Iraqi Journal of Information and Communications Technology(sss)
Vol. 7, Issue 2, August 2024
www.ijict.edu.iq
ISSN:2222-758X
e-ISSN: 2789-7362

VOXEL VIDEO STREAMING OVER WIRELESS NETWORKS

Estabraq H. Makiyah [1], **Nassr N. Khamees** [2]
[1] University of Information Technology and Communications, Baghdad, Iraq
[2] Department of Information and Communication Engineering, College of Information Engineering, Al-Nahrain University, Jadriya, Baghdad, Iraq
estabraq.h.m@uoitc.edu.iq[1], nassrnafea@gmail.com[2]
Corresponding Author: **Nassr N. Khamees**
Received:23/02/2023; Revised:10/06/2023; Accepted:15/08/2023
DOI:10.31987/ijict.7.2.244

***Abstract*- High-density point clouds expressing attractive 3D images are attracting attention. These gigantic media require large bandwidth allocations, making them problematic to stream to resource-constrained hand-held devices. This paper proposes a method for point cloud compression and streaming of large point clouds using a web server. Storing large point cloud videos on a web server allows users to publish data sets without using additional applications or sending large amounts of data ahead of time. HTTP/2 improves transfer efficiency by compressing headers into binary format and reduces latency by sending many multiplexed requests using a single TCP session. The large size of the point cloud data and the current limitations of wireless channels demand point cloud compression methods. MATLAB is used for the purpose of analysis, at results show an average compression ratio of 30:1. Our proposed system achieves lower average latency between successive frames resulting in an increased frame rate to 72.46 fps when streaming 4M points, and 82.51 fps in the case of streaming 800K points compared to much lower rates in conventional work.**

keywords: **Streaming, Point cloud, volumetric video, compression, web service, http/2.**

I. INTRODUCTION

Due to the increasing fascination of virtual and augmented reality activities, there is a growing interest in embracing and integrating users into multidimensional worlds. This proliferation of representations will enable everyone to navigate multi-sensory 3D media experiences [1]. Collecting and managing 3D scene geometry data is necessary for many cutting-edge applications, such as: Interactive VR/AR video, robotic or car navigation, diagnostic imaging. The simplest form of this data is a set of collected points called a point cloud [1]. The term "point cloud" refers to a collection of points consisting of x, y, z coordinates and associated properties like color, normal, reflectance [2]. The geometry or position of individual points and the attributes or other data associated with each of those points can be separated into his two halves or parts of the point cloud. The terms "dynamic point cloud" and "static point cloud" are used to describe point clouds with and without the time dimension [2].

Voxel video applications are used in different applications in healthcare, industry, and academic education, as it provides excellent display capabilities and optional viewing angles from different angles. This has certainly attracted attention in many areas of industry and education [3]. Up to 4Gbps bandwidth may be required to transmit the raw point cloud data of the humanoid. Sending complete scenes is more demanding and not feasible with current network capacity [4]. To effectively transmit volume point cloud video over wireless channel, researchers must overcome many challenges due to high video rates and decoding difficulties [3]. In this paper, we propose a spatial projection method that is a method of compression for point cloud that is required for new immersive voxel video applications.

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

II. RELATED WORK

Researchers mostly depend on the latest standard published in the MPEG Call for Proposals regarding with point cloud compression. The best standards were classified as G-PCC (Geometry Point Cloud Compression) and V-PCC (Video Point Cloud Compression) [1]. In contrast to previous work using binomial projections, the authors of [5] used geometric point cloud compression using a single projection, which outperformed the latter. The author of [6] suggested skipping some frames and applying the filtering method, but for data processing and filtering generated by Kinect he uses Point Cloud Library (PCL). A deep learning-based method for geometrically lossy compression of point clouds was proposed in [7] by adopting a hybrid representation of the point cloud. Researchers in [8] proposed a normal-based intra-prediction method that provides stronger lossless compression for attributes by including point cloud normals. In the same scope, a designed network in [9] was presented with a source of real-time video streaming using a professional video camera connected to a virtual network of routers to deliver the multimedia content wirelessly to the intended classroom. Specific Quality of Service (QoS)- based compression techniques were evaluated.

The authors of [10] and [3] presented a DASH-based method for streaming point cloud videos. The server works by splitting the point cloud video into equal tiles and applying encoding to each tile with different levels of quality at multiple source rates. Authors of [11] have compressed the size of coordinates data collected from many devices attached to an object for the purpose of tracking. Only selected data for the tracked vehicle were sent to the remote http server via constrain the transmission of information with the movement of the vehicles. The author of [12] proposed a real-time point cloud streaming system adapted to the octree compression process, while the author of [13] has used a web server for storing huge size point clouds and changed the methodology of octree for the purpose of streaming, and added a point cloud viewer and presented the streamer. On the other hand, the authors of [14] used this particular protocol for streaming adaptive resolution video. Similarly, the author of [15] also uses her HTTP/2 to demonstrate faster download times for small, latency-sensitive elements in the presence of a large number of concurrent downloads.

The proposed point cloud compression technique suggests a geometry projection method by converting 3D voxelized points into 2D density grid which resulted in a 30:1 compression ratio of voxelized cubes. No one of the above authors have addressed our proposed hybrid system of using HTTP/2 protocol for streaming point cloud voxel video over internet.

III. POINT CLOUD DATA

The collection of different 3D points forms a point cloud. Each point has a 3D location of (x, y, z), which are spatial properties, and in addition to additional qualities like color, surface normal, and reflectance, individual points have defined spatial relationships and orderings. For high quality processing of point cloud data, each entity is quantized into a diced voxel matrix of size $2^{-d} \times 2^{-d} \times 2^{-d}$. Voxels are generated from volume sub-partitions down to (d) Level of Detail (LoD) of the root $1\times1 \times1$ 3D seed voxel [1].

Geometry and attributes are two subcategories of point clouds. Points at specific locations make up the geometry, and attributes provide useful data for each of those points. These two elements can be encoded separately. Signal distribution

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

within the geometry is one of the major challenges. A regular grid of voxels can be thought of as a binary signal of geometry. The relation of number of points to number of voxels is a way to define sparsity. Generating coarse LoD from a point cloud usually requires reducing this precision through quantization [16].

A. Point Cloud Capture and Acquisition

An array of sensors in a studio set up, provides a representation of any object in 3D space to obtain real-time point clouds of objects in motion, such as humans, at the very high resolution required for scenarios such as augmented/mixed reality. It's a way to surround and capture. Volume content and conference call video. Techniques such as photogrammetry, infrared depth cameras, scene lighting and stereoscopic parallax, structured light, and lasers can be used to record the position of objects in 3D space. Collected objects are reflected in a sparse cloud of voxelized points generated by both real-time and computational post-processing of the data [17].

A laser scanning method was developed to obtain the shape of the article and a very high-resolution 3D point cloud. This artefact measurement method is likely to be used in a wide range of applications. The actual field of view of a terrestrial laser scanner (TLS) and its placement are limited. This allows portable digital cameras to capture objects from different angles and positions with higher resolution. However, using multiple shots may not guarantee full exposure of the scene. Also, TLS data cannot be used to distinguish fine details with sub-pixel accuracy as in camera images [18].

Recently, point cloud mapping and acquisition applications are being developed for collecting the feature points by using depth cameras built in smart mobile phones. A tool kit mapping application is developed using Unity platform and is used in this paper.

B. Point Cloud Compression

The Moving Pictures Expert Group / Video-Point Cloud Compression (MPEG V-PCC) standard uses video coding methods to compress point cloud sequences. The point cloud is separated into patches, which are then crammed into a 2D grid and individually video codec-compressed for geometry and attributes. To enhance the quality of the reconstruction, preprocessing and post-processing methods including smoothing, padding, and interpolation are used [19]. Two-dimensional surfaces that have been fitted to point clouds can be used to depict them. Using fitted surfaces or predetermined surfaces, projection onto 2D planes may also be made to those points (like planes aligned on axis). The point cloud transforms into a point sequence when the dimensionality is reduced to one dimension [2].

Octree modeling involves repeatedly dividing the cubic voxel space into eight smaller cubes of equal size, as shown in Fig.1. At least one occupied voxel is required to split a sub cube. A 2D surface is projected with a point cloud. 2D projections are encoded using conventional methods [5].

A 3D binary array R(x,y,z) is used to represent the entire voxelized region. R(x,y,z) is set to 1 for occupied voxels and 0 for unoccupied voxels. Elements of G at (X_a Y_b Z_c) are equal to 0 if there are no occupied voxels at those coordinates.

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

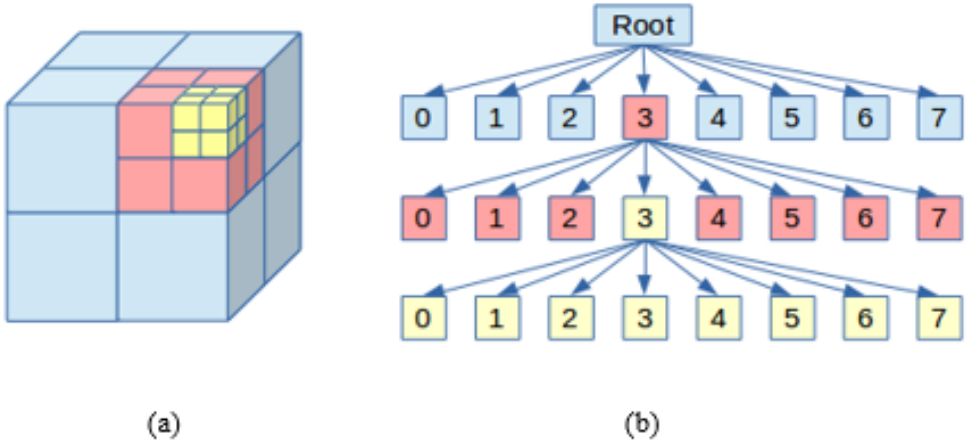

Figure 1: (a) Octree division. (b) A voxel subdivided as octree [20].

Projection helps reduce the dimensionality of the point cloud. In this work, a 3D voxel point cloud frame is transformed into his 2D density mesh using the proposed geometry projection method, and the resulting binary mesh is compressed. We use projection for the benefit of being based on a purely geometric, octree-based putrefaction methodology and is valuable for vast ranges of use cases that are meant to address a variety of point clouds going from human replicas to wide-scale maps of cities [21]. A block diagram of the proposed method is clarified in Fig.2.

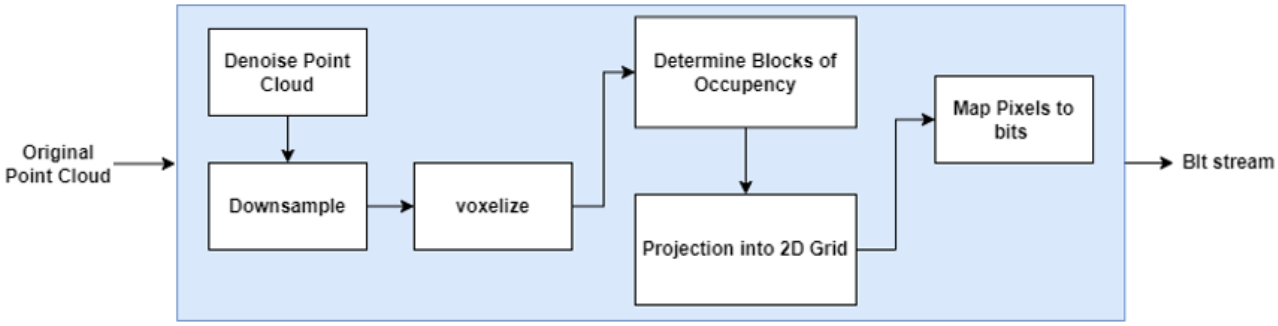

Figure 2: Block Diagram of Proposed Geometry Projection Compression.

IV. HTTP/2

HTTP/1.1 has some performance limitations [22]. The two main strategies for overcoming the mentioned limitations: are using multiple HTTP connections and making fewer but larger HTTP requests. Using multiple connections allows a client to send multiple requests simultaneously, resulting in faster response times. Reducing the number of requests and

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

increasing the size reduces the number of transfers between the client and server, reducing latency. Additionally, the server can handle larger requests more efficiently, resulting in better performance [23].

HTTP/2 was designed to maximize efficiency of network resources, send and receive data, and receive data with minimal latency [24]. Of particular interest to multimedia science is the push feature, which allows content to be delivered before the client requests it. Additionally, HTTP/2 includes a new mechanism for multiplexing the delivery of structured data called HTTP/2 streams [14]. HTTP/2 is a binary protocol that simplifies the framing process. In HTTP 1.1 it can be difficult to determine when a frame begins and ends. In HTTP/2, a connection carries many streams at the same time and both sides send frames from different streams at the same time [25].

A. Header Compression

One of the drawbacks of HTTP/1.1 is that redundant headers are sent for each request and response between the streaming server and the client-side browser. Since HTTP/1.1 is a stateless protocol [8], the server does not need to store data from previous requests [25] and each request contains all the page information for the server to satisfy it. must be Cookies are commonly used in request headers to maintain state. This can lead to large header sizes averaging 800 bytes and wasted bandwidth. This is one of the most expensive resources in communication between browsers and servers [8].

B. Server Push

By enabling the push feature, the client receives the pushed content before it is requested by the server [14]. Without server push, the browser would have to fetch and parse the initial HTML page before requesting any additional objects that the page might need. This means that it takes him two trips to load all the content of a simple web page, and more trips for larger pages. The results of [26] suggest that servers can transfer more than a few small files, which positively impacts network performance.

V. EXPERIMENTAL RESULTS

The captured point clouds are formatted by (.ply), which are plain text files. Examples of these are dense point cloud datasets are presented in Fig.3. Each of the samples consists of a cubic spatial resolution of $1024 \times 1024 \times 1024$ voxels known as depth 10. Samples (a-d) are acquired by [27] which is a project authorized by jpeg and known as 8i Voxelized Full Bodies (8iVFB). Each sample presents the whole model of a human and is captured by 42 RGB cameras categorized in 14 clusters at 30 fps. Table I shows details of each frame.

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

Figure 3: Sample frames:[(a) Long dress (b) Red and Black (c) Loot (d) Soldier][25] (e) Lion[13].

TABLE I
SAMPLE POINT CLOUD DATASETS

Sample Frame	No. of points	No. of Frames
Long dress	765,821	300
Red and Black	729,133	300
Loot	784,142	300
Soldier	1,059,810	300
Lion	4,000,000	300

A. Implementation of Proposed Geometry Projection Compression

We use MATLAB for the compression process. The steps that we apply to the sample data set of (soldier) is depicted in Fig.4.

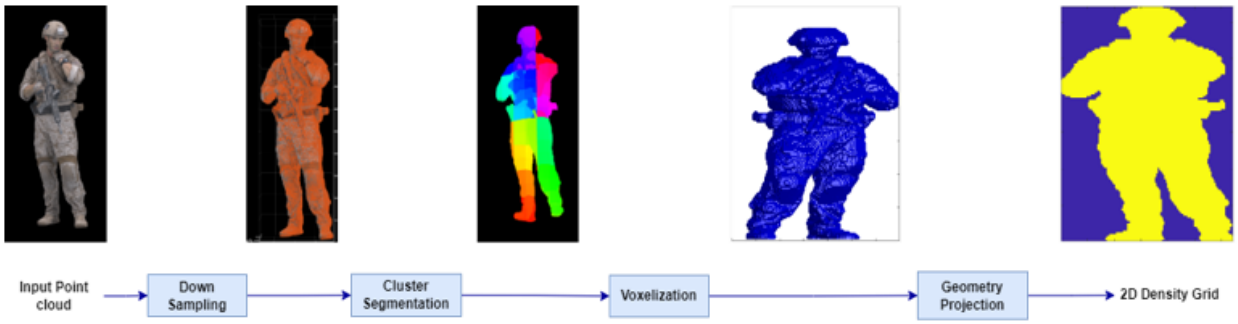

Figure 4: Compression process steps for a sample point cloud frame.

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

The compression of the point cloud was conducted in the following steps, firstly, we find occupancy 3D Grid, where voxels with empty data are neglected. Secondly, the 3D surface representation of points with equal values are presented using (iso_surface) function. . Next step is to transform 3D Occupancy Grid to 2D Density Grid using the proposed geometry projection method. Finally, we map each pixel to a single bit instead of bytes in the packed binary output image using the (bwpack) function. Table II shows the results of compressing samples of data sets using our proposed compression methodology of geometry projection.

TABLE II
SAMPLE POINT CLOUD DATASETS

Sample Frame	**No. of points**	**Original size of 3D** Occupancy Grid (in Bytes)	**Compressed Size for 2D** Density Grid (in Bytes)
Long dress	765,821	3,618,392	117,016
Loot	784,142	3,630,384	129,008
Red and Black	729,133	3,437,744	106,368
Soldier	1,059,810	3,944,752	143,376
Lion	4,000,000	5,883,201	210,114

In order to find the compression ratio when mapping 3D occupancy grid to 2D Density grid as:

$$\text{compression ratio} = \frac{\text{original size}}{\text{compressed size}} \tag{1}$$

This results in an average ratio ranging from 28:1 to 30:1 depending on the number of points and size of data.

B. Implementation of Proposed Point Cloud Streaming Methodology

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.5.

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

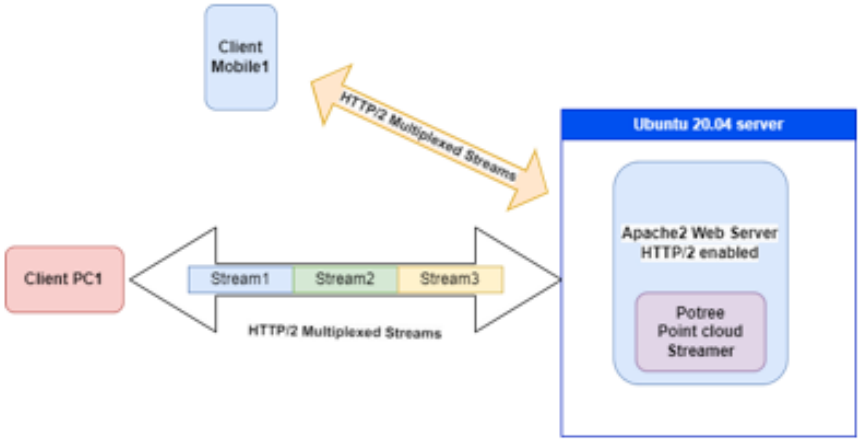

Figure 5: Proposed methodology explaining HTTP/2 stream multiplexing.

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.6.

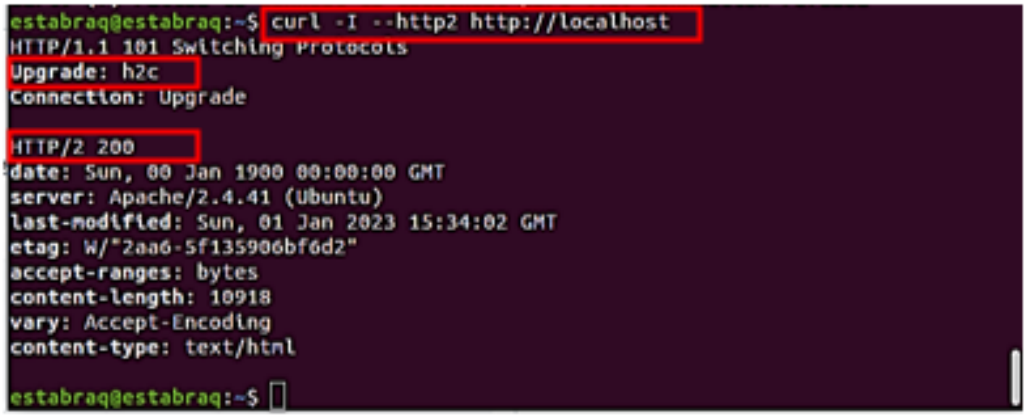

Figure 6: HTTP/2 enabled for apache2 server.

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.7.

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

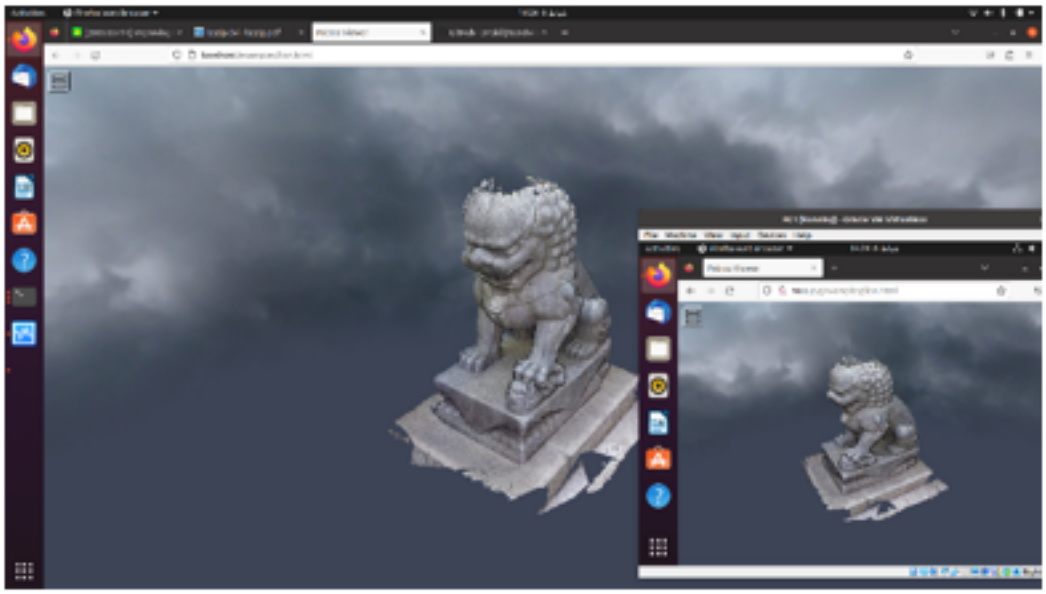

Figure 7: Accessing stream of sample frame of point cloud (lion) from a virtual box.

Packets carrying HTTP/2 protocols are captured using Wireshark and shown in Fig.8, where the upgrade to h2 is clarified.

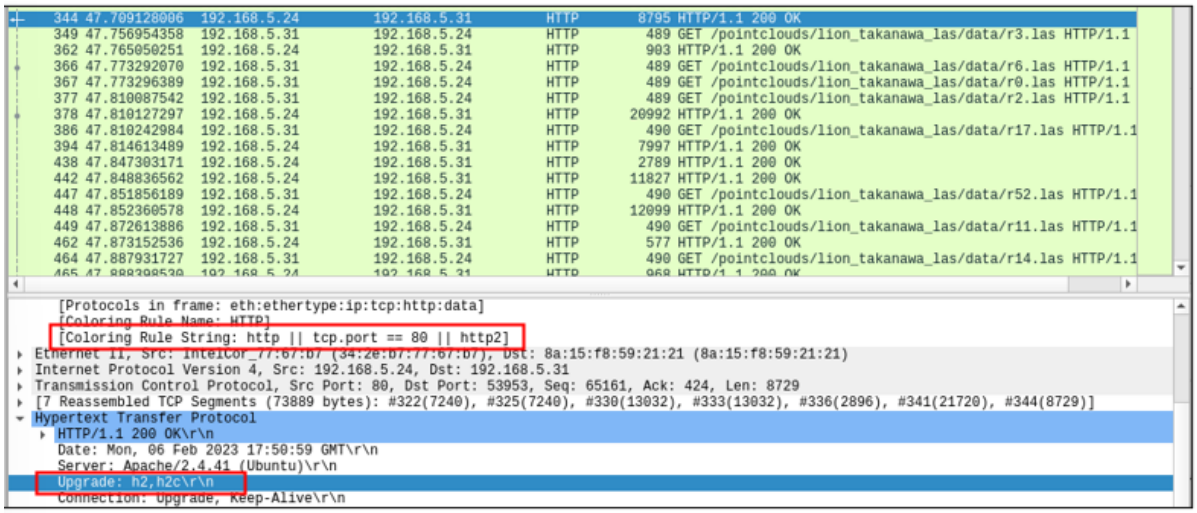

Figure 8: HTTP/2 packets captured by Wireshark.

Fig.9 shows an example frame captured using Wireshark and clarifies the measurement of inter-frame time delta, and table III shows statistics of 20 frames as an example of the overall data collected using Wireshark while streaming three of the sample point cloud data sets (lion) and (red and black). Columns show the samples of time delta from previous

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

displayed frame measured in milliseconds, along with data carried by that frame and measured in Bytes.

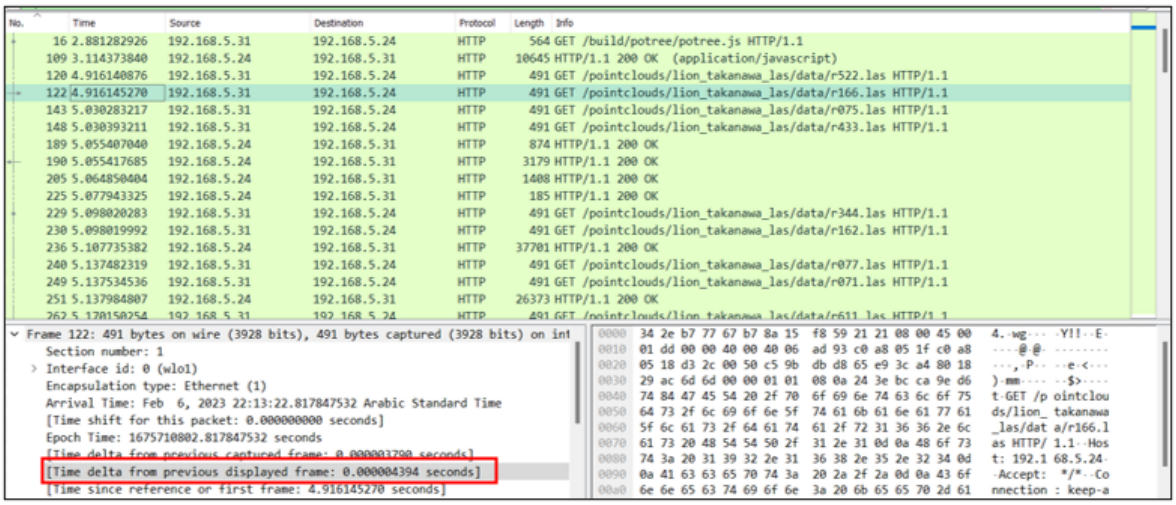

Figure 9: Delta time captured by Wireshark.

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

Lion		Red and Black	
Time Delta (ms)	**Size (Bytes)**	**Time Delta (ms)**	**Size (Bytes)**
0.678	874	1.74	827
0.00439	185	0.984	1396
114.1379	3179	135.98	485
0.10999	491	1.979	495
2.00769	1408	14.58	2182
0.0002	491	0.043	963
25.013	26373	12.83	491
0.0106	9082	5.462	473
9.432	37701	0.037	501
13.092	5737	0.0004	498
9.715	491	0.86	1573
29.746	43097	0.259	4744
0.052	2654	0.116	4694
0.4502	7132	19.8	484
7.724	10823	0.122	473
18.96	491	0.002	483
0.0004	491	0.195	1814
11.743	2937	0.076	2417
6.138	18246	3.25	1147
9.225	491	44.05	118

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

In table IV, we calculate the average Time delta between successive frames and the average bitrate of point cloud data transmission.

TABLE IV
AVERAGE TIME DELTA AND AVERAGE BITRATE

Lion		Red and Black	
Avg. Time Delta (ms)	**Avg. Bitrate (KB_{ps})**	**Avg. Time Delta (ms)**	**Avg. Bitrate (KB_{ps})**
13.8154	625.027	12.1183	105.032

In order to calculate frames rate per second, we use equation (2) that divides one second over the time that within which each frame is displayed.

$$\text{Frame Rate} = \frac{1}{\text{Avg Time Delta}}\ (fps) \tag{2}$$

Table V 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 [13] and for (Red and Black) presented by [27].

TABLE V
COMPARISON OF RESULTS SHOWING THAT OUR SYSTEM OUTPERFORMS PREVIOUS WORK

System	**Lion**	**Red and Black**
[13]	Frame Latency 16.6 ms	
	Frame Rate 60 fps	
[27]		Frame Latency 33.3 ms
		Frame Rate 30 fps
Our Proposed System	Frame Latency 13.815 ms	Frame Latency 12.118 ms
	Frame Rate 72.46 fps	Frame Rate 82.51 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 [13] and [27] in a very similar environment stated by [13] using a Notebook with Core i7 CPU 2.30 GHz, 16 GB RAM, GTX 860 M GPU and 1TB disk. Our results show achieving 72.46 fps, while authors in [13] have achieved 60 fps streaming (lion), and we have achieved a frame rate of 82.51 fps for (Red and Black) compared to the 30 fps, stated by authors in [27].

Figs. 10 and 11 show our proposed system's result compared to previous work's.

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

ISSN:2222-758X
e-ISSN: 2789-7362

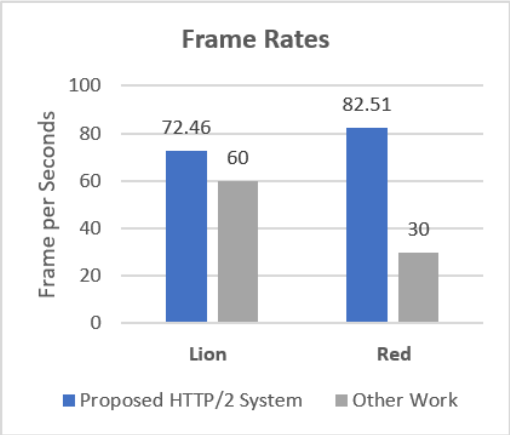

Figure 10: Frame Rates of proposed system compared to other work.

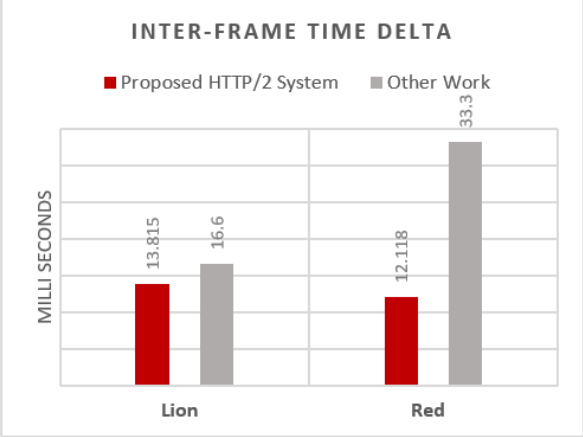

Figure 11: Average Time delta between successive frames.

VI. CONCLUSIONS

In this work, we capture a large number of point cloud data stored in a web server. Applying our proposed geometrical projection compression that results in preparing the volumetric video for streaming over HTTP/2 internet protocol.

The proposed compression methodology has shown very good results in compressing 3D point cloud video frames by about 30:1 compression ratio. By comparing our proposed system's result to previous conventional works, 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 and 63.6% for the case of (Red and Black) with 729K points. Frame rates also increased when receiving the stream by 20.7%, 175% for the cases of (Lion) and (Red), respectively.

Funding

None

ACKNOWLEDGEMENT

The author would like to thank the reviewers for their valuable contribution in the publication of this paper.

CONFLICTS OF INTEREST

The author declares no conflict of interest

REFERENCES

[1] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P.A. Chou, et al. "Emerging MPEG Standards for Point Cloud Compression". IEEE J. Emerg. Sel. Top. Circuits Syst. 9, 133-148, 2019, doi:10.1109/JETCAS.2018.2885981.
[2] M. Quach, J. Pang, D. Tian, G. Valenzise, F. Dufaux, "Survey on Deep Learning-Based Point Cloud Compression", 2022, Front. Sig. Proc. 2:846972. doi: 10.3389/frsip.2022.846972.
[3] 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.
[4] T.K. Hung, "On Error Concealment of Dynamic 3D Point Cloud Streaming", National Tsing Hua University, Master Thesis, MM '22: Proceedings of the 30th ACM International Conference on Multimedia, October 2022, Pages 3134-3142 https://doi.org/10.1145/3503161.3548384.
[5] D. Tzamarias, K. Chow, I. Blanes, J. S. SagristÃ , "Compression of point cloud geometry through a single projection", Universitat Autonoma de Barcelona - Data Compression Conference (DCC) 2021, USA, March 24 to 25, 2021.

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

[6] C. Moreno, M. Li, "Frame Filtering and Skipping for Point Cloud Data Video Transmission", 28 January, 2017, https://dx.doi.org/10.25046/aj020109.
[7] X. Wen, X. Wang, J. Hou, L. Ma, Y. Zhou and J. Jiang, "Lossy Geometry Compression Of 3d Point Cloud Data Via An Adaptive Octree-Guided Network," 2020 IEEE International Conference on Multimedia and Expo (ICME), London, UK, 2020, pp. 1-6, doi: 10.1109/ICME46284.2020.9102866.
[8] Q. Yin, Q. Ren, L. Zhao, W. Wang, J. Chen, "Lossless Point Cloud Attribute Compression with Normal-based Intra Prediction", Accepted by the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting 2021, arXiv:2106.12236v1 [eess.IV] 23 Jun 2021.
[9] A. Mustafa, M. I. Aal-Nouman, and O. A. Awad, "Cloud-Based Vehicle Tracking System", Iraqi Journal of ICT, vol. 2, no. 4, pp. 21-30, Feb. 2020. DOI: https://doi.org/10.31987/ijict.2.4.81
[10] M. Hosseini, Ch. Timmerer, "Dynamic Adaptive Point Cloud Streaming",6 pages, 23rd ACM Packet Video (PV'18) Workshop, June 12–15, 2018, Amsterdam, Netherlands arXiv:1804.10878v2.
[11] E. H. Al-Hemiary and S. H. Majid, "Performance Evaluation of Multimedia Content over MPLS e-Learning Network", Iraqi Journal of ICT, vol. 2, no. 1, pp. 1-9, Aug. 2019. DOI: https://doi.org/10.31987/ijict.2.1.42
[12] 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
[13] M. Schuetz, "Potree: Rendering Large Point Clouds in Web Browsers", Faculty of Informatics at the Vienna University of Technology, submitted in partial fulfillment of the requirements for the degree of Diplom-Ingenieur in Visual Computing Vienna, 2016
[14] 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
[15] 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
[16] W. Jiang, J. Tian, K. Cai, F. Zhang, T. Luo. (2012). "Tangent-plane continuity Maximization Based 3D point Compression," in 2012 19th IEEE International Conference on Image Processing (IEEE), 1277-1280. doi:10.1109/ ICIP.2012.6467100
[17] C. Loop, C. Zhang, and Z. Zhang, âReal-time high-resolution sparse voxelization with application to image-based modeling,â in Proc. 5th High-Perform. Graph. Conf., 2013, pp. 73â79.
[18] M. Dahne, Partovi, M. Maboudi, D. Krueger, M. Gerke, "Automatic Integration of Laser Scanning and Photogrammetric Point Clouds: From Acquisition to Co-registration", https://doi.org/10.5194/isprs-archives-XLIII-B1-2021-85-2021
[19] ITU-T (2019). ITU-T Recommendation "H.265. High efficiency video coding (HEVC)".
[20] A. David, "Advanced Octrees 1: preliminaries, insertion strategies and maximum tree depth", 2014, < https://geidav.wordpress.com/2014/07/18/advanced-octrees-1-preliminaries-insertion-strategies-and-max-tree-depth/> (last visited on 20/5/2023).
[21] Ch. Cao. "3D point cloud compression". Multimedia [cs.MM]. Institut Polytechnique de Paris, 2021. English. NNT : 2021IPPAS015. tel-03524521
[22] J. L. Wagner, E. Marcotte, "Web Performance in Action Building Fast Web Pages", Manning Publication Co., First Edition, Jan 2017, ISBN 9781617293771
[23] B. Pollard, "HTTP/2 in Action", March 2019 by Manning Publications Co., First Edition, ISBN 978-1617295164
[24] C. Muller, 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.
[25] D. Stenberg, "HTTP2 Explained", ACM SIGCOMM Computer Communication Review 120 Volume 44, Number 3, July 2014
[26] 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
[27] 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

This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0