Encapsulation Methods for Stable Free-viewpoint Video Streaming Services

Minjae Seo
Department of Computer
Seoul Women's University
Seoul, Republic of Korea
seominjae@swu.ac.kr

Jong-Ho Paik
Department of Software Convergence
Seoul Women's University
Seoul, Republic of Korea
paikjh@swu.ac.kr

***Abstract*— In this paper, we propose encapsulation methods for free-viewpoint video streaming services. The proposed methods are more efficient for view synthesis, which is one of the essential techniques to implement free-viewpoint services. By using these methods, users can reliably receive free-viewpoint video streaming services.**

Keywords—free-viewpoint video; view synthesis; multiview service; multiviewpoint video

I. INTRODUCTION

Free-viewpoint video is video content in which users can change the viewpoints from which they watch the video. Due to user requirements of immersive media, free-viewpoint video is attracting attention. Therefore, various methods for streaming free-viewpoint video services have been considered; several methods have been proposed to provide free-viewpoint video service based on dynamic adaptive streaming over HTTP (DASH) to transmit videos adaptively to users in various network environments [1]. In addition, a streaming service method using MPEG media transport (MMT) has been studied to provide free-viewpoint videos that can be transmitted in a hybrid environment [2]. However, there are still many obstacles to its commercialization such as difficulty in content production and restrictive bandwidth, which is not sufficient for a large capacity service. In addition to the bandwidth restriction, it is also physically impossible to capture and store complete videos from all viewpoints. Therefore, a technique for view synthesis is essential to provide free-viewpoint video content. The quality of service (QoS) for a free-viewpoint video service can be improved depending on how quickly the viewpoints are synthesized and how short the time delays are once the user request is sent.

Basically, views can be synthesized on both the client and server side. Performing view synthesis on the client side reduces request process and requires less time as compared to the server. To synthesize one view on the client side, two original texture videos, related texture videos for each, and additional information should be received. For video streaming, more information is required to solve the complicated synchronization problem of free-viewpoint video. View synthesis should consider temporal and spatial changes for every content.

However, view synthesis is still performed on the server side because the quality and duration of view synthesis depends on the performance of the computer. The server decides how many viewpoints can be provided; providing too many viewpoints can increase the delay of the service. The service server should also consider that user network environments may not always be in good condition.

In this paper, we propose encapsulation methods for media and additional information for stable free-viewpoint video streaming services. First, we propose encapsulation methods for the client side to solve the synchronization problem. Then, we propose an encapsulation method for the server side to control the number of viewpoints and user movement. Users are expected to receive reliable and stable services with these methods.

II. RELATED WORK

A. Free-Viewpoint Video (FVV)

FVV is a video service that not only provides users with multiple viewpoints but also allows them to choose the viewpoint [3].

The viewpoint synthesis technique plays a very important role in FVV. To create a viewpoint, two texture videos, i.e., two depth videos, are required; additional information is also required to synthesize them [5]. The depth image is used to represent the distance and can be extracted from the texture video. Because it is not possible to store all views on the server, it is important to create natural compositing viewpoints; therefore, generating synthesized viewpoints quickly is an important issue to be addressed.

B. Media Presentation Description (MPD)

MPD is a document format in DASH that defines rules that allow clients to access and download media segments [4]. The MPD does not control the actual playback; however, media files are segmented according to the MPD so that the user can adaptively receive the service.

Fig. 1 shows the data model of the MPD [4]. Through this structure, DASH dynamically solves problems such as synchronization and playback time of videos. Therefore, considering that the FVV streaming service is based mainly

on DASH, it may be very effective to include information that can facilitate view synthesis in MPD.

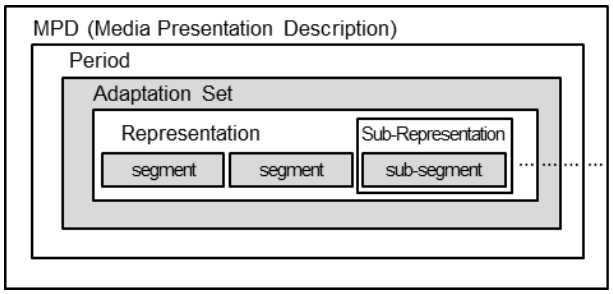

Fig. 1. Data Model of MPD in DASH [4]

III. Encapsulation Methods for Stable Free-viewpoint Video Services

The overall service configuration information may vary depending on where the service provider decides to perform view synthesis. Recent streaming services are considered to synthesize views on the client side [6]. This is because when a user requests a viewpoint for which original video does not exist, the time required for the server to synthesize and transmit the video is expected to be longer than that required by the client. However, because the performance of the synthesis program is continuously improving and the processing speed depends on the performance of the user network environment and the device, it is necessary to consider performing view synthesis on the server side. Moreover, if there are more tasks to be performed on the client side, the service processing speed can be slower and delays can occur. Therefore, in this study, we propose methods for encapsulating the information required by the client to synthesize viewpoints efficiently, and a method for encapsulating additional information so that the video can be stably provided from the server.

A. Encapsulation Methods for View Synthesis on the Client side

In order to perform view synthesis on the client side, it is important to map the relationship between the original texture video and depth video. If individual files without reference information are delivered to the client, fitting temporal and spatial information becomes difficult and time-consuming. Therefore, we propose two encapsulation methods for the client side to provide relationship information between two videos.

1) AdaptationSet in MPD

The inclusion of video information in the AdaptationSet in MPD is the most applicable method because free-viewpoint video streaming service based on DASH is mainly considered.

The AdaptationSet in MPD provides information such as screen information, accessibility, and the ContentComponent, which indicates the feature content that is included in the AdaptationSet. Table 1 shows the elements of the ContentComponent.

TABLE I. Elements and attributes of ContentComponent [4]

Element or Attribute Name	Use
ContentComponent	
@id	O
@lang	O
@contentType	O
@par	O
Accessibility	0…N
Role	0…N
Rating	0…N
Viewpoint	0…N

Elements such as *id* and *contentType* can be used to indicate whether original texture video and depth video exist, and they can be distinguished by specifying their type. Elements such as *Role* are used to identify the role schema that defines the role of the media in the ContentComponent [4].

Depth video can be generated on the client side by using original texture video; however, usually the server stores and manages texture video and depth video separately to reduce the view synthesizing time. When providing a free-viewpoint video streaming service based on DASH, the MPD document can be configured for each view. Therefore, it is the most stable method in the commercial system that does not require additional design or methods to provide video information using the AdaptationSet information.

2) Sub Track in ISOBMFF

When texture video is encapsulated with ISO base media file format (ISOBMFF), the depth video can be considered to be encapsulated at the same time as the sub-structure. To be suitable for transmission, the media is encapsulated in ISOBMFF, commonly called MP4. Sub track box in ISOBMFF is a user data box that matches the track box proposed for media, replacing the main track or the conversion track [7].

Table 2 shows the sub track information box *stri* contained in the sub track box *strk*. If the depth video is included as a sub track, the values of the *alternate_group* and *sub_track_id* should be set. In case of the *sub_track_id*, it is necessary to specify when sub track exists. The *Alternate_group* specifies a set of sub tracks or tracks. If it has a non-zero value, it implies that the track contains alternate data. A description of the sub track can also be added through the value of the *attribute_list*. The sub track box is designed to fit media such as SVC or MVC, which is effective to solve synchronization problems.

TABLE II. SYNTAX OF SUB TRACK INFORMATION BOX [7]

```
aligned(8) class SubTrackInformation
    extends FullBox('stri', version =0, 0) {
    template int(16) switch_group = 0;
    template int(16) alternate_group = 0;
    template unsigned int(32) sub_track_id = 0;
    unsigned int(32) attribute_list[];  // to the end of the box
}
```

B. *Encapsulation Method for View Synthesis on the Server Side*

It takes time to create and deliver an intermediate point on the server. The more the points that need to be synthesized, the more will be the delay. Therefore, if the network condition is not good, reducing the number of synthesis points is effective for providing a stable service. In this case, the number of positions constituting the entire service information map may be variable. If the number of viewpoints is fixed, the network situation is not good, and when the number of intermediate viewpoints is reduced, there is no image of the value to be referenced. Therefore, by specifying relative values for each position, the user can receive an appropriate value according to the network situation when selecting the desired reference position.

Table 3 shows the syntax of the box designed for the FVV service in a previous study [8]. We refer to the IDs of five locations in total. The synthesis has already been performed on the server side; it can be utilized as the service location map information, which can be referred to when moving the viewpoint. Depending on the situation of the network, the reference ID value of a view, which can be accessed, can be changed. The ID rule of the video is very important once the position of the entire position map information is determined. It is expected that the box will be used effectively if the ID rule is set specifically.

TABLE III. SYNTAX OF PROPOSED FREEVIEWPOINTVIDEOBOX

```
aligned(8) class FreeViewpointVideoBox
    extends Box('FVVB') {
    unsigned int(8) viewSelection;
    unsigned int(8)[16] subViewId;
    unsigned int(32)orgVideoId;
    if (orgVideoId == 0) {
        //it is for the synthesized Video
        unsigned int(32) left_VideoId;
        unsigned int(32) right_VideoId;
        unsigned int(32) top_VideoId;
        unsigned int(32) bottom_VideoId;
     }
     unsigned int(8) objectFocused;
     unsigned int(8) zoomFlag;
}
```

IV. CONCLUSIONS

In this paper, we proposed three methods to encapsulate media and information for free-viewpoint video streaming services. Two methods were proposed for providing view synthesis on the client side. Including information in the AdaptationSet is a reliable method to provide video information using elements and attributes in a commercial system without requiring additional design or methods. We also proposed a method to include depth video and related information in the sub track box of ISOBMFF, which is effective in solving the synchronization problem.

In the case of synthesizing on the server side, a method that includes the proposed box was suggested. Reducing the number of intermediate synthesis points adaptively can be effective in providing stable service if the network is not in good condition. Through this, users can access stable free-viewpoint video streaming services.

In future study, we will perform experiments using the above three methods to verify their efficiency in providing stable services.

ACKNOWLEDGMENT

This research is supported by the Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2018 [R2017030041, Experience Maximization Technology for Cultural Contents at Free Selection View].

REFERENCES

[1] Ahmed Hamza, Mohamed Hefeeda, "A DASH-based Free Viewpoint Video Streaming System", in Proc. of the ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, March, 2014, pp. 55–60.

[2] J.K. Yun, S.W. Park, K.J. Koo, M.K. Han, J.H. Jang, "The Technical Trend of Super Multi-View Media Transmission System", in 2016 Electronics and Telecommunications Trends, 2016, pp. 70-79.

[3] Arpad Huszak, "Advanced Free Viewpoint Video Streaming Techniques", Multimedia Tools and Applications, January, 2017, pp.373-396.

[4] ISO/IEC 23009-1, "Information technology—Dynamic adaptive streaming over HTTP (DASH)—Part 1: Media presentation description and segment formats", 2014.

[5] Tianyu Su, Ashkan Sobhani, Abdulsalam Yassine, Shervin Shirmohommadi, Abbas Javadtalab, "A DASH Based HEVC Multi-View Video Streaming System", Journal of Real-Time Image Processing, May, 2015, pp.1-25.

[6] Aljoscha Smolic, "3D video and free viewpoint video—From capture to display", Pattern Recognition vol.44, September, 2010, pp-1958-1968.

[7] ISO/IEC 14496-12, "Information technology—Coding of audio-visual objects—Part 12: ISO base media file format", 2015.

[8] Minjae Seo, Jong-Ho Paik, "An Encapsulation Method with Media Relation Information for Free-Viewpoint Video Service", 2017 Korea Internet Information Society Fall Conference Proceedings Vol. 18, Issue 2, 2017.