Research on the Display of Ultra-Large Point Cloud Data Using a 3DWebGIS Distributed Rendering System

Tomohiro KAWANABE[1], Satoshi TSURUTA[2], Minehito AGENO[2], Yoshihide KAGATANI[2], Kenji ONO[3]

[1] RIKEN Center for Computational Science, Kobe, Japan – tkawanabe@riken.jp
[2] GEO Solutions Inc, Nishinomiya, Japan - tsuruta@geo-sol.co.jp, ageno@geo-sol.co.jp, kagatani@geo-sol.co.jp
[3] Research Institute for Information Technology, Kyushu University, Fukuoka, Japan – keno@cc.kyushu-u.ac.jp

Keywords: FOSS4G, iTowns, Point cloud, 3DTiles, WebGL, Himawari

Abstract

This paper presents a method for ultra-high-resolution visualization of large-scale 3DWebGIS data using ChOWDER, a web-based scalable display system, combined with the open-source 3DWebGIS application iTowns. Web browsers impose a heap memory limit of about 4 GB, which restricts conventional WebGIS from handling very large 3D datasets. ChOWDER distributes 3DWebGIS rendering across multiple browser-based display clients, expanding both memory space and screen space. As a case study, we convert cloud data from the Himawari meteorological satellite into approximately 500 million 3D points, generate 3DTiles (about 6.1 GB, eight-level octree), and render them on a tiled display wall composed of fifteen 4K displays (20K horizontal resolution). Each display browser renders only its portion of the scene, enabling detailed inspection of cloud structures while preserving an overview.

During visualization, we observe triangular non-rendered regions in the point cloud. Analysis shows that these artifacts arise from the use of an Earth-centered Cartesian coordinate system (EPSG:4978), which causes 3DTiles bounding voxels to intersect the Earth's surface, combined with WebGIS behavior that occasionally fails to display tiles at the requested zoom level. We argue that such issues are inherent when mapping global-scale 3D data onto WebGIS platforms. As a mitigation strategy, we propose pre-segmenting global data into multiple regions and generating 3DTiles separately so that bounding volumes do not span the Earth's surface. Future work includes implementing and validating this region-segmented workflow, and comparing 3DTiles with alternative point-cloud formats such as Potree and COPC for performance and artifact behavior.

1. Introduction

In recent years, many software applications have migrated to cloud-based platforms, with web browsers serving as the primary user interface. In the geographic information field, the ease of 3D rendering in browsers using WebGL has led to the widespread adoption of 3DWebGIS applications in both commercial and open-source software. However, web browsers have limitations on the heap memory available to user programs, potentially causing instability or crashes when rendering large-scale 3D data that consumes significant memory. Our proposed scalable display system, "ChOWDER," utilizes the open-source 3DWebGIS iTowns (iTowns, 2025) as middleware to achieve distributed rendering of 3DWebGIS across multiple displays (web browsers). This mechanism enables ultra-high-resolution display of 3DWebGIS by overcoming the heap memory limitations of web browsers. We reported at FOSS4G ASIA 2024 on the distributed rendering technique for massive building polygon data in 3DTiles format on a 3x3 tile-based display composed of nine 4K displays using ChOWDER, along with its actual memory consumption and distributed performance (Kawanabe et al., 2024).

This paper is structured as follows: Section 2 provides an overview of ChOWDER itself and its 3DWebGIS distributed rendering functionality, Section 3 introduces a method to convert cloud data captured by a weather satellite over the Pacific Ocean into approximately 500 million 3DTiles format 3D point cloud data, and to distribute and render it at ultra-high resolution across fifteen 4K resolution displays (browsers) using ChOWDER. Section 4 discusses current challenges in rendering global-scale point clouds caused by unnatural artifacts and proposes solutions for these artifacts. Section 5 outlines the immediate development plan to address the challenges identified in the previous section. Finally, Section 6 concludes this paper.

2. ChOWDER: A Web Based Scalable Display System

2.1 Overview

ChOWDER (COoperative Workspace DrivER) (Kawanabe et al., 2018; ChOWDER, 2025) is a web-based open-source tiled display driver jointly developed by the RIKEN Center for Computational Science and Research Institute for Information Technology, Kyushu University. It achieves an ultra-high-resolution pixel space by arranging multiple full-screen web browsers in a tiled configuration, without requiring specialized hardware or software.

Its core concept is the Virtual Display Area (VDA), a virtual two-dimensional (x, y) coordinate space managed as metadata on the ChOWDER server, as shown in Figure 1. For each content item, the server stores metadata, such as its position on the VDA, zoom level, content type, and a pointer to the content (e.g., a URI), in a key-value store (KVS). Depending on the content type, the data itself may also be stored in the KVS and linked to its metadata. Furthermore, the server maintains the position and zoom level of each display client (browser window) on the VDA.

This contribution has been peer-reviewed.
https://doi.org/10.5194/isprs-archives-XLVIII-4-W20-2025-39-2026

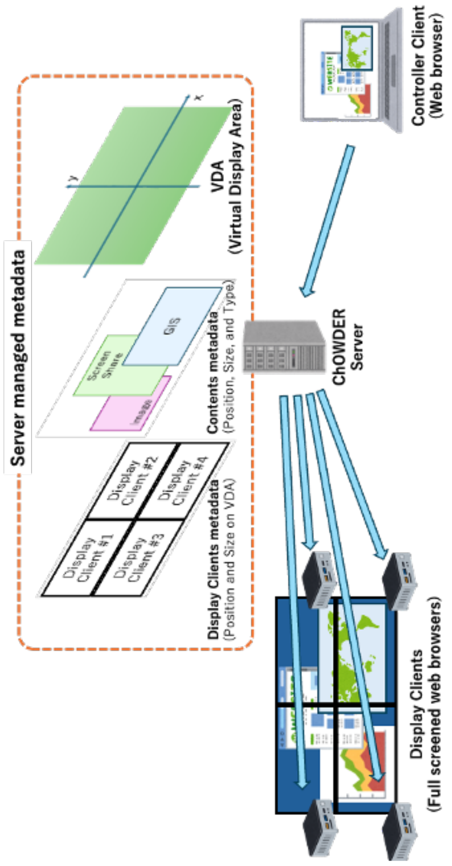

Figure 1. Virtual Display Area (VDA) concept.

ChOWDER is a server-client web application with two types of clients: display clients and control clients. Display clients are full-screen web browsers, and multiple display clients placed on a VDA form a tiled display. Overlapping multiple display clients on the VDA enables mirrored display, allowing the same content to be shared across multiple locations even when displays are in remote locations. The controller client is a web application running on the user's PC. It registers content and controls its position and zoom level on the VDA. It is used by the ChOWDER operator.

Static content, such as text, images, and PDFs, can be uploaded from the controller client to the server along with the desired position and magnification ratio on the VDA. Based on this information, the server selects the appropriate display client and transfers the content data along with the VDA metadata (display position and magnification ratio). Each display client calculates the relative position and magnification ratio of the content based on its own placement on the VDA and renders the content within the browser window. The magnification ratio is defined as the ratio between the content's original size and its size on the VDA, while preserving the aspect ratio. Since rendering is delegated to the web browser, raster images become pixelated when enlarged, whereas vector content like PDFs remains smooth.

ChOWDER supports dynamic content, including video, webcam feeds, and screen sharing. When a user specifies dynamic content and its placement on the controller client, WebRTC-based streaming communication begins (WebRTC, 2025). The server determines the responsible display client, and the dynamic content is delivered via peer-to-peer streaming between the controller client and the designated display client. Figure 2 shows an example of displaying various types of content (3DWebGIS, webcam feed, PDF, and text) on a 20K horizontal-resolution tiled display (consisting of 15 4K displays) using ChOWDER.

Figure 2. An example of displaying various types of content on a 20K horizontal resolution tiled display driven by ChOWDER.

2.2 3DWebGIS Distributed Rendering Function

3DWebGIS distributed rendering function was added in 2020 (Kawanabe et al., 2020). To use this function, the user launches both a 3DWebGIS client and a controller client on their PC. The 3DWebGIS client is a standard 3DWebGIS application extended to communicate with the ChOWDER server, and is implemented using the open-source iTowns framework (iTowns, 2025). The motivation for choosing iTowns is discussed later in this section.

First, the user accesses the ChOWDER server from a web browser on their PC to start the 3DWebGIS client. The server sends the iTowns-based application code to the browser and registers metadata for the launched 3DWebGIS content, including its initial position on the VDA, magnification ratio, and initial camera parameters for 3D rendering of GIS content. This metadata is propagated to both the display clients and the controller client. Each display client checks whether the new 3DWebGIS content falls within its display region on the VDA. If it does, the client creates an iframe (an HTML inline frame for embedded content) with the appropriate display size and initially draws the 3DWebGIS content there. The same iframe and content are also rendered on the controller client. The user can move, enlarge, or reduce the 3DWebGIS content on the controller, and the corresponding content on each display client is updated accordingly.

When the user changes the viewpoint on the 3DWebGIS client running on their PC (for example, by zooming in), the map server receives requests for new map tiles at the updated zoom level, and the 3DWebGIS client display is refreshed, as with a regular 3DWebGIS application. At the same time, metadata describing the change in zooming ratio is sent via the ChOWDER server to each display client. Each display client then requests the relevant map tiles within its own screen area from the map server and displays them. In this way, independently running 3DWebGIS instances on multiple display clients can operate synchronously with only lightweight metadata communication.

To distribute 3DWebGIS rendering across multiple displays, the view frustum must be divided appropriately. ChOWDER's

3DWebGIS function relies on iTowns, which itself uses Three.js, a cross-browser JavaScript library and API for WebGL-based 3D graphics. Three.js provides an API to offset the view frustum (Three.js, 2025). ChOWDER exploits this API to split a single iTowns scene into multiple view frustums, allowing several web browsers to render different portions of the single 3DWebGIS content cooperatively.

This paper primarily focuses on large-scale data visualization using distributed displays. However, ChOWDER also enables the simultaneous display of multiple 3DWebGIS contents. As shown in Figure 3, it can display multiple 3DWebGIS contents controlled by multiple controller clients on a large-scale tiled display. Utilizing this feature allows multiple operators to simultaneously view distinct GIS scenes, expected to foster discussions that differ from those held thus far.

Figure 3. Use case where multiple people simultaneously view and discuss WebGIS.

2.3 Memory Distribution Effect of 3DWebGIS Distributed Rendering Function

In our previous report (Kawanabe et al., 2024), we measured the heap memory consumption of each browser when ChOWDER distributes and renders 3DWebGIS content across multiple web browsers, and we confirmed that this method achieves effective distribution of memory load.

The maximum user heap memory allowed by modern web browsers, such as Google Chrome and Mozilla Firefox, is currently 4GB. This means that regardless of the amount of memory installed on the PC, the total size of the application program running in the web browser and its associated data cannot exceed 4GB. This is one reason why WebGIS cannot handle large-scale data.

By utilizing the functionality we propose, users gain two key advantages: handling large-scale data through memory distribution effects and ultra-high-resolution display using multiple monitors.

3. 3D Visualization of Meteorological Satellite Data

This section introduces a method for converting Earth-scale meteorological satellite data into 3D point clouds and visualizing them on distributed displays.

3.1 Meteorological Satellite Himawari

The Himawari geostationary satellite system, operated by the Japan Meteorological Agency, is critical for meteorological monitoring across the Asia-Pacific region. Utilizing the Advanced Himawari Imager (AHI), Himawari-8 and 9 provide high-resolution multispectral data with 10-minute full-disk scanning. This high-frequency observation is indispensable for analysing rapid atmospheric dynamics, significantly enhancing numerical weather prediction accuracy and contributing to international disaster risk reduction strategies.

The two-dimensional cloud image data is published in quasi-real time as a multilingual web application, accessible to anyone (Himawari Web, 2025) (Figure 4). Furthermore, the Solar Radiation Consortium (Amaterass.org, 2025) provides research institutions with highly precise, geometrically corrected data in a 4km grid format, totaling 3000x3000 grids (Takenaka et al., 2020; Yamamoto et al., 2020). This publicly available grid data includes not only raw data but also derived data, such as cloud top height and cloud thickness.

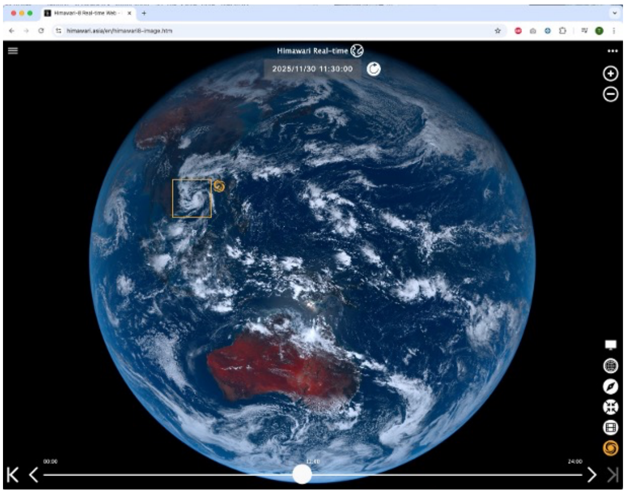

Figure 4. Himawari Real-time Web.

3.2 Data Conversion Workflow

Using this publicly available grid data, voxels are defined within each grid cell, with the cloud top height as the maximum height and the height obtained by subtracting cloud thickness from the cloud top height as the minimum height. By randomly generating multiple points within these voxels, the 2D observation data is represented as a 3D cloud shape within the point cloud data. This conversion program was developed as an open-source program distributed on GitHub (Point Cloud Converter, 2020).

The generated point cloud data is converted to binary LAS format using txt2las from LAStools (LAStools, 2025), and then converted to 3DTiles using py3dtiles (py3dtiles, 2025). This study used data captured by the satellite at 00:00 UTC on October 10, 2019. The generated 3DTiles data has an octree structure with 8 levels, a total data size of approximately 6.1GB, about 75,000 files, and a total of approximately 500 million points.

Figure 5 shows an example of visualizing this data in 3D using the standard iTowns application. The point cloud color corresponds to the cloud top height. Whiter areas indicate higher cloud top heights. It can be seen that the cloud top height is high around the typhoon's eye and low inside the eye.

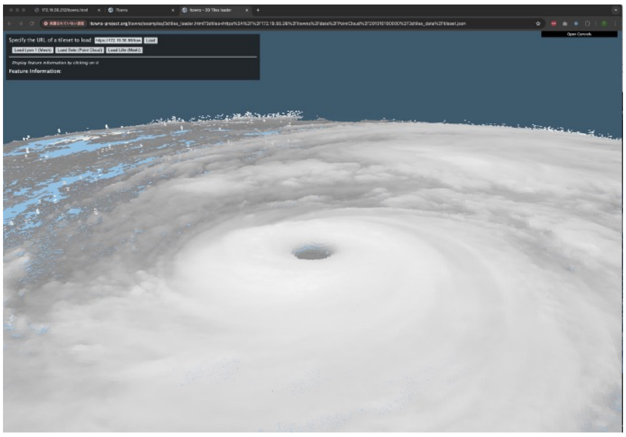

Figure 5. Visualization example of generated 3D cloud point cloud data.

3.3 Visualization Device

This subsection describes the distributed rendering system used to visualize the data. As shown in Figure 6, it consists of a tiled display with a horizontal resolution of 20K, formed by arranging fifteen 4K resolution displays in a 5x3 grid. Each display is connected to a separate Windows PC. On the local network, one Mac mini serves as the ChOWDER server, and another serves as the map server. The map server stores Japanese Geospatial Information Authority map tiles (including DEM) and OpenStreetMap tiles. This setup is used to minimize network latency during performance testing. For regular use, it is possible to access map servers on the internet instead of using this local map server. The controller client utilizes a laptop connected via Wi-Fi.

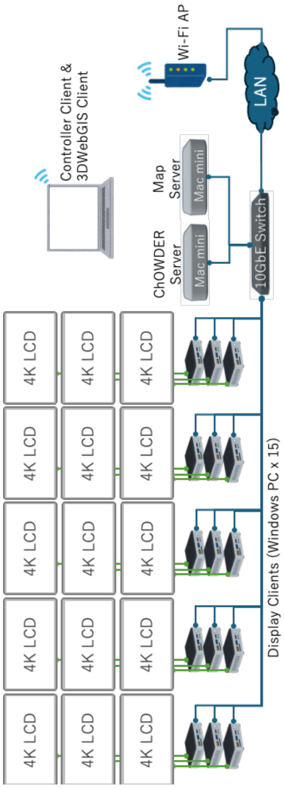

Figure 6. A diagram of the visualization device.

3.4 Advantage of Distributed Rendering

The data size used in this visualization example is manageable for standard 3DWebGIS tools such as Cesium (Cesium, 2025). However, when viewing the entire dataset in a web browser on a standard desktop environment (typically with a maximum resolution of 4K-5K), the point cloud appears at a coarse granularity. To observe the detailed structure of the cloud, zooming in is necessary, making it impossible to display the entire area and examine details simultaneously. However, by using ChOWDER, the rendering of 3DWebGIS can be distributed across multiple displays (web browsers), enabling an ultra-high-resolution display that would be unachievable on a single display. An example of this visualising is shown in Figure 7.

Furthermore, even if the amount of data does not reach the user's heap memory limit, the rendering cost per display is reduced by distributing the data to be rendered across multiple displays (web browsers), which can result in faster rendering. This is particularly effective when a high performance GPU (Graphics Processing Unit) is not installed.

Figure 7. An example of distributed rendering of the Earth-scale point cloud data.

4. Challenges and Reflections

Visualizing the data shown in Section 3 reveals triangular artifacts, as depicted in Figure 8. This phenomenon occurs not only with ChOWDER's distributed rendering feature but also in both iTowns and Cesium on a single display. The primary reason is that the generated point cloud uses an Earth-centered, orthogonal coordinate system (EPSG:4978). Since the point cloud covers nearly half of Earth's surface, converting it to 3DTiles in a single batch results in a boundary voxel encompassing approximately half of Earth's surface. When this is subdivided into an octree structure, the voxel is split along orthogonal coordinate axes. This causes the intersection points between the subdivided voxel and the Earth's surface to form triangular shapes at specific locations (Figure 9).

Furthermore, in WebGIS, a phenomenon occurs where map tiles corresponding to the display zoom level fail to render for some reason. This is a common issue that occurs not only in iTowns but also in other WebGIS systems. This issue arises not only with 2D map tiles but also with 3DTiles data. In the case of this data, it manifests as non-rendered triangular areas. Typically, 3DTiles are generated for relatively small areas, so boundary voxels rarely intersect the Earth's surface. Consequently, even if tiles for a zoom level that should be displayed are not rendered, rendering artifacts do not occur.

Based on the above reflections, when converting global-scale data, such as the cloud data used in this study, to 3DTiles, it is expected that applying a method where the data is pre-divided into multiple regions and the 3DTiles conversion is performed region by region, ensuring the bounding boxes do not intersect with the Earth's surface, can suppress the occurrence of unnatural artifacts.

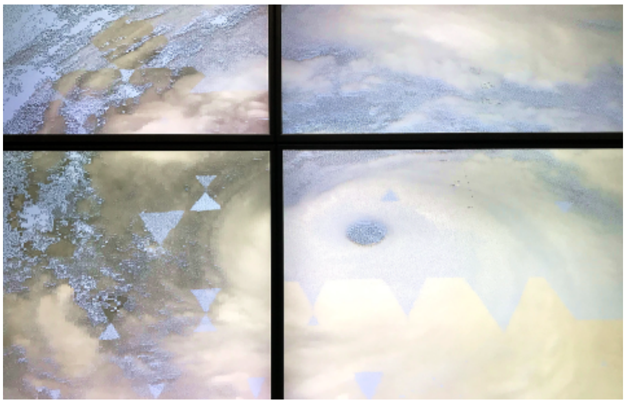

Figure 8. Triangular artifacts appearing during cloud's point cloud display.

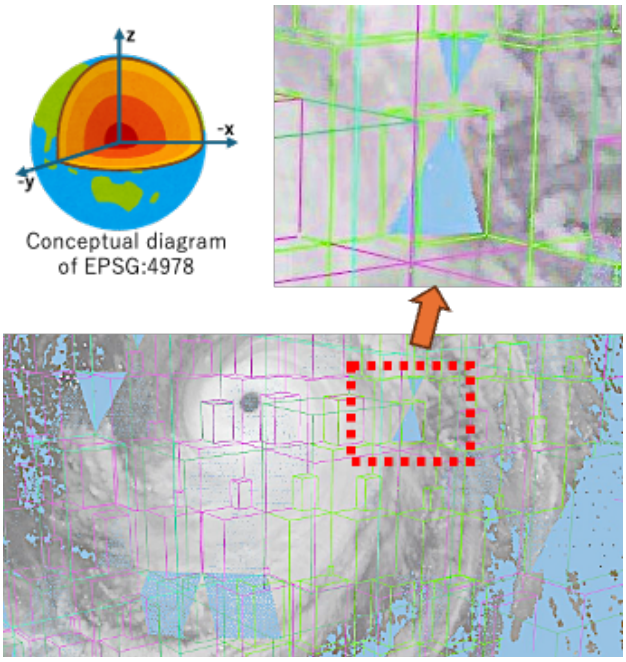

Figure 9. 3DTiles bounding voxel and ground surface intersect in triangular shape.

5. Future Work

We plan to verify whether the pre-segmentation method for regions during 3DTiles generation proposed in the previous section can suppress the occurrence of unnatural artifacts. Furthermore, while current methods estimate cloud three-dimensional structures based on two-dimensional observation data, next-generation meteorological satellite will be expected to provide three-dimensional observation data from infrared sounder (Himawar-10, 2025), enabling scientifically accurate, high-resolution three-dimensional structure visualization.

Additionally, we plan to compare the performance of 3DTiles, Potree (Potree, 2025), and COPC (COPC, 2025) as point cloud data formats. Preliminary experiments using cloud data in this paper showed that Potree was the fastest, but triangular artifacts occurred similarly across all data formats. Therefore, verifying the aforementioned domain pre-dividing method takes priority.

6. Conclusion

This paper introduced a method for distributed rendering of 3DWebGIS across multiple displays (web browsers) using the open-source scalable display system ChOWDER. As a large-scale data application example, it demonstrated converting cloud data acquired by meteorological satellite covering a hemispheric scale of the Earth into point cloud data in the 3DTiles format. and then distributed rendering it across a tiled display system comprising fifteen 4K resolution displays, achieving a horizontal resolution of 20K visualization.

In this visualization case, triangular non-rendering areas were observed within portions of the cloud point cloud. We identified this as a compound issue: the conversion procedure to point cloud data uses an Earth-centered origin orthogonal coordinate system, causing intersections between 3DTiles' bounding voxels and the Earth's surface to form triangles. Additionally, the WebGIS application often cannot display data at the requested zoom level. This issue demonstrates that when attempting to overlay global-scale 3D data onto WebGIS, it is an unavoidable problem that must always be considered, not limited to this specific dataset.

The issue on the data conversion side is expected to be mitigated by performing region segmentation during data conversion to suppress intersections between 3DTiles' bounding voxels and the Earth's surface. Therefore, the next step involves developing and validating a data conversion workflow that incorporates region segmentation.

Acknowledgements

Himawari 8/9 gridded data are distributed by the Center for Environmental Remote Sensing (CEReS), Chiba University, Japan.

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

References

Amaterass.org, 2025: The Solar Radiation Consortium, http://www.amaterass.org/index-e.html (Accessed 30 Nov. 2025).

Cesium, 2025: Software, The Platform for 3D Geospatial, https://cesium.com/ (Accessed 30 Nov. 2025).

ChOWDER, 2025: Software, COoperative Workspace DrivER, RIKEN Center for Computational Science, https://github.com/SIPupstreamDesign/ChOWDER (Accessed 30 Nov. 2025).

COPC, 2025: Software, Cloud Optimized Point Cloud, https://copc.io/ (Accessed 30 Nov. 2025).

Himawari-10, 2025: Kotaro Bessho, Akiyoshi Andou, Yasuhiko Sumida, Miki Abe, Himawari-10 Project Overview, SOLA, 2025, Volume 21, Pages 251-257, Released on J-STAGE July 05, 2025, Advance online publication May 16, 2025, Online ISSN 1349-6476, https://doi.org/10.2151/sola.2025-031

Himawari Web, 2025: Himawari Real-time Web, https://himawari.asia/ (Accessed 30 Nov. 2025).

iTowns, 2025: Software, Version 2.46.0, MATIS research laboratory, IGN, https://www.itowns-project.org/ (Accessed 30 Nov. 2025).

Kawanabe, T., Nonaka, J., Hatta, K., Ono, K. (2018). ChOWDER: An Adaptive Tiled Display Wall Driver for Dynamic Remote Collaboration. In: Luo, Y. (eds) Cooperative Design, Visualization, and Engineering. CDVE 2018. Lecture Notes in Computer Science(), vol 11151. Springer, Cham. https://doi.org/10.1007/978-3-030-00560-3_2

Kawanabe, T., Tsuruta, S., Ageno, M., Kagatani, Y., & Ono, K. (2024). On the Performance of Distributed Rendering System for 3DWebGIS Application on Ultra-High-Resolution Display. International Journal of Geoinformatics, 21(1), 15–25. https://doi.org/10.52939/ijg.v21i1.3785

LAStools, 2025: Software, Tools to process LiDAR data files, https://lastools.github.io/ (Accessed 30 Nov. 2025).

Point Cloud Converter, 2020: Software, RIKEN Center for Computational Science, https://github.com/SIPupstreamDesign/ChOWDER/blob/master/server/parser/amaterass_convert_test.js (Accessed 30 Nov. 2025).

Potree, 2025: Software, WebGL point cloud viewer for large datasets, https://github.com/potree/potree (Accessed 30 Nov. 2025).

py3dtiles, 2025: Software, A python library and command-line tool designed for generating and managing 3D Tiles datasets, https://py3dtiles.org/ (Accessed 30 Nov. 2025).

Takenaka, H., T. Sakashita, A. Higuchi, T. Nakajima (2020): Development of geolocation correction for geostationary satellite observation by phase only correlation method using visible channel, Remote Sensing, 12 (15), 2472 doi:10.3390/rs12152472.

Three.js, 2025: Software, Java Script 3D Library, https://threejs.org/docs/#PerspectiveCamera.setViewOffset (Accessed 30 Nov. 2025).

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

Yamamoto, Y., K. Ichii, A. Higuchi, H. Takenaka (2020): Geolocation accuracy assessment of Himawari-8/AHI imagery for application to terrestrial monitoring, Remote Sensing, 12 (9), 1372 doi:10.3390/rs12091372.

WebRTC, 2025: Software, Real-time communication for the web, https://webrtc.org/ (Accessed 30 Nov. 2025).