Point Cloud Streaming over HTTP/2

Estabraq H. Makiyah* and Nassr N. Khamees+
*Information Technology Academy, University of Information Technology and Communications,Baghdad, Iraq
+College of Information Engineering, Al-Nahrain University, Baghdad, Iraq
Emails: estabraq.h.m@uoitc.edu.iq, nassrnafea@gmail.com

2023 6th International Conference on Advanced Communication Technologies and Networking (CommNet) | 979-8-3503-2939-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/CommNet60167.2023.10365247

***Abstract*—Point clouds with high density have been drawing attention for expressing attractive 3D visuals. Since these volumetric huge media need extensive bandwidth allocation, it would be problematic to stream to devices with restrained resources. In this paper, we propose the techniques of point cloud compression and streaming for large point clouds using a web server. To achieve reliability of streaming HTTP/2, protocol is suggested. Such techniques are built over web server. Storing large point cloud volumetric video in a web server allows users to present their data sets to the public without having to use additional applications or to send large volume data beforehand. HTTP/2 increases transmission efficiency by compressing headers in a binary format, and lowers latency by using one TCP session for sending many multiplexed requests. Our proposed system achieves lower average latency between successive frames resulting of an increased frame rate to 72.46 fps when streaming 4M points, 82.51 and 61 fps streaming 800K points compared to much lower rates in conventional work.**

Keywords— ***Streaming, Point cloud, volumetric video, web service, http/2.***

I. Introduction

Volumetric point cloud is the collection of points expressed in three-dimensional space, each of which is associated with a number of properties such as location and color. Point clouds are used to reassemble a 3D formation or a complete scene consisting of different points. 3D Point clouds can be collected using groups of cameras and depth sensors in a variety of configurations and can include billions of points to reproduce objects after reconstructing them in a dense resolution appearance [1]. Performance enhancement and creativity can be achieved by enabling user interaction with physical environment scenes through smart digital appliances in an immersive style [2]. 3D point clouds have lately spurt as the new interface between user and immersive media by presenting many techniques for communication between real and unreal augmented objects [1].

Usually, point clouds are too large be processed by normal algorithms with normal memory allocation. Out-of-the box methodologies are needed such as splitting the data into various stacks and process one or a few stacks during one unit of time. This method is acceptable for processing, but in case of visualization it is often advisable to display the entire dataset and not just a few cells at once [3].

3D immersive media are used in wide variety, such as healthcare, industry and academic education applications, because they provide great display capabilities and different angle viewing optionality. This, for sure, has drawn attention in many aspects of industry and education. In order to effectively transmit volumetric point cloud video over wireless network, researchers must investigate to overcome many challenges, due to the high video rates and decoding difficulty [4].

In this paper, we aim to show the benefits point cloud compression and using HTTP/2 protocol for streaming point clouds over internet networks. Experimental results outperform regular web-based streaming methods and show lower inter-frame latency rates.

II. Related Work

Authors in both [1] and [4] have presented a DASH-based technique for streaming point cloud video. The server in their work divides point cloud video into equal tiles and then apply encoding to each tile with diverse quality degrees at multiple source rates. While the authors of [5] have proposed a real-time point cloud streaming system with adjustments to the octree compression process, the author of [3] has also modified the octree methodology and presented a point cloud viewer and streamer based on web service.

No one of the above authors have used HTTP/2 as a streaming protocol over internet.

On the other hand, authors in [6] have used this specific protocol for streaming adaptive resolution video. On the same track, authors in [7] have also used HTTP/2 for demonstrating download time reduction for small delay-sensitive items when a simultaneous heavy download is present.

No one of the above researches have considered streaming point cloud volumetric video using HTTP/2 as proposed in this paper.

III. Point Cloud Streaming

Unlike 2D video, volumetric video includes 3D data, allowing the user to view the video with six degrees of freedom (6DoF). Recent evolvements in Virtual Reality (VR) and Augmented Reality (AR), volumetric video suggests many intriguing applications [5].

Viewers may watch parts of the overall volumetric video in VR technology, this is the same case for point cloud users, which is called the field of view (FoV). In order to avoid losing bandwidth during transmission, it is better to transmit packets containing only FoV without the need to send the complete video data. In point cloud streaming, tile can have a different impact on the quality of the video at the receiver, due to the distance between the client and the physical environment [8]. By simply concatenating individual points without compression, you can render in full parallelism at the expense of a lot of data. The transmission of a single raw volumetric video frame, with the size of 4 MB to 15 MB, would demand about 1 Gbps to 3.6 Gbps bit rate, along with the delivering the frame in raw condition. This is definitely not possible for the wireless networks in their current capabilities [5].

Some researchers process point cloud videos in the raw form, by projecting them into two dimensional images. In that

case, third-party applications are required to be additionally installed along with large amount of data, for the purpose of presenting these volumetric videos to interested viewers. At some points, hard disks carrying very large amount of data, should be sent physically by mail [3].

In this paper we propose storing point clouds in an HTTP server and then use characteristics presented by HTTP/2 to stream the content at a lower bitrate using a renderer that is capable of streaming and viewing high density point cloud videos with millions of points, without needing to transmit the complete volumetric video beforehand installing additional viewers.

A. Point Cloud Compression

Up to 4 Gbps of bandwidth might be demanded for transmitting raw point cloud data in human formation. Transmission of a complete scene is more demanding and cannot be conducted within current network capabilities [9].

Due to the cumulative advance of using point cloud, storage of large point cloud data and methods of transmission have become more challenging. In order to complete the compression process, different requirements are demanded. For each type of point cloud, sparse or dense, suitable compression algorithms with necessary adaptations should be assigned.

Other types of point clouds are static and dynamic, according to their alterability during a time interval. The compression can also lossy or lossless, pointing to the ability of restoring complete data at the receiver [10].

Experts in MPEG developed existing typical PCC methodologies [11]. Two popular designs were presented, the Geometry-based PCC (G-PCC) and Video-based PCC (V-PCC). G-PCC depends on 3D models (e.g., octree or triangle surface) for encoding the volumetric data directly, whereas V-PCC projects 3D to 2D in order to run one of the conventional 2D video codecs [12].

Our methodology for streaming is to compress 3D point cloud content using octree geometrical projection which saves processing time.

IV. HTTP/2

The two main strategies for overcoming the performance limitations of HTTP/1.1 [13] are using multiple HTTP connections and making fewer but larger HTTP requests. By using multiple connections, the client can send multiple requests at once, allowing for faster response times. Making fewer but larger requests also helps to reduce latency by reducing the number of transmissions between the client and server. Additionally, larger requests can be more efficiently processed by the server, resulting in improved performance [14].

HTTP/2 was designed to maximize the efficiency of network resources in order to send and transmit data and receive them with lowest latency measures [15]. The push function, which grants the delivery of contents the client before it is requested, has attracted particular interest among multimedia scholars. In addition, HTTP/2 includes a new mechanism for multiplexing the delivery of structured data called HTTP/2 streams [6]. HTTP/2 is a binary protocol that simplifies the process of framing. In HTTP 1.1, it can be difficult to determine the beginning and end of frames. With HTTP/2, one connection transmits many streams simultaneously, with both sides sending frames from different streams at once [16].

A. Header Compression

One of the drawbacks of HTTP/1.1 is the redundant headers that are sent between the streaming server and the browser at the client side for each request and response. In HTTP/1.1, every request must include all the side information for the server to fulfill it, without the server having to store data from previous requests [16] because it is a stateless protocol [17]. Cookies are often used in request headers to maintain state, which can lead to large header sizes, an average of 800 bytes, resulting in wasted bandwidth, which is one of the most-costly resources in browser-server communication [17].

B. Server Push

By enabling the push function, the client receives content pushed by server before requesting it [6]. Without Server Push, the browser must first retrieve the initial HTML page and parse it before requesting any additional objects that the page requires. This means two trips are required for all the contents to be loaded for a basic web page, and more trips for heavier pages. The findings of [18] suggested that a server could transmit more than just a few small files, and that this had a positive effect on network performance.

V. Proposed Methodology Implementation

Our experiment requires storing and rendering 3d point cloud data in web server before starting the process of streaming. This web server is installed on an MSI laptop with Intel(R) Core i7-10750H CPU 2.60GHz, 16 GB RAM, and running Ubuntu 20.04. In this work, apache2 web server is used for its ability to support http/2. Potree is installed inside the apache2 web server and is used as point cloud streamer. Clients can access a point cloud stream inside the web server and receive service over internet network as explained in Fig.1.

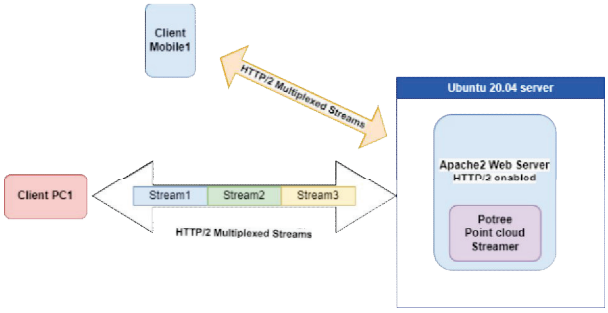

Fig. 1. HTTP/2 stream multiplexing

A. Data Acquisition and Conversion

Point cloud data sets acquired from sensors, such as videogrammetry and 3D laser scanning, RGB-D cameras, and stereo cameras can instantly acquire the geometry of 3D surface of the demanded object in an efficient way [19]. Each point includes required geometry properties i.e. coordinates (x, y, z) with optional attributes like RGB color, reflectance and normal. Usual cameras that were used for 2D video recordings capture only (x,y) coordinates and attributes of

RGB colors. A stereo image added a third z-coordinate and brings the depth attribute using epipolar geometry using two infrared (IR) cameras as least requirements [9].

We capture point cloud in a huge number of (.ply) format which are plain text files. In order to stream point cloud, we convert acquired data into a format that is suitable for storing in the http server and then make necessary compression methodologies so that it can be transmitted over internet.

We experiment streaming samples of dense point cloud datasets. Each sample consists of one spatial resolution for a cube of 1024x1024x1024 voxels, which is known as depth 10 [20]. Fig. 2 shows snapshots of sample frames, while Table I shows details of each.

(a) (b) (c)

Fig. 2. Sample point cloud datasets. (a): loot. (b) red and black [20] (c): lion [3]

TABLE I. SAMPLE POINT CLOUDS

Sample Frame	No. of points	No. of Frames
Lion	4,000,000	200
Loot	784,142	300
Red and Black	729,133	300

We use Python and the accompanying LAS packages to convert point cloud datasets. LAS stands for laser file format [21]; LAS format is a data format that is better use for exchanging three-dimensional point cloud data between data users. LAS supports the transaction of any three-dimensional x,y,z coordinates. Additionally, this format is a binary file format that retains specific information about the nature of the data without being overly complicated. The data conversion methodology is depicted in Fig. 3.

B. Octree Compression

Point cloud shapes are usually preprocessed with voxelization, a type of coordinate quantization. Octree enactment is the process of iterative partitioning of a cubic voxel space into 8 small blocks of identical size. Partitioning of a subblock occurs provided that at least one voxel is occupied within it [22].

In this paper, we use LASzip, which is considered a lossless compressor for LiDAR stored in the LAS format. It provides high compression ratio at unparalleled speed and supports streaming and random-access decompression. LASzip compressors are called streaming compressors. The schema of streaming starts compressing points and create a compressed file without having to read the entire file, and starts decompressing points after reading a part of the received file only and creates a decompressed file. Memory capacity remains minimal compared to the data they process [23]. The LASzip compressor always encodes points within blocks of points to allow searching within the compressed file. The compressor stores a block table at the end of the file because each compressed block has a different size, so this table helps specifying the start byte of each block [24].

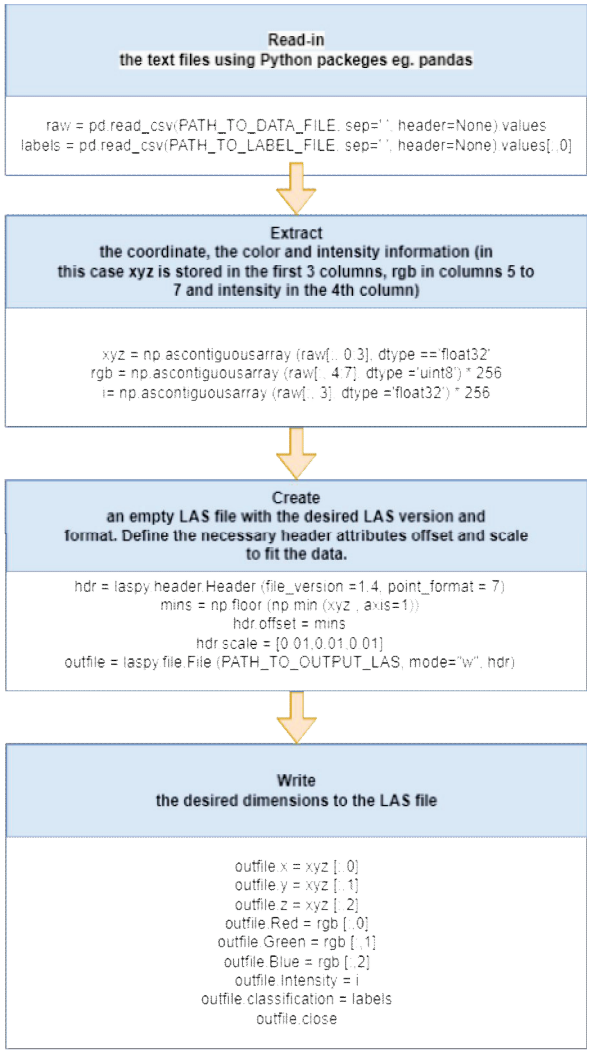

Fig. 3. Point Cloud Dataset Conversion

VI. EXPERIMENTAL RESULTS

By setting up apache2 web server on Ubuntu 20.04 server and enabling HTTP/2 protocol, we can verify the settings as shown in Fig. 4.

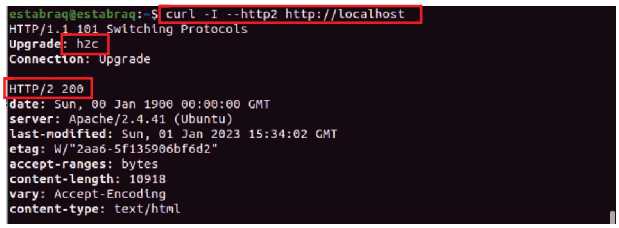

Fig. 4. HTTP/2 enabled for apache2 server

The point cloud data stored in web server can be accessed form an internal virtual box or from any external client, as shown in Fig. 5.

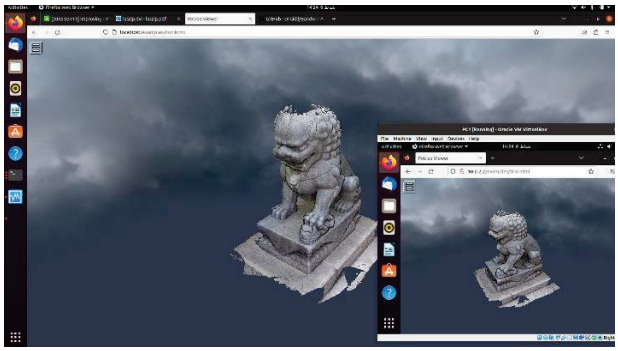

Fig. 5. Accessing stream of sample frame of point cloud (lion) from a virtual box.

Packets carrying HTTP/2 protocols are captured using Wireshark and shown in Fig.6, table where the upgrade to *h2* is clarified.

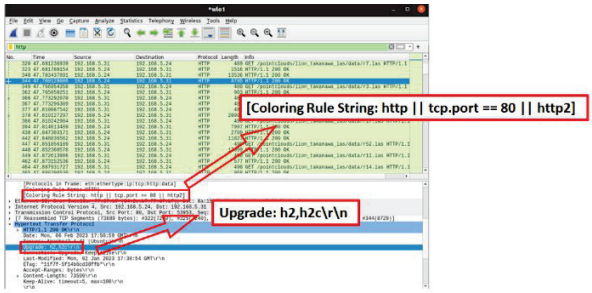

Fig. 6. HTTP/2 packets captured by Wireshark

Using (tcpdump), we are able to capture each of the 200 frames, in the case of (lion) sample, and 300 frames, in the case of (loot and Red and Black) samples, loaded within the page.

Table II shows statistics of 20 frames as an example of the overall data collected using Wireshark while streaming each of the sample point cloud data sets. Columns show the samples of time delta from previous displayed frame measured in milliseconds, along with data carried by that frame and measured in Bytes.

TABLE II. STATISTICS OF TIME DELTA BETWEEN EACH TWO SUCCESSIVE FRAMES AND SIZE OF DATA

Lion		Red and Black		Loot	
Time Delta (ms)	**Size (Bytes)**	**Time Delta (ms)**	**Size (Bytes)**	**Time Delta (ms)**	**Size (Bytes)**
0. 678	874	1.74	827	1.8188	840
0. 00439	185	0.984	1396	0.969	1409
114.1379	3179	135.98	485	0.0233	250
0.10999	491	1.979	495	0.0374	582
2.00769	1408	14.58	2182	0.506	256
0.0002	491	0.043	963	11.683	579
25.013	26373	12.83	491	0.6361	256
0.0106	9082	5.462	473	0.0007	643
9.432	37701	0.037	501	0.0115	632
13.092	5737	0.0004	498	0.4897	247
9.715	491	0.86	1573	0.0242	247
29.746	43097	0.259	4744	206.02	643
0.052	2654	0.116	4694	0.4705	250
0.4502	7132	19.8	484	49.057	643
7.724	10823	0.122	473	0.4843	250
18.96	491	0.002	483	1.232	643
0.0004	491	0.195	1814	0.4711	250
11.743	2937	0.076	2417	53.353	643
6.138	18246	3.25	1147	0.0064	641
9.225	491	44.05	118	0.479	250

In Table III, we calculate the average Time delta between successive frames and the average bitrate of point cloud data transmission

TABLE III. AVERAGE TIME DELTA AND AVERAGE BITRATE

Lion		Red and Black		Loot	
Avg. Time Delta (ms)	**Avg. Bitrate (KBps)**	**Avg. Time Delta (ms)**	**Avg. Bitrate (KBps)**	**Avg. Time Delta (ms)**	**Avg. Bitrate (KBps)**
13.8154	625.027	12.1183	105.032	16.3887	31

In order to calculate frames rate per second, we use equation (1) that divides one second over the time that within which each frame is displayed.

$$Frame\ Rate = {}^{1}/_{Avg\ Time\ Delta} \quad \text{(fps) (1)}$$

Table IV shows that the results of latency of frame display time delta for our proposed system when streaming over HTTP/2 is much less than the results of related experiments considering (lion) point cloud datasets presented by [3] and both (Red and Black) and (Loot) presented by [20].

TABLE IV. COMPARISON OF RESULTS SHOWING THAT OUR SYSTEM OUTPERFORMS PREVIOUS WORK

System	Lion	Red and Black	Loot
[3]	Frame Latency **16.6 ms**	--	--
	Frame Rate **60 fps**		
[20]	--	Frame Latency **33.3 ms**	Frame Latency **33.3 ms**
		Frame Rate **30 fps**	Frame Rate **30 fps**
Our Proposed system	Frame Latency **13.815 ms**	Frame Latency **12.118 ms**	Frame Latency **16.388 ms**
	Frame Rate **72.46 fps**	Frame Rate **82.51 fps**	Frame Rate **61 fps**

The time consumed for rendering objects is usually different for each server, because running the application in a browser environment acts as an extra layer for the GPU. Other reasons might be GPU miss use, or that the communication between the CPU and GPU is inefficient. We conducted the experiment in a normal environment available for most of the users using an MSI Laptop with Intel(R) Core i7-10750H CPU 2.60GHz, 16 GB RAM, running Ubuntu 20.04, a 300 Mbps router, and mobile devices running IOS 16.3 and Android 12. And yet, results of our proposed system of streaming point cloud over HTTP/2 clearly outperform the conventional results of experiments in [3] and [20], by achieving 72.46 fps, while authors in [3] have achieved 60 fps streaming (lion), and we have achieved a frame rate of 82.51

fps, and 61 fps for (Red and Black) and (Loot), respectively, compared to the 30 fps, stated by authors in [20].

Fig. 7 and Fig. 8 show our proposed system's result compared to previous work's.

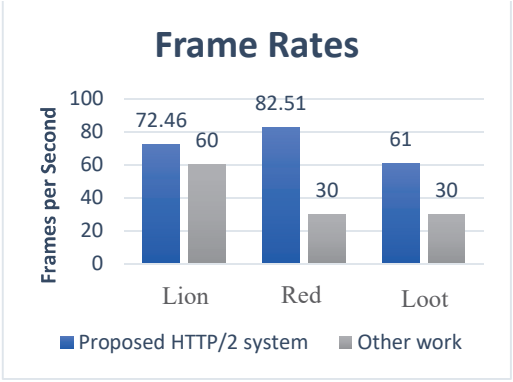

Fig. 7. Frame Rates of proposed system compared to other work

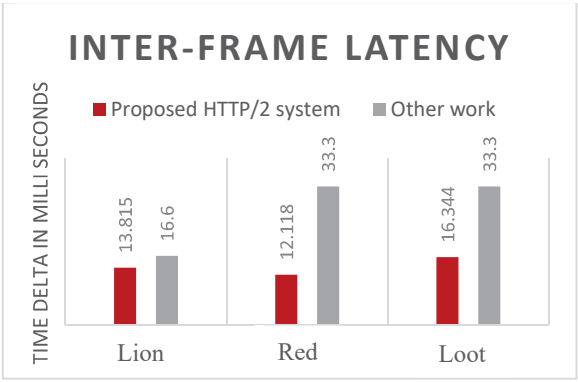

Fig. 8. Average Time delta between successive frames

VII. Discussion and Conclusions

In this work, we capture large number of point cloud data, store them in a web server, apply necessary compression, and then prepare the volumetric video for streaming over HTTP/2 internet protocol.

By comparing our proposed system's result to previous conventional work, we conclude that our system outperforms others' when streaming point cloud volumetric videos in a web-based system using HTTP/2 protocol. The outperformance is clear by decreasing inter-frame latency for large point cloud streaming by 16.8% for the case of (lion) with 4M points, 63.6% for the case of (Red and Black) with 729K points and by 51% for the case of (loot) that consists of 783K points. Frame rates are also increased when receiving the stream by 20.7%, 175% and 103% for the cases of Lion, Red and Loot, respectively.

The experiment has some limitations since it is conducted in an environment that is interference-free. One WIFI connection device is used and one user is receiving the stream. More advanced scenarios can be discussed in a future work.

References

[1] M. Hosseini and C. Timmerer, "Dynamic Adaptive Point Cloud Streaming," *arXiv (Cornell University)*, Jun. 2018, doi: https://doi.org/10.1145/3210424.3210429.

[2] J. Hu, A. Shaikh, A. Bahremand, R. LiKamWa, "Characterizing Real-Time Dense Point Cloud Capture and Streaming on Mobile Devices", University, HotEdgeVideo'21, January 31-February 4, 2022, New Orleans, LA, USA, https://doi.org/10.1145/3477083.3480155

[3] M. Schuetz, "Potree: Rendering Large Point Clouds in Web Browsers", Faculty of Informatics at the Vienna University of Technology, Vienna, 2016.

[4] J. Li, C. Zhang, W. Liu, W. Sun, W. Hu, O. Li, "Narwhal: a DASH-based Point Cloud Video Streaming System over Wireless Networks", Conference: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops, DOI:10.1109/INFOCOMWKSHPS50562.2020.9162937

[5] K. Lee, J. Yi, Y. Lee, S. Choi, Y. Min Kim, "GROOT: A Real-time Streaming System of High-Fidelity Volumetric Videos", MobiCom '20, September 21–25, 2020, London, United Kingdom, https://doi.org/10.1145/3372224.3419214

[6] M. M. Awad, N. N. Khamiss, "Low Latency UHD Adaptive Video Bitrate Streaming Based on HEVC Encoder Configurations and Http2 Protocol", Iraqi Journal of Science, 2022, Vol. 63, No. 4, pp: 1836-1847, DOI: 10.24996/ijs.2022.63.4.40

[7] X. Mi, F. Qian, X. Wang, "SMig: Stream Migration Extension For HTTP/2", CoNEXT '16, Irvine, CA, USA, 2016, DOI: http://dx.doi.org/10.1145/2999572.2999583.

[8] Z. Liu et al., "Point Cloud Video Streaming: Challenges and Solutions," in IEEE Network, vol. 35, no. 5, pp. 202-209, September/October 2021, doi: 10.1109/MNET.101.2000364.

[9] T.K. Hung, "On Error Concealment of Dynamic 3D Point Cloud Streaming", National Tsing Hua University, Master Thesis, 2022.

[10] Ch. Cao, "3D Point cloud compression Multimedia", Institut Polytechnique de Paris,2021, NNT:2021IPPAS015.

[11] S. Schwarz, M. Preda, V. Baroncini, et al., "Emerging mpeg standards for point cloud compression," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, pp. 133–148, 2019.

[12] J. Wang, D. Dingy, Z. Liz, Z. Ma, "Multiscale Point Cloud Geometry Compression", arXiv:2011.03799v1 [eess.IV] 7 Nov 2020.

[13] J. L. Wagner, E. Marcotte, "Web Performance in Action Building Fast Web Pages", December 2016 ISBN 9781617293771.

[14] B. Pollard, "HTTP/2 in Action", 2019 by Manning Publications Co.

[15] C. Müller, S. Lederer, C. Timmerer, and H. Hellwagner. Dynamic adaptive streaming over HTTP/2.0. In Proceedings of the IEEE International Conference on Multimedia and Expo, ICME, pages 1–6, 2013.

[16] D. Stenberg, "HTTP2 Explained", ACM SIGCOMM Computer Communication Review 120 Volume 44, Number 3, July 2014

[17] K. Yamamoto, T. Tsujikawa, K. Oku, "Exploring HTTP/2 Header Compression", CFI'17, June 14-16, 2017, Fukuoka, Japan, 2017 Association for Computing Machinery, https://doi.org/10.1145/3095786.3095787

[18] S. Rosen, B. Han, Sh. Hao, Z. M. Mao, F. Qian, "Push or Request: An Investigation of HTTP/2 Server Push for Improving Mobile Performance", 2017 International World Wide Web Conference Committee (IW3C2), http://dx.doi.org/10.1145/3038912.3052574

[19] Q. Wang, Y. Tan, Z. Mei "Computational Methods of Acquisition and Processing of 3D Point Cloud Data for Construction Applications". Arch Computat Methods Eng 27, 479–499 (2020), https://doi.org/10.1007/s11831-019-09320-4

[20] E. d'Eon, B. Harrison, T. Myers, Ph. A. Chou, "8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva, January 2017

[21] SC. Charleston, "Lidar 101: An Introduction to Lidar Technology, Data, and Applications", National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center. 2012.

[22] D. E. Tzamarias, K. Chow, I. Blanes, J. Sagristà, "Compression of point cloud geometry through a single projection", DCC 2021, Snowbird (UT), USA.

[23] M. Isenburg, LAStools , "LASzip: lossless compression of LiDAR data", 2013, DOI:10.14358/PERS.79.2.209

[24] LAStools, "Efficient tools for LiDAR processing." [Online]. Available: http://www.lastools.org/ (Last visited on 21/11/2023).