On the Performance of Distributed Rendering System for 3DWebGIS Application on Ultra-High-Resolution Display

Kawanabe, T.,[1*] Tsuruta, S.,[2] Ageno, M.,[2] Kagatani, Y.[2] and Ono. K.[3]
[1]Center for Computational Science, RIKEN, Japan, E-mail: tkawanabe@riken.jp*
[2]GEO Solutions Inc., Japan, E-mail: tsuruta@geo-sol.co.jp, ageno@geo-sol.co.jp, kagatani@geo-sol.co.jp
[3]Research Institute for Information Technology, Kyushu University, Japan, E-mail: keno@cc.kyushu-u.ac.jp
**Corresponding Author*

DOI: https://doi.org/10.52939/ijg.v21i1.3785

Abstract
While the amount of data to be displayed on GIS continues to glow, WebGIS faces challenges in handling large datasets due to the heap memory limitations of web browsers. Meanwhile, the resolution of desktop displays is reaching its limit, making it challenging to display ultra-high-resolution content on the desktop, so using a Tiled Display Wall (TDW) offers a more feasible solution. In this context, we introduce ChOWDER, an open-source software that enables the distributed rendering of various contents, including 3DWebGIS on a TDW. It provides a detailed explanation of its distributed rendering functionality, utilizing iTowns, an open-source GIS platform, and Three.js, a JavaScript library for 3D graphics in a web browser using WebGL. As a practical example, using an ultra-high-resolution TDW system running ChOWDER, we will show how to display 3D WebGIS distributed rendering on a TDW with a horizontal resolution of 20K, composed of fifteen 4K displays and 15 PCs. Furthermore, we conducted experiments of distributed rendering of large-scale geographic information, including 3DTiles city building data on multiple displays, and confirmed that memory consumption was effectively distributed. ChOWDER is a middleware that can build ultra-high-resolution displays without special hardware. It can display not only 3DWebGIS distributed rendering but also various content simultaneously and in a bird's-eye view on ultra-high-resolution displays, making it applicable to applications such as wide-area GIS display and simultaneous display of a wide variety of data that cannot be overlaid on GIS.

Keywords: Distributed Rendering, FOSS4G, Geospatial, OSS, WebGL

1. Introduction

The total amount of data that geographic information systems must display is accelerating due to the spread of sensing devices with location information used daily, such as the IoT (Internet of Things), and the increase in spatial and temporal resolution of observation equipment. On the other hand, the increase in resolution of display devices used to analyze and visualize such data is reaching its limit due to several physical constraints described later. The maximum resolution of commercially available display devices is 8K; 4K or 5K is considered the upper limit for desktop use. Using the tiled display driver provided by the GPU (Graphic Processing Unit) manufacturer, it is possible to create a display environment with an even larger area and higher resolution. However, the middleware provided by the GPU manufacturer currently has a maximum resolution limitation of 16K [1], which is the maximum resolution that can be achieved on a single PC. However, even if these mechanisms are used to create an ultra-high-resolution display environment, it is only possible to render data within the limitation of heap memory on web browsers in the case of WebGIS (Web-based Geographic Information System) applications.

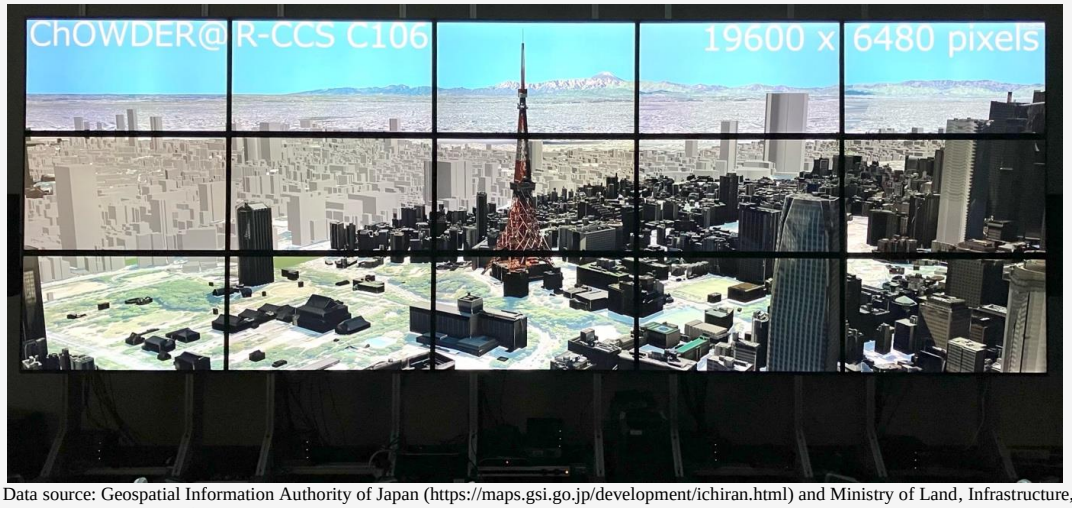
In this paper, we propose a method for distributing and rendering 3DWebGIS (3-Dimensional WebGIS) content across multiple PCs (web browsers) and its implementation as a solution to the above issues, which are memory limitation of a web application and resolution limitation of a display device. The structure of the rest of this paper is designed to provide a clear and comprehensive understanding of our proposal. First, in Section 2, we clarify the target issues.

Then, Section 3 introduces an overview of ChOWDER, a web-based tiled display driver that enables the distributed rendering of 3DWebGIS, as shown in Figure 1. Section 4 explains ChOWDER's 3DWebGIS distributed rendering method. Section 5 introduces other related research that realizes distributed GIS rendering and explains the differences with ChOWDER's functions. Section 6 explains the experiment for the memory load distribution when ChOWDER is used for distributed 3DWebGIS rendering and reports the results. Section 6 describes the experimental process and the improvements to ChOWDER that are considered based on the results. Finally, Section 7 summarizes this paper.

2. Limitations of the WebGIS Applications

The tiled web map adopted by Google Maps in 2005 was simple and revolutionary, and most GIS that appeared afterward adopted the mechanism. The tiled web map is the process of distributing map image files, which have been divided into square tiles in advance from a server to a client application, and the map is displayed by laying out the map image tiles on the screen of the client application. Map tiles have zoom levels according to their magnification, and in the case of Google Maps, there are 23 zoom levels, from zoom level 0, which shows the entire earth in a single tile, to zoom level 22, which is the most detailed. When a user zooms in or out on a map in a GIS application, the GIS application retrieves map tiles from the server for the zoom level corresponding to the screen resolution and displays them. This system is helpful in most cases, but the map provider predetermines the level of information reduction at each zoom level, and the tile images are created with that in mind, so it may not match the level of information reduction desired by the user (For example, if the user wants to see an overview of an entire city, not all roads of interest for the user may be represented at that zoom level.).

A possible solution to these issues is to increase the screen resolution of the PC that displays the GIS application. Currently, the maximum resolution of PC displays on the market is 8K (7680 x 4320 pixels), but the most commonly available resolutions are around 4K (3840 x 2160 pixels) or 5K (5120 x 2880 pixels). Such high-resolution displays allow map tiles to be displayed at more detailed zoom levels. Furthermore, by using a multi-display driver provided by the GPU manufacturer, it is possible to increase the resolution of the display area further. However, even when using the multi-display driver provided by the GPU manufacturer, the maximum resolution is 16K (15360 x 15360 pixels) [1]. There is a limit to how high resolution can be achieved in a desktop environment. The maximum effective resolution of a desktop PC display is reported to be 186 dpi (dots per inch) [2]. This corresponds to a 24-inch 4K resolution display. If a 16K x 16K environment is created using sixteen 24-inch 4K displays using the multi-display function of a GPU, the diagonal size will be 96 inches, which is an unrealistic size for a desktop display. For this reason, such ultra-high-resolution displays are generally used not as desktops but as so-called tiled display walls (TDW), in which flat display devices are arranged in tiles on a wall.



Data source: Geospatial Information Authority of Japan (https://maps.gsi.go.jp/development/ichiran.html) and Ministry of Land, Infrastructure, Transport and Tourism of Japan (https://www.mlit.go.jp/plateau/open-data/)

Figure 1: An example of viewing 3D geographic information by ChOWDER's 3DWebGIS distributed rendering method on a 20K horizontal resolution tiled display wall consisting of fifteen 43-inch 4K liquid crystal displays and fifteen PCs

Displaying in high resolution also increases the memory required by GIS applications. In particular, for WebGIS applications, maximum heap memory size is limited for Web browsers. In the case of Google Chrome Version 128.0.6613.86, the maximum heap memory that an application can use is about 4GB. This can be confirmed by executing 'performance.memory.jsHeapSizeLimit' in the Chrome developer tool's console. This memory limit is often not an issue if the only purpose is simply to display map image tiles at high resolution, but large amounts of memory are required to display various data overlaid on the map, particularly 3D polygon data of features such as 3DTiles [3] and textures. For example, the 3DTiles building data for all 23 wards of Tokyo provided by the Tokyo Digital Twin Project [4] cannot all be rendered at once in their 3DWebGIS viewer (Wards where textured data is provided, the textured data is used.) The demand for overlaying a wide variety of user data on GIS is expected to continue increasing. Still, the amount of data that can be displayed on a limited screen resolution and memory space is limited.

To solve the abovementioned issues of limited screen resolution and limited memory space, related to increasing the resolution of 3DWebGIS applications, distributing the display of WebGIS content across multiple PCs (multiple web browsers) can be applied. In fact, it makes possible to display WebGIS content at a resolution that exceeds the upper limit of a single PC and distributes the memory load required for the display to each PC (web browser). This paper proposes a method for distributed rendering of 3DWebGIS applications across multiple web browsers using the scalable display system ChOWDER.

3. ChOWDER: A Web-based Scalable Display System

The scalable display system ChOWDER (COoperative Workspace DrivER) [5] and [6], jointly developed by RIKEN Center for Computational Science and Kyushu University, is a web-based, open-source tiled display driver that can create an ultra-high-resolution pixel space by arranging multiple displays that display a web browser in full-screen mode in tiles. ChOWDER does not require any specialized hardware or software. The central characteristic concept is the virtual two-dimensional display space, called VDA (Virtual Display Area), which enables dynamic and flexible display system configuration. As shown in Figure 2, the VDA is a simple two-dimensional (x,y) coordinate space, and is part of a metadata collection managed by the ChOWDER server.

For content shown on the display, Metadata such as the position on the VDA, magnification ratio, content type, and pointers to the content itself (such as URIs) are managed in a Key-Value Store (KVS) on the ChOWDER server. The content data itself is also managed in the KVS in association with the metadata. The server also manages the position and magnification ratio of displays (web browser windows) placed on the VDA. ChOWDER is a server-client type Web application, and clients are divided into "display clients" and "controller clients." The display client is a Web browser in full-screen display mode. Multiple display clients can be arranged in tiles on the VDA to achieve a tiled display.

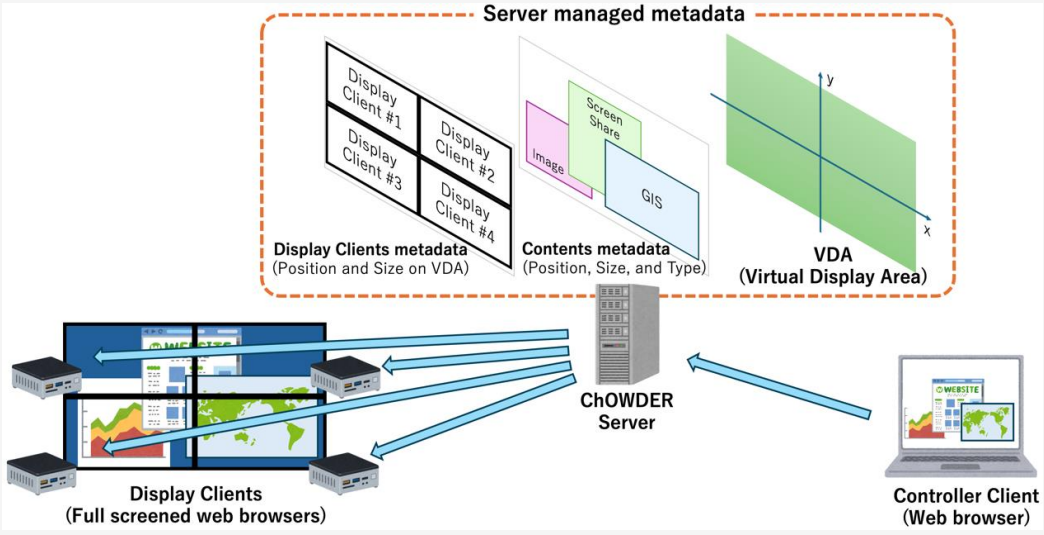


Figure 2: Overview of ChOWDER's VDA Concept

By overlapping multiple display clients on the VDA, a mirror display can be achieved between the display clients. If each display client is in a remote location, display information can be shared between the remote locations. The controller client is an application that runs on a web browser and can register the content to be displayed and adjust the position and magnification ratio of the content on the VDA. It runs on the PC of the user who administer/control ChOWDER.

Displayable static content are text, images, and PDFs. Users can upload the content from the controller client to the server, and the position and magnification ratio on the VDA can be specified. The server determines which display client should display the content by its position on the VDA and sends the content data and VDA metadata (display position, magnification ratio) to that display client. The receiving display client calculates the content's relative position and magnification ratio from its own position and magnification ratio on the VDA and draws the content in its browser window. The magnification ratio described here is the ratio between the original size of the content and the size on the VDA. The aspect ratio is maintained even when the content is magnified on the VDA. The content is rendered by the display client's web browser. The rendering process for each content type depends on the web browser's processing. For example, with raster content such as images, pixel roughness becomes noticeable when magnified. On the other hand, vector content such as PDF maintains its smoothness even when magnified. In addition, video, webcam footage, and screen sharing can be displayed as dynamic content.

When the user specifies the content to be displayed and the position and magnification ratio on the controller client, streaming communication using the WebRTC [7] protocol is started. The server determines the display client that should display the content. Then, the dynamic content is displayed through peer-to-peer streaming communication between the controller client and the display client. The display function for 3DWebGIS was added in 2020 [8]. The following section will explain this in more detail. Figure 3 shows a typical use case of ChOWDER. Two users are using ChOWDER's controller client on each PC to show GIS content based on the OpenStreetMap (https://openstreetmap.org/copyright) to each other for discussion. Text, PDF, and JPEG images are also displayed on the TDW. Each user can freely move, resize, and specify the stacking order of contents.

4. Distributed Rendering Method for 3DWebGIS by using ChOWDER

This section explains ChOWDER's 3DWebGIS distributed rendering function. To use this function, the user must launch a 3DWebGIS client and the controller client on their PC. The 3DWebGIS client is a regular 3DWebGIS application with added functionality for communicating with the ChOWDER server. It uses the open-source iTowns [9] as the 3DWebGIS application. The reason for using iTowns will be explained later in this section. Figure 4 shows the primary sequence of this function. In this diagram, the ChOWDER system has already been started, and two display clients and one controller client have already been connected to the ChOWDER server.

Figure 3: A typical use case of ChOWDER. It is designed to share content on an ultra-high-resolution display and facilitate discussions among multiple people, offering experiences that are not achievable on a conventional PC display

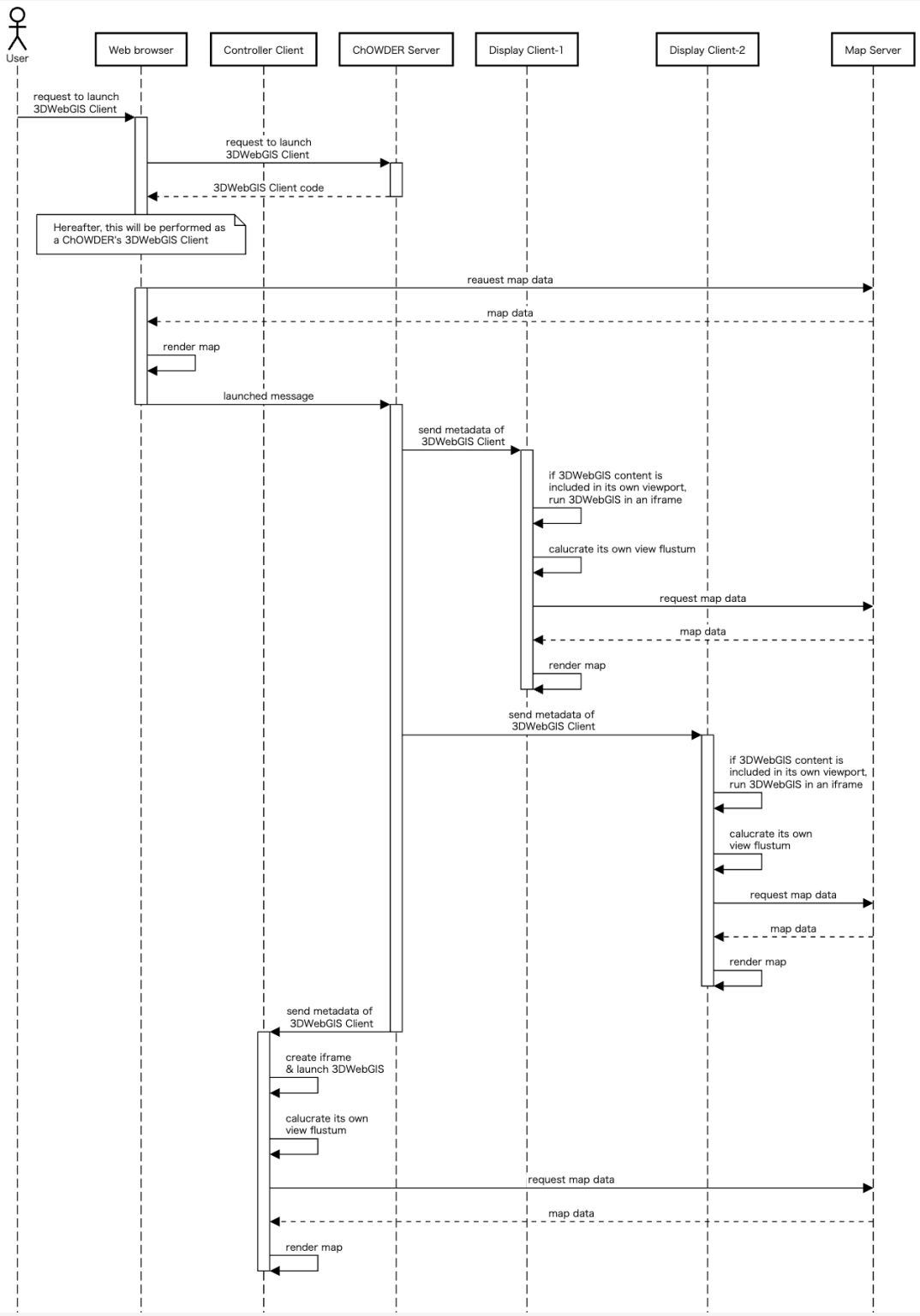


Figure 4: Startup sequence diagram of 3DWebGIS content distributed rendering function

First, the user requests the ChOWDER server from the web browser on their PC to start the 3DWebGIS client. The ChOWDER server then sends the iTowns-based application code to the browser, and the browser then performs as a 3DWebGIS client. The launch of the 3DWebGIS client and its metadata (initial position on the VDA, magnification ratio, initial camera position for the 3D rendering of GIS

content) is notified to the display client and controller client. The display client evaluates the metadata of the launched 3DWebGIS content, and if it is included in its own display area, it creates an iframe (inline frame: an HTML tag for embedded content) according to the content display size and initially draws the 3DWebGIS content in it. The iframe and 3DWebGIS content are also drawn on the controller client. The user can move, enlarge, or reduce the 3DWebGIS content displayed on the controller, and the 3DWebGIS content is also moved, enlarged, or reduced on the display client in response to the operation. As shown in Figure 5, when the viewpoint is moved (for example, zoomed in) on the 3DWebGIS client on the user's PC, the map server receives a request for map tile data for the new zoom level, and the 3DWebGIS client screen is updated, just as with a regular 3DWebGIS application.

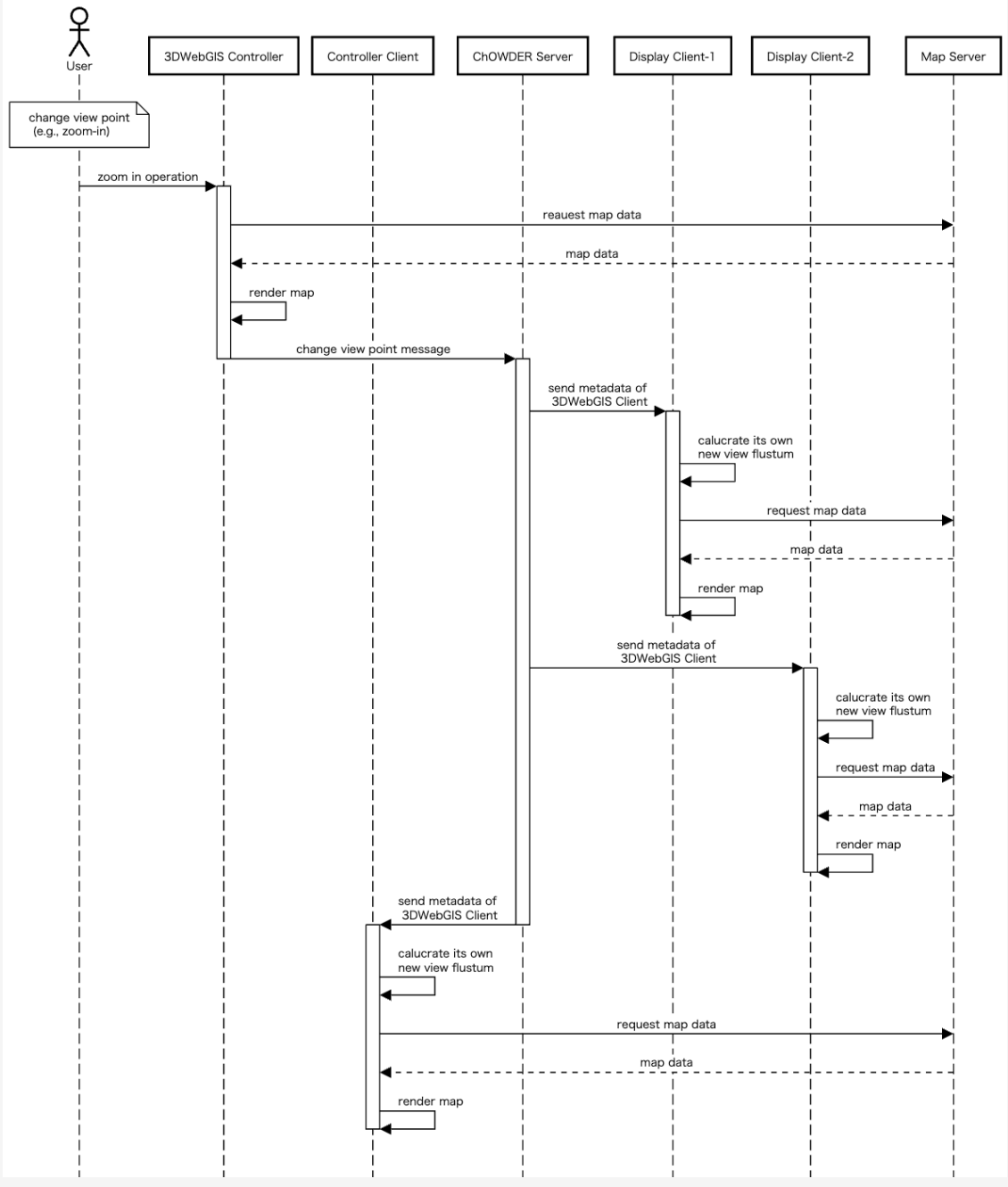


Figure 5: Viewpoint movement sequence diagram for 3DWebGIS content

In addition, metadata about the viewpoint change is notified to the display client via the ChOWDER server, and each display client requests map tiles within its own display range from the map server and displays them. This function allows 3DWebGIS applications running independently on each display client to work together with minimal metadata communication.

The view frustum must be appropriately divided to split and render 3DWebGIS content across multiple displays. As mentioned above, ChOWDER's 3DWebGIS display function uses iTowns. iTowns uses Three.js, a cross-browser JavaScript library and application programming interface (API) used to create and display 3D graphics in a web browser using WebGL (Web Graphics Library), and Three.js has an API that can offset the view frustum [10]. ChOWDER utilizes Three.js's view frustum offset API to divide a single iTowns content into multiple view frustums, allowing multiple web browsers to split and render 3DWebGIS content, as shown in Figure 6. However, in our first report on the 3DWebGIS distributed rendering feature in ChOWDER [8], when iTowns performed a 3DTiles load, it appeared to load all data without considering whether it was within or outside the view frustum, and we did not measure the memory utilization efficiency of distributed rendering. Since then, the 3DTiles load process has been improved in iTowns Release 2.42.0. In this paper, we measured the amount of heap memory consumed by each browser when distributing and rendering 3DWebGIS content using ChOWDER across multiple web browsers and confirmed the effective memory load distribution achieved by this method.

5. Related Works

SAGE2 [11] is the *de facto* standard for web-based tiled display drivers. It can perform distributed rendering of Google Maps' 2D maps but does not support 3D map display. SAGE3 [12], the successor to SAGE2, was released in 2023, and the map display function was replaced with a dedicated application based on Leaflet [13]. Leaflet is a JavaScript library for 2D map viewing, so it does not support 3D map display. Liquid Galaxy [14] is a panoramic viewer system that performs distributed rendering of Google Earth on multiple displays. The Liquid Galaxy application employs a master-slave architecture, and slave nodes render adjacent view frustums based on the orientation of the view frustum of the Google Earth content displayed by the master node. In other words, the view frustum of each node only changes its orientation while maintaining the exact same shape of a rectangular pyramid. Figure 7 shows the difference in the shapes of the view frustums handled by Liquid Galaxy and ChOWDER. Liquid Galaxy's distributed rendering method is simple and effective. Still, in principle, the distributed displays must be arranged in an arc shape centered on the viewer person, and applying it to a general TDW arranged on a flat surface will result in rendering with an unnatural perspective. To achieve distributed rendering of 3DWebGIS with a general planar TDW, a multi-display driver provided by a GPU manufacturer is used, making it possible to display ultra-high-resolution images in a single Web browser window.

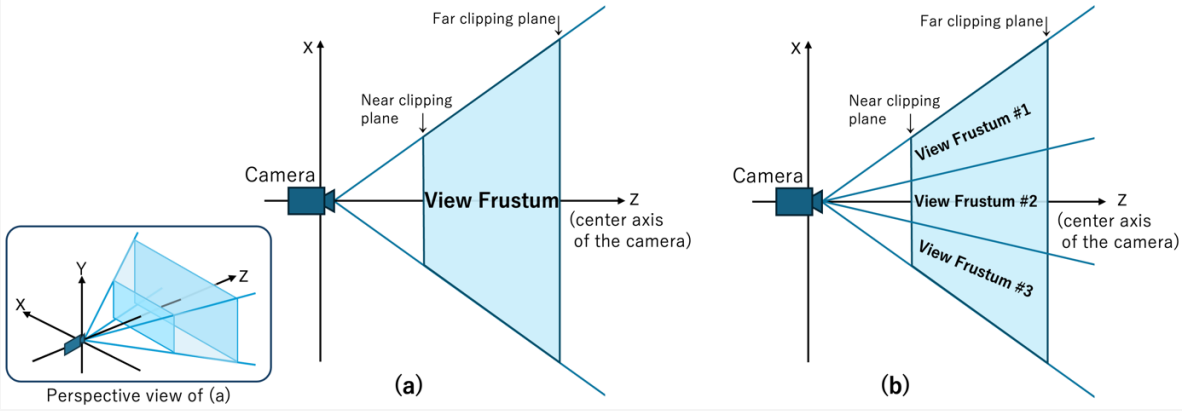


Figure 6: Diagram of the view frustum (a) standard view frustum (b) segmented view frustum

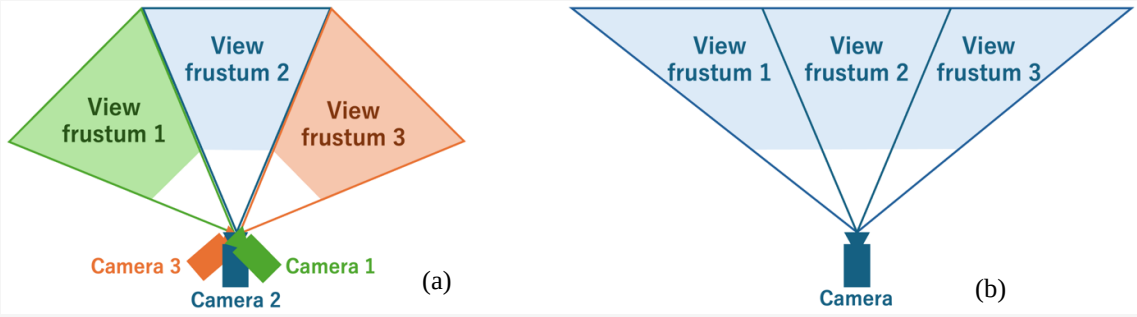


Figure 7: Differences in view frustum division methods (a) Liquid Galaxy (b) ChOWDER. Liquid Galaxy is a simultaneous rendering of multiple cameras and does not divide a single camera frustum

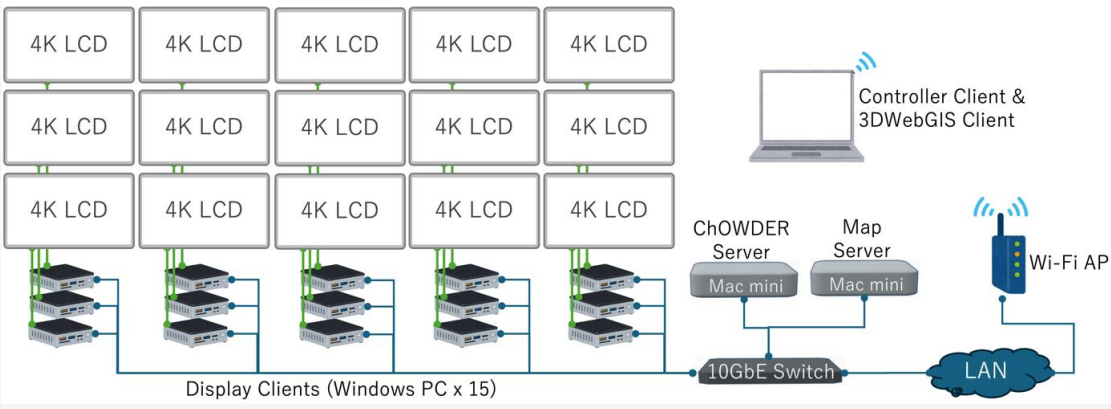


Figure 8: A configuration diagram of the experimental apparatus. In this experiment, all fifteen displays were set as ChOWDER distributed displays, of which nine displays on the left were used to display 3DWebGIS content

In this method, the view frustum is a single rectangular pyramid; no special distributed processing is required for 3D graphics. Each display shows a segmented 2D image from the result of a single 3D rendering. However, as mentioned in Section 2, due to the upper limit of the heap memory size of Web browsers, it is unsuitable for large-scale display of overlay data that consumes a lot of memory, such as 3D building data.

6. Experimental Procedures and Results

We experimented to confirm that the 3D WebGIS distributed rendering method proposed in Section 4 distributes the memory consumption of web browsers on a distributed rendering display. Figure 8 shows the experimental environment used. The TDW, owned by R-CCS, consists of fifteen 4K displays, with one Windows PC connected to each display. Each PC used a Lenovo ThinkCentre M79 Gen2 (CPU: AMD Ryzen 7 PRO 5750GE, Memory: 32GB). In addition, the ChOWDER server and the map server that distributes map tiles and 3DTiles data are connected to a GbE (Gigabit Ethernet) switch. Each display shows a full-screen web browser connected to the ChOWDER server and acts as a display client. The experimenter's laptop PC (MacBook Air 15 inch, 2023, CPU: M2, Memory: 24GB) runs the client controller and the 3DWebGIS controller in separate web browsers. The client controller adjusts the position and magnification of the 3DWebGIS content on the VDA. The laptop PC is connected to the ChOWDER server via the in-house Wi-Fi. All web browsers used were Google Chrome, and the display client's heap memory consumption recorded the values immediately after scene rendering was completed using Chrome's developer tools. The experimental data used was the textured building data for Chiyoda, Minato, and Chuo wards in Tokyo, from the 3DTiles data provided by the PLATEAU project [15] of the Ministry of Land, Infrastructure, Transport and Tourism of Japan. In addition, we have independently adjusted the SSE (Screen Space Error) threshold when drawing 3DTiles to ensure that all 3DTiles data is drawn at the most detailed level.

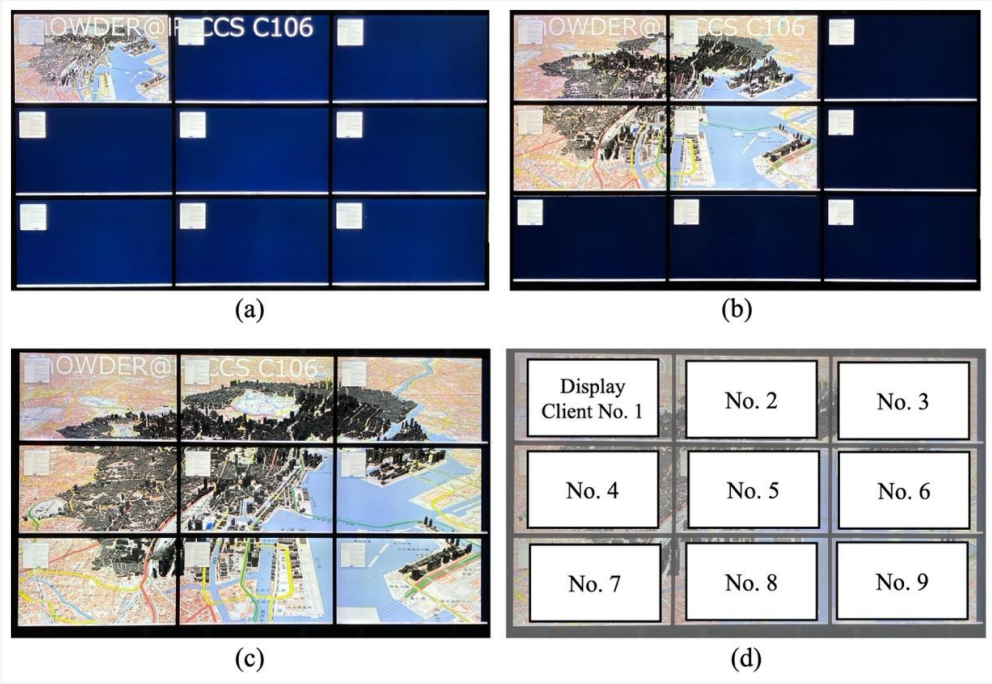


Figure 9: Distributed rendering images of the experimental results (a), (b), (c) and the display client numbers (d) To measure heap memory consumption, a Google Chrome developer tools window was displayed in the upper left corner of each display

Table 1: Heap memory consumption per display client in each experiment [MB]

		Display Client No.								
		1	2	3	4	5	6	7	8	9
Experiment No.	1	268.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	2	133.0	188.0	N/A	68.3	37.7	N/A	N/A	N/A	N/A
	3	66.8	122.0	140.0	84.3	87.2	56.9	41.2	38.4	33.6

In the first experiment (Experiment No. 1), the 3DWebGIS content displaying the 3DTiles building data for the above three wards was scaled to fit the entire screen of a single 4K resolution display (Figure 9(a)). At this time, the heap memory size of the web browser was 268.0 MB. In the next experiment (Experiment No. 2), the same content was scaled to fit on four displays, two vertically and two horizontally (Figure 9(b)). The heap memory sizes of each web browser were 133.0MB, 188.0MB, 68.3MB, and 37.7MB, respectively. In the final experiment (Experiment No. 3), the same content was scaled to fit across nine displays, three vertically and three horizontally (Figure 9(c)). The heap memory sizes of each web browser were 66.8MB, 122.0MB, 140.0MB, 84.3MB, 87.2MB, 56.9MB, 41.2MB, 38.4MB, and 33.6MB, respectively. In ChOWDER, 3DWebGIS content is scaled while maintaining the aspect ratio, so in this case, a 3x3 display usage is the maximum magnification in this experiment. Table 1 shows the heap memory consumption for each display client. The Display Client Numbers are indicated in Figure 9(d). This table shows that Display Client No.1's memory consumption decreases as the experiment progresses. This is because the number of buildings Display Client No.1 needs to display decreases. Also, at Display Client No.5, the memory consumption increases more than twice from Experiment No.2 to Experiment No.3. This is because most of the screen was sea in Experiment No.2, and land, including buildings, was drawn in Experiment No.3.

From these experimental results, distributed rendering of 3DWebGIS using ChOWDER achieves excellent memory load balancing. During distributed rendering, the heap memory size of each web browser is different because the amount of 3DTiles data contained in each responsible drawing area is different. Also, the total heap memory size of all browsers is larger than when rendering in a single browser because iTowns in the iframe (described in Section 4.) loads 3DTiles data wider than its view frustum, and data loading in overlapping areas occurs during distributed rendering.

As explained in Section 4, the 3DWebGIS client that runs on the user's PC is a regular iTowns application with a communication function with the ChOWDER server added. Loading a large amount of 3DTiles data, as in the experiment conducted in this study, may slow down the operation of this client application or cause it to crash. There were various error messages reported by Google Chrome when it crashes, but in many cases it was "Out of memory". To make ChOWDER's 3DWebGIS distributed rendering function practical, it is urgent to solve this problem. Specifically, when a 3DTiles load request occurs, it is necessary to reduce the amount of data loaded into memory. For 3DTiles, the LoD (Level of Detail) of the data can be defined, and a method of loading coarser LoD data can be considered. However, since the 3DTiles data provided often does not have a sufficiently coarser LoD defined, we are considering a method of displaying only the bounding box of the 3DTiles data without loading it. In addition, this experiment measured the heap memory consumption of the web browser on the display client, but rendering 3D data also consumes GPU memory.

We want to perform a more detailed analysis by adding the GPU memory consumption to the measurement. Furthermore, during this experiment, we found that the 3DTiles data that iTowns retrieves from the server is larger than the rendered view frustum. We verified this experimentally: loading 3DTiles while displaying a nearby area that does not contain 3DTiles increases memory usage. Although we have not confirmed the specific difference in size, making this range adjustable to a smaller size will allow for more memory-efficient distributed rendering, so we are considering modifying the iTowns itself to further investigate this hypothesis.

7. Conclusion

In this paper, we have shown the limitations of current 3DWebGIS when handling increasing amounts of data, proposed a distributed rendering method to address this issue, and introduced the view frustum offset API of Three.js, iTowns, a 3DWebGIS to which it can be applied, and ChOWDER. This web-based tiled display driver incorporates these technologies to implement the method. Furthermore, we presented an excellent memory load balance using distributed rendering for 3DWebGIS from display experiments. We demonstrated that the proposed method provides an effective solution for managing the increased data displayed in 3DWebGIS. ChOWDER does not require any special equipment, and can achieve ultra-high resolution tiled displays using only a standard PC and open source software. In this paper, we have focused on explaining how to display ultra-high-resolution images of a single 3DWebGIS using a distributed rendering method, but ChOWDER can display multiple contents, including 3DWebGIS, on an ultra-high-resolution display at the same time. By displaying many data contents in formats that cannot be superimposed on GIS simultaneously with 3DWebGIS content, it provides a bird-view visualization environment that is impossible with the resolution of a standard single display. This system may be used for wide-area geographic space monitoring applications, such as the primary display of a local government disaster response headquarters. The challenge is finding potential users and testing use based on specific use cases.

In addition, the distributed rendering method, which uses the view frustum offset API of Three.js, can be applied not only to 3DWebGIS but also to ultra-high-resolution distributed rendering of general 3D graphics written in WebGL. However, there are issues with displaying dynamic content since there is currently no synchronization between display clients.

Acknowledgments

The authors thank DeepL (https://www.deepl.com/) and Grammarly (https://grammarly.com/) for editing English.

References

[1] Limitations. About NVIDIA Mosaic. [Online]. Available: https://www.nvidia.com/content/Control-Panel-Help/vLatest/en us/mergedProjects/nvwks/SLI_Mosaic_Mode.htm. [Accessed Jul. 29, 2024].

[2] Baxter, B. and Corriveau, P., (2005). PC Display Resolution Matched to the Limits of Visual Acuity. *Journal of the Society for Information Display*, Vol. 13(2), 169-174. https://doi.org/10.1889/1.2012600.

[3] 3DTiles, (2024). *The Open Specification for 3D Data*. [Online]. Available: https://cesium.com/why-cesium/3d-tiles/ [Accessed Jul. 29, 2024].

[4] Tokyo Digital Twin Project, (2024). [Online]. Available: https://info.tokyo-digitaltwin.metro.tokyo.lg.jp/. [Accessed Jul. 29, 2024].

[5] Kawanabe, T., Nonaka, J., Hatta, K. and Ono, K., (2018). ChOWDER: An Adaptive Tiled Display Wall Driver for Dynamic Remote Collaboration. In *nternational Conference on Cooperative Design, Visualization and Engineering*, Cham: Springer International Publishing, 11-15. https://doi.org/10.1007/978-3-030-00560-3_2.

[6] ChOWDER GitHub Repository, (2024). [Online]. Available: https://github.com/SIPupstreamDesign/ChOWDER. [Accessed Jul. 29, 2024].

[7] WebRTC, (2024). *Real-Time Communication for the Web*. [Online]. Available: https://webrtc.org/. [Accessed Jul. 29, 2024].

[8] Kawanabe, T., Hatta, K. and Ono, K., (2020). ChOWDER: A New Approach for Viewing 3D Web GIS on Ultra-High-Resolution Scalable Display. *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. 412-413. https://doi.org/10.1109/CLUSTER49012.2020.00055.

[9] iTowns (in French). [Online]. Available: https://www.itowns-project.org/. [Accessed Jul. 29, 2024].

[10] three.js API Reference. [Online]. Available: https://threejs.org/docs/#api/en/cameras/PerspectiveCamera.setViewOffset. [Accessed Jul. 29, 2024].

[11] Marrinan, T., Aurisano, J., Nishimoto, A., Bharadwaj, K., Mateevitsi, V. A., Renambot, L., Long, L., Johnson, A. E. and Leigh, J., (2014). SAGE2: A New Approach for Data Intensive Collaboration Using Scalable Resolution Shared Displays. *IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 177-186. https://doi.org/10.4108/icst.collaboratecom.2014.257337.

[12] Harden, J., Kirshenbaum, N., Tabalba Jr, R. S., Leigh, J., Renambot, L. and North, C., (2023). Sage3 for Interactive Collaborative Visualization, Analysis, and Storytelling. *Companion Proceedings of the 2023 Conference on Interactive Surfaces and Spaces*, 50-52. https://doi.org/10.1145/3626485.3626541.

[13] Leaflet. an open-source JavaScript Library for Mobile-Friendly Interactive Maps. [Online]. Available: https://leafletjs.com/. [Accessed Sep. 3, 2024].

[14] Liquid Galaxy, (2024). *A Cluster of Computers Running Google Earth, Street View, and other Panoramic Applications to Create an Immersive Experience*. [Online]. Available: https://liquidgalaxy. org/. [Accessed Sep. 3, 2024].

[15] Project PLATEAU Portal Site (in Japanese), (2024). [Online]. Available: https://www.geospatial.jp/ckan/dataset/plateau. [Accessed Jul. 29, 2024].