Title: METHOD FOR REFERENCE PICTURE MANAGEMENT INVOLVING MULTIVIEW VIDEO CODING

Abstract: A series of memory management operation commands are described for the memory management of decoded reference pictures that are stored within a memory (1110), a multiview video coding operation. The video coding operation will consider the view for which a picture is to be coded as compared against the view associated with the stored reference picture (1120), where a memory management operation command is enabled affecting the memory status of the stored reference pictures where such an effect may be designation of a reference picture (1125) being a short term reference picture, a long term reference picture, or designating the reference picture as not being needed.
Published: without international search report and to be republished upon receipt of that report.
METHOD FOR REFERENCE PICTURE MANAGEMENT INVOLVING MULTIVIEW VIDEO CODING

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application Serial No. 60/851,522, filed October 13, 2006, which is incorporated by reference herein in its entirety and U.S. Provisional Application Serial No. 60/851,589, filed on October 13, 2006, which is incorporated by reference herein in its entirety, as well.

TECHNICAL FIELD

The present invention relates to the field of moving pictures, especially the issue of the memory maintenance of moving pictures associated with multiview video coding.

BACKGROUND

Many interframe encoding systems make use of reference pictures where the use of such reference pictures helps reduce the size of an encoded bit stream. This type of result is encoding efficiency is better than just using intraframe encoding techniques, by themselves. Many encoding standards therefore incorporate both intraframe and interframe encoding techniques to encode a bit stream from a series of moving images. As known in the art, different types of reference pictures are used for encoding standards such as an "I" picture which is encoded only by using elements within the picture itself (intraframe), a "B" picture which is encoded by using elements from within the picture itself and/or elements from two previous reference pictures (interframe), and a "P" picture which is encoded by using elements from within the picture itself and/or elements from one previous reference picture (interframe). Both "B" and "P" pictures can use multiple reference pictures, but the difference between both of these type of pictures is that "B" allows the use of inter prediction with at most two motion-compensated prediction signals per block while "P" allows the use of one only predictor per predicted block.

When the "B" or "P" pictures are being encoded and/or decoded, such pictures are therefore dependent on other reference frames so that such pictures may be properly encoded or constructed during a decoding operation. The encoding/decoding system should provide some type of memory location so that reference picture can be stored while other pictures are being encoded or decoded in
view of such reference pictures. Obviously, after a while, a reference picture cannot be used for a coding operation because no more pictures to be coded will use the reference picture during the future coding operation.

Although, one could store all the reference pictures permanently in a storage device, such a solution would be an inefficient use of memory resources. Therefore, memory techniques such as using a First in First Out (FIFO) or Last in First Out (LIFO) memory operations, as known in the art, could be used in the case of operating a memory device with the storage of reference pictures to help reduce the space required for such reference pictures (by discarding unnecessary reference pictures).

Such memory operations however may produce undesirable results when considering the use of an multiview coding system where pictures that are encoded and/or decoded have both a temporal and a view inter-relationship. That is, the multiview coding system introduces the aspect of having multiple views of moving pictures, where each view represents a different view of a respective object/scene. Now, a reference picture may be used in the encoding or decoding of pictures associated with two different views. Therefore, simple memory techniques can not be used in such an environment.

SUMMARY

These and other drawbacks and disadvantages of the prior art are addressed by the present principles, which are directed to a method and apparatus for reusing available motion information as a motion estimation predictor for video encoding.

According to an aspect of the present principles, there is provided a coder that performs memory management operations on a reference picture stored in a memory device in view of information from a picture being decoded by the decoder, where such information is related to view information associated with that reference picture.

These and other aspects, features and advantages of the present principles will become apparent from the following detailed description of exemplary embodiments, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The present principles may be better understood in accordance with the following exemplary figures, in which:
FIG. 1 presents an exemplary embodiment multiview coding of video picture views at different times, where such video pictures are coded using reference pictures in the manner indicated in the figure.

FIG. 2 presents an exemplary embodiment of a video coder in accordance with the principles of the present invention.

FIG. 3 presents an embodiment of pseudo code for a syntax element dec_ref_pic_marking() used in accordance with the principles of the present invention.

FIG. 4 presents the pseudo code for a syntax element seq_parameter_set_mvc_extension() used in accordance with the principles of the present invention.

FIG. 5 presents an embodiment of pseudo code for a syntax element dec_ref_pic_marking_rnvc_extension() used in accordance with the principles of the present invention.

FIG. 6 presents an embodiment of a sample picture header used in accordance with the principles of the present invention.

FIG. 7 presents an embodiment of pseudo code for a syntax element dec_ref_pic_marking_mvc_extension() used in accordance with the principles of the present invention.

FIG. 8 presents an embodiment of a sample picture header used in accordance with the principles of the present invention.

FIG. 9 presents an embodiment of pseudo code for a syntax element dec_ref_pic_marking_mvc_extension() used in accordance with the principles of the present invention.

FIG. 10 presents an embodiment of pseudo code for a syntax element dec_ref_pic_marking_mvc_extension() used in accordance with the principles of the present invention.

FIG. 11 presents a block diagram of an embodiment of a picture marking method in accordance with principles of the present invention.
DETAILED DESCRIPTION

The principles of the invention can be applied to any intra-frame and inter-frame based encoding standard. The term "picture" which is used throughout this specification is used as a generic term for describing various forms of video image information which can be known in the art as a "frame", "field", and "slice", as well as the term "picture" itself.

Also, in the description of the present invention, various commands (syntax elements) which use the C language type of formatting are detailed in the figures that use the following nomenclature for descriptors in such commands:

u(n): unsigned integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits( n ) interpreted as a binary representation of an unsigned integer with most significant bit written first.

ue(v): unsigned integer Exp-Golomb-coded syntax element with the left bit first.
se(v): signed integer Exp-Golomb-coded syntax element with the left bit first.
C: represents the category for which a syntax element applies to, i.e. to what level should a particular field apply.

The present description illustrates the present principles. It will thus be appreciated that those skilled in the art will be able to devise various arrangements that, although not explicitly described or shown herein, embody the present principles and are included within its spirit and scope.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the present principles and the concepts contributed by the inventor(s) to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions.

Moreover, all statements herein reciting principles, aspects, and embodiments of the present principles, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as
equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

Thus, for example, it will be appreciated by those skilled in the art that the block diagrams presented herein represent conceptual views of illustrative circuitry embodying the present principles. Similarly, it will be appreciated that any flow charts, flow diagrams, state transition diagrams, pseudo code, and the like represent various processes which may be substantially represented in computer readable media and so executed by a computer or processor, whether or not such computer or processor is explicitly shown.

The functions of the various elements shown in the figures may be provided through the use of dedicated hardware as well as hardware capable of executing software in association with appropriate software. When provided by a processor, the functions may be provided by a single dedicated processor, by a single shared processor, or by a plurality of individual processors, some of which may be shared. Moreover, explicit use of the term "processor" or "controller" should not be construed to refer exclusively to hardware capable of executing software, and may implicitly include, without limitation, digital signal processor ("DSP") hardware, read-only memory ("ROM") for storing software, random access memory ("RAM"), and non-volatile storage.

Other hardware, conventional and/or custom, may also be included. Similarly, any switches shown in the figures are conceptual only. Their function may be carried out through the operation of program logic, through dedicated logic, through the interaction of program control and dedicated logic, or even manually, the particular technique being selectable by the implementer as more specifically understood from the context.

In the claims hereof, any element expressed as a means for performing a specified function is intended to encompass any way of performing that function including, for example, a) a combination of circuit elements that performs that function or b) software in any form, including, therefore, firmware, microcode or the like, combined with appropriate circuitry for, executing that software to perform the function. The present principles as defined by such claims reside in the fact that the functionalities provided by the various recited means are combined and brought
together in the manner which the claims call for. It is thus regarded that any means that can provide those functionalities are equivalent to those shown herein.

Reference in the specification to "one embodiment" or "an embodiment" of the present principles means that a particular feature, structure, characteristic, and so forth described in connection with the embodiment is included in at least one embodiment of the present principles. Thus, the appearances of the phrase "in one embodiment" or "in an embodiment" appearing in various places throughout the specification are not necessarily all referring to the same embodiment.

FIG. 1 represents an exemplary embodiment of a reference picture structure used in a Multiview Coding system. Specifically, the presented structure pertains to the use of eight different views (S0-S7) for times (T0-T100) in accordance with the multiview encoding (MVC) scheme proposed in A. Vetro, Y. Su, H. Kimata, A. Smolic, "Joint Multiview Video Model (JMVM) 1.0", JVT-T208.doc, Klagenfurt, Austria, July, 2006. This multiview encoding standard is based on coding in the Advanced Video Coding (AVC) standard (G. Sullivan, T. Wiegand, A. Luthra, "Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding)", Palma de Mallorca, ES 18-22, October 2004). The large difference between both standards is AVC does not addresses coding multiview pictures while MVC does.

Referring back to FIG. 1, it can be seen for example that when coding a picture associated with view S1 at T1, that the picture to be coded is related to pictures (reference pictures) from the same view (S1 at TO and S1 at T2), and that the picture to be coded is related to pictures from a different view (SO at T1 and S2 at T1). Hence, when coding the picture associated with S1, T1, it would make sense to keep reference pictures (S1 at TO, S1 at T2, SO at T1 and S2 at T1) in a memory device such as a buffer, register, RAM, and the like which may be implemented either in hardware, software, or a combination thereof. Such reference pictures however would not be that useful when considering the coding of a picture S7 at T98, which would rely on the use of different reference pictures than used for picture S1 at T1.

One solution for effective memory management of a buffer for a coding operation is disclosed in the use of the decoded picture buffer (DPB) which is associated with the AVC video standard. In a simplified version of block diagram 200 in FIG. 2, the operation between a coder 205, coding buffer 210, and decoded picture
buffer 215 is shown. During a coding operation (either encoding or decoding), a picture that is currently being coded by coder 205 is present in coding buffer 210, while previously coded reference pictures are stored in decoded picture buffer 215. AVC discloses the use commands known as memory management control operations (MMCO) which allow the coder 205 to specify how the reference pictures in decoder picture buffer 215 should be maintained. That is, when a picture is being encoded, such MMCOs are inputted into the header of the picture presently being encoded as to specify what should be done with the reference pictures that came before such a picture. This operation is known as “marking”. These commands then can be used by the coder 205 in the future as to determine what should be done with a reference picture that is present in decoder picture buffer 215. It should be noted that although the term picture is being used to represent various elements video information, AVC refers to the use of slices where such reference pictures may use slices from the same picture as a “reference picture”, and regardless of how a picture may be sub-divided, the principles of the present invention apply.

FIG. 3 represents the command in AVC (dec_ref_pic_marking) that is used to implement MMCO commands. Specifically, a reference picture is marked as being a short term reference picture, a long term reference picture, or the picture is marked as not being a reference picture (in which case the reference picture would be discarded if memory is needed) when employing a MMCO command. The statuses of reference pictures may be changed as more pictures are coded, for example a reference picture that is designated as being a short term as one picture is being code picture can be identified as being a long term reference picture when a second picture is being coded.

FIG. 3 also presents a command flag known as the adaptive_ref_pic_marking_mode_flag which is used between two different modes for performing the marking of picture headers (slice headers). When the flag is set to "0", a sliding window reference marking mode is activated that provides a FIFO mechanism for short-term reference pictures. When the flag is set for "1", an adaptive reference picture marking mode is activated which provides syntax elements to mark reference pictures as “unused for reference” and to assign long-term-frame indices to reference pictures, etc. The various assignments for reference frames via MMCO commands used in AVC are shown in Table 1 below:
<table>
<thead>
<tr>
<th>memory_management_control_operation</th>
<th>Memory Management Control Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>End memory_management_control_operation syntax element loop</td>
</tr>
<tr>
<td>1</td>
<td>Mark a short-term reference picture as &quot;unused for reference&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Mark a long-term reference picture as &quot;unused for reference&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Mark a short-term reference picture as &quot;used for long-term reference&quot; and assign a long-term frame index to it</td>
</tr>
<tr>
<td>4</td>
<td>Specify the maximum long-term frame index and mark all long-term reference pictures having long-term frame indices greater than the maximum value as &quot;unused for reference&quot;</td>
</tr>
<tr>
<td>5</td>
<td>Mark all reference pictures as &quot;unused for reference&quot; and set the MaxLongTermFrameIdx variable to &quot;no long-term frame indices&quot;</td>
</tr>
<tr>
<td>6</td>
<td>Mark the current picture as &quot;used for long-term reference&quot; and assign a long-term frame index to it</td>
</tr>
</tbody>
</table>

TABLE 1

One problem with the design of AVC is that pictures can be identified by their respective frame number (framejium) value which represents the actual coding order of a picture (in a sequence of pictures) and by a picture's respective picture order count (POC) which is the order in which a picture is to be displayed. MVC however is more complex than AVC because multiple views must be considered in MVC while
AVC is concerned with just a single view. Hence, in MVC, an additional value view id (view_id) is used to associate a particular picture to a particular view.

When combining therefore the MMCO of AVC with the usage of viewjds from MVC, the current usage of MMCOs of the prior art only allow a user to supply MMCOs according to pictures of the same view_id. That is, a picture that is being coded can only refer to other pictures of the same viewjd type (view 1 type pictures can only supply MMCO commands for other view 1 type pictures).

Having to keep track of all of these different views with the current usage of MMCO commands prevents an inefficient usage of memory management for operating a DPB.

Specifically, FIG. 4 represents the current MVC, where an additional syntax has been added in the SPS (which is used to signal cross view references). The added syntax is used to indicate the cross-view references to be used for anchor (i.e. I pictures) and non-anchor pictures in the way described below.

Please note that an anchor picture typically represents a coded picture in which all slices reference only slices with the same picture order count, i.e., only slices in other views, and not slices in the current view. Such a picture is signaled by setting the anchor_pic_flag to 1. After decoding the anchor picture, all of the following coded pictures in display order shall be capable of being decoded within using inter prediction from any other picture decoder prior to the anchor picture. If a picture in one view is an anchor picture, then all pictures within the same temporal index in other views shall also be known as anchor pictures.

The following procedure shall be conducted to place reference pictures from a view that is different from a current view into the reference prediction lists:

- For each value of T from 0 to num_multiview_ref_forJistX-1 :
  - The reconstructed picture from view reference_view_for_list_X[i] that is temporally aligned with the current picture shall be obtained and inserted into the decoded picture buffer (DPB).
  - An index to that picture shall be inserted into the next empty slot in RefPicListX.

In this specified implementation, the MMCO commands are associated directly with the individual views only and cannot mark pictures in other views. As a direct consequence, cross-view reference pictures could stay in the DPB longer than
necessary (as indicated above) because such a picture can only be marked "unused for reference" by a picture in its own view later in the bitstream. For example, referring to FIG. 1, pictures for T0-T11 in view S0 are only needed for views S1, S2, and thereafter are marked as used for reference. Hence, the DPB storing such pictures would require a large storage area. Hence, the only way to clear the DPB, without consideration of what view a picture is associated with, either a picture associated with the start of a new group of pictures (GOP) or an instantaneous decoding refresh (IDR) picture would indicate to completely clear the DPB of reference pictures.

The present invention therefore proposes a solution to the DPB problem by providing MMCO which can be applied cross-views, which means that when a picture is being coded, such a picture will contain information as to how consider reference pictures across views (views which are not the same as the picture currently being coded).

Several embodiments of the present invention are presented in view of the AVC standard where new high level syntax elements are defined and discussed, although it is to be understood that the principles also apply to other coding standards that make use of multiple view pictures.

In one embodiment presented in FIG. 5, a new syntax element `dec_ref_pic_marking_mvc_extension()` which is to be used to mark pictures across views. This function is called from the corresponding `slice_header_mvc_extension()` function presented in FIG. 6 which represents the picture header syntax of a picture being coded (specifically this command is adapted from the slice header shown in AVC).

Since this new syntax is only used to mark picture that are in a view other than the current view, it must also be considered to provide the option of allowing the system to mark pictures within the same view. The marking of pictures within the same view is enabled by calling the AVC compatible function `dec_ref_pic_marking` (see FIG. 5) after the new marking process. It is noted that such a function may be called either before or after the activation of the MVC based marking.

An additional constraint is placed on the AVC based `dec_ref_pic_marking()` syntax for MVC, where the AVC syntax only assumes a single view because multiview systems were not addressed initially in the AVC standard. Hence, the AVC syntax has to be applied only to the view to which a current picture being coded belongs.
Referring back to FIG. 5, several additional syntaxes are added which are defined below:

- **mvc_adaptive_ref_pic_marking_mode_flag**: This flag is used to select between the reference marking mode of a picture that is currently being coded. The flag at "0" represents a sliding window reference picture marking mode, where short term reference pictures are assigned a FIFO basis in the DPB. The flag at "1" represents an adaptive reference picture marking mode, where elements may be provided to mark reference pictures in a view other than the view associated with a picture currently being coded. Such statuses for references pictures in the other views include "unused for reference" and "long-term frame indices".

  The flag shall be equal to 1 when the number of pictures (frames, complementary field pairs, and non-field fields) that are currently marked as "used for long-term reference" is equal to \( \text{Max} (\text{Num_ref_frames}, 1) \).

- **memory_management_control_operation**: This operation specifies a control operation (MMCO) to be applied to affect the reference picture marking operation by a coder. The memory_management_control_operation syntax element is followed by data necessary for the operation specified by the value of the control operation. The values and control operations associated with the MMCOs for multiview are specified below in Table 2. The memory_management_control_operation syntax elements are processed by coding processing in the order in which such commands appear in a picture header (e.g., slice header), and the semantic constraints expressed for each MMCO applies in the specific position in which that individual MMCO is processed.
<table>
<thead>
<tr>
<th>memory_management_control_operation</th>
<th>Memory Management Control Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>End memory_management_control_operation syntax element loop</td>
</tr>
<tr>
<td>1</td>
<td>Mark a short-term reference picture in a view other than itself as &quot;unused for reference&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Mark a long-term reference picture in a view other than itself as &quot;unused for reference&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Mark a short-term reference picture in a view other than itself as &quot;used for long-term reference&quot; and assign a long-term frame index to it</td>
</tr>
<tr>
<td>4</td>
<td>Specify the maximum long-term frame index and mark all long-term reference pictures having long-term frame indices greater than the maximum value as &quot;unused for reference&quot;</td>
</tr>
<tr>
<td>5</td>
<td>Mark all reference pictures in a view as &quot;unused for reference&quot;</td>
</tr>
<tr>
<td>6</td>
<td>Mark all reference pictures in all views other than itself as &quot;unused for reference&quot; and set the MaxLongTermFrameIdx variable to &quot;no long-term frame indices&quot;</td>
</tr>
<tr>
<td>7</td>
<td>Mark a long-term reference picture in a view other than itself as &quot;used for short-term reference&quot;</td>
</tr>
</tbody>
</table>

**TABLE 2**

memory_management_control_operation shall not be equal to 1 in a picture header (e.g., slice header) unless the specified reference picture is marked as "used for short-term reference" when the memory_management_control_operation is processed by a coding process.

memory_management_control_operation shall not be equal to 2 in a slice header unless the specified long-term picture number refers to a reference picture that is marked as "used for long-term reference" when the memory_management_control_operation is processed by the decoding process.
memory_management_control_operation shall not be equal to 3 in a slice header unless the specified reference picture is marked as "used for short-term reference" when the memory_management_control_operation is processed by the decoding process.

memory_management_control_operation shall not be equal to 3, 5 or 6 if the value of the variable MaxLongTermFrameldx is equal to "no long-term frame indices" when the memory_management_control_operation is processed by the decoding process.

Not more than one memory_management_control_operation equal to 4 shall be present in a picture header.

Not more than one memory_management_control_operation equal to 5 shall be present in a picture header.

Not more than one memory_management_control_operation equal to 6 shall be present in a picture header.

When decoding a field and a memory_management_control_operation command equal to 3 is present that assigns a long-term frame index to a field that is part of a short-term reference frame or part of a short-term complementary reference field pair, another memory_management_control_operation command to assign the same long-term frame index to the other field of the same frame or complementary reference field pair shall be present in the same decoded reference picture marking syntax structure.

Note, the above requirement must be fulfilled even when the field referred to by the MMCO is equal to 3 and is subsequently marked as "unused for reference" as for example when a MMCO is equal to 2 in a picture header, which causes a field to be marked as "unused for reference".

When the first field (in decoding order) of a complementary reference field pair includes a long_term_reference_flag equal to 1 or a memory_management_control_operation command equal to 6, the decoded reference picture marking syntax structure for the other field of the complementary reference field pair shall contain a memory_management_control_operation command equal to 6 that assigns the same long-term frame index to the other field.
Note, the above requirement must be fulfilled even when the complementary reference field pair is subsequently marked as "unused for reference" as for example when a MMCO is equal to 2 in a picture header of a second field that causes which causes a field to be marked as "unused for reference".

`difference_of_view_id` is used to derive the viewjd to which the current `memory_management_control_operation` is applicable.

`difference_of_pic_nums` is used to assign a long-term frame index to a short-term reference picture in a view other than itself or to mark a short-term reference picture in a view other than itself as "unused for reference". When the associated `memory_management_control_operation` is processed by the decoding process and the resulting picture number derived from `difference_of_pic_nums` shall be a picture number assigned to one of the reference pictures marked as "used for reference" and not previously assigned to a long-term frame index.

The resulting picture number is to be constrained as follows:

- If `field_pic_flag` is equal to 0, the resulting picture number shall be one of the set of picture numbers assigned to reference frames or complementary reference field pairs.

Note, when the `field_pic_flag` is equal to 0, the resulting picture number must be a picture number assigned to a complementary reference field pair in which both fields are marked as "used for reference" or a frame in which both fields are marked as "used for reference". In particular, when `field_pic_flag` is equal to 0, the marking of a non-paired field or a frame in which a single field is marked as "used for reference" cannot be affected by a `memory_management_control_operation` equal to 1.

- Otherwise, (field_pic_flag is equal to 1), the resulting picture number shall be one of the set of picture numbers assigned to reference fields.

`long_term_frame_idx` is used (with `memory_management_control_operation` equal to 2) to assign a long-term frame index to a picture with viewjd different from the current picture's viewjd. When the associated `memory_management_control_operation` is processed by the decoding process, the
value of long_term_frame_idx shall be in the range of 0 to MaxLongTermFrameldx, inclusive.

The syntax **difference_of_pic_nums**, allows to select pictures with a picNum that is larger that the picNum of the current picture. This makes marking more efficient.

The application of the different functions shown in TABLE 2 are shown below:

When the MMCO equals 1, this represents that a short-term reference picture is defined as being "unused for reference". Therefore, let picNumX be specified by:

\[ \text{picNumX} = \text{CurrPicNum} - (\text{difference_of_pic_nums}) \]

Let viewld be specified by

\[ \text{viewldX} = \text{CurrViewld} - (\text{difference_of_view_id}) \]

Depending on field_pic_flag the value of picNumX is used to mark a short-term reference picture as "unused for reference" as follows.

- If field_pic_flag is equal to 0, the short-term reference frame or short-term complementary reference field pair specified by picNumX in the view specified by viewldX and both of its fields are marked as "unused for reference".

- Otherwise (field_pic_flag is equal to 1), the short-term reference field specified by picNumX in the view specified by viewldX is marked as "unused for reference". When that reference field is part of a reference frame or a complementary reference field pair, the frame or complementary field pair is also marked as "unused for reference", but the marking of the other field is not changed.

When the MMCO equals 2, this situation represents that a long term reference picture is being changed to be "unused for reference". Depending on field_pc_flag, the value of LongTermPicNum is used to mark a long-term reference picture as "unused for reference" as follows:

- If field_pic_flag is equal to 0, the long-term reference frame or long-term complementary reference field pair having LongTermPicNum equal to long_term_pic_num and both of its fields are marked as "unused for reference".

- Otherwise (field_pic_flag is equal to 1), the long-term reference field specified by LongTermPicNum equal to long_term_pic_num is marked as "unused for
reference”. When that reference field is part of a reference frame or a complementary reference field pair, the frame or complementary field pair is also marked as "unused for reference", but the marking of the other field is not changed.

When the MMCO equals 3, this situation represents the process of assigning a LongTermFrameldx to a short-term reference picture (making a short term reference picture a long term reference picture).

Given the syntax element difference_of_pic_nums and difference_of_view_id, the variable picNumX and viewldX are obtained as specified above. picNumX shall refer to a frame or complementary reference field pair or non-paired reference field marked as "used for short-term reference" and not marked as "non-existing" in the view specified by viewldX.

When LongTermFrameldx equal to long_term_frame_idx is already assigned to a long-term reference frame or a long-term complementary reference field pair, that frame or complementary field pair and both of its fields are marked as "unused for reference".

When LongTermFrameldx is already assigned to a non-paired reference field, and the field is not the complementary field of the picture specified by picNumX, that field is marked as "unused for reference".

Depending on field_pic_flag the value of LongTermFrameldx is used to mark a picture from "used for short-term reference" to "used for long-term reference" as follows:

- If field_pic_flag is equal to 0, the marking of the short-term reference frame or short-term complementary reference field pair specified by picNumX in the view specified by viewldX and both of its fields are changed from "used for short-term reference" to "used for long-term reference" and assigned LongTermFrameldx equal to long_term_frame_idx.

- Otherwise (field_pic_flag is equal to 1), the marking of the short-term reference field specified by picNumX in the view specified by viewldX is changed from "used for short-term reference" to "used for long-term reference" and assigned LongTermFrameldx equal to long_term_frame_idx. When the field is part of a reference frame or a complementary reference field pair, and the other field of the same reference frame or complementary reference field pair is also marked as "used
for long-term reference", the reference frame or complementary reference field pair is also marked as "used for long-term reference" and assigned LongTermFrameldx equal to long_term_frame_idx.

When the MMCO equals 4, such a situation is invoked to change the status of a reference picture from a "used for long term reference" to "unused for reference" when a LongTermFrameldx value is larger than the value associated with max_long_term_frame_idx_plus1-1.

The variable MaxLongTermFrameldx is determined as follows:

If max_long_term_frame_idx_plus1 is equal to 0, MaxLongTermFrameldx is set equal to "no long-term frame indices".

Otherwise (max_long_term_frame_idx_plus1 is greater than 0), MaxLongTermFrameldx is set equal to max_long_term_frame_idx_plus1 - 1.

Note, the memory_management_control_operation command equal to 4 can be used to mark long-term reference pictures as "unused for reference". The frequency of transmitting max_long_term_frame_idx_plus1 is not specified by this specification, however and may be selected by the designer of a coder. However, the encoder should send a memory_management_control_operation command equal to 4 upon receiving an error message, such as an intra refresh request message.

The MMCO being equal to 5 represents a situation where all of the reference pictures in a view which are specified by a viewldX (as derived above), are marked as "unused for reference". That is, this MMCO provides the coder with the function of changing all of the pictures for a particular view (without having to identify each reference picture specifically). This type of function may be invoked to change the status of all the reference pictures that are of the same view as a picture currently being coded. Similarly, this command can be invoked to change the statuses of reference pictures of a particular view which are not the same as the view associated with a picture currently being coded.

Having the MMCO being equal to 6 represents the circumstance of having all reference pictures in all views other than the view associated with a current view having their statuses changed to "unused for reference" and having the MaxLongTermFrameldx variable being set to "no long-term frame indices". The
command effectively controls the DPB to eventual clear out all of the reference pictures which are for views not associated with the view for a picture currently being coded. As noted above, the MMCO being equal to 5 is to change the status of the reference pictures associated with a particular view, while the present MMCO (being equal to 6) affects all of the reference pictures which are associated with views which are not the same as the view of a picture being coded.

The MMCO being equal to 7 represents the situation of having the status of a reference picture being changed from "long-term reference picture" to "used for short-term reference". Such a reference picture is associated with a view which is different than the view associated with a picture currently being coded.

FIG. 7 presents an alternative embodiment of the principles of the present invention where a syntax element difference_of_pics_nums_minus1 is presented (instead of using the syntax element difference_of_pic_nums). The implication of such a change is for the situation when one cannot select a picture with a picNumX that is larger than the picNumX of the current picture. The MMCOs associated with this embodiment operate the same as identified above (in Table 2).

FIG. 8 presents an alternative embodiment of the principles of the present invention where the picture header (such as a slice header) commands are modified as to call out a syntax element command slice_header_mvc_extension() during the time when a picture is being coded in an AVC operation. That is, the MMCO commands for multiviews may take place in this embodiment during the AVC encoding (where all reference pictures different views may be considered) instead of what was presented previously were references pictures of a view different than a picture being considered would be considered.

FIG. 9 discloses the composition of the syntax element command dec_ref_pic_marking_mvc_extension(). This new syntax (as called in the picture header/slice header as shown in FIG. 8) is used to mark pictures that are in a view other than the current view by setting the appropriate difference_of_view_id syntax. In order to allow the reference pictures associated with view associated with a picture being coded to have their memory statuses changed, the difference_of_view_id syntax is set to 0. This proposed syntax element will replace the existing AVC function for DPB management using the command def_ref_pic_marking. Various syntax
elements associated with dec_ref_pic_marking_mvc_extension() are explained below.

**mvc_adaptive_ref_pic_marking_mode_flag** which is used to select between the reference marking mode of a picture that is currently being coded. The flag at "0" represents a sliding window reference picture marking mode, where short term reference pictures are assigned a FIFO basis in the DPB. The flag at "1" represents an adaptive reference picture marking mode, where elements may be provided to mark reference pictures as being "unused for reference" and to assign "long-term frame indices".

**memory_management_control_operation** (MMCO) specifies a control operation to be applied to affect the reference picture marking. The MMCO syntax element is followed by data necessary for the operation specified by the value of MMCO. The values and control operations associated with called MMCO are shown in Table 3 (below). The MMCO syntax elements in the present embodiment are processed by the decoding process in the order in which they appear in the slice header, and the semantics constraints expressed for each

**memory_management_control_operation** apply at the specific position in that order at which that individual MMCO is processed.

For interpretation of **memory_management_control_operation**, the term reference picture is interpreted as follows.

- If the current picture is a frame, the term reference picture refers either to a reference frame or a complementary reference field pair.

- Otherwise (the current picture is a field), the term reference picture refers either to a reference field or a field of a reference frame.

**memory_management_control_operation** shall not be equal to 1 in a slice header unless the specified reference picture is marked as "used for short-term reference" when the **memory_management_control_operation** is processed by the decoding process.
memory_management_control_operation shall not be equal to 2 in a slice header unless the specified long-term picture number refers to a reference picture that is marked as "used for long-term reference" when the memory_management_control_operation is processed by the decoding process.

memory_management_control_operation shall not be equal to 3 in a slice header unless the specified reference picture is marked as "used for short-term reference" when the memory_management_control_operation is processed by the decoding process.

memory_management_control_operation shall not be equal to 3, 5 or 6 if the value of the variable MaxLongTermFrameIdx is equal to "no long-term frame indices" when the memory_management_control_operation is processed by the decoding process.

When decoding a field and a memory_management_control_operation command equal to 3 is present that assigns a long-term frame index to a field that is part of a short-term reference frame or part of a short-term complementary reference field pair, another memory_management_control_operation command to assign the same long-term frame index to the other field of the same frame or complementary reference field pair shall be present in the same decoded reference picture marking syntax structure.

Note, the above requirement must be fulfilled even when the field referred to by the memory_management_control_operation equal to 3 is subsequently marked as "unused for reference" (for example when a memory_management_control_operation equal to 2 is present in the same slice header that causes the field to be marked as "unused for reference").

When the first field (in decoding order) of a complementary reference field pair includes a long_term_reference_flag equal to 1 or a memory_management_control_operation command equal to 6, the decoded reference picture marking syntax structure for the other field of the complementary reference field pair shall contain a memory_management_control_operation command equal to 6 that assigns the same long-term frame index to the other field.

Note, that the above requirement must be fulfilled even when the first field of the complementary reference field pair is subsequently marked as "unused for reference" (for example, when a memory_management_control_operation equal to 2
is present in the slice header of the second field that causes the first field to be marked as "unused for reference").

difference_of_view_id is used to derive the view_id to which the current memory_management_control_operation is applicable.

difference_of_pic_nums is used to assign a long-term frame index to a short-term reference picture or to mark a short-term reference picture as "unused for reference". When the associated memory_management_control_operation is processed by the decoding process, the resulting picture number derived from difference_of_pic_nums shall be a picture number assigned to one of the reference pictures marked as "used for reference" and not previously assigned to a long-term frame index. The resulting picture number is constrained as follows:

- If field_pic_flag is equal to 0, the resulting picture number shall be one of the set of picture numbers assigned to reference frames or complementary reference field pairs. Note, when field_pic_flag is equal to 0, the resulting picture number must be a picture number assigned to a complementary reference field pair in which both fields are marked as "used for reference" or a frame in which both fields are marked as "used for reference". In particular, when field_pic_flag is equal to 0, the marking of a non-paired field or a frame in which a single field is marked as "used for reference" cannot be affected by a memory_management_control_operation equal to 1.

- Otherwise (field_pic_flag is equal to 1), the resulting picture number shall be one of the set of picture numbers assigned to reference fields.

long_term_pic_num is used (with memory_management_control_operation equal to 2) to mark a long-term reference picture as "unused for reference". When the associated memory_management_control_operation is processed by the decoding process, long_term_pic_num shall be equal to a long-term picture number assigned to one of the reference pictures that is currently marked as "used for long-term reference".

The resulting long-term picture number is constrained as follows:

- If field_pic_flag is equal to 0, the resulting long-term picture number shall be one of the set of long-term picture numbers assigned to reference frames or complementary reference field pairs.
Note, when field_pic_flag is equal to 0, the resulting long-term picture number must be a long-term picture number assigned to a complementary reference field pair in which both fields are marked as "used for reference" or a frame in which both fields are marked as "used for reference". In particular, when field_pic_flag is equal to 0, the marking of a non-paired field or a frame in which a single field is marked as "used for reference" cannot be affected by a memory_management_control_operation equal to 2.

- Otherwise (field_pic_flag is equal to 1), the resulting long-term picture number shall be one of the set of long-term picture numbers assigned to reference fields.

long_term_frame_idx is used (with memory_management_control_operation equal to 3 or 6) to assign a long-term frame index to a picture. When the associated memory_management_control_operation is processed by the decoding process, the value of long_term_frame_idx shall be in the range of 0 to MaxLongTermFrameIdx, inclusive.

The syntax difference_of_pic_nums allows to select pictures with a picNum that is larger than the picNum of the current picture. This makes marking more efficient.

The decoded reference picture marking process is described below for the different MMCO commands:

Let viewld be specified by

viewldX = CurrViewld - ( difference_of_view_id ).

All the MMCO commands (as shown in Table 3 below) are applied to the viewld derived above as viewldX.
When the MMCO is equal to 0, the marking of picture headers (slice headers, for example), is ended.

When the MMCO is equal to 1, a particular reference frame will have the status associated with it changed from being a "short-term reference picture" to being "unused for reference".

Let $\text{picNumX}$ be specified by

$$\text{picNumX} = \text{CurrPicNum} - (\text{difference_of_pic_nums}).$$

Additionally, depending on the field_pic_flag, the value of $\text{picNumX}$ is used to mark a short-term reference picture as "unused for reference" as follows:
- If field_pic_flag is equal to 0, the short-term reference frame or short-term complementary reference field pair specified by picNumX in the view specified by viewldX and both of its fields are marked as "unused for reference".

- Otherwise (field_pic_flag is equal to 1), the short-term reference field specified by picNumX in the view specified by viewldX is marked as "unused for reference". When that reference field is part of a reference frame or a complementary reference field pair, the frame or complementary field pair is also marked as "unused for reference", but the marking of the other field is not changed.

When the MMCO is equal to 2, a particular reference picture will have the status associated with it changed from being "long-term reference picture" to being "unused for reference". Depending on field_pic_flag the value of LongTermPicNum is used to mark a long-term reference picture as "unused for reference" as follows.

- If field_pic_flag is equal to 0, the long-term reference frame or long-term complementary reference field pair having LongTermPicNum equal to long_term_pic_num and both of its fields are marked as "unused for reference".

- Otherwise (field_pic_flag is equal to 1), the long-term reference field specified by LongTermPicNum equal to long_term_pic_num is marked as "unused for reference". When that reference field is part of a reference frame or a complementary reference field pair, the frame or complementary field pair is also marked as "unused for reference", but the marking of the other field is not changed.

When the MMCO is equal to 3, a particular reference frame is assigned to a LongTermFrameldX assigning a short-term reference picture to a long-term reference picture. Given the syntax element difference_of_pic_nums and difference_of_view_id, the variable picNumX and viewldX are obtained as specified above. picNumX shall refer to a frame or complementary reference field pair or non-paired reference field marked as "used for short-term reference" and not marked as "non-existing" in the view specified by viewldX.

When LongTermFrameldX equal to long_term_framejdx is already assigned to a long-term reference frame or a long-term complementary reference field pair, that frame or complementary field pair and both of its fields are marked as "unused for reference". When LongTermFrameldX is already assigned to a non-paired reference field, and the field is not the complementary field of the picture specified by picNumX,
that field is marked as "unused for reference". Depending on field_pic_flag the value of LongTermFrameIdx is used to mark a picture from "used for short-term reference" to "used for long-term reference" as follows.

- If field_pic_flag is equal to 0, the marking of the short-term reference frame or short-term complementary reference field pair specified by picNumX in the view specified by viewIdX and both of its fields are changed from "used for short-term reference" to "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx.

- Otherwise (field_pic_flag is equal to 1), the marking of the short-term reference field specified by picNumX in the view specified by viewIdX is changed from "used for short-term reference" to "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx. When the field is part of a reference frame or a complementary reference field pair, and the other field of the same reference frame or complementary reference field pair is also marked as "used for long-term reference", the reference frame or complementary reference field pair is also marked as "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx.

When the MMCO is equal to 4, a maximum long-term frame index value is specified, whereby all of reference frames which are identified as long term reference frames and have frame indices greater than the maximum value are categorized as being "unused for reference". Specifically (within the nomenclature of the function call), all pictures for which LongTermFrameIdx is greater than max_long_term_frame_idx_plus1 - 1 and that are marked as "used for long-term reference" are marked as "unused for reference".

The variable MaxLongTermFrameIdx is derived as follows:

- If max_long_term_frame_idx_plus1 is equal to 0, MaxLongTermFrameIdx is set equal to "no long-term frame indices".

- Otherwise (max_long_term_frame_idx_plus1 is greater than 0), MaxLongTermFrameIdx is set equal to max_long_term_frame_idx_plus1 - 1. It is noted that the present MMCO can be used to mark long-term reference pictures as "unused for reference". The frequency of transmitting
max_long_term_frame_idx_plus1 should be decided by the designer of the coder which complies with this invention. However, the coder should send a memory_management_control_operation command equal to 4 upon receiving an error message, such as an intra refresh request message.

When the MMCO is equal to 5, the coder will mark all reference pictures in a view specified by viewIDx (a specific view) as "unused for reference" and set the MaxLongTermFrameldx variable equal to "no long-term frame indices". This means, the reference pictures identified with a particular view are set to being "unused for reference", where before such reference pictures were marked as being "long-term".

When the MMCO is equal to 6, the current picture being coded is marked as being "used for long-term" and a long-term frame index is assigned to the picture. Specifically, when a variable LongTermFrameldx equal to long_term_frame_idx is already assigned to a long-term reference frame or a long-term complementary reference field pair, that frame or complementary field pair and both of its fields are marked as "unused for reference". When LongTermFrameldx is already assigned to a non-paired reference field, and the field is not the complementary field of the current picture, that field is marked as "unused for reference".

The current picture is marked as "used for long-term reference" and assigned LongTermFrameldx equal to long_term_frame_idx.

When field_pic_flag is equal to 0, both its fields are also marked as "used for long-term reference" and assigned LongTermFrameldx equal to long_term_frame_idx.

When field_pic_flag is equal to 1 and the current picture is the second field (in decoding order) of a complementary reference field pair, and the first field of the complementary reference field pair is also currently marked as "used for long-term reference", the complementary reference field pair is also marked as "used for long-term reference" and assigned LongTermFrameldx equal to long_term_frame_idx.

After marking the current decoded reference picture, the total number of frames with at least one field marked as "used for reference", plus the number of complementary field pairs with at least one field marked as "used for reference", plus the number of non-paired fields marked as "used for reference" shall not be greater
than Max( num_ref_frames, 1 ). Please note, that under some circumstances, the
above statement may impose a constraint on the order in which a
memory_management_control_operation syntax element equal to 6 can appear in the
decoded reference picture marking syntax relative to a
memory_management_control_operation syntax element equal to 1, 2, or 4.

When the MMCO is equal to 7, a long term reference picture identified by the
long_term_pic_num in a view identified by viewldX is marked as "used for short-term reference". This means, that a particular frame (identified by a pic number) and a
specified view, will have its status changed from being a "long-term reference picture"
to being a "short-term reference picture".

An alternative embodiment of the present invention is disclosed in accordance
with the syntax element shown in FIG. 10, where a difference_of_pic_nums or
difference_of_pic_nums_minus1 is transmitted based on the value of the
difference_of_view_id. This solution is proposed when using MMCO commands
just for the temporal case instead of using the difference_of_pic_nums syntax element
described for FIG 8.

FIG. 11 discloses a block diagram 1100 of a generalized reference picture
marking method in accordance with principles of the present invention.

Step 1105 represents the generalized concept of coding a picture (where the
coding operation typically is encoding the picture from a group of video based moving
pictures). This operation may represent the decoding of a coded picture, too.

In the present step however, the picture being coded is associated with a
particular view from a plurality of views which are used in a multi-view video coding
system. The preferred embodiment makes use of the principles disclosed in the MVC
video standard within the context of an AVC coder, but it is to be appreciated that other
multiview video standards may be used. Importantly, the coded picture will
associated with a picture ID number and a view ID number. The picture ID represents
the coded pictures number in within a sequence of coded pictures. The coded picture
will also have a view ID number which corresponds to the view in which the coded
picture is associated with (between 1 to n, n=the total number of views). For example,
a coded picture that is associated with a view "2" will be known as having a view ID
number of "2".
In step 1110, the coded picture is stored in a memory device (such as the DPB) and the coded picture is assigned a memory status. The coded picture is stored so that the picture may be used as a reference picture. As described above, a coded picture may have at least three different memory statuses associated with:

"Long-Term Reference Picture" represents when a coded picture is supposed to be stored as a reference picture. The coded picture which is designated as a long-term reference picture is assigned an index number (in a long-term picture index). This picture is supposed to be retained for the time being, so that such a picture may be used as a reference picture when coding future pictures.

"Short-Term Reference Picture" represents a coded picture as is supposed to be retained for a short time, as a reference picture. In this case, the reference picture will be moved out of the memory device (DPB) when room is required.

"Unused as Reference" represents when a coded picture is not meant to be used as a reference picture. In this case, the DPB may remove the reference picture when space is required (using LIFO/FIFO), or erased from the DPB directly.

It is possible that a picture designated as being unused as reference may be directly deleted and never stored in the DPB during steps 1110 and 1115.

In step 1120, a second video picture is being coded. At this time, the video picture is marked with a memory management command operation (as described with other embodiments), where the MMCO affects the reference picture stored in the memory device used for storing reference pictures such as the DPB.

As described earlier, different MMCOs may be employed to decide how to change the memory status associated with a reference picture. Such a change may be made based upon whether the picture currently being coded has a same/different view from the stored reference picture (for example, all the pictures with the same view ID of are changed at the same time, as this represents a global change). The change of the memory status associated with a picture may be done directly, where a particular stored reference picture is identified and the MMCO specifies that the memory status of the picture will change (for example, changing between the memory statuses of "long-term reference", "short-term reference", and "unused as reference", this representing a local change).

Additionally, as described above, the various implementations of the invention allow for situations where the view ID of the picture currently being coded will affect
whether pictures can only be changed for reference pictures with a different view ID
(across different views) or for reference pictures which have the same view ID as the
picture being coded (for the specific view).

Step 1125 is the implementation of the MCCO command, where the memory
storage device storage the reference picture implements the memory status
associated with the reference picture. These types of operations are also described
above.

These and other features and advantages of the present principles may be
readily ascertained by one of ordinary skill in the pertinent art based on the teachings
herein. It is to be understood that the teachings of the present principles may be
implemented in various forms of hardware, software, firmware, special purpose
processors, or combinations thereof.

Most preferably, the teachings of the present principles are implemented as a
combination of hardware and software. Moreover, the software may be implemented
as an application program tangibly embodied on a program storage unit. The
application program may be uploaded to, and executed by, a machine comprising any
suitable architecture. Preferably, the machine is implemented on a computer platform
having hardware such as one or more central processing units ("CPU"), a random
access memory ("RAM"), and input/output ("I/O") interfaces. The computer platform
may also include an operating system and microinstruction code. The various
processes and functions described herein may be either part of the microinstruction
code or part of the application program, or any combination thereof, which may be
executed by a CPU. In addition, various other peripheral units may be connected to
the computer platform such as an additional data storage unit and a printing unit.

It is to be further understood that, because some of the constituent system
components and methods depicted in the accompanying drawings are preferably
implemented in software, the actual connections between the system components or
the process function blocks may differ depending upon the manner in which the
present principles are programmed. Given the teachings herein, one of ordinary skill
in the pertinent art will be able to contemplate these and similar implementations or
configurations of the present principles.

Although the illustrative embodiments have been described herein with
reference to the accompanying drawings, it is to be understood that the present
principles is not limited to those precise embodiments, and that various changes and modifications may be effected therein by one of ordinary skill in the pertinent art without departing from the scope or spirit of the present principles. All such changes and modifications are intended to be included within the scope of the present principles as set forth in the appended claims.
CLAIMS:

1. A method for memory management of a reference picture used multiview video coding comprising the steps of:
   - storing a reference picture in a memory (1110), where the reference picture is associated with a memory status and a view;
   - coding a video picture with information (1120) which affects the memory status of said stored reference picture.

2. The method of Claim 1, where said memory status change is implemented using a memory management operation command.

3. The method of Claim 2, wherein said coding step is implemented when the view associated with said reference picture is different than a view associated with said coded video picture.

4. The method of Claim 2, wherein said coding step is implemented when the view associated with said reference picture is the same as the video associated with the coded video picture.

5. The method of any one of Claims 2-4, wherein a second stored reference picture is of a second view which is different than said view; and
   - said memory management operation command affects all of the reference pictures associated with said view without affecting the reference pictures associated with said second view.

6. The method of any one of Claims 2-5, where said memory status associated with said stored reference frame is changed from a status selected from: long term reference frame, short term reference frame, and non-used for reference to a status selected from: long term reference frame, short term reference frame, and non-used for reference.

7. The method of any one of Claims 2-6, wherein said reference picture is initially coded using an H.264 based coding operation and said memory status change is performed during a multiview coding operation.

8. The method of any one of Claims 2-6, wherein said reference picture is coded and said change in said memory status is performed in a video coding operation that does both temporal and inter-view coding.
9. The method of any one of Claims 1-8 where a marking mode syntax element flag is called to select between the reference marking modes of said picture that is currently being coded.

10. A coding apparatus for performing any of the methods claimed in Claims 1-9.
FIG. 2
dec_ref_pic_marking() {
    if (nal_unit_type == 5 || nal_unit_type == 21) {
        /* nal_unit_type 21 is specified in Annex Error!
         
         Reference source not found. */
    
    no_output_of_prior_pics_flag 215 u(1)
    long_term_reference_flag 215 u(1)
    } else {
        adaptive_ref_pic_marking_mode_flag 215 u(1)
        if (adaptive_ref_pic_marking_mode_flag)
            do {
                memory_management_control_operation 215 ue(v)
                if (memory_management_control_operation == 1 ||
                    memory_management_control_operation == 3)
                    difference_of_pic_nums_minus] 215 ue(v)
                if (memory_management_control_operation == 2)
                    long_term_pic_num 215 ue(v)
                if (memory_management_control_operation == 3 ||
                    memory_management_control_operation == 6)
                    long_term_frame_idx 215 ue(v)
                if (memory_management_control_operation == 4)
                    max_long_term_frame_idx_plus1 215 ue(v)
            } while (memory_management_control_operation != 0)
        }
    }
}

FIG. 3
seq_parameter_set_mvc_extension() {
    num_views_minus_1
    for(i = 0; i <= num_views_minus_1; i++) {
        num_multiview_refs_for_list0[i]
        num_multiview_refs_for_list1[i]
        for(j = 0; j < num_multiview_refs_for_list0; j++) {
            anchor_reference_view_for_list_0[i][j]
            non_anchor_reference_view_for_list_0[i][j]
        }
        for(j = 0; j < num_multiview_refs_for_list1; j++) {
            anchor_reference_view_for_list_1[i][j]
            non_anchor_reference_view_for_list_1[i][j]
        }
    }
}
```c
dec_ref_pic_marking_mvc_extension() {
    mvc_adaptive_ref_pic_marking_mode_flag
    if( mvc_adaptive_ref_pic_marking_mode_flag )
        do {
            memory_management_control_operation
            if( memory_management_control_operation == 1 ||
                memory_management_control_operation == 3 ||
                memory_management_control_operation == 7 ) {
                difference_of_view_id
                difference_of_pic_nums
            }
        }
        if( memory_management_control_operation == 2 ||
            memory_management_control_operation == 7 )
            long_term_pic_num
        if( memory_management_control_operation == 4 )
            max_long_term_frame_idx_plus1
        if( memory_management_control_operation == 5 )
            difference_of_view_id_minus1
        if( memory_management_control_operation == 3 )
            long_term_frame_idx
    } while( memory_management_control_operation != 0 )
    dec_ref_pic_marking()
}
```

**FIG. 5**
FIG. 6
```c
slice_header_mvc_extension() {
  first_mb_in_slice
    2  ue(v)
  slice_type
    2  ue(v)
  pic_parameter_set_id
    2  ue(v)
  frame_num
    2  u(v)
  if (frame_mbs_only_flag) {
    field_pic_flag
      2  u(1)
    if (field_pic_flag)
      bottom_field_flag
        2  u(1)
  }
  if (nal_unit_type == 21)
    IDR_pic_id
      2  ue(v)
  if (pic_order_cnt_type == 0) {
    pic_order_cnt_lsb
      2  u(v)
    if (pic_order_present_flag && !field_pic_flag)
      delta_pic_order_cnt_bottom
        2  se(v)
  }
  if (pic_order_cnt_type == 1 && !delta_pic_order_actual_zero_flag) {
    delta_pic_order_cnt [0]
      2  se(v)
    if (pic_order_present_flag && !field_pic_flag)
      delta_pic_order_cnt [1]
        2  se(v)
  }
  if (redundant_pic_cnt_present_flag)
    redundant_pic_cnt
      2  ue(v)
  if (slice_type == EB)
    direct_spatial_mv_pred_flag
      2  u(1)
  if (slice_type == EB || slice_type == EB) {
    num_ref_idct_active_overide_flag
      2  u(1)
    if (num_ref_idct_active_overide_flag)
      num_ref_idct[0] active_minus1
        2  ue(v)
    if (slice_type == EB)
      num_ref_idct[1] active_minus1
        2  ue(v)
  }
}
```
<table>
<thead>
<tr>
<th>Function/Condition</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref_pic_list_reordering()</td>
<td>2</td>
</tr>
<tr>
<td>if((weighted_pred_flag &amp;&amp; (slice_type == EP))</td>
<td></td>
</tr>
<tr>
<td>pred_weight_table()</td>
<td>2</td>
</tr>
<tr>
<td>if(nal_ref_idc != 0)</td>
<td>2</td>
</tr>
<tr>
<td>dec_ref_pic_marking_mvc_extension()</td>
<td>2</td>
</tr>
<tr>
<td>if(entropy_coding_mode_flag &amp;&amp; slice_type != 1 &amp;&amp; slice_type != SI)</td>
<td>2</td>
</tr>
<tr>
<td>cabac_init_idc</td>
<td>ue(v)</td>
</tr>
<tr>
<td>slice_qp_delta</td>
<td>se(v)</td>
</tr>
<tr>
<td>if(slice_type == SP</td>
<td></td>
</tr>
<tr>
<td>if(slice_type == SP)</td>
<td></td>
</tr>
<tr>
<td>sp_ltor_switch_flag</td>
<td>u(1)</td>
</tr>
<tr>
<td>slice_qs_delta</td>
<td>se(v)</td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>if(deblocking_filter_control_present_flag)</td>
<td>2</td>
</tr>
<tr>
<td>disable_deblocking_filter_idc</td>
<td>ue(v)</td>
</tr>
<tr>
<td>if(disable_deblocking_filter_idc != 1)</td>
<td>2</td>
</tr>
<tr>
<td>slice_alpha_c0_offset_div2</td>
<td>se(v)</td>
</tr>
<tr>
<td>slice_beta_offset_div2</td>
<td>se(v)</td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>if(num_slice_groups_minus1 &gt; 0 &amp;&amp; slice_group_map_type &gt;= 3 &amp;&amp; slice_group_map_type &lt;= 5)</td>
<td>2</td>
</tr>
<tr>
<td>slice_group_change_cycle</td>
<td>u(v)</td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

FIG. 6B
<table>
<thead>
<tr>
<th>Function</th>
<th>C</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>dec_ref_pic_marking_mvc_extension() {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvc_adaptive_ref_pic_marking_mode_flag</td>
<td>215</td>
<td>u(1)</td>
</tr>
<tr>
<td>if( mvc_adaptive_ref_pic_marking_mode_flag )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>do {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>memory_management_control_operation</td>
<td>215</td>
<td>ue(v)</td>
</tr>
<tr>
<td>if( memory_management_control_operation == 1</td>
<td></td>
<td>memory_management_control_operation == 3 )</td>
</tr>
<tr>
<td>difference_of_view_id</td>
<td>215</td>
<td>se(v)</td>
</tr>
<tr>
<td>difference_of_pic_nums_minus1</td>
<td>215</td>
<td>se(v)</td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if( memory_management_control_operation == 2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>long_term_pic_num</td>
<td>215</td>
<td>ue(v)</td>
</tr>
<tr>
<td>if( memory_management_control_operation == 4 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_long_term_frame_idx_plus1</td>
<td>215</td>
<td>ue(v)</td>
</tr>
<tr>
<td>if( memory_management_control_operation == 5 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>difference_of_view_id</td>
<td>215</td>
<td>se(v)</td>
</tr>
<tr>
<td>if( memory_management_control_operation == 3 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>long_term_frame_idx</td>
<td>215</td>
<td>ue(v)</td>
</tr>
<tr>
<td>} while( memory_management_control_operation != 0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dec_ref_pic_marking( )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 7
slice_header_mvc_extension( ) {
  first_mb_in_slice
  slice_type
  pic_parameter_set_id
  frame_num
  if( frame_mbs_only_flag ) {
    field_pic_flag
    if( field_pic_flag )
      bottom_field_flag
  }
  if( nal_unit_type == 21 )
    idr_pic_id
  if( pic_order_cnt_type == 0 ) {
    pic_order_cnt_lsb
    if( pic_order_present_flag && !field_pic_flag )
      delta_pic_order_cnt_bottom
  }
  if( pic_order_cnt_type == 1 && !delta_pic_order_always_zero_flag ) {
    delta_pic_order_cnt [ 0 ]
    if( pic_order_present_flag && !field_pic_flag )
      delta_pic_order_cnt [ 1 ]
  }
  if( redundant_pic_cnt_present_flag )
    redundant_pic_cnt
  if( slice_type == EB )
    direct_spatial_mv_pred_flag
  if( slice_type == EP && slice_type == EB ) {
    num_ref_idx_active_override_flag
    if( num_ref_idx_active_override_flag ) {
      num_ref_idx_10_active_minus1
      if( slice_type == EB )
        num_ref_idx_11_active_minus1
    }
  }
}
FIG. 8B
dec_ref_pic_marking_mvc_extension() { 
  C Descriptor
  if(nal_unit_type == 5 || nal_unit_type == 21) {
    no_output_of_prior_pics_flag = 215 u(1)
    long_term_reference_flag = 215 u(1)
  } else {
    mvc_adaptive_ref_pic_marking_mode_flag = 215 u(1)
    if( mvc_adaptive_ref_pic_marking_mode_flag )
      do {
        if( memory_management_control_operation != 0 ||
            memory_management_control_operation != 6 )
          difference_of_view_id = 215 se(v)
        memory_management_control_operation = 215 ue(v)
        if( memory_management_control_operation != 1 ||
            memory_management_control_operation != 3 ) {
          difference_of_pic_nums = 215 se(v)
        }
        if( memory_management_control_operation != 2 ||
            memory_management_control_operation != 7 )
          long_term_pic_num = 215 ue(v)
        if( memory_management_control_operation != 3 ||
            memory_management_control_operation != 6 ) {
          long_term_frame_idx = 215 ue(v)
          if( memory_management_control_operation == 4 )
            max_long_term_frame_idx_plus1 = 215 ue(v)
        } while( memory_management_control_operation != 0 )
      }
  } 
} 

FIG. 9
dec_ref_pic_marking_mvc_extension() {
  mvc_adaptive_ref_pic_marking_mode_flag = 215
  if (mvc_adaptive_ref_pic_marking_mode_flag)
    do {
      memory_management_control_operation = 215
      if (memory_management_control_operation == 1 ||
          memory_management_control_operation == 3) {
        difference_of_view_id = 215
        difference_of_pic_nums_minus1 = 215
      }
      if (memory_management_control_operation == 2)
        long_term_num = 215
      if (memory_management_control_operation == 4)
        max_long_term_frame_idx_plus1 = 215
      if (memory_management_control_operation == 5)
        difference_of_view_id = 215
      if (memory_management_control_operation == 3)
        long_term_frame_idx = 215
      } while (memory_management_control_operation != 0)
  dec_ref_pic_marking()
}
CODING A FIRST PICTURE FROM A GROUP OF MOVING PICTURE TO BE CODED, WHERE SAID CODED PICTURE IS ASSOCIATED WITH A VIEW AND A PICTURE NUMBER.

STORING SAID CODED FIRST PICTURE IN A MEMORY.

ASSOCIATING A REFERENCE PICTURE MEMORY STATUS WITH SAID CODED FIRST PICTURE.

CODING A SECOND PICTURE FROM A GROUP OF MOVING PICTURES TO BE CODED, WHERE SAID CODING OPERATION MARKS THE SECOND PICTURE WITH A MEMORY MANAGEMENT COMMAND AFFECTING THE REFERENCE PICTURE MEMORY STATUS ASSOCIATED WITH SAID CODED FIRST PICTURE.

OPERATING THE MEMORY THAT IS STORING SAID CODED FIRST PICTURE IN VIEW OF SAID MEMORY MANAGEMENT COMMAND.

FIG. 11