AVC-762

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION

ISO/IEC JTC1/SC29/WG11 **N0930**

MPEG 95/
March 1995

Source: Video Subgroup

Title: Proposed corrigendum for ISO/IEC 13818-2 (MPEG-2 Video)
Lausanne 20, March 1995

Introduction

Each corrigendum item is followed by an informative background.

1. Fix Ambiguous Dual-prime Vertical Vector Range Restriction

In section 8.3, replace note at the bottom of Table 8.8 :

This restriction applies to the final reconstructed motion vector. In the case of dual prime motion vectors it applies before scaling is performed, after scaling is performed and after the small differential motion vector has been added.

By the following text :

This restriction applies to the final reconstructed motion vector. In the case of dual prime motion vectors this restriction applies to all the following values :

vector'[0][0][1]
*((vector'[0][0][1] * m[parity_ref][parity_pred])//2)*
*((vector'[0][0][1] * m[parity_ref][parity_pred])//2) + e[parity_ref][parity_pred]*
*((vector'[0][0][1] * m[parity_ref][parity_pred])//2) + dmvector[1]*
*((vector'[0][0][1] * m[parity_ref][parity_pred])//2) + e[parity_ref][parity_pred] + dmvector[1]*

Background :

An ambiguity has been pointed-out regarding the restriction on the vertical range of the final reconstructed motion vectors when dual-prime prediction is used.

In section 8.3, the note at the bottom of Table 8.8 says:

<<This restriction applies to the final reconstructed motion vector. In the case of dual prime motion vectors it applies before scaling is performed, after scaling is performed and after the small differential motion vector has been added.>>

The final dual prime motion vectors are computed according to the equations specified in section 7.6.3.6. In those equations, the vertical motion vector coordinate is computed as follows:

```
vector'[r][0][1] = ((vector'[0][0][1] * m[parity_ref][parity_pred])//2)
                                    + e[parity_ref][parity_pred] + dmvector[1];
```

The "*e[parity_ref][parity_pred]*" is the adjustment necessary to reflect the vertical shift between the lines of the top field and the bottom field.

The problem is the ambiguity of the expression "*before scaling is performed, after scaling is performed and after the small differential motion vector has been added*", since there are in fact 3 operations involved:

a) scaling i.e., the operation ((*vector'*[0][0][1] * *m[parity_ref][parity_pred]*)//2)

b) adding *e[parity_ref][parity_pred]* (forgotten in the note describing the restriction)

c) adding *dmvector*[1]

The order of the operations is not specified by the standard (only the result counts). In particular b) and c) can be performed in any order, and *e[parity_ref][parity_pred]* and *dmvector*[1] can be added first together and the result be then added to the scaled vector.

2. Change Semantics when Chromaticity Parameters are not Coded

After table 6-7, replace :

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the chromaticity is assumed to be that corresponding to colour_primaries having the value 1.

By the following text :

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the chromaticity is assumed to be implicitely defined by the application.

After table 6-8, replace :

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the transfer characteristics are assumed to be those corresponding to transfer_characteristics having the value 1.

By the following text :

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the transfer characteristics are assumed to be implicitely defined by the application.

After table 6-9, replace :

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the matrix coefficients are assumed to be those corresponding to matrix_coefficients having the value 1.

By the following text and NOTE:

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the matrix coefficients are assumed to be implicitely defined by the application.

NOTE - In applications which may have signals with more than one set of colour primaries, transfer characteristics, and/or matrix coefficients, it is recommended to transmit a sequence display extension with colour_description set to one, and to specify the appropriate values for the values colorimetry parameters.

Background :

In applications where only one set of colour primaries, transfer characteristics, and matrix coefficients is used, there is not a need for the codec to pass colour description parameters. The current syntax includes default color description parameters which could cause improper interpretation of many bitstreams.

A solution to this problem which embraces all existing implementations without introducing a new problem for other implementations is to not have a default definition, but instead to let the application define the default, as described in this corrigendum.

3. Define signed_level = -2048 Reserved

In Annex B, section B.5, replace the right part of Table B-16 by:

fixed length code	signed_level
1000 0000 0000	reserved
1000 0000 0001	-2047
1000 0000 0010	-2046
...	...
1111 1111 1111	-1
0000 0000 0000	forbidden
0000 0000 0001	+1
...	...
0111 1111 1111	+2047

Background :

It was pointed-out that in the right part of Table B-16, the entry "1000 0000 0000" corresponding to *signed_level* = -2048 was omitted but never explicitly forbidden.

This corrigendum fixes the problem and guarantees that *signed_level* can always be coded with sign/value where the value fits on 11-bit, thus avoiding the risk of breaking any existing implementation. It was decided that this value should be "reserved" rather than "forbidden". "forbidden" is usually used for values that would cause start-code emulation, which is not the case here.

4. First Frame after any sequence_header() cannot be a B-Frame

In section 6.1.1.6 (sequence header), replace the sentence :

In the coded bitstream, a repeat sequence header may precede either an I-picture or a P-picture, but not a B-picture.

by:

In the coded bitstream, the first picture following a sequence header or a repeated sequence header shall be either an I-picture or a P-picture, but not a B-picture.

Background :

This was an editorial oversight.

5. Lift Restriction on frame_pred_frame_dct in Progressive Frames

In section 6.3.10, under the description of the syntax element frame_pred_frame_dct, replace the statement:

frame_pred_frame_dct shall be 1 if progressive_frame is 1

by :

frame_pred_frame_dct shall be 1 if progressive_sequence is 1

In section 6.3.10, under the description of the syntax element progressive_frame, replace the statement:

- *frame_pred_frame_dct shall be 1*

by :

- *if progressive_sequence is equal to one, then frame_pred_frame_dct shall be 1*

Background:

The following text from MPEG95/044 explains the background for this corrigendum :

It appears that the restriction <<frame_pred_frame_dct shall be 1 if progressive_frame is 1>> causes problem when editing bitstreams bitstreams, as demonstrated by the following example.

Let's say sequence 1 ends with a top field first interlaced frame, i.e.

progressive_sequence=0

picture_structure=3

progessive_frame=0

top_field_first=1

repeat_first_field=0

while sequence 2 starts with a bottom field first interlaced frame, i.e.

progressive_sequence=0

picture_structure=3

progessive_frame=0

top_field_first=0

repeat_first_field=0

Now, to put them together we need a "glue" frame with repeat_first_field as follows:

progressive_sequence=0

picture_structure=3

progessive_frame=1

top_field_first=1

repeat_first_field=1

The situation is depicted graphically as follows:

```
1T    g0 g2   2T
 1B    g1    2B
```

where (1T,1B) is the last frame of sequence 1 and (2B,2T) is the first frame of sequence 2. The "g"s signify the 3 fields of the "glue" frame. When editing bitstream it is often necessary to insert "glue" frames, not only to match parity but also to introduce a delay in order to match VBV buffer fullness.

Ideally, we would like the "g"s to repeat the last field of sequence 1, i.e. g0 and g1 are both predicted by 1B while g2 is a repeat of g0.

It appears, however, that this is impossible since repeat_first_field can be 1 only if progressive_frame is 1, and field prediction is not allowed since progressive_frame = 1 implies frame_pred_frame_dct = 1.

That means we'll get an inevitable jerk going from sequence 1 to sequence 2.

Our view is that progressive_frame applies to the current frame, while whether it can be coded well with frame_pred_frame_dct depends on both the current frame and the reference frame. The fact that the current frame is progressive does not necessary mean that it can be coded well with frame_pred_frame_dct = 1.

6. VBV

In Clause C.3.1 of Annex C, after definition of t(n), remove :

For the bits preceding the first picture start code and following the final picture start code $R(n) = R_{max}$

and add following text:

Ambiguity at the beginning of a sequence :

> *The interval of time* t_{n+1}-t_n *between removal of two consecutive pictures can normally be derived from the bitstream as described in clauses C.9, C.10, C.11 and C.12.*
>
> *When random access is made in a sequence,* $t_{n+1} - t_n$ *cannot be determined from the video bitstream alone for the first picture(s) after the sequence header since the previous coded P- or I-Frame does not exist in the decoded sequence.. If the bitstream is multiplexed as part of a systems bitstream according*

to Recommendation ITU-T H.220.0 | ISO/IEC 13818-1 then it is possible (but not certain) that information in the systems bitstream may be used to determine unambiguously this interval of time. This information is available if decoding time stamps (DTS) are transmitted for picture n and n+1.

If the rate R(n) cannot be determined unambiguously, it is not possible for the VBV to precisely determine the fullness in trajectories in the VBV buffer during a limited period (always less than the maximum value for vbv_delay, which is approx. 0.73 seconds), therefore strict VBV verification of the entire bitstream is not always possible. Note that an encoder always knows the values of tn+1 - tn after each repeated sequence headers and therefore knows how to generate a bitstream that does not violate the VBV constraints at those points.

The ambiguity may become a problem when the video bitstream is remultiplexed and delivered at a rate different from the intended piecewise constant rate R(n).

It should also be noted that the input rate for the bits preceding the first picture header cannot be determined from the bitstream.

Ambiguity at the end of a sequence :

The input of all the bits following the picture start code of the picture preceding an end of sequence code cannot be determined from the bitstream. There shall exist an input rate for these bits that does not lead to an overflow or, if low_delay is equal to 1, an underflow of the VBV buffer. This rate shall be less than the maximum rate specified in the sequence header.

In Clause C.9 of Annex C, remove :

If t_{n+1}-t_n cannot be determined with any of the previous paragraphs because the previous P- or I-Picture does not exist (which can occur at the beginning of a sequence), then the time interval is arbitrary with the following restrictions:

*The time interval between removing one frame (or the first field of a frame) and removing the next frame can be arbitrarily defined equal to T, 2*T or 3*T. In this case the delivery rate of the data for the first frame is ambiguous. Therefore the VBV buffer status until after this data has been removed from the VBV buffer may have more than one value. At least one of the valid choices for the decoding time shall lead to a set of VBV buffer states that meet the requirements of this annex on overflow and underflow. If the bitstream is multiplexed as part of a systems bitstream according to Recommendation ITU-T H.220.0 | ISO/IEC 13818-1 then information in the systems bitstream may be used to determine unambiguously the VBV buffer state after removing the first picture.*

In Clause C.11 of Annex C, remove :

If t_{n+1}-t_n cannot be determined with any of the previous paragraphs because the previous coded P- or I frame does not exist (which can occur at the beginning of a sequence), then the time interval is arbitrary with the following restrictions:

*The time interval between removing one frame (or the first field of a frame) and removing the next frame (or the first field of a frame) can be arbitrarily defined equal to 2*T or 3*T. Therefore the VBV buffer status until after this data has been removed from the VBV buffer may have more than one value. At least one of the valid choices for the decoding time shall lead to a set of VBV buffer states that meet the requirements of this annex on overflow and underflow. If the bitstream is multiplexed as part of a systems bitstream according to Recommendation ITU-T H.220.0 | ISO/IEC 13818-1 then information in the systems bitstream may be used to determine unambiguously the VBV buffer state.*

Background :

The sentence <<*For the bits preceding the first picture start code and following the final picture start code R(n) = R_{max}*>> was incorrect, and the ambiguities in the VBV specification were not well documented and sometimes documented incorrectly. In particular the existence of a system stream is not sufficient to lift the ambiguity in all cases.

7. Fix Problem in profile_and_level_indication Description

Replace paragraph preceding Table 8-1:

The profile_and_level_indication in the sequence_extension indicates the profile and level to which the bitstream complies. The meaning of the bits in this parameter is defined in Table 8-1.

By :

The profile_and_level_indication in the sequence_extension indicates the profile and level to which the bitstream complies. The most significant bit of profile_and_level_indication is called "escape bit". When the escape bit is set to zero, the profile and level are derived from profile_and_level_indication according to Tables 8-1, 8-2 and 8-3.

Background :

In the course of study for the 4:2:2 profile amendment, it has been found that the descriptions of the profile_and_level_indication syntax referring to Table 8-1 and Table 8-4 are not aligned. We should state that

if the escape bit is '0', then profile and level are structured as in Tables 8-1,2,3,

if the escape bit is '1', then profile and level are structured as in Table 8-4.

8. Fix Typing Mistakes

In the definition for intra_vlc_format, the reference to "7.2.1" should be changed to "7.2.2.1".

In section 7.2.1:

- All "dc_dct_size" should be changed to "dct_dc_size", to be consistent with the syntax described in section 6.2.6.

- All "dc_dct_differential" should be changed to "dct_dc_differential", to be consistent with the syntax described in section 6.2.6.

- All the "dc_dct_pred" should be changed to "dct_dc_pred" to be consistent with the previous changes.

9. Define slice_picture_id in Reserved Bits of Slice Extension

In section 6.2.4, replace the syntax table for slice by :

slice() {	No. of bits	Mnemonic
slice_start_code	32	bslbf
if (vertical_size > 2800)		
slice_vertical_position_extension	3	uimsbf
if (*<sequence_scalable_extension() is present in the bitstream>*) {		
if (scalable_mode == "data partitioning")		
priority_breakpoint	7	uimsbf
}		
quantiser_scale_code	5	uimsbf
if (nextbits() == '1') {		
slice_extension_flag	1	bslbf
intra_slice	1	uimsbf
slice_picture_id_enable	1	uimsbf
slice_picture_id	6	uimsbf
while (nextbits() == '1') {		
extra_bit_slice **/* with the value '1' */**	1	uimsbf
extra_information_slice	8	uimsbf
}		
}		
extra_bit_slice **/* with the value '0' */**	1	uimsbf
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

In section 6.3.16, replace:

intra_slice_flag -- This flag shall be set to '1' to indicate the presence of intra_slice and reserved_bits in the bitstream.

By:

slice_extension_flag-- This flag shall be set to '1' to indicate the presence of intra_slice, slice_picture_id_enable, slice_picture_id and extra_bit_slice in the bitstream.

In section 6.3.16, replace:

reserved_bits -- This is a 7 bit integer, it shall have the value zero, other values are reserved.

By:

slice_picture_id_enable -- This flag controls the semantics of slice_picture_id. If slice_picture_id_enable is set to "0", slice_picture_id is not used by this specification and shall have the value zero. If slice_picture_id_enable is set to "1", slice_picture_id may have a value different from zero.

slice_picture_id_enable must have the same value in all the slices of a picture. slice_picture_id_enable may be omitted from the bitstream (by setting slice_extension_flag to '0') in which case it shall be assumed to have the value zero.

slice_picture_id_enable is not used by the decoding process.

slice_picture_id -- This is a 6 bit integer. If slice_picture_id_enable is set to "0", slice_picture_id is not used by this specification and shall have the value zero. If slice_picture_id_enable is set to "1", slice_picture_id is application defined and may have any value, with the constraint that slice_picture_id shall have the same value in all the slices of a picture.

slice_picture_id is not used by the decoding process. slice_picture_id is intended to aid recovery on severe bursts of errors for certain types of applications. For example the application may increment slice_picture_id with each transmitted picture, so that in case of severe burst error, when several slices are lost, the decoder can know if the slice following the burst error belongs to the current picture or to another picture, which may be the case if at least a picture header has been lost.

Background:

In some networks that do not have guaranteed bandwidth, eg, ethernet, FDDI, large bursts of errors may occur occasionally. If the errors are localized to one picture, then recovery can happen upon reception of the very next slice header. However, if the errors are spread over several pictures, then recovery may take a long time, since the decoder does not know what the decoding parameters are for the next available slice.

If the header information is available from a separately transmitted Data Partitioning layer, or available from the application, then error recovery can be speeded up considerably if the slice header contains information that an application can use to identify which picture the slice belongs to. The **slice_picture_id** provides this information.

10. Fix Text in Copyright Extension

In section 6.3.15 (Copyright extension):

Replace :

copyright_identifier *-- This is a 8-bit integer which identifies a Registration Authority as designated by ISO/IEC JTC1/SC29.*

By :

copyright_identifier *-- This is a 8-bit integer given by a Registration Authority as designated by ISO/IEC JTC1/SC29.*

Replace :

In this case, the value of copyright_number identifies uniquely the copyrighted work marked by the copyrighted extension and is provided by the Registration Authority identified by copyright_identifier.

By :

In this case, the value of copyright_number identifies uniquely the copyrighted work marked by the copyrighted extension.

Background :

This corrigendum is to clear a confusion of who issues the copyright_identifier or the copyright_number