Implementation and Demonstration of Real-time Point Cloud Streaming System using HoloLens

Yumeka Chujo[1)], Kenji Kanai[2)] and Jiro Katto[1)2)]

[1)]*Department of Computer Science and Communication Engineering, Waseda University*
[2)]*Waseda Research Institute for Science and Engineering*
3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan

ytay14.08e-w75@ruri.waseda.jp, k.kanai@aoni.waseda.jp, katto@waseda.jp

2022 IEEE International Conference on Consumer Electronics - Taiwan | 978-1-6654-7050-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCE-TAIWAN55306.2022.9869044

***Abstract*—Demands for high immersive communications using 3D content are increasing. In order to contribute to development of this immersive communications, this paper reports a prototype implementation of real-time point cloud streaming system and shows live-action demonstrations of live point cloud streaming using HoloLens. From the demonstration, we confirm that the prototype system could have a potential capability of realization of live streaming without any point cloud playback interruptions.**

Keywords—Point Cloud Streaming, Mixed Reality, HoloLens, Immersive Communication

I. Introduction

Recently, demands for high immersive communications using 3D content are increasing to improve user experience of various services, such as entertainment, remote work, and remote education. To address this fact, 3D point cloud streaming is one of promising technologies.

Regarding point cloud streaming, many researchers have studied rate adaptation methods and frameworks [1, 2, 3] extended from 2D video streaming, like MPEG-Dynamic Adaptive Streaming over HTTP (DASH) [4]. Although these previous works could improve Quality of Service (QoS) and Quality of Experience (QoE), their evaluations are limited to theoretical models and computer simulations. In addition, there are some reports of real-time point cloud streaming system [5, 6], but they do not publish their source codes, and we cannot customize their systems freely.

Therefore, inspired by the previous research efforts, we develop an open framework of real-time point cloud streaming system and contribute to the development of high immersive communications. Based on this motivation, as a preliminary step, this paper reports a prototype implementation of real-time point cloud streaming system and shows live-action demonstrations of both on-demand and live point cloud streaming using HoloLens.

II. Implementation of Point Cloud Streaming System

Figure 1 shows a system architecture of our prototype. As shown in the figure, in the server side, there are three functionalities. A capture handler captures RGB and Depth (RGB-D) images from a LiDAR camera connected via a USB cable or a network. Once the server obtains the RGB-D image(s), a point cloud handler generates point cloud data and performs some 3D data processing. For example, this handler performs voxel down-sampling and removes wall areas to decrease data size of the point cloud frame. Because the necessity of wall area depends on application, this process is optional, and the user can select it. After point cloud data is generated, it is saved as a PLY format. In this paper, we defined the PLY file as a point cloud "frame." At the same time, meta information of point cloud frame is also saved as a JSON format. This meta file contains a frame number, data size, frame rate and so on, and this information will be used for point cloud streaming, like a media presentation description (MPD) file for DASH. All these functionalities are written in Python and implemented by using Open3D [9] which is a well-known library for 3D data processing. After the point cloud frames are ready, they are published on a point cloud streaming server (e.g., Apache HTTP server).

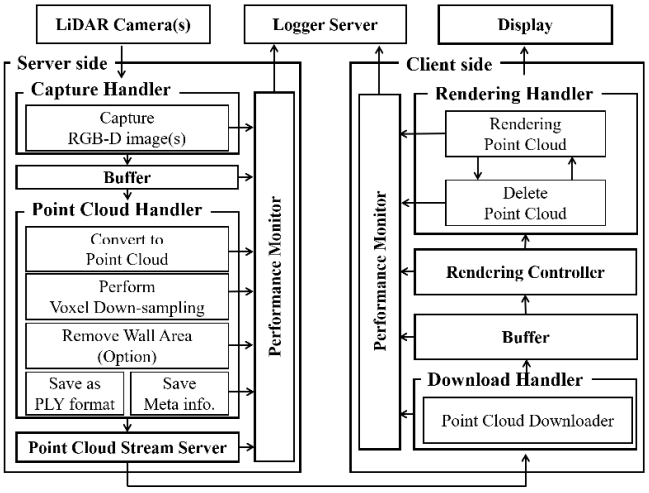

Fig. 1 System architecture of our `point cloud streaming` system.

In the client side, there are three functionalities. The download handler firstly obtains the meta file from the streaming server and sets the configuration of the client. Then, this handler requests the point cloud frames according to the meta information. Once the download handler obtains the point cloud frame, the rendering controller will get the point cloud frame from a buffer and render it. In the rendering handler, the client updates the displayed point cloud by performing rendering and deleting point cloud alternatively. The update rate (refresh rate) is controlled by the rendering controller. For the client, we use a Microsoft HoloLens 2, so the client is developed by using Unity and written in C#. We

can confirm that the client application can also be executed on Unity running on Windows PC as well as HoloLens 2. We use [7] for point cloud rendering.

In addition, in order to monitor server/client performance, both the server and client have the functionality of performance monitoring, which collects performance information, such as processing time generated by each handler, and outputs them to a logger server. The collected information is visualized in real time on a web browser. We use Grafana for data visualization.

III. Demonstration of Point Cloud Streaming

A. Demonstration environment

In order to validate our prototype system, we demonstrate both on-demand and live point cloud streaming using the protype system. Figure 2 shows an image of demonstration environment. As shown in the figure, the streaming server and the logger server are installed at our laboratory, and the LiDAR camera (Intel RealSense L515) is connected to the streaming server via a USB Type-C cable and captures a laboratory scene. A client user wears HoloLens 2 at home and requests point cloud streaming to the streaming server via the Internet. In this demonstration, we configure the streaming parameters in order not to interrupt the playback of point cloud frame sequence. We carried out the system performance analysis in a preliminary experiment and collected the performance test results. Although the detailed explanation is omitted due to the page limitation, from the test results, we tune the streaming parameters, such as voxel size, frame rate and initial buffer size for on-demand and live streaming. It should be noted that voxel size is a parameter for voxel down-sampling. Data size of point cloud (and number of points) becomes small (sparse) as the voxel size becomes large. Detailed definition is found in the document of Open3D [9].

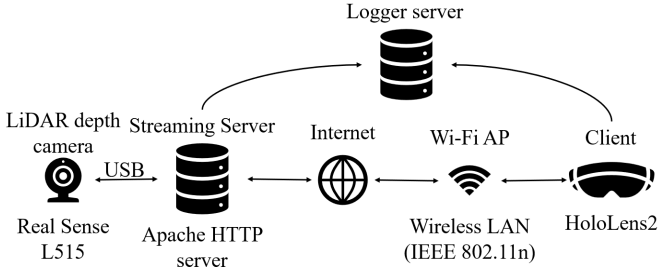

Fig. 2 Demonstration environment.

B. Live streaming demonstration

Table I shows the examples of setting parameters and results of initial delay for the live streaming case. In our demonstration environment, the setting parameters can achieve the smooth point cloud streaming without any playback interruptions. Figure 3 shows a snapshot of HoloLens view while the user enjoys the live point cloud streaming. Demonstration videos for the live-action HoloLens view are published on our GitHub repository [8], and they can help us to understand how the user experience is. The demonstration videos for on-demand streaming case are also published on the same GitHub repository.

Fig. 3 Snapshot of HoloLens view of live point cloud streaming (Demo A).

TABLE I
SETTING PARAMETERS AND RESULTS OF INITIAL DELAY

Demo	Voxel size (m)	Data size (MB)	Frame rate (fps)	Initial delay (sec)
A	0.005	0.603	5	0.685
B	0.007	0.367	10	0.518

V. Conclusion and Future Work

In this paper, to develop an open framework of real-time point cloud streaming system, we implemented a prototype point cloud streaming system and demonstrated live point cloud streaming using our prototype and HoloLens.

In the future, we will consider using WebRTC instead of HTTP streaming protocol to gain download speed and implement adaptive rate control methods based on related work. In addition, we will adopt point cloud compression technologies to our system to reduce data size. Finally, we will publish our system as an open framework on GitHub.

Acknowledgment

This work was partially supported by NICT, Grant Number 03801, Japan.

References

[1] M. Hosseini, et al., “Dynamic Adaptive Point Cloud Streaming,” Proc. of Packet Video Workshop 2018, pp.25 – 30, Jun. 2018.
[2] C. Wu, et al., “Dynamic 3D point cloud streaming: distortion and concealment,” Proc. of ACM Workshop on NOSSDAV 2021., pp.98-105, Sep. 2021.
[3] J. Li, et al., “Joint Communication and Computational Resource Allocation for QoE-driven Point Cloud Video Streaming,” Proc. of IEEE ICC 2020, pp. 1-6, Jun. 2020.
[4] DASH Industry Forum [online]: https://dashif.org/
[5] S. Orts-Escolano, et al., “Holoportation: Virtual 3D Teleportation in Real-time,” Proc. of UIST 2016, pp.741-754, Oct. 2016.
[6] D. Laskos, et al., “Real-time Upper Body Reconstruction and Streaming for Mixed Reality Applications,” Proc. of International Conference on Cyberworlds (CW) 2020, pp. 129-132, 2020 ·
[7] GitHub, “Point cloud importer & renderer for Unity” [online]: https://github.com/keijiro/Pcx
[8] GitHub,“iccetw-chujo”[online]:https://github.com/yumekaC/iccetw-chujo
[9] Open3D, “Point cloud — Open3D 0.15.1 documentation” [online]: http://www.open3d.org/docs/release/tutorial/geometry/pointcloud.html#Voxel-downsampling