International Journal of Internet, Broadcasting and Communication Vol.12 No.2 90-97 (2020)
http://dx.doi.org/10.7236/IJIBC.2020.12.2.90

IJIBC 20-2-13

Implementation of AR Remote Rendering Techniques for Real-time Volumetric 3D Video

Daehyeon Lee[1], Munyong Lee[1], Sang-ha Lee[2], Jaehyun Lee[3], Soonchul Kwon[1*]

[1]*Researcher, Graduate School of Smart Convergence, Kwangwoon University, Seoul, Korea*
[1*]*Professor, Graduate School of Smart Convergence, Kwangwoon University, Seoul, Korea*
[2]*Researcher, Dept. of Electronic Engineering, Kwangwoon University, Seoul, Korea*
[3] *Researcher, Dept. Plasma Bio Display, Kwangwoon University, Seoul, Korea*
*{dleogus13813, ansdyd0803, kcv456, noa6142, *ksc0226}@kw.ac.kr*

Abstract

Recently, with the growth of mixed reality industrial infrastructure, relevant convergence research has been proposed. For real-time mixed reality services such as remote video conferencing, the research on real-time acquisition-process-transfer methods is required. This paper aims to implement an AR remote rendering method of volumetric 3D video data. We have proposed and implemented two modules; one, the parsing module of the volumetric 3D video to a game engine, and two, the server rendering module. The result of the experiment showed that the volumetric 3D video sequence data of about 15 MB was compressed by 6-7%. The remote module was streamed at 27 fps at a 1200 by 1200 resolution. The results of this paper are expected to be applied to an AR cloud service.

Keywords: *augmented reality; mixed reality; remote rendering; volumetric 3d video; WebRTC*

1. Introduction

AR (augmented reality), along with AI, Big-data, and 5G Network, is a core technology that is leading the 4th industrial revolution, and it realizes mixed reality services. With the construction of 5G network infrastructure, it is possible to provide ultra-high speed and ultra-connected AR services with an ultra-low delay. It is expected to grow vastly with real-time services of mixed reality [1][2]. Traditionally, directly the AR service implementation methods load the 3D data stored in client device. Due to the limitation of client device hardware spec, optimization of the 3D data is required, so the quality may be degraded. Therefore, recently, a variety of cloud-based AR remote rendering technologies have been proposed. This is a new mixed reality service technology that enables high-quality interactive 3D data to be rendered in the cloud server and streamed to a client device in real time [3-5].

2. BACKGROUND THEORY

2.1 Volumetric 3D Video

Manuscript Received: March. 17, 2020 / Revised: March. 24, 2020 / Accepted: March. 29, 2020
Corresponding Author: ksc0226@kw.ac.kr
Tel: +82-2-940-8637, Fax: +82-50-4174-3258
Graduate School of Smart Convergence, Kwangwoon University, Korea

Volumetric 3D Video is a technology that captures 3D space. There is a method of using multiple RGBD cameras as a volumetric 3D video capture system [5][8]. Figure 1 (a) shows the volumetric 3D video acquisition process. First, since it uses multiple RGBD cameras, it is a genlock sync process. In a system using two or more cameras, the slave sync generator is synchronized with the master sync generator. In order to prevent the time difference between devices, it is connected to each video signal generator through a sync generator [9]. Secondly, RGB and depth information is obtained from a synchronized multiple RGBD camera, and extrinsic parameters (R, t) are obtained through a calibration process. Calibration is the process of matching the coordinate system of each camera to one coordinate system [10]. In general, a charuco board that combines a QR code and a chess board is used. To obtain (R, t) values based on the reference camera, pose estimation is performed on the charuco board. After calculating the affine transformation between the camera and the board, a matched 3D point cloud is obtained. Figure 1 (b) shows the calibration process using a charuco board. Figure 1 (c) shows the generated 3D point clouds.

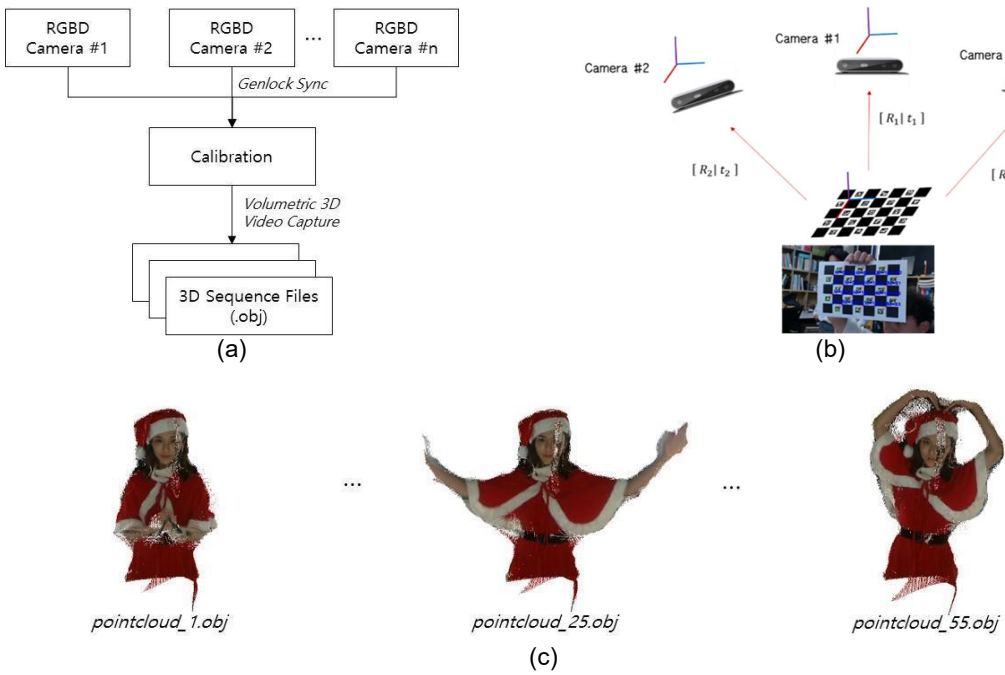

Figure 1. Volumetric 3D Video System (a) Workflow (b) Calibration (c) 3D Point Cloud Sequence Files from Volumetric 3D Video Capture

2.2 AR Remote Rendering

In general, a mixed reality implementation is a method in which 3D data is stored in a client device and loaded. In this case, since the specifications of the client are limited, it is implemented through mesh optimization of 3D data. As another method, a method of rendering 3D data on a server and rendering it to a remote client has been proposed. This can improve rendering quality by using a powerful server. This method is generally expressed in terms of cloud rendering, server rendering, and remote rendering. S. Shi et al. [3] shows the concept of the remote rendering method according to the 3D data type as shown in Table 1. Model-based rendering is a method of transmitting 3D data (mesh or point cloud) to a client. The client needs powerful hardware for 3D graphics rendering, and the server computes 3D data processing. There are original model, partial model, simplified model, and point cloud [11-13]. Image-based rendering is a method of rendering all 3D models on the server and sending only the information of the images to the

client. Therefore, hardware for 3D graphic rendering is not required on the client. There are multiple depth images, depth image, environment map, and image impostor methods [14-15].

Table 1. Classify remote rendering based on 3d data types

Rendering Type	Data Type	Network Bandwidth Computation On Clients
Model-based Rendering	Original Model	High
	Partial Model	↑
	Simplified Model	
	Point Cloud	
Image-based Rendering	Multiple Depth Images	
	Depth Image	
	Environment Map	↓
	Imaged Impostor	Low

3. PROPOSED METHOD

In this paper, we propose two methods. The first is for parsing volumetric 3D video data in real time to a game engine. The second method is to render peer to peer from a game engine through WebRTC. Figure 2 shows the AR remote rendering workflow from a volumetric capture system to clients. The volumetric capture system generates point cloud binary data at a rate of 30 fps. The game engine running on the server performs real-time parsing and creates a virtual camera looking at the object. At this time, the virtual camera was able to work with the client's location value. The virtual camera is streamed to a mobile client web browser by using peer-to-peer WebRTC. Its streaming data is the 2D scene video information viewed by the client.

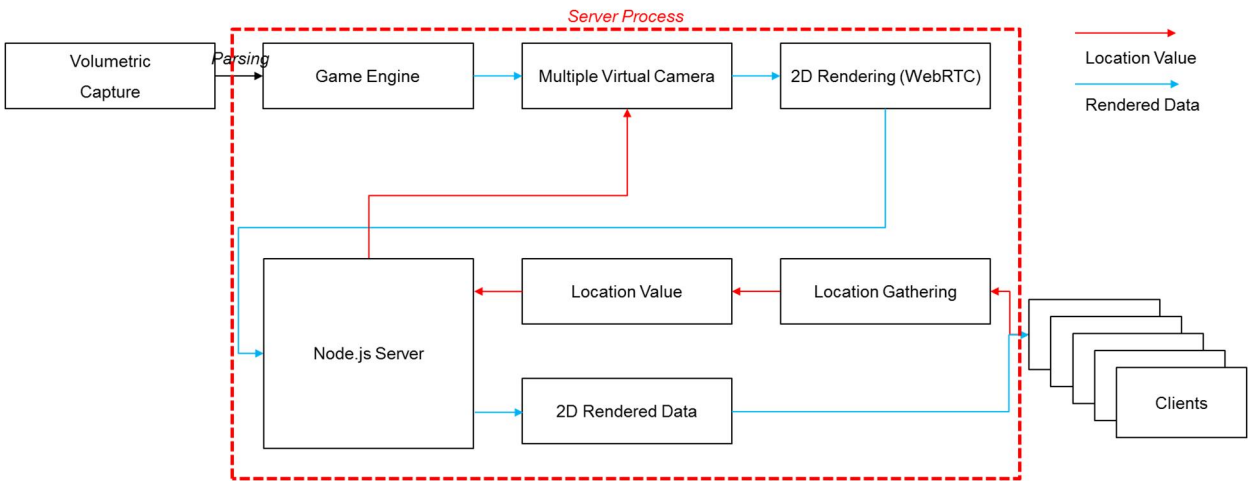

Figure 2. Proposed Method

3.1 Parsing Module

Data output from the volumetric capture system requires the compression process for real-time parsing. Figure 3 shows the proposed parsing module. The proposed system compresses in the divider module and

generates binary data. And these transmit to the game engine parsing module.

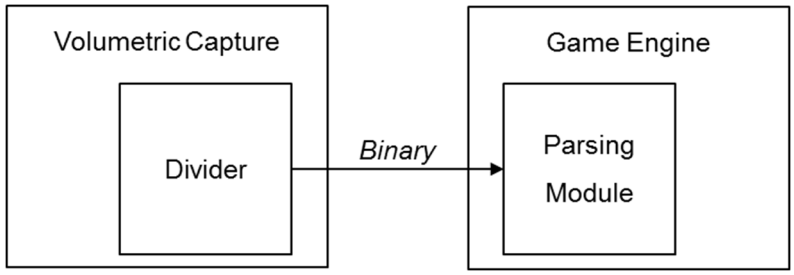

Figure 3. Proposed 'Parsing' Method

The divider module compresses through the data comparison operation. The 3D data sequence is divided into an overlap section and an add section by comparing previous frame data and current frame data. The overlap part is not updated in the parsing module, and the data parsed in the previous frame is used. Only the add part is parsed. Figure 4 shows the concept of the overlap part and the add part.

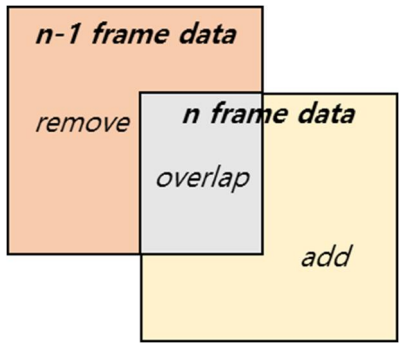

Figure 4. Proposed 'Divider' Method

One 3D frame data is composed of multiple point clouds. Each point cloud compares the x, y, z coordinate values and the r, g, b color values. For the data comparison operation, the $p_{feature}$ value is obtained. The $p_{feature}$ is a representative value of each 3D frame data. This is the calculation method of the $p_{feature}$ in Equation 1.

$$p_{feature} = 10^{33}x + 10^{25}y + 10^{17}z + 10^{9}r + 10^{6}g + 10^{3}b \ \ (x, y, z, r, g, b \geq 0) \tag{1}$$

Here, x, y, z each have vertex values represented by 8 digits, and r, g, b each have color values represented by 3 digits. Therefore, the $p_{feature}$ is the only feature value. Equations 2 and 3 show the comparison operation of the divider module.

$$p_f = \{p_0, p_1, \ldots p_n\} \tag{2}$$

$$\{p'_{f-1} - p'_f = 0 \; p'_{f-1} - p'_f \neq 0 \quad (p'_{f-1} \in p_{f-1}, p'_f \in p_f) \tag{3}$$

p_f in equation 2 is a set consisting of n points in the f-th frame. p'_f is an arbitrary element value of p_f. Equation 3 defines the overlap part and the add part. If the difference between p'_f and p'_{f-1} is 0, it is the overlap part. Conversely, if the operation result is not 0, it is the add part.

3.2 Server Rendering Module

Figure 5 shows that video output from the game engine's virtual camera is streamed to the client based on WebRTC. WebRTC is an open framework for the web that supports Real-Time Communications (RTC) functionality in the browser. Supports transmission of video, voice and general data in peer to peer method. It is also implemented as an open web standard and is provided as a JavaScript API in the browser [16-17]. The virtual camera streams the video by applying the client's location value. At this time, the video is 2D scene information. The location value is divided into position data and rotation data. The position data transfers the client's position value using SLAM (Simultaneous localization and mapping). The rotation data transmits accelerometer data and gyroscope data of the IMU sensor embedded in the client.

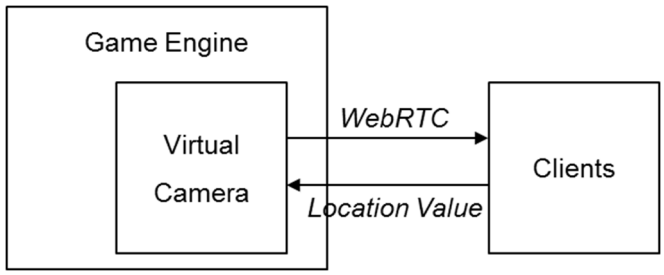

Figure 5. Proposed 'Rendering' Method

4. EXPERIMENT AND RESULT

4.1. Experiment Environment

Table 2 shows the experimental environment. The communication environment used Wi-Fi. Server specifications are as follows. The processor used Intel Core i7-8700 CPU @ 3.20GHz, and the GPU used NVIDIA GeForce RTX 2060. The memory used was 16GB RAM. The 3D data used in the experiment has a size of 13 to 14 MB per frame. The client used Samsung Galaxy Note 10+ (SM-N976 N) for experiments in the mobile environment. The rendered video was implemented in the Chrome Mobile Web Browser, and the experimental measurement tool used 'Google DevTools' in a mobile debugging environment [18-21].

Table 2. Experiment Environment

Classification		Information
Server	CPU	Intel i7-8700 CPU@ 3.20GHz
	GPU	NVIDIA GeForce RTX 2060
	RAM	16GB
3D Model Data	Type	Binary
	Structure	(x, y, g, r, g, b)
	Size	13-14MB
Rendering Resolution		1200 by 1200
Network		Wi-Fi
Used Browser		Google "Chrome"
Mobile Device		SM-N976N (Samsung Galaxy Note 10+)
Analysis Program		Google DevTools

4.2. Experiment Result

The results of the experiment are split into three parts. The first part (figure 6) shows the results of the divider module experiment. The second (figure 7) explains the results of the initial delay time, whilst the

final section (figure 8) introduces the results of the server rendering speed time.

The line graph below (figure 6) shows two lines. The lower line represents the parsing data with divide module (PDDM) and the above live shows the original data (OD). The result showed that the PDDM line is more compressed by between 1-2 MB for each frame than that of the OD line.

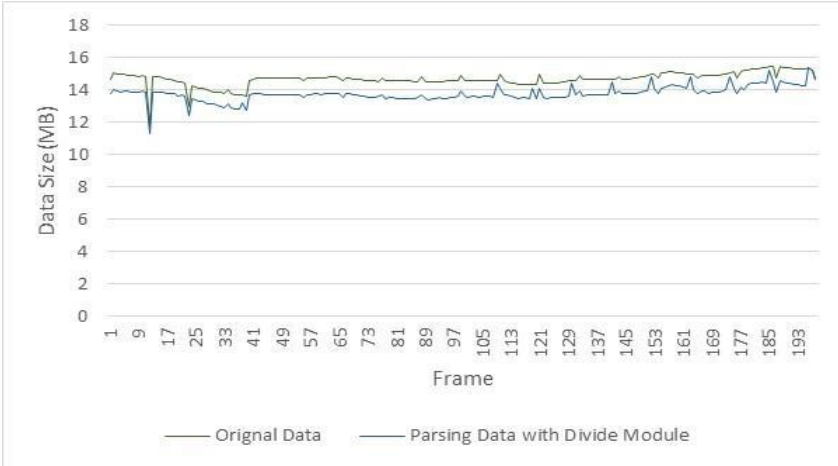

Figure 6. Divider Module Experiment Result

Figure 7 shows the result of measuring the initial delay time. The experiment was conducted in Google Chrome browser of the Samsung Galaxy Note 10+ 5G. Wireshark was used to analyze the packet transmission. The I/O graph shows packets per second over time. The experimental result showed that the initial delay time was 4 seconds.

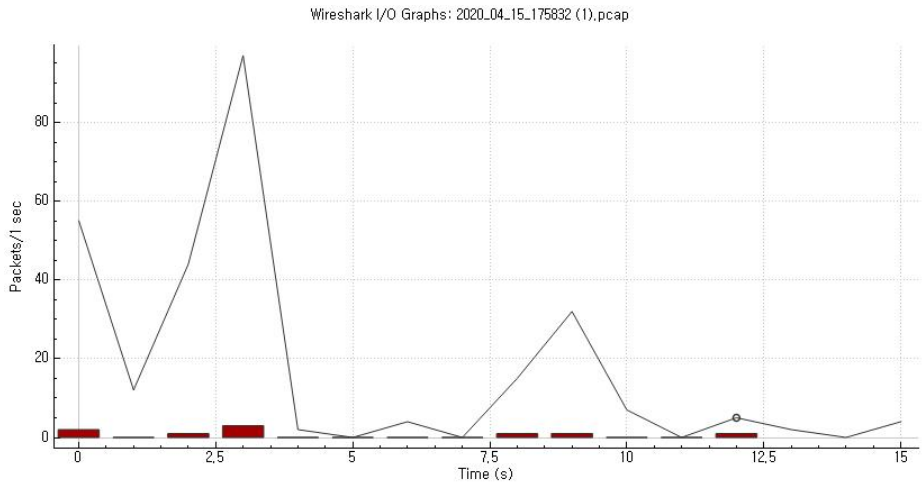

Figure 7. Initial Delay Time Result (I/O Graph)

Figure 8 shows the streaming speed results using the WebRTC. As a result of the experiment, it was confirmed that the mobile web browser rendered at a rate of about 25 -27 fps.

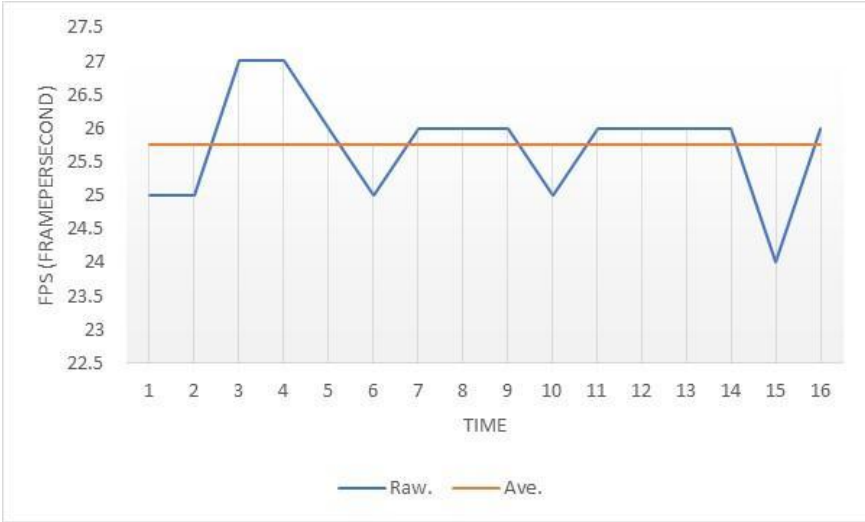

Figure 8. Server Rendering Speed Result

5. CONCLUSION

There have been previous studies, and the result of this study shows the possibility of the applicability of AR cloud service. This study is meaningful for a server-client service of volumetric 3D capture system data. The proposed method is as follows. The first is a server real-time parsing method of sequence 3D data obtained from volumetric 3D capture. The second is a method of rendering from a server to a mobile web client. Additional research is required for the complete augmented reality service in the mobile web browser. First, it is necessary to implement the camera permission of the smart mobile phone in the browser. Second, cross platform support is required. Finally, multi-client should be supported.

ACKNOWLEDGEMENT

This research was supported by Cultural Technology R&D Project through the Korea Creative Content Agency (R2019050033_00000001121082086772010135 1).

REFERENCES

[1] Luis Muñoz-Saavedra, Lourdes Miró-Amarante, and Manuel Domínguez-Morales, "Augmented and Virtual Reality Evolution and Future Tendency," Applied Sciences, Vol. 10, No. 1, 2020. DOI: http://doi.org/10.3390/app10010322.

[2] Kiara Ottogalli, Daniel Rosquete, Aiert Amundarain, Iker Aguinaga, and Diego Borro, "Flexible Framework to Model Industry 4.0 Processes for Virtual Simulators," Applied Sciences, Vol. 9, No. 23, 2019. DOI: http://doi.org/10.3390/app9234983.

[3] Shu Shi, Klara Nahrstedt, and Roy Campbell, "A Real-Time Remote Rendering System for Interactive Mobile Graphics," ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 8, No. 3s, Article 46, 2012. DOI: http://doi.org/10.1145/2348816.2348825.

[4] Lee, K., Chu, D., Cuervo, E., Kopf, J., Degtyarev, Y.,Grizan, S., Wolman, A., and Flinn, J. Outatime, "Using speculation to enable low-latency continuous interaction for mobile cloud gaming," In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services, pp. 151–165, 2015. DOI: http://doi.org/10.1145/2867070.2867076.

[5] Shahram Izadi et al., "Holoportation: Virtual 3D Teleportation in Real-time," Microsoft Research, 2016. DOI: http://doi.org/10.1145/2984511.2984517.
[6] Li Lin, Xiaofei Liao, Guang Tan, Hai Jin, Xiaobin Yang, Wei Zhang, and Bo Li, "LiveRender: A Cloud Gaming System Based on Compressed Graphics Streaming," Conference Paper in IEEE/ACM Transactions on Networking, 2014.
[7] Teemu Kämäräinen, Matti Siekkinen, Jukka Eerikäinen, Antti Ylä-Jääski, "CloudVR: Cloud Accelerated Interactive Mobile Virtual Reality," ACM Multimedia Conference, 2018. DOI: http://doi.org/10.1145/3240508.3240620.
[8] Andrés Fuster-Guilló, Jorge Azorín-López, Juan Miguel Castillo Zaragoza, Luis Fernando Pérez Pérez, Marcelo Saval-Calvo, and Robert B. Fisher, "3D Technologies to Acquire and Visualize the Human Body for Improving Dietetic Treatment," Proceedings, Vol. 31, No. 1, 2019. DOI: http://doi.org/10.3390/proceedings2019031053.
[9] J. Kovacs, "An Overview of Genlock," MicroImage Video Systems, 2001.
[10] Chenyang Zhang, Teng Huang, and Qiang Zhao, "A New Model of RGB-D Camera Calibration Based on 3D Control Field," Sensors, Vol. 19, No. 23, 2019. DOI: http://doi.org/10.3390/s19235082.
[11] Eisert, P. and Fechteler, P., "Low delay streaming of computer graphics," In Proceedings of the 15th IEEE International Conference on Image Processing, pp. 2704–2707, 2008. DOI: http://doi.org/10.1109/ICIP.2008.4712352.
[12] Duguet, F. and Drettakis, G., "Flexible point-based rendering on mobile devices," IEEE Trans. Comput. Graph. Appl., Vol. 24, No. 4, pp. 57–63, 2004. DOI: http://doi.org/10.1109/MCG.2004.5.
[13] Baratto, R. A., Kim, L. N., and Nieh, J., "Thinc: A virtual display architecture for thin-client computing," In Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP'05). ACM, pp. 277–290, 2005. DOI: http://doi.org/10.1145/1095809.1095837.
[14] Lamberti, F. and Sanna, A., "A streaming-based solution for remote visualization of 3D graphics on mobile devices," IEEE Trans. Vis. Comput. Graph., Vol. 13, No. 2, pp. 247–260, 2007. DOI: http://doi.org/10.1109/TVCG.2007.29.
[15] Noimark, Y. and Cohen-Or, D., "Streaming scenes to mpeg-4 video-enabled devices," IEEE Comput. Graph. Appl., Vol. 23, pp. 58–64, 2003. DOI: http://doi.org/10.1109/MCG.2003.1159614.
[16] Alan Johnston, John Yoakum, and Kundan Singh, Avaya Inc, "Taking on WebRTC in an Enterprise" IEEE Communications Magazine, Vol. 51, pp. 48-54, 2013. DOI: http://doi.org/10.1109/MCOM.2013.6495760.
[17] WebRTC, *https://webrtc.org/.*
[18] Geonhee Lee, Pyeong-ho Choi, Hwa-seop Han, Seunghyun Lee, and Soonchul Kwon, "A Study on the Performance Comparison of 3D File Formats on the Web," International Journal of Advanced Smart Convergence (IJASC), Vol. 8, No. 1, pp. 65-74, 2019. DOI: http://doi.org/10.7236/IJASC.2019.8.1.65.
[19] Geonhee Lee, Seunghyun Lee, and Soonchul Kwon, "A study on Compression of 3D Model Data and Optimization of Website," Journal of Engineering and Applied Sciences (JEAS), Vol. 14, No. 1, pp. 3934-3937, 2019. DOI: 10.3923/jeasci.2019.3934.3937.
[20] Duckkyoun Nam, Daehyeon Lee, Seunghyun Lee, and Soonchul Kwon, "Performance Compression of 3D File Formats on a Mobile Web Brower," International Journal of Broadcasting and Communication (IJIBC), Vol. 11, No. 2, pp. 31-42, 2019. DOI: http://doi.org/10.7236/IJIBC.2019.11.2.31.
[21] Duckkyoun Nam, Daehyeon Lee, Seunghyun Lee, and Soonchul Kwon, "A Comparative Study 3D Data Performance in Mobile Web Brosers in 4G and 5G Environments,," International Journal of Broadcasting and Communication (IJIBC), Vol. 11,No. 3,pp. 8-19, 2019. DOI: http://doi.org/10.7236/IJIBC.2019.11.3.8.