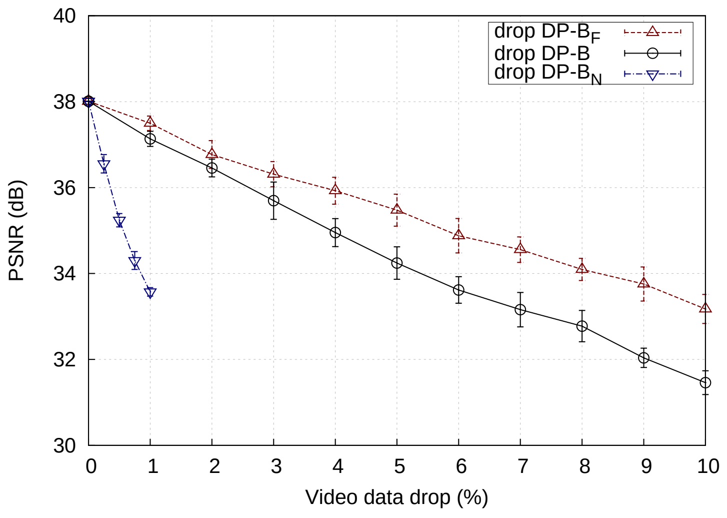
(**b**)

Figure 11. Percentage packet size distribution when using data partitioning and a packet size limit of 1 kB. (**a**) for Paris test sequence; (**b**) for Stefan test sequence.

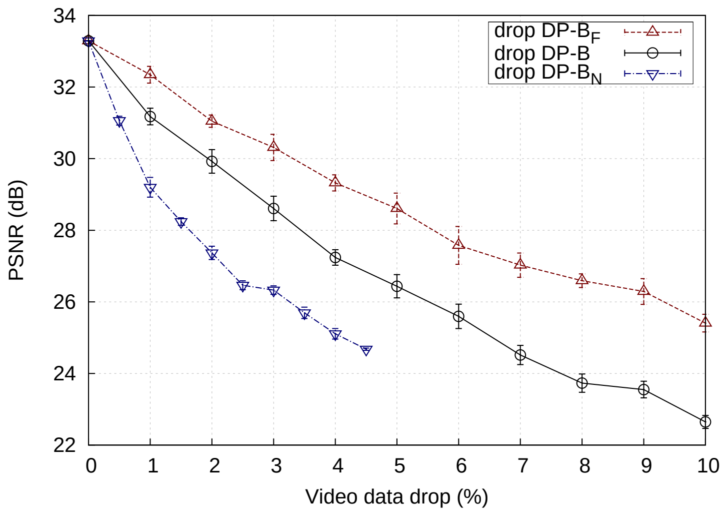
To test the effect of data loss from different parts of the video stream on PSNR, the packets of the tested video sequences were subject to selective uniform drop tests. Three set of tests were performed. In the first set of tests, packets were dropped from DP-B_F only. In the second set of tests, only DP-B_N packets were dropped. Finally, DP-B packets were dropped without any consideration for DP-B sub-parts. These tests allow the effect of the classification scheme to be assessed, for applications such as unequally protecting more significant partitions/sub-partitions or selectively dropping packets during congestion. To allow a more general interpretation, the results are presented in terms of data loss.

Figure 12a shows the results of these tests for the Paris test sequence. Each point is the average of 20 simulations. The results show that selectively dropping DP-B_F can result in a quality gain of 1.5 dB over dropping DP-B. Clearly, selectively dropping DP-B_N data has a damaging impact on video quality, so long as sufficient packets exist to be dropped (hence the way the plots for DP-B_N data loss stop at early percentages). Figure 12b shows results for the more active Stefan test sequence. It can be concluded here that selectively dropping DB-B_F packets can give up to 3 dB gain over dropping DP-B packets. Clearly, naturally-coded MBs are very important for picture decoding, as their loss cannot be easily concealed.

Figure 13 shows the percentage of DP-B$_N$ of the total size of DP-B for the tested sequences. Notice the generally low percentages of DP-B$_N$ data, confirming the above comment about sufficient packet percentages needing to exist in order for them to be dropped. Referring to Figure 14, it is clear that the DP-B$_N$ contribution is very closely connected to the sequence motion activity. For the largely static Akiyo, DP-B$_N$ contributes only 0.02% while for Soccer 27.7% of the intra-coded MBs are naturally encoded. Consequently, the gain over dropping DP-B, Table 3, tends to be more for more active sequences and with increased loss rates, though there is no strict relationship. There is a gain of as much as 5 dB for Soccer for data loss rates of 6% or more. In fact, the results have shown that there is always some gain (provided data loss is occurring) whatever the activity level of the sequence.



(**a**)



(**b**)

Figure 12. Peak Signal-to-Noise Ratio (PSNR) versus percentage video data loss. (**a**) for Paris test sequence; (**b**) for Stefan test sequence.

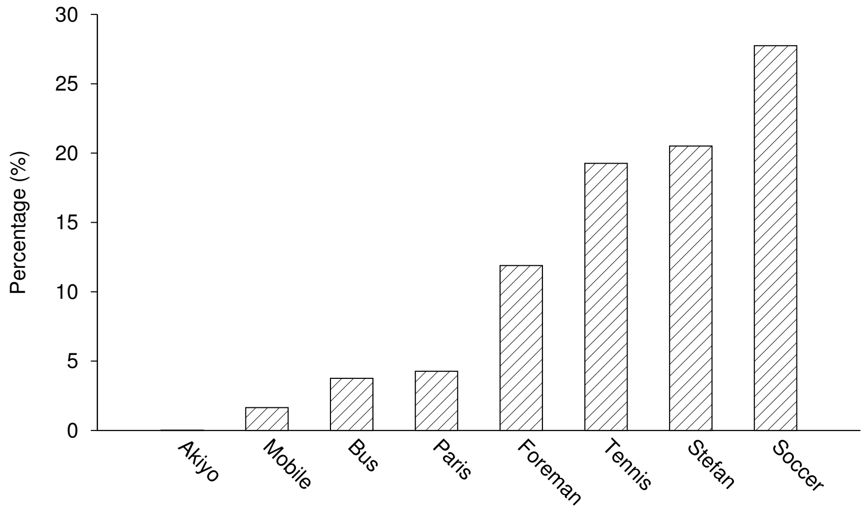


Figure 13. Percentage contributions of B_N of total data partition-B size for different test sequences.

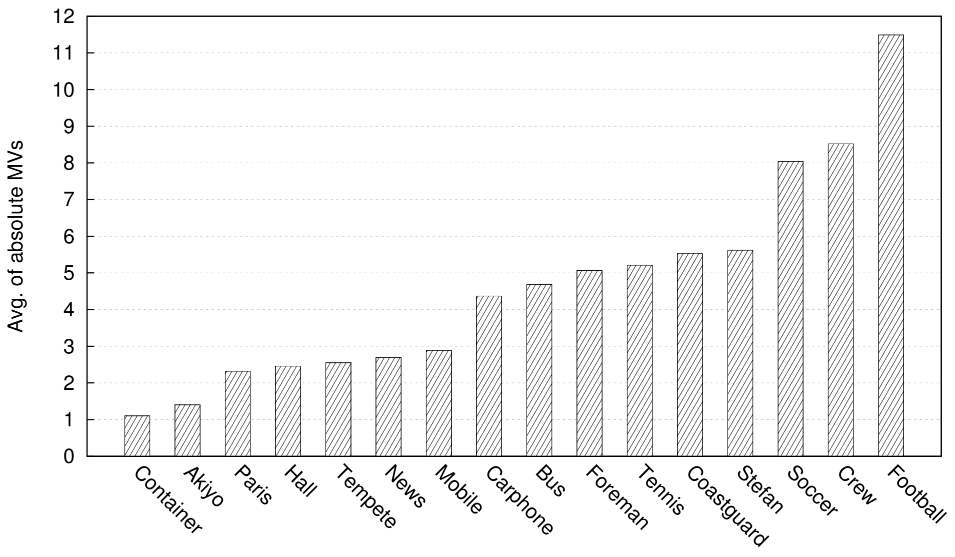


Figure 14. Average of absolute motion vectors for a range of test sequences.

Table 3. Peak Signal to Noise Ratio (PSNR) gain in dB when dropping data partition (DP)-B_F over dropping the unclassified DP-B for different test sequences and data loss rates of 2%, 4%, 6%, 8% and 10%.

Data Loss Rates:	**Video Quality Gain (dB)**				
	2%	**4%**	**6%**	**8%**	**10%**
Foreman	1.29	1.91	2.29	3.17	2.54
Bus	0.88	2.27	3.51	2.96	3.89
Tennis	0.95	2.06	3.19	3.50	4.44
Soccer	3.46	3.72	5.02	5.43	5.47

6. Conclusions

This paper has introduced the concept of refining the data-partitioning facility of H.264/AVC, herein applied to data-partition-B. Our work can also be extended still further by sub-dividing data-partition-C, with some further gain in objective video quality. This paper has demonstrated the substantial potential gain in video quality, which increases with the percentage data drop rate and the coding complexity of the video content. Data-partitioning is an attractive form of prioritization that can be deployed without the need for the scalable extension of H.264. Apart from its lack of support for error resiliency and high energy consumption, compared to H.264/AVC, HEVC tends to rely on transport mechanisms such as Dynamic Adaptive Streaming over HTTP (DASH) [43]. DASH is a form of client-controlled adaptive progressive download. Unfortunately, as TCP is the underlying HTTP protocol, without sufficiently large buffers, start-up delays and stalls are likely to occur [44] as TCP struggles to maintain its reliability during periods of heavy congestion, despite dynamic rate adaptation, which can lead to disconcerting quality fluctuations. In addition, DASH relies on the creation of multi-rate streams of video segments in the manner of simulcast, which may be difficult to achieve or organize on mobile devices, whatever DASH's utility within content distribution networks. The implication is that bit-stream error resilience methods, such as those developed in this paper, will continue to be required for any user-to-user video services, especially if the video encoder is on a mobile device.

Author Contributions: Ismail Ali and Sandro Moiron designed and performed the experiments, while Martin Fleury analyzed the results and presented the paper. Mohammed Ghanbari supervised the experiments and acted as a consultant. All authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wiegand, T.; Sullivan, G.; Bjontegaard, G.; Luthra, A. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 560–576.
2. Talluri, R. Error-resilient coding in the ISO MPEG-4 standard. *IEEE Commun. Mag.* **1998**, *36*, 112–119.
3. Takashima, Y.; Wada, M.; Murakami, H. Reversible variable length codes. *IEEE Trans. Commun.* **1995**, *43*, 158–162.
4. Stockhammer, T.; Bystrom, M. H.264/AVC data partitioning for mobile video communication. In Proceedings of the International Conference on Image Processing, Singapore, 24–27 October 2004; pp. 545–548.
5. Mys, S.; Dhondt, Y.; Van der Walle, D.; De Shrivjer, D.; Van der Walle, R. A performance evaluation of the data partitioning tool in H.264/AVC. *SPIE Multimed. Syst. Appl.* **2006**, *6391*, 639102.
6. Stockhammer, T. Independent data partitions A and B. In Proceedings of the 3rd Meeting of JVT, Fairfax, VA, USA, 6–10 May 2002.
7. Ye, Y.; Chen, Y. Flexible data partitioning mode for streaming video. In Proceedings of the 4th Meeting of JVT, Klagenfurth, Austria, 22–26 July 2002; Doc. JVT-D136.
8. Sullivan, G. Seven steps toward a more robust codec design. In Proceedings of the 3rd Meeting of JVT, Fairfax, VA, USA, 6–10 May 2002; Document: JVT-C117.
9. Lambert, P.; De Neve, W.; Dhondt, Y.; Van de Walle, R. Flexible macroblock ordering in H.264/AVC. *J. Vis. Commun. Image Represent.* **2006**, *17*, 358–375.
10. Schreier, R.; Rahman, A.; Krishnamurthy, G.; Rothermel, A. Architecture analysis for low-delay video coding. In Proceedings of the IEEE International Conference on Multimedia and Expo, Toronto, ON, Canada, 9–12 July 2006; pp. 2053–2056.
11. Ali, I.; Moiron, S.; Fleury, M.; Ghanbari, M. Congestion Resiliency for Data-Partitioned H.264/AVC Video Streaming over IEEE 802.11e Wireless Networks. *Int. J. Handheld Comput. Res.* **2012**, *3*, 55–73.
12. Satyan, R.; Nyamweno, S.; Labeau, F. Comparison of intra updating methods for H.264. In Proceedings of the 10th International Symposium on Wireless Personal Multimedia Communications, Sydney, Australia, 7–10 September 2007; pp. 996–999.

13. Haskell, P.; Messerschmitt, D. Resynchronization of motion compensated video affected by ATM cell loss. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, CA, USA, 23–26 March 1992; pp. 545–548.
14. Schreier, R.; Rothermel, A. Motion adaptive intra refresh for the H.264 video coding standard. *IEEE Trans. Consum. Electron.* **2006**, *52*, 249–253.
15. Stuhlmüller, K.; Färber, N.; Link, M.; Girod, B. Analysis of video transmission over lossy channels. *IEEE J. Sel. Areas Commun.* **2000**, *18*, 1012–1032.
16. Stockhammer, T.; Kontopodis, D.; Wiegand, T. Rate-distortion optimization for JVT/H.26L video coding in packet loss environment. In Proceedings of the Packet VideoWorkshop, Pittsburgh, PY, USA, 24 April 2002.
17. Zhang, R.; Regunathan, S.; Rose, K. Video coding with optimal inter/intra-mode switching for packet loss resilience. *IEEE J. Sel. Areas Commun.* **2000**, *18*, 966–976.
18. Zhang, Y.; Gao, W.; Lu, Y.; Huang, Q.; Zhao, D. Joint source-channel rate-distortion optimization for H.264 video coding over error-prone networks. *IEEE Trans. Multimed.* **2007**, *9*, 445–454.
19. Schierl, T.; Hannuksela, M.; Wang, Y.K.; Wenger, S. System Layer Integration of High Efficiency Video Coding. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1871–1884.
20. Ali, I.; Moiron, S.; Fleury, M.; Ghanbari, M. Refined Data Partitioning for Improved Video Prioritization. In Proceedings of the IEEE 8th Computer Science and Electronic Engineering Conference, Colchester, UK, 28–30 September 2016.
21. Cisco. *Cisco Visual Networking Index: Forecast and Methodology, 2012–2017*; Technical Report; Cisco Systems, Inc.: San Jose, CA, USA, 2012.
22. Stockhammer, T.; Hannuksela, M. H.264/AVC video for wireless transmission. *IEEE Wirel. Commun.* **2005**, *12*, 6–13.
23. Wenger, S. H.264/AVC over IP. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 645–656.
24. Marpe, D.; Schwarz, H.; Wiegand, T. Context-adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 620–636.
25. Dhondt, Y.; Mys, S.; Vermeirsch, K.; Van de Walle, R. Constrained inter prediction: Removing dependencies between different data partitions. In *Advanced Concepts for Intelligent Vision Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 720–731.
26. Worrall, S.; Fabri, S.; Sadka, A.H.; Kondoz, A.M. Prioritisation of data partitioned MPEG-4 video over mobile networks. *Eur. Trans. Telecommun.* **2001**, *12*, 169–174.
27. Barmada, B.; Ghandi, M.; Jones, E.; Ghanbari, M. Prioritized transmission of data partitioned H.264 video with hierarchical QAM. *IEEE Signal Process. Lett.* **2005**, *12*, 577–580.
28. Yang, G.; Shen, D.; Li, V.O.K. UEP for video transmission in space-time coded OFDM systems. In Proceedings of the IEEE INFOCOM, Twenty-Third AnnualJoint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, 7–11 March 2004; pp. 1200–1210.
29. Wenger, S.; Stockhammer, T. H.26L over IP and H.324 framework. In Proceedings of the 14th Meeting of ITU-T VCEG, Santa Barbara, CA, USA, 24-27 September, 2001; Document: VCEG-N52.
30. Varsa, V.; Hannuksela, M.; Wang, Y.K. Non-normative error concealment algorithms. In Proceedings of the 14th Meeting of ITU-T VCEG, Santa Barbara, CA, USA, 24-27 September, 2001; Document: VCEG-N62.
31. Tan, W.; Setton, E.; Apostolopoulos, J. Lossless FMO and Slice Structure Modification for Compressed H.264 Video. In Proceedings of the IEEE International Conference on Image Processing, San Antonio, TX, USA, 16 September–19 October 2007; Volume 4, pp. 285–288.
32. Mazataud, C.; Bing, B. A new lossless FMO removal scheme for H.264 videos. In Proceedings of the 18th International Conference on Computer Communications and Networks, San Francisco, CA, USA, 3–6 August 2009; pp. 1–6.
33. Im, S.; Pearmain, A. An optimized mapping algorithm for classified video transmission with the H.264 Flexible Macroblock Ordering. In Proceedings of the IEEE International Conference on Multimedia and Expo, Toronto, ON, Canada, 9–12 July 2006; pp. 1445–1448.
34. Argyropoulos, S.; Tan, A.; Thomos, N.; Arikan, E.; Strintzis, M. Robust transmission of multi-view video streams using flexible macroblock ordering and LT codes. In Proceedings of the 3DTV Conference, Kos Island, Greece, 7–9 May 2007; pp. 1–4.

35. Vu, T.; Aramvith, S. An error-resilience technique based on FMO and error propagation for H.264 video coding in error prone channels. In Proceedings of the IEEE International Conference on Multimedia and Expo, New York, NY, USA, 28 June–3 July 2009; pp. 205–208.
36. Xu, J.; Wu, Z. Joint adaptive intra refreshment and unequally error protection algorithms for robust transmission of H.264/AVC video. In Proceedings of the IEEE International Conference on Multimedia and Expo, Toronto, ON, Canada, 9–12 July 2006; pp. 693–696.
37. Wenger, S.; Horowitz, M. FMO: Flexible Macroblock Ordering. In Proceedings of the 3rd Meeting of Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Munich, Germany, 15–19 March, 2004; Document: JVT-C089.
38. Bing, B. *3D and HD Broadband Video Networking*; Artech House: Boston, MA, USA, 2010.
39. Wenger, S. Coding Performance when not using In-Picture Prediction. In Proceedings of the 3rd Meeting of Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Munich, Germany, 15–19 March, 2004; Document: JVT-B024.
40. Richardson, I. *H.264 and MPEG-4 Video Compression*; Wiley & Sons Ltd.: Chichester, UK, 2004.
41. Hannuksela, M.; Wang, Y.; Gabbouj, M. Isolated regions in video coding. *IEEE Trans. Multimed.* **2004**, *6*, 259–267.
42. Ali, I.; Moiron, S.; Fleury, M.; Ghanbari, M. Packet Prioritization for H.264/AVC Video with Cyclic Intra-Refresh Line. *J. Vis. Commun. Image Represent.* **2013**, *24*, 486–498.
43. Stockhammer, T. Dynamic adaptive streaming over HTTP –: Standards and design principles. In Proceedings of the Second Annual ACM Conference on Multimedia Systems, San Jose, CA, USA, 23–25 February 2011; pp. 133–144.
44. Seufert, M.; Egger, S.; Slanina, M.; Zinner, T.; Hoßfeld, T.; Tran-Gia, P. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 469–492.