Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2009, Article ID 786015, 13 pages
doi:10.1155/2009/786015

Review Article

The Emerging MVC Standard for 3D Video Services

Ying Chen,[1] Ye-Kui Wang,[2] Kemal Ugur,[2] Miska M. Hannuksela,[2] Jani Lainema,[2] and Moncef Gabbouj[1]

[1] *Department of Signal Processing, Tampere University of Technology, 33720 Tampere, Finland*
[2] *Nokia Research Center, Visiokatu 1, 33720 Tampere, Finland*

Correspondence should be addressed to Ying Chen, ying.chen@tut.fi

Received 1 October 2007; Revised 7 February 2008; Accepted 5 March 2008

Recommended by Aljoscha Smolic

Multiview video has gained a wide interest recently. The huge amount of data needed to be processed by multiview applications is a heavy burden for both transmission and decoding. The joint video team has recently devoted part of its effort to extend the widely deployed H.264/AVC standard to handle multiview video coding (MVC). The MVC extension of H.264/AVC includes a number of new techniques for improved coding efficiency, reduced decoding complexity, and new functionalities for multiview operations. MVC takes advantage of some of the interfaces and transport mechanisms introduced for the scalable video coding (SVC) extension of H.264/AVC, but the system level integration of MVC is conceptually more challenging as the decoder output may contain more than one view and can consist of any combination of the views with any temporal level. The generation of all the output views also requires careful consideration and control of the available decoder resources. In this paper, multiview applications and solutions to support generic multiview as well as 3D services are introduced. The proposed solutions, which have been adopted to the draft MVC specification, cover a wide range of requirements for 3D video related to interface, transport of the MVC bitstreams, and MVC decoder resource management. The features that have been introduced in MVC to support these solutions include marking of reference pictures, supporting for efficient view switching, structuring of the bitstream, signalling of view scalability supplemental enhancement information (SEI) and parallel decoding SEI.

1. Introduction

Three-dimensional video has gained significant interest recently. Furthermore, with the advances in acquisition and display technologies, 3D video is becoming a reality in consumer domain with different application opportunities. Given a certain maturity of capture and display technologies and with the help of multiview video coding (MVC) techniques, a number of different envisioned 3D video applications are getting feasible [1]. 3D video applications can be grouped under three categories: free-viewpoint video, 3D TV, and immersive teleconferencing. The requirements of these applications are quite different and each category has its own challenges to be addressed.

1.1. Application Scenarios. To illustrate these challenges, consider Figure 1, where the end-to-end architecture of different applications is shown. In this illustration, a multiview video is first captured and then encoded by a multiview video coding (MVC) encoder. A server transmits the coded bitstream(s) to different clients with different capabilities, possibly through media gateways. The media gateway is an intelligent device, also referred to as a media-aware network element (MANE), which is in the signaling context and may manipulate the incoming video packets (rather than simply forward packets). At the final stage, coded video is decoded and rendered with different means according to the application scenario and capabilities of the receiver. To provide smoothly immersive experience when a user adjusting its viewing position, view synthesis [2, 3] may be required at the client to generate "virtual" views of a real-world scene. However, till now, this process is out of the scope of any existing coding standard.

In free-viewpoint video, the viewer can interactively choose his/her viewpoint in 3D space to observe a real-world scene from preferred perspectives [4]. It provides realistic impressions with interactivity, that is, the viewer can navigate freely in the scene within a certain range, and

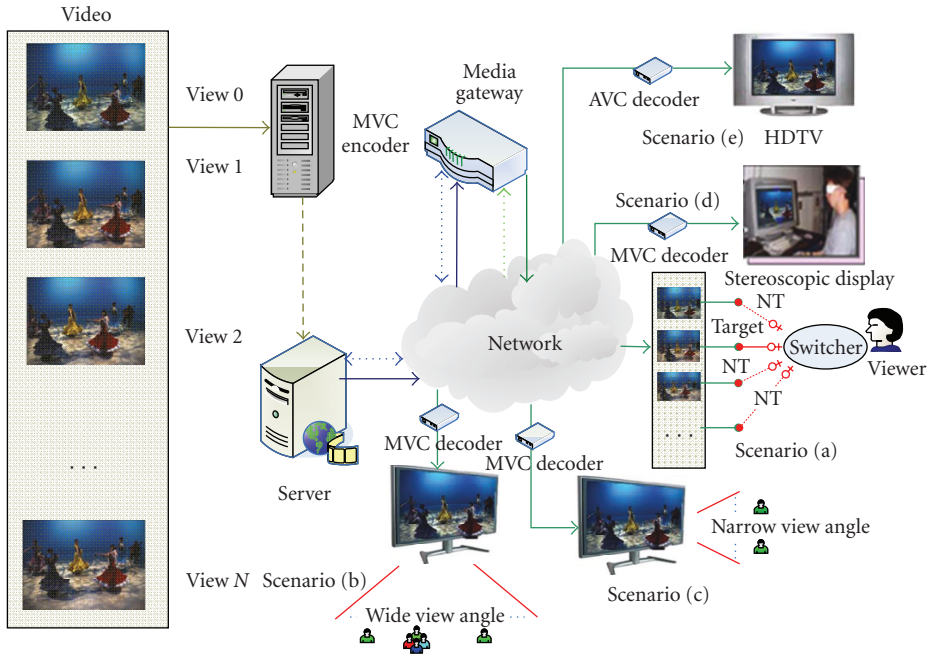


Figure 1: MVC system architecture.

analyze the 3D scene from different viewing angles. Such a video communication system has been reported in [5]. Unlike holography, which generates 3D representation and requires changing of the relative geometry position of a viewer to switch view point, this scenario is actually realized by switching between rendered view(s) using interface such as remote controller. In case the desired viewpoint is not available, interpolating a virtual view from other available views can be employed. Scenario (a), in Figure 1, illustrates this application, where there exist several candidate views for the viewer, and one of them is selected as the target view that is displayed (views that are not targeted and thus are not outputted are denoted as "NT" for simplicity in Figure 1). In this scenario, not all the candidate views are required to be decoded, thus the decoder can focus its resources only on decoding of the target view. For this purpose, the target view needs to be efficiently extracted from the bitstream and thus only the packets that are required for successfully decoding the desired views are transmitted. To enable navigation in a scene, important functionality to be achieved by the system is efficient switching between different views.

3D TV refers to the extension of traditional 2D TV displays-to-displays capable of 3D rendering. In this application, more than one view is decoded and displayed simultaneously [6]. A simple 3D TV application can be realized by stereoscopic video. Stereoscopic display can be achieved by using data glasses or other means. However, it is nicer for the user to get the 3D feeling directly through 3D appliances with added feature of rendering binocular depth cures [7], which can be realized by autostereoscopic displays. Advanced autostereoscopic displays can support head-motion parallax, by decoding and displaying multiple views from different viewpoints simultaneously. That is, a viewer without extra facilities like data glasses can move to different geometry angle ranges, each of which contains typically two views rendered and shed by 3D displays. 3D TV displays are discussed in [8]. The viewer then can experience a slightly different scene by moving his/her head (for example, user may look what is behind a certain object in the scene). In this scenario, multiple views need to be decoded simultaneously; therefore parallel processing of different views is very important to realize this application. In addition, displaying multiple views is important also to realize wide viewing angle as shown in Figuer 1(b). This scenario is also referred to as autostereoscopic 3D TV for multiple viewers [7]. However, if the decoder capability is limited or the transmission bandwidth decreases, the client at a receiver may simply decode and render just a subset of the views but still provide 3D display with a narrow view angle, as shown in Figure 1(c). The media gateway plays an important role to provide the adaptation functionality to support this use case. Such 3D TV broadcast or multicast system must then support flexible stream adaptation. Stream adaptation can be achieved at the server or media gateway, where only the sub-bitstreams, with less bandwidth and desired by the client are transmitted and other packets are discarded. After bitstream extraction, the sub-bitstream must be decodable for by MVC decoders.

Free-viewpoint video focuses on its functionality in free navigation while 3D TV emphasizes on 3D experience. In immersive teleconference, both interactivity and virtual reality may be preferred by the participants and thus free viewpoint or 3DTV style can be both supported. In the immersive teleconferencing, where there is interactivity among viewers, immersiveness can be achieved either in

a free-viewpoint video or 3D TV manner. So, the problems or requirements in free-viewpoint video or 3D TV are still existing and valid.

Typically, two mechanisms can make people perceptually feel immersed in a 3D environment. A typical technique, known as head-mounted display (HMD), needs a device worn on the head, as a helmet, which has a small display optic in front of each eye. This scenario is shown in Figure 1(d). Substitutions for HMD need to introduce head tracking [9] or gaze tracking [10] techniques, as shown in the solutions discussed in [7]. In 3D TV, however, each stereoscopic display can have effect on a certain small range of a view angle, thus, a viewer can change his/her viewing position when he/she is trying to view the scene in another viewpoint, as if there was a natural object.

For rendering of 3D TV content or view synthesis, depth information is needed. Depth-images storing the depth information as a monoscopic color video can be coded with existing coding standards, for example, as auxiliary pictures in H.264/AVC [11].

As the normal 2D TV or HDTV applications are still dominating the market, the MVC content will provide a way for those 2D decoders, for example, H.264/AVC decoder in the set-top box (STB) of digital TV to generate a display from an MVC bistream, as shown in Figure 1(e). This requires MVC bitstreams to be backward compatible, for example, to H.264/AVC.

1.2. Requirements of MVC. Due to the huge amount of data, particularly when the number of views to be decoded is large, transmission of multiview video applications relies heavily on the compression of the video captured by cameras. Therefore, efficient compression of multiview video contents is the primary challenge for realizing multiview video services.

A natural way to improve compression efficiency of multiview video content is to exploit the correlation between views, in addition to the use of inter prediction in monoview coding. This requires buffering of additional decoded pictures. When the number of views is large, the required memory buffer may be prohibitive. In order to make efficient implementations of MVC feasible, the codec design should include efficient memory management of decoded pictures.

The above challenges and requirements, among others [12], are the basis of the objectives for the emerging MVC standard, which is under development by the joint video team (JVT), and will become the multiview extension of H.264/AVC [11]. MVC standardization in the JVT started in July 2006 and is expected to be finalized in mid-2008. The most recent draft of MVC is available in [13].

In the MVC standard draft, redundancies among views are utilized to improve compression efficiency compared to independent coding of views. This is allowed with the so-called interview prediction, in which decoded pictures of other views can be used as reference pictures when coding a picture as long as they all share the same capturing or output time. View dependencies for interview prediction are defined for each coded video sequence.

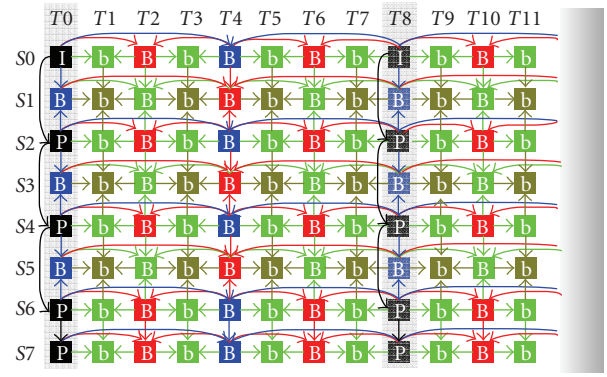


Figure 2: Typical MVC prediction structure.

With the exception of interview prediction, pictures of each view are coded with the tools supported by H.264/AVC. In particular, hierarchical temporal scalability was found to be efficient for multiview coding [14]. A typical prediction structure of MVC, utilizing both interview prediction and hierarchical temporal scalability, is shown in Figure 2. It is noted that the MVC standard provides a greater deal of flexibility than depicted in Figure 2 for arranging temporal or view prediction references [15].

Except the coding efficiency requirement, the following important aspects of the MVC requirements [12] for the design of the MVC standard are listed.

1.2.1. Scalabilities. View scalability and temporal scalability are considered in the MVC design for the adaptation of user preference, network bandwidth, and decoder complexity. View scalability is useful in the scenario shown in Figure 1(c), wherein some of the views are not transmitted and decoded.

1.2.2. Decoder Resource Consumption. In 3D TV scenarios, as shown in Figures 1(b) and 1(c), a number of views are to be decoded and displayed, an optimal decoder in terms of memory and complexity is of vital importance to make the real-time decoding of MVC bitstreams possible.

1.2.3. Parallel Processing. In the 3D TV scenarios, since multiple views need to be decoded simultaneously, parallel processing of different views is very important to realize this application and to reduce the computation time to achieve real-time decoding.

1.2.4. Random Access. Besides temporal random access, view random access is to be supported to enable accessing a frame in a given view with minimal decoding of frames in the view dimension. For example, free-viewpoint video described in Figure 1(a) needs advanced view random access functionality to support smooth navigation.

1.2.5. Robustness. When transmitted in a lossy channel, the MVC bitstream will have error resiliency capabilities. There are error resilient tools in H.264/AVC which can benefit the

MVC applications. Other techniques, which are designed only for MVC and discussed later, can also be utilized to improve error resilience of MVC bitstreams.

1.3. Contributions of this Paper. JVT has recently finalized the scalable extension of H.264/AVC, also known as scalable video coding (SVC) [16]. MVC shares some design principles with SVC, such as backward compatibility with H.264/AVC, temporal scalability, and network friendly adaptation, and many features in SVC have been reused in MVC.

However, new mechanisms are needed in MVC at least related to view scalability, interview prediction structure, coexisting of decoded pictures from multiple dimensions (i.e., both the temporal and view dimensions) in the decoded picture buffer, multiple representations in the display, and parallel decoding at the decoder.

These mechanisms cover the challenges and requirements, identified above, for 3D video services, except for the compression efficiency challenge. In this paper, we will describe how these mechanisms are realized in the existing draft MVC standard.

The main MVC features discussed in this paper include reference picture management to achieve optimal memory consumption at the decoder, time-first coding to support consistent system level design, SEI messages, and other features for view and scalability information provisioning, adaptation, random access, view switching, and reference picture list construction.

The rest of this paper is organized as follows. In Section 2, we discuss the MVC bitstream structure and the backward compatibility which is mentioned in Scenario (e). In Section 3, with a typical application scenario, we discuss how adaptation works when connectivity between server and client or decoder capacity varies. Then, view scalability information SEI message, which is designed to facilitate the storage, exaction, and adaptation of MVC bitstream, is reviewed. The features discussed in this section are of importance for efficient file composition, bitstream exaction, and stream adaptation in intermediate media gateways, which has been mentioned in Scenario (c). Random access and view switching functionalities are described in Section 4, which is desirable in Scenario (a). In Section 5, the decoded picture buffer management is discussed. This topic is crucial to enable a system to minimize the required memory for decoding MVC bitstreams. In Section 6, the parallel decoding SEI message, which is important for real-time MVC decoder solutions, is discussed. Other related issues are summarized in Section 7. Finally, Section 8 concludes the paper.

2. Structure of MVC Bitstreams

This section reviews the concept of network abstraction layer units (NAL units) and summarizes how the NAL unit types defined in H.264/AVC and SVC are reused for MVC. Syntax elements in the NAL unit header in the MVC context are also discussed.

In H.264/AVC, the coded video bits are organized into NAL units. NAL units can be categorized to video coding layer (VCL) NAL units and non-VCL NAL units. The supported VCL NAL unit types and non-VCL NAL units in H.264/AVC are defined in [11] and well categorized in [17].

In MVC, there is a base view, which is coded independently and is compliant with H.264/AVC, this meets the requirement in Scenario (e) of the MVC system architecture, as shown in Figure 1. Consequently, coded picture information for the base view is included in the VCL NAL units specified in H.264/AVC. A new NAL unit type, called coded slice of MVC extension, is used for containing coded picture information for nonbase views. When an MVC bitstream containing NAL units of the new NAL unit type is fed to an H.264/AVC decoder, NAL units of any new NAL unit type can be ignored and the decoder only decodes the bitstream subset containing NAL units of the existing NAL unit types defined in H.264/AVC.

There are useful properties of the coded pictures in the H.264/AVC-compliant base view, such as temporal level, which are not indicated in the VCL NAL units of H.264/AVC. To indicate those properties for the base view-coded pictures, the prefix NAL unit, of another new NAL unit type, has been introduced. Note that prefix NAL unit is also specified in SVC. A prefix NAL unit precedes each H.264/AVC VCL NAL unit and contains its essential characteristics in multiview context. As H.264/AVC decoders ignore prefix NAL units, the backward compatibility to H.264/AVC is still maintained.

Non-VCL NAL units include parameter set NAL units and SEI NAL units among others. Parameter sets contain the sequence-level header information (in sequence parameter sets (SPS)) and the infrequently changing picture-level header information (in picture parameter sets (PPS)). With parameter sets, this infrequently changing information needs not to be repeated for each sequence or picture, hence coding efficiency is improved. Furthermore, the use of parameter sets enables out-of-band transmission of the important header information, avoiding the need of redundant transmissions for error resilience. In "out-of-band" transmission, parameter set NAL units are transmitted in a more different channel than the ones for transmission of other NAL units. More discussions on parameter sets can be found in [18].

In MVC, coded pictures from different views may use different sequence parameter sets. An SPS in MVC can contain the view dependency information for interview prediction. This enables signaling-aware media gateways to construct the view dependency tree. Therefore, each view can be mapped to the view dependency tree and view scalability can be fulfilled, without any extra signaling inside NAL unit headers [19].

The scalable nesting SEI message [19], which was also introduced in SVC with the same name, is set apart from other SEI messages in that it contains one or more ordinary SEI messages, but in addition it indicates the scope of views or temporal levels for which the messages apply. In doing so, it enables the reuse of the syntax of H.264/AVC SEI messages for a specific set of views and temporal levels.

Some of the other SEI messages specified in MVC are related to the indication of output views, available operation points, and information for parallel decoding.

In H.264/AVC, an NAL unit consists of a 1-byte header and an NAL unit payload of varying size. In MVC, this structure is retained except for prefix NAL units and MVC-coded slice NAL units, which consist of a 4-byte header and the NAL unit payload. New syntax elements in MVC NAL unit header include priority_id, temporal_id, anchor_pic_flag, view_id, idr_flag and inter_view_flag.

anchor_pic_flag indicates whether a picture is an anchor picture or nonanchor picture. Anchor pictures and all the pictures succeeding in output order (i.e., display order) can be correctly decoded without decoding of previous pictures in decoding order (i.e., bitstream order) and thus can be used as random access points. Anchor pictures and nonanchor pictures can have different dependencies, both of which are signaled in the sequence parameter set.

More discussions on anchor pictures will be given in Section 4. idr_flag is introduced in Section 4, inter_view_flag is discussed in Section 5, and the other new MVC NAL unit header fields are introduced in Section 3.

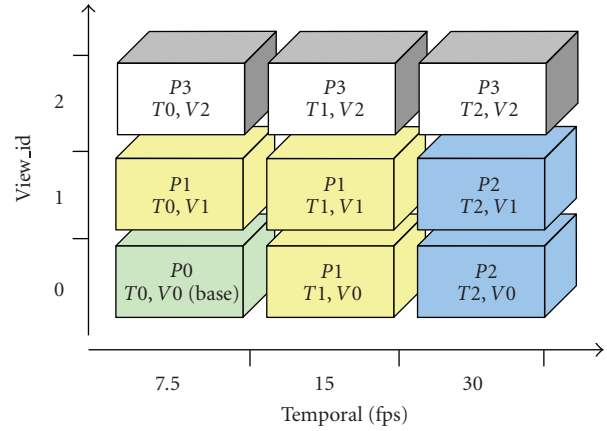
3. Extraction and Adaptation of MVC Bitstreams

MVC supports temporal scalability and view scalability. A portion of an MVC bitstream can correspond to an operation point that gives output representation for a certain frame rate and a number of target views. Data representing higher frame rate, views closer to the leaves of the dependency tree, or views that are not preferred by the client can be truncated during the stream bandwidth adaptation at the server or media gateway, or ignored at the decoder for complexity adaptation.

The bitstream structure defined in MVC is characterized by two syntax elements: view_id and temporal_id. The syntax element view_id indicates the identifier of each view. This indication in NAL unit header enables easy identification of NAL units at the decoder and quick access of the decoded views for display. The syntax element temporal_id indicates the temporal scalability hierarchy or, indirectly, the frame rate. An operation point including NAL units with a smaller maximum temporal_id value has a lower frame rate than an operation point with a larger maximum temporal_id value. Coded pictures with a higher temporal_id value typically depend on the coded pictures with lower temporal_id values within a view, but never depend on any coded picture with higher temporal_id.

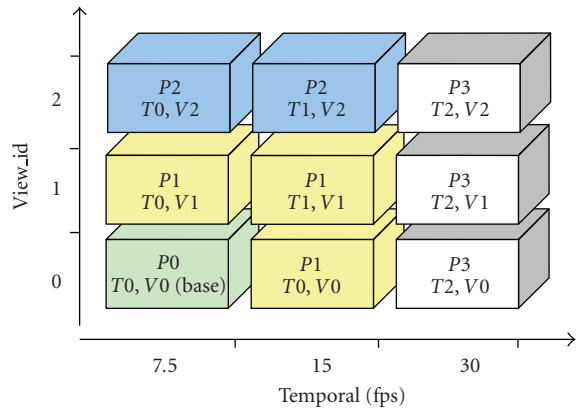
The syntax elements view_id and temporal_id in the NAL unit header are important for both bitstream extraction and adaptation. Another important syntax element in the NAL unit header is priority_id [19], which is mainly used for the simple one-path bitstream adaptation process.

Whenever the operation point contains only a subset of the entire MVC bitstream, such as in Scenario (a) and Scenario (c) shown in Figure 1, a bitstream extraction process is then needed to exact the required NAL units from the entire bitstream. The bitstream extraction process should be a lightweight process without heavy parsing of the bitstream. For this purpose, the mapping between each operation point (identified by the combination of required view_id values and temporal_id values) and the required NAL units is specified as part of the view scalability information SEI message (VSSEI) [20]. After the operation point is agreed upon, the server can simply extract the required bitstream subset by discarding nonrequired NAL units by checking the view_id and temporal_id values in the fixed-length coded NAL unit headers.



Path:
$P = 0$: view 0/7.5
$P = 1$: view 0, 1/15
$P = 2$: view 0, 1/30
$P = 3$: view 0, 1, 2/30

(a)



Path:
$P = 0$: view 0/7.5
$P = 1$: view 0, 1/15
$P = 2$: view 0, 1, 2/15
$P = 3$: view 0, 1, 2/30

(b)

Figure 3: Assignment of priority_id for NAL units of a 3-view bitstream with two levels of temporal resolution. *T: temporal level; V: view identifier; P: priority identifier. Temporal level equal to 0 corresponds to 7.5 fps (frame per second), it equal to 1 corresponds to 15 fps, and it equal to 2 corresponds to 30 fps.*

Media gateways can perform single-path adaptation by simply discarding NAL units with priority_id greater than a certain value. The priority_id has no normative effect on the decoding process. The only constraint to priority_id values is that any bitstream subset extracted based on any value of priority_id must be a conforming MVC bitstream. It is the encoder responsibility to set priority_id values for the NAL units and the values can be rewritten, for example, when the preference of the decoder changes.

Figure 3 depicts two examples of priority_id assignments which yield two different adaptation paths for the same MVC bitstream that contains 3 views with 3 temporal levels. In Figure 3(a), the priority_id is assigned such that the 7.5 Hz base view is with priority_id equal to 0, and then frame rate of 15 Hz including both view 0 and view 2 is with priority_id equal to 1, and then higher frame rate is preferred to more views. In Figure 3(b), the first two steps are the same as in Figure 3(a), while in the last two steps, more views are preferred to higher frame rate.

Although a simple media gateway may perform stream adaptation exclusively based on priority_id, more intelligent implementations may jointly employ the values of priority_id, view_id, and temporal_id, in order to perform combined adaptation. For example, for the bitstream discussed in Figure 3, there can be two adaptation steps, the first step is to have NAL units with temporal_id equal to 1 (15 Hz) and view_id through 0 to 1; the second step is to increase frame rate directly to 30 Hz and include all the NAL units in view 2. Note that in this case, the NAL units corresponding to each adaptation step can have different values of priority_id, for example, when the priority_id assignment follows Figure 3(a).

An MVC bitstream may contain a large number of views (the view_id in the current MVC draft specification is of 10 bits). This makes the possible number of combinations of view_id values and temporal_id values huge. However, in practical applications, typically only limited combinations, that is, operation points, would be used. The VSSEI has been designed to be flexible to signal any subset of all the possible operation points. Beside the mapping of operation points and NAL units, the following information for each indicated operation point is also included in the VSSEI, either to enable the establishment of the communication session or more efficient bitstream extraction or adaptation.

Profile and level: This information describes the capacity a decoder requires to decode a bitstream. Profile and level can be signaled in the SPS. However, the total number of SPS is limited to a certain value in the bitstream and it may happen that for all the operation points, many of them share the same SPS, the level inside which is not accurate enough to describe the minimum required capacity of the decoders for different operation points. Therefore, profile and level are signaled in the VSSEI for each operation point.

Bit rate: Similar as profile and level, this information is needed in the session negotiation process for the server and the client to agree upon a certain operation point. This information is also useful in rate adaptation by MANEs. For example, to better adapt the bandwidth, it is necessary for intelligent media gateways to know the bandwidth of a session when it switches to another operation point.

Operation point dependencies: In the VSSEI, each operation point is identified by the view_id values of the target views and the temporal_id values. The dependent views as well as the dependent pictures may be known from the active SPS which contains the view dependency information. However, within the view dependency, pictures may have more flexible relationship. For example, assume in a two-view bitstream with 30 fps, 4 temporal levels and according to the SPS MVC extension anchor pictures and nonanchor pictures in view 1 are, respectively, dependent on anchor and nonanchor pictures in view 0. And if we have two operation points (OPs), OP 0 has the pictures in view 0 with temporal level up to 3, that is, 15 fps and OP 1 has pictures with all the pictures in view 1, however, the pictures with the highest temporal level in view 1 do not really rely on interview pictures for reference. Then, OP 1 actually depends only on OP 0, which cantinas half of the pictures in view 0 and the highest temporal level pictures in view 0 can be neglected for transmission and decoding. However, with only the view dependency signaled in the SPS MVC extension, those pictures are still required to be transmitted and decoded. Thus, operation point dependency information included in the VSSEI would enable simply identification and discarding of the nonrequired NAL units that are not indicated by the view dependency information signaled in SPS.

In the following are some MVC stream adaptation examples in a broadcasting system (see Figure 1). Assume that the entire bitstream contains coded pictures of 8 views.

For Scenario (e), NAL units are filtered by the MANE so that only the NAL units that can be recognized by H.264/AVC decoders (by checking the NAL unit type) are fed to the STB of an HDTV.

For Scenario (d), an operation point containing, for example, only view 0 and view 1 is in use. The MANE controls the bitstream in a way that only allows the NAL units with view_id (by checking the view_id in the NAL unit header) equal to 0 or 1 to be sent to the client.

Depending on the bandwidth, a client with enough decoding capability for 3D TV may switch between Scenarios (b) and (c), wherein the sub-bitstream corresponding to Scenario (b) forms an operation point that contains only a subset of the views within a narrow view angle. The MANE filters out the views outside the view angle.

4. Random Access and View Switching

4.1. Random Access. Random access refers to starting decoding of a bitstream from a point other than the beginning. The support of random access is required for traditional trick play modes such as fast forward and fast backward. In streaming applications, random access is used to seek the desired playback position requested by the users. In broadcast and multicast applications, random access points are required to allow for newcomers to tune in or switching of program channels.

Random access with MVC for the above purposes is not much different from that with single-view coding, as

all the target views of an operation point are accessed simultaneously. The only difference is that there may be views dependent on by the target views; hence these dependent views need also to be accessed and decoded.

To access to a picture in a given view at a specified time, the decoder should first find the closest preceding temporal locations that are random access points to the specific target view and all the dependent views, collectively referred to as the required views. Then the decoder starts decoding the required views from a found location. In average, how many view pictures need to be decoded to access to a specific target picture is therefore proportional to the random access period (i.e., the length of the temporal dependency chain) and the number of dependent views (i.e., the length of the interview dependency chain).

Instantaneous decoding refresh (IDR) pictures are natural random access points. In an MVC bitstream, IDR pictures in the base view have NAL units of type 5. If the bitstream also contains NAL units that are unknown to plain H.264/AVC decoders, then the base view IDR picture NAL units are each preceded by a prefix NAL unit, which has idr_flag equal to 1. IDR pictures of nonbase views, also referred to as view-IDR (V-IDR) pictures in the draft MVC standard, all have idr_flag equal to 1. V-IDR pictures may rely on pictures from other views but only within the same access units though interview prediction [21].

An access unit contains all the NAL units pertaining to a certain time instance. According to the draft MVC standard, an IDR access unit is an access unit wherein the pictures of all the views are IDR pictures. Such an IDR access units provide random access support at the time instance to all the views. Note that the draft MVC standard allows for such access unit wherein pictures of some views are IDR pictures while pictures of other views are non-IDR pictures.

IDR pictures disallow any picture succeeding the IDR picture in decoding order (i.e., bitstream order) to be inter-predicted from earlier pictures in the same view. This leads to a reduced compression efficiency compared to the typical open GOP (group of pictures) coding structures such as the IBBP structure, where the B pictures after the I picture in decoding order precede the I picture in display order, and can use pictures before the I picture in decoding order for inter prediction. The I pictures in such open GOP coding structures are defined as anchor pictures in the draft MVC standard and are identified by the NAL unit header syntax element anchor_pic_flag equal to 1. Anchor pictures can therefore also be used as random access points, while application implementers must bear in mind that a few pictures after such random access points may not be correctly decoded when random access is carried out at these points. Actually, in this situation these pictures can be dropped from the bitstream sent to the user. Like V-IDR pictures, anchor pictures in nonbase views can also use interview prediction.

It is also possible to perform random access at non-intra pictures, for example, using gradual decoding refresh (GDR) based on the isolated regions technology [22]. In this case, the GDR random access points can be indicated by the recovery point SEI message as specified in H.264/AVC, but included in the scalable nesting SEI message that tells to which views the semantics apply.

4.2. View Switching. View switching refers to changing the target view(s). The number of target view(s) may be one or more. In case the number of target view(s) change or any of the target view is changed from one view to another, a view switching occurs. View switching must happen at view-switching points, after which the new target view(s) can be correctly decoded. A typical application for view switching is free-viewpoint video, which has been shown in Scenario (a) of Figure 1.

All random access points can also be used as view switching points. There is another type of switching points that are not random access points. For example, if at picture X the target views can be switched to view subset C from view subset A but not from view subset B, then picture X is a view-switching point from view subset A to view subset C. This type of switching points can be realized by specifically setting the interview prediction relationship, or by using the SP/SI coding technology [23].

5. Decoded Picture Buffer Management

In this section, we first introduce the decoding order arrangement of coded view pictures, which is closely related to decoded picture buffer management. After that, we present an analysis of the buffer requirement for decoding of MVC bitstreams, which has been discussed in more details in [24]. Finally, reference picture management methods both inside a view and related to interview pictures are discussed.

5.1. Decoding-Order Arrangement. In H.264/AVC, the order how NAL units are placed inside the bitstream is referred to as the decoding order. In multiview video, where two dimensions, time and view, are involved, prescription of the decoding order gets more complicated.

Two fundamentally different decoding order arrangements, view-first coding and time-first coding, have been considered by the JVT. In view-first coding [25], within each group of pictures (GOP), pictures of each view are contiguous in decoding order, as shown in Figure 4, where the horizontal direction denotes time (each time instance is represented by Tm), and the vertical direction denotes view (each view is represented by Sn). Pictures of each view are grouped into GOPs, for example, pictures T1 to T8 for any view in Figure 2 form a GOP.

View-first codingcauses a fundamental problem for storage of multiview video bitstreams in media container files based on ISO base media file format [26]. Coded pictures belonging to different views but with the same time instance are interleaved with pictures of other time instances in a bitstream, and thus cannot be in the same access unit. These different access units, when composed into a file according to the ISO base media file format, correspond to different samples. The ISO base media file format requires samples to be ordered in their decoding order. According to the ISO base media file format, the decoding time of a sample is an

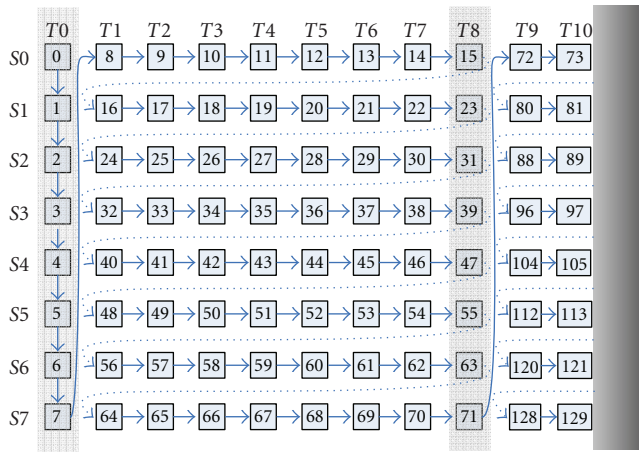


Figure 4: View-first coding.

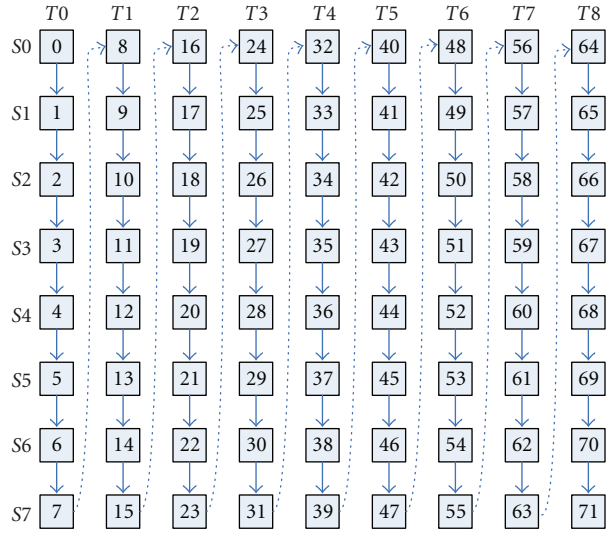


Figure 5: Time-first coding.

increasing function of sample number, and the composition time (also used as presentation time) of a sample is indicated as a nonnegative increment compared to its decoding time. Consequently, view-first coding would require a composition time offset proportional to the GOP size multiplied by the number of views, which would be perceived as significant initial buffering delay. Furthermore, possibility for parallel decoding would be hard to realize when view-first coded streams are included in files compliant with ISO base media file format, because the indicated decoding and composition times assumed single-processor operation.

To overcome the mentioned problems, time-first coding was introduced in MVC [27]. In time-first coding, pictures of any temporal location are contiguous in decoding order, as shown in Figure 5. In this case, we can define pictures of the same time instance but belonging to different views as one access unit. Note that the decoding order of access units may not be identical to the presentation order.

With time-first coding, an access unit contains NAL units continuous in decoding order. This definition is similar to the access unit definition in SVC. Therefore, many mechanisms designed in the SVC file format, such as extractors and aggregators, are useful for MVC too. Some design principle for MVC file format can be found in [28].

The following subsections on buffer requirement analysis and buffer management are all for time-first coding only.

5.2. Buffer Requirement Analysis. In MVC, pictures in the same time instance are assumed to be outputted simultaneously. Decoded pictures used for prediction or future output are buffered in the decoded picture buffer (DPB). To efficiently utilize the buffer memory, the DPB management processes have been specified, which include a storage process of decoded pictures into the DPB, a marking process of reference pictures, and an output and removal process of decoded pictures from the DPB.

Assume that we have a prediction structure similar to the one shown in Figure 2, where each GOP includes a number of views (nv) and in each view *gl* (GOP length) pictures.

The optimal DPB size, as discussed in [29], is TL + 1, where TL is the highest temporal level of all the pictures and $\mathrm{TL} = \lceil \log_2(gl) \rceil$.

The DPB sizes for time-first coding in different scenarios are summarized in the following, while more details can be found in [24, 30].

5.2.1. DPB When Output is not Taken into Consideration. In time-first coding, the pictures in the same time instance will be stored in the DPB longer and each view preserves the hierarchical B coding structure. So there are two steps to reach the maximum DPB size for time first:

(1) take the pictures in the same time instance as a whole and form a hierarchical B coding structure, the DPB size would then be $nv \cdot (\mathrm{TL} + 1)$;

(2) for the nonreference pictures in the highest temporal level, interview prediction requires them to be stored in the DPB.

These two steps are shown in Figure 6.

So, in the typical prediction structure, the maximum DPB size for time-first coding is $nv \cdot (\mathrm{TL} + 1) + 2$.

In both results, there is a "2", which actually means the maximum interview reference pictures in the typical prediction structure.

5.2.2. DPB When the Output is Taken into Consideration. When the output is considered, thecase is even worse for view-first coding, especially for 3D TV application scenario, which requires the display of all the views. The reason is that, in view-first coding, all the pictures of the already coded view in a GOP must be kept in the buffer at least till the last view starts decoding.

For simplicity, we give the DPB buffer sizes for view-first coding and time-first coding in both 3D TV and free-viewpoint video scenarios without detailed analysis, which can be found in [24].

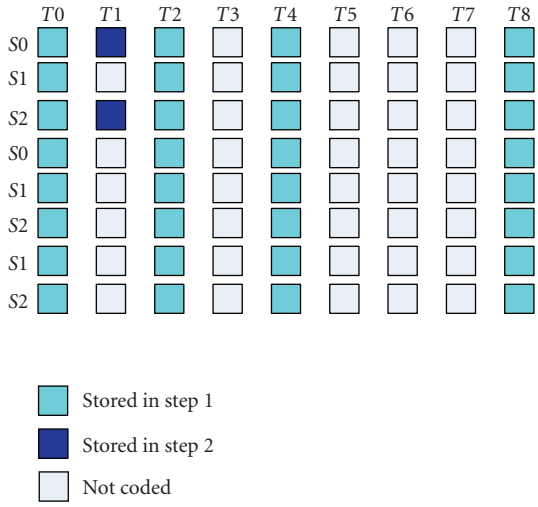


Figure 6: DPB status for time-first coding.

In 3D TV scenario, the total DPB sizes for time-first coding are $(nv-1)\,gl+\mathrm{TL}+1$ and $nv(2\cdot\mathrm{TL}-\log_2\lfloor\mathrm{TL}-1\rfloor)$, respectively.

In free-viewpoint video, the total DPB size for time-first coding is $(nv\cdot(\mathrm{TL}+1)+3)/2+\mathrm{TL}-1-\log_2\lfloor\mathrm{TL}-1\rfloor$.

Table 1 gives the example values for all the compared scenarios when the GOP length is 16 and number of views are 8. Time-first coding, as shown by the formula as well as the example values, requires less DPB size.

Note. Scenarios through (1) to (4) are the following scenarios, respectively: (1) DPB w/o output; (2) 3D TV DPB with output; (3) Free viewpoint video DPB with output, maximum; (4) Free viewpoint video DPB with output, average *gl* is 16 and *nv* is 8.

5.3. Buffer Management Inside a View. Because of the time-first coding structure, whether a picture is a reference picture or nonreference picture can be decided only by its temporal prediction structure. Because for any two pictures in a view, if picture A follows picture B, then in the whole bitstream, picture A also follows any picture with the same time instance as picture B. This is not the case in view-first coding, so it may require cross-view explicitly or implicit marking to make those pictures with the same time instance as B but with early decoding time as A as "unused for reference".

So, all the memory management control operation commands, if present, are effective inside a view. And the sliding window also takes effect inside a view, which was proposed into JVT in the same time in [30–32].

5.4. Buffer Management for Interview Reference Pictures. In each time instance, if dependency exists, for one current decoding picture, there can be one or more interview reference pictures. Those interview reference pictures, although there are not used for temporal prediction within a view, are required to be somehow stored in the decoded picture buffer.

Table 1: Comparison examples between view-first and time-first when different scenarios are utilized.

	(1)	(2)	(3)	(4)
view-first	44	117	44	32
time-first	44	56	56	23.5

However, whether to store these pictures as "used for reference" or "unused for reference" is still an issue. In the AVC specification, if a picture is not used as a reference picture for others, it is with a nal_ref_idc value equal to 0 and is a nonreference picture. Those pictures, however, in MVC context can be used for interview reference picture, for example, the highest temporal level pictures in view 0 when view 1 is decoded.

If there are stored as a reference picture, when only base view sub-bitstream is decoded, it is definitely an extra memory burden for the H.264/AVC decoder and the encoder may need to design extra memory management control operation (MMCO) commands. So, in [33], we proposed that those pictures are not required to be stored as a reference picture. This solution solves the problem we mentioned above and another question arises: how would those pictures used only for interview prediction be managed to reach the optimal buffer management. One argument is if an interview picture is not used for temporal prediction and is a nonreference picture, it may be not available in the DPB.

Because of the time-first coding structure and the assumption that pictures are outputted at the same time, the concern mentioned above is solved.

So there is no extra marking process for those pictures if all views are required for output. If some views are not required for output, those pictures can be implicitly removed from the DPB earlier. The implicit removal is based on the view dependency defined in the MVC SPS extension [19, 34]. The implicit removal is defined in the hypothetical reference decoder (HRD) part of the MVC specification. The current HRD design of MVC focuses mostly on output conformance.

Although the interview prediction structure is in the scope of MVC SPS extensions, for each time instance, a picture can be used as interview picture or not based on real uses. For example, pictures in higher temporal levels may be more helpful for the efficiency while picture in lower temporal levels may be less helpful. The decoding of those pictures, if they are nonreference pictures and belong to the views that are not required for output, can be avoided. So an inter_view_flag was proposed by [35, 36] and was introduced into the MVC specification.

6. Parallel Coding of Multiple Views

One of the key identified requirements for the MVC standard is its ability to support parallel processing of different views [11]. The parallel processing of different views is especially important for 3D broadcasting use cases, where the displays need to output many views simultaneously to support head-motion parallax. However, interview dependencies between

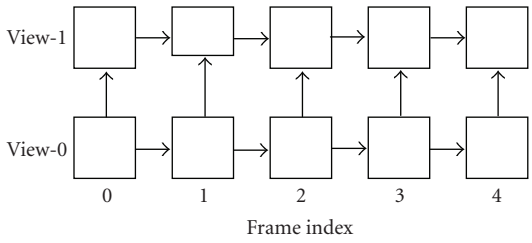


Figure 7: Sample prediction structure for two views.

pictures may impose serious parallelism issues to the video system, because two pictures at different views need to be decoded sequentially. Let us consider a 3DTV system displaying simultaneously two views, and views are coded with the coding structure as illustrated in Figure 7.

In order to decode a picture in view 1 at any temporal instant, the picture in view 0 at the same temporal instant will be decoded first. The only way to display two views at the same time is by having an MVC decoder running two times faster than a regular single-view decoder. Even though two independent decoders running on different platforms might be available, both decoders need to run twice faster than the single-view decoder because decoding has to be performed sequentially. The situation gets worse, with the increasing number of views that is supported by the 3D display. Currently there are commercial displays which can display 100 views simultaneously, and if all the views depend on each other, then the decoder must run 100 times faster, which is very challenging.

One way to increase the parallelism is to code each view independently. However, this kind of simulcast approach results in a significant penalty in coding efficiency as interview redundancies are not exploited at all. The draft MVC standard includes a more efficient method that allows parallel decoding/encoding operation of multiple views with high coding efficiency. This is achieved by utilizing the parallel decoding information SEI message that indicates that the views are encoded with systematic constraints, so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of macroblocks in other views [37, 38].

In order to describe how parallel processing is achieved using parallel decoding information SEI message, let us consider an example, where two pictures from view 1 and view 0 are going to be decoded. Assume view 1 picture references view 0 picture as illustrated in Figure 8 (for simplicity, the sizes of the frames are five macroblocks both horizontally and vertically). Parallel decoding information SEI message indicates the video is encoded in a way that macroblocks in view 1 picture could only use reconstruction values of macroblocks that belong to certain rows in view 0 picture. For example, the macroblocks in the first macroblock row of view 1 picture could only use reconstruction values from the first two macroblock rows in view 0 picture. In other words, the available reference area for the first macroblock row of view 1 picture constitutes only data from the first

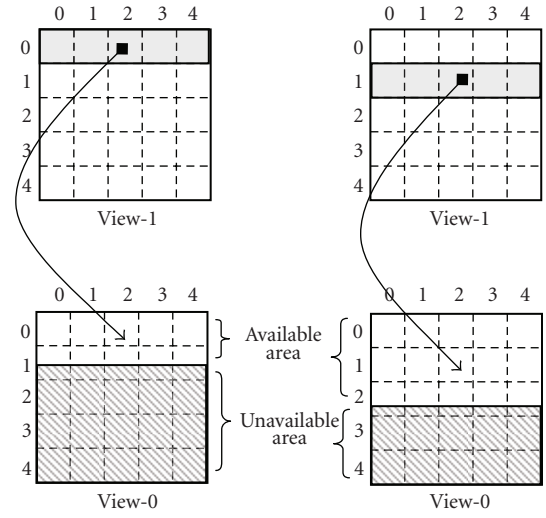


Figure 8: Systematic restriction of reference area.

two macroblock rows of view 0 picture (i.e., the motion vectors for the view 1 macroblocks are restricted). Similarly, the second macroblock row of view 1 picture only uses reconstruction values of the first three macroblock rows of the view 0 picture. This systematic restriction of reference area enables parallel decoding of first row of view 1 with any row below the second of view 0, as they are not referring each other.

In order to illustrate how this feature is used, let us assume an MVC decoder running on two processors (or processor cores) and decoding a bitstream containing two-views, where view 1 references view 0. Further assume that the bitstreams are coded with the restrictions as described above. The parallel decoding operation of these two views is illustrated in Figure 9, where processor P0 is decoding view 0 pictures and processor P1 is decoding view 1 pictures. The decoding operations for both views start simultaneously, but decodingf the first row of macroblocks in view 1 picture does not start before view 0 notifies the view 1 decoder. This notification is done after all the macroblocks in the first two macroblock rows in view 0 are decoded, and their reconstruction data are placed in the memory. This notification tells decoder of view 1 that all data required to decode first macroblock row in view 1 are ready. This way, the decoder of view 1 could start decoding the macroblocks of the first row, while the decoder of view 0 proceeds with decoding macroblocks in the third row and two decoders run in parallel. This parallel operation continues with two macroblock rows of delay between two views till the decoding of all the macroblocks is finished.

The benefit of using parallel decoding information SEI message is that significant coding gain is achieved over simulcast, while maintaining almost the same desirable parallelism characteristics. When compared to anchor method, where encoding happens without utilizing the SEI message and systematic restrictions, it is seen that parallel operation is achieved with almost no penalty on coding efficiency:

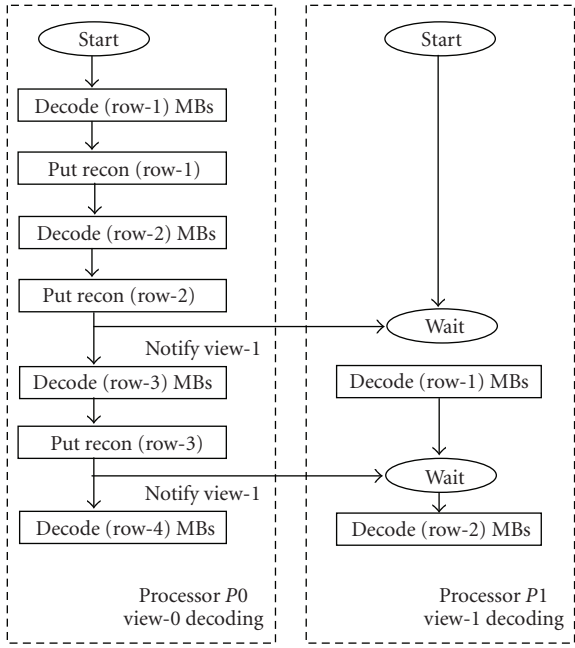


Figure 9: Sample parallel decoding process for two views.

maximum 0.08 and in average 0.03 dB loss for all the test sequences defined in the comment test condition for MVC [39]. Compared to simulcast, similar parallelism is achieved with 0.9 dB gain on coding efficiency as interview prediction is still utilized [37, 38].

In addition to using this SEI message, parallel processing could also be achieved by using simpler prediction structures. For example, consider the case where nonanchor pictures do not use interview prediction but only temporal prediction [40]. This approach achieves parallel operation as nonanchor pictures can be independently decoded without referencing other views. The parallelism of this structure could be further improved by using the parallel decoding information SEI message for anchor pictures.

It should be noted that parallel decoding information SEI message does not change the worst case complexity of MVC decoders. This means that the MVC decoders need to be designed to handle bitstreams where the encoding restrictions have not been applied, and parallel decoding information SEI message is not present. However, system standards such as DVB-H [41] can mandate the usage of this SEI message on their respective environments. This would ensure parallel operation for all the decoders operating on these services.

7. Other Related Techniques

Beside those discussed in earlier sections, the joint draft of MVC includes the following related techniques as summarized below.

7.1. Reference Picture List Construction. The reference picture list construction process can flexibly arrange temporal and view prediction references. This provides not only potential coding efficiency gain but also error resilience, since reference picture section and redundant picture mechanisms can then be extended to the view dimension [15]. This strengthens the error robustness of the MVC bitstreams.

7.2. Active View Information SEI Message. The decoder may prefer to display a subset of the views encoded in an MVC bitstream. If this preference can be known by the decoder, then only the output views and the dependent views need to be decoded and stored in the DPB. The active view information SEI message was introduced to indicate the views that are to be output [42].

7.3. Multiview Scene Information and Multiview Acquisition Information SEI Messages. Two SEI messages related to acquisition and rendering were introduced in MVC, namely multiview scene information SEI message and multiview acquisition information SEI message, to signal camera parameters, which are helpful in view interpolation by a renderer [43].

8. Concluding Remarks

In this paper, we reviewed the key aspects of the system, transport interface, and decoder designs of MVC. We also introduced techniques crucial in meeting the requirements of typical 3D services and system architectures. These solutions, as adopted to the draft MVC standard, focus on two parts: features to facilitate storage and transport of MVC bitstreams and features to achieve minimum decoder resource consumption.

For the MVC system and transport interface, bandwidth adaptation, decoder capability adaptation, view random access, and view switching are the main concerns of the design. For the MVC decoder, minimizing the memory consumption and computational complexities are addressed.

The following key points of the MVC design are highlighted.

(i) MVC shared the same network abstraction layer (NAL) unit types designed in SVC, while differs a little in some specific syntax elements.

(ii) The base view of an MVC bitstream was designed to be H.264/AVC compatible in a way that it can be reconstructed by a standard H.264/AVC decoder. At the same time, backward-compatible extensions allow to utilize MVC-specific features with an MVC-compliant decoder.

(iii) New supplemental enhancement information (SEI) messages have been introduced to signal operation points as well as their dependency information. It is designed for adaptation and bitstream extraction. In addition, a mechanism has been specified that allows reusing all the original H.264/AVC SEI messages.

(iv) Time-first coding order was introduced to facilitate the file format design of MVC. This coding order is essential to achieve optimal buffer management at the decoder.

(v) Parallel decoding information SEI message was introduced to enable parallel encoder/decoder operation

for different views. This is especially important for 3D broadcast systems that support head-motion parallax, where the receiving end needs to decode and display multiple views simultaneously.

Acknowledgment

This work was supported in part by Nokia and the Academy of Finland, Finnish Centre of Excellence Program 2006-2011 under Project 213462.

References

[1] A. Smolic, H. Kimata, and A. Vetro, "Development of MPEG standards for 3D and free viewpoint video," in *Three-Dimensional TV, Video, and Display IV*, vol. 6016 of *Proceedings of SPIE*, Boston, Mass, USA, October 2005.

[2] H.-Y. Shum, S. B. Kang, and S.-C. Chan, "Survey of image-based representations and compression techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 11, pp. 1020–1037, 2003.

[3] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV," in *Stereoscopic Displays and Virtual Reality Systems XI*, vol. 5291 of *Proceedings of SPIE*, pp. 93–104, San Jose, Calif, USA, May 2004.

[4] H. Kimata, M. Kitahara, K. Kamikura, Y. Yashima, T. Fujii, and M. Tanimoto, "System design of free viewpoint video communication," in *Proceedings of the 4th International Conference on Computer and Information Technology (CIT '04)*, pp. 52–59, Wuhan, China, September 2004.

[5] A. Smolic and P. Kauff, "Interactive 3-D video representation and coding technologies," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 98–110, 2005.

[6] A. Vetro, W. Matusik, H. Pfister, and J. Xin, "Coding approaches for end-to-end 3D TV systems," in *Proceedings of the 23rd Picture Coding Symposium (PCS '04)*, pp. 319–324, San Francisco, Calif, USA, December 2004.

[7] C. Fehn, R. de la Barré, and S. Pastoor, "Interactive 3-DTV-concepts and key technologies," *Proceedings of the IEEE*, vol. 94, no. 3, pp. 524–538, 2006.

[8] J. G. Eden, "Information display early in the 21st century: overview of selected emissive display technologies," *Proceedings of the IEEE*, vol. 94, no. 3, pp. 567–574, 2006.

[9] B. Fröba and C. Küblbeck, "Face detection and tracking using edge orientation information," in *Visual Communications and Image Processing*, vol. 4310 of *Proceedings of SPIE*, pp. 583–594, San Jose, Calif, USA, January 2001.

[10] L. Young and D. Sheena, "Methods & designs: survey of eye movement recording methods," *Behavior Research Methods & Instrumentation*, vol. 7, no. 5, pp. 397–429, 1975.

[11] ITU-T Rec. H.264—ISO/IEC IS 14496-10, "Advanced video coding for generic audiovisual services," v3, 2005.

[12] ISO/IEC JTC1/SC29/WG11, "Requirements on multi-view video coding v.5," N7539, Nice, France, October 2005.

[13] "Joint draft 6.0 on multi-view video coding," JVT-Z209, Antalya, Turkey, January 2007.

[14] D. Tian, M. M. Hannuksela, and M. Gabbouj, "Sub-sequence video coding for improved temporal scalability," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 6, pp. 6074–6077, Kobe, Japan, May 2005.

[15] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "On MVC reference picture list construction," JVT-V043, Marrakech, Morocco, January 2007.

[16] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. Wien, Eds., "Joint draft 11 of SVC amendment," JVT-X201, Geneva, Switzerland, June-July 2007.

[17] Y.-K. Wang, M. M. Hannuksela, S. Pateux, A. Eleftheriadis, and S. Wenger, "System and transport interface of SVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1149–1163, 2007.

[18] S. Wenger, "H.264/AVC over IP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, 2003.

[19] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "Comments on MVC JD 2.0," JVT-W035, San Jose, Calif, USA, April 2007.

[20] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "View scalability information SEI message for MVC," JVT-W037, San Jose, Calif, USA, April 2007.

[21] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "MVC comments on JD 3.0," Geneva, Switzerland, July 2007.

[22] M. M. Hannuksela, Y.-K. Wang, and M. Gabbouj, "Isolated regions in video coding," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 259–267, 2004.

[23] M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 637–644, 2003.

[24] Y. Chen, Y.-K. Wang, and M. Gabbouj, "Buffer requirement analyses for multi-view video coding," in *Proceedings of the 26th Picture Coding Symposium (PCS '07)*, Lisbon, Portugal, November 2007.

[25] "Joint multiview video model (JMVM) 1.0," JVT-T208, Klagenfurt, Austria, July 2006.

[26] ISO/IEC IS 14496-2, "Information technology—coding of audio-visual objects—part 12: ISO base media file format," 2005.

[27] Y.-K. Wang, Y. Chen, and M. M. Hannuksela, "Time-first coding for multi-view video coding," JVT-U104, Hangzhou, China, October 2006.

[28] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "On MVC file format," M14634, Lausanne, Switzerland, July 2007. 1pt0.8pt

[29] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B pictures and MCTF," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '06)*, pp. 1929–1932, Toronto, Ontario, Canada, July 2006.

[30] Y. Chen, Y.-K. Wang, and M. M. Hannuksela, "MVC reference picture management," JVT-U105, Hangzhou, China, October 2006.

[31] A. Vetro and S. Yea, "Comments on MVC reference picture marking," JVT-U062, Hangzhou, China, October 2006.

[32] P. Pandit, Y. Su, P. Yin, and C. Gomila, "Comments on high-level syntax for MVC," JVT-U026, Hangzhou, China, October 2006.

[33] Y. Chen, Y.-K. Wang, M. M. Hannuksela, S. Liu, and H. Li, "On MVC reference picture marking," JVT-V044, Marrakech, Morocco, January 2007.

[34] A. Vetro and S. Yea, "MVC clarification of marking process," JVT-V085, Marrakech, Morocco, January 2007.

[35] Y.-K. Wang, Y. Chen, and M. M. Hannuksela, "Comments to JMVM 1.0," JVT-U103, Hangzhou, China, October 2006.

[36] J. Choi, W. Shim, H. Song, and Y. Moon, "Inter-view prediction reference picture marking," JVT-W056, San Jose, Calif, USA, April 2007.

[37] K. Ugur, H. Liu, J. Lainema, M. Gabbouj, and H. Li, "Parallel encoding-decoding operation for multi-view video coding with high coding efficiency," in *Proceedings of the Conference on True Vision, Capture, Transmission, and Display of 3D Video (3DTV '07)*, pp. 1–4, Kos Island, Greece, May 2007.

[38] K. Ugur, J. Lainema, H. Liu, and Y.-K. Wang, "Parallel decoding info SEI message for MVC," JVT-V098, Marrakech, Morocco, January 2007.

[39] "Common test conditions for multiview video coding," JVT-T207, Klagenfurt, Austria, July 2006.

[40] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Comparative study of MVC structures," JVT-V132, Marrakech, Morocco, January 2007.

[41] G. Faria, J. A. Henriksson, E. Stare, and P. Talmola, "DVB-H: digital broadcast services to handheld devices," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 194–209, 2006.

[42] Y.-K. Wang, M. M. Hannuksela, and Y. Chen, "MVC output related conformance," JVT-W036, San Jose, Calif, USA, April 2007.

[43] A. Vetro, S. Yea, W. Matusik, H. Pfister, and M. Zwicker, "Anti-aliasing for 3D displays," JVT-W060, San Jose, Calif, USA, April 2007.