IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 June 2023, accepted 29 July 2023, date of publication 2 August 2023, date of current version 10 August 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3301188*

RESEARCH ARTICLE

# Large-Scale Oceanic Dynamic Field Visualization Based on WebGL

**DONGLIN FAN[1,2], TIANLONG LIANG[1,2], HONGCHANG HE[1], MENGYUAN GUO[3], AND MENGHUI WANG[1,2]**

[1]College of Geomatics and Geoinformation, Guilin University of Technology, Guilin 541006, China
[2]Ecological Spatiotemporal Big Data Perception Service Laboratory, Guilin University of Technology, Guilin 541006, China
[3]College of Arts, Guilin University of Technology, Guilin 541006, China

Corresponding author: Hongchang He (hhe@glut.edu.cn)

This work was supported in part by the Natural Science Foundation of Guangxi Province under Grant 2022GXNSFBA035637, in part by the Basic Scientific Research Ability Improvement Project for Young and Middle-Aged Teachers of Universities in Guangxi under Grant 2021KY0255, and in part by the "BaGui Scholars" Program of the Provincial Government of Guangxi.

**ABSTRACT** The dynamic visualization of ocean dynamics data is essential for effectively presenting oceanic information data. However, with the increasing spatial resolution of ocean remote sensing data, performing global-scale visualization in a single pass has become challenging. Directly rendering ocean data with high spatial resolution can result in problems such as increased communication time, blocked resource loading, and page lag. To address this issue, this paper proposes a common ocean data standard based on the characteristics of massive ocean element data, constructs an efficient three-level LOD model for ocean dynamic field rendering, and achieves particle rendering of ocean dynamic data by converting ocean data into image slices to complete the rendering of high-volume ocean data on the Web side. Compared with traditional methods, the proposed method significantly improves rendering performance. The average window construction time (FT) is reduced to 51.77ms, enhancing the overall rendering speed by approximately 64%. Meanwhile, the average frames per second (FPS) increase to 57.45fps, augmenting rendering stability and smoothness by around 18%. The peak memory consumption of the highest resolution data used in this paper is about 90MB, which is only 1/4 of the original. The proposed method effectively compensates for the disadvantage of slow rendering of large-scale ocean data visualization in some systems, enabling fast rendering of dynamic ocean data on the Web.

**INDEX TERMS** Oceanic engineering and marine technology, oceanography, application software.

## I. INTRODUCTION

The oceanic dynamic field generally refers to a field of force generated by natural phenomena such as ocean currents, wind, and tides in the ocean [1]. Visualizing this field allows for the extraction and intuitive display of information, which is crucial for understanding patterns of change in the marine environment [2], [3]. With advancements in remote sensing techniques and vector visualization methods, along with the power of HTML5 technology, it has become possible to visualize oceanic dynamic fields on a large scale [4], [5], [6], [7]. To achieve dynamic visualization of oceanic dynamic data, it is necessary to transmit the data directly to the browser for rendering [8], [9]. However, global network congestion during the transmission of oceanic dynamic data can cause webpage lagging and negatively impact the efficiency and user experience of data rendering [10]. Additionally, rendering data directly using the CPU in the browser can further contribute to webpage lagging and delays [11]. To address these challenges, this paper proposes a method for dynamically visualizing large-scale oceanic dynamic data based on WebGL.

Advancements in oceanic visualization technology and network application architectures in recent decades have laid the technological groundwork for large-scale oceanic data visualization [12], [13], [14]. Some researchers have approached oceanic data processing from a data perspective and proposed methods to reduce network transmission for

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Wang.

individual pieces of oceanic data, such as data compression [15] and data partitioning [16]. If we consider only one aspect, even with the latest remote sensing data compression technology, it isn't easy to meet the requirements for large-scale ocean data visualization. [17], [18], [19]. Thus, it becomes necessary to address the issue of data volume from alternative perspectives.

Level Of Detail (LOD) techniques [20] can tackle the problem by focusing on the ''transmission'' aspect. By transmitting data with varying levels of precision based on user requirements, the efficiency and quality of oceanic remote sensing data transmission and utilization can be significantly improved [21]. There is existing evidence to suggest that using LOD technology can effectively enhance the rendering performance of ocean data [10]. Furthermore, by enhancing existing LOD models, LOD models specifically designed for ocean data are more targeted and can significantly improve rendering efficiency and quality [22]. Therefore, establishing a generalized LOD oceanic data processing model based on LOD technology, combined with methods such as data compression and downsampling, can substantially improve rendering efficiency.

Within the realm of visualization, oceanic dynamic fields are classified as vector fields, encompassing phenomena such as ocean currents, winds, and related factors [23]. Traditional methods for visualizing vector fields include the use of point glyphs and line glyphs, which map geometric shapes, as well as the point noise method that relies on texture generation mapping [24], and the line integral convolution method [25]. However, these approaches encounter challenges in effectively capturing the continuity and complex structural characteristics of vector fields, often resulting in high computational overhead and low rendering efficiency [26]. To overcome these limitations, a particle system-based multilevel ocean flow dynamic visualization method [5] has been proposed to address the shortcomings of traditional arrow symbols and streamline methods. This method can clearly and efficiently represent ocean vector field data while improving rendering efficiency. This method has become a leading commercial approach for vector field visualization and is currently widely used in visualization systems related to flow fields [27].

In the web application architecture, the Browser/Server (B/S) architecture has become a superior choice for modern web applications due to its advantages, such as cross-platform compatibility, security, scalability, maintainability, and user experience [28]. HTML5, a key component of the B/S architecture, plays a significant role in oceanic visualization research [29]. The Canvas API within HTML5 enables the depiction of oceanic features and dynamic changes through the rendering of static graphics and dynamic animations [30]. However, when dealing with large-scale oceanic data, the Canvas API's performance and data processing speed may become limiting factors, making real-time visualization challenging [11]. This limitation will be discussed further in the third section of this paper. WebGL, a web graphics library based on OpenGL ES, provides optimizations such as shader program optimization, GPU parallel computing, data batching, texture usage optimization, and caching, which can enhance the efficiency and quality of oceanic visualization [7]. Despite this, traditional ocean flow visualization methods based on WebGL and GPU acceleration, while improving rendering efficiency, have failed to achieve the desired visual effect [31]. To address this limitation, combining the aforementioned particle system visualization method with certain WebGL extensions can significantly enhance the visualization of ocean flow and improve the performance and quality of GPU-accelerated particle systems [32]. Consequently, there is significant potential for large-scale oceanic dynamic field visualization based on WebGL.

Oceanic dynamic field visualization can help fishermen, oceanographers, coastal managers, and others to obtain information about oceanic dynamic fields better, allowing for further analysis and understanding of their spatiotemporal patterns [33]. Despite significant achievements in remote sensing technology, visualization technology, and web application architecture [4], [28], [34], there are still significant challenges in achieving large-scale, high-precision, and efficient oceanic dynamic field visualization. Therefore, based on the advantages of the aforementioned technologies, this article proposes a large-scale oceanic dynamic field visualization method. By utilizing this method, it is possible to significantly reduce the size of individual transmissions of ocean data within a limited window, which can be leveraged to render visualizations of ocean dynamics efficiently. The specific contributions to the field of ocean visualization research are as follows:

- A universal LOD ocean data processing model has been established. By utilizing LOD techniques in combination with data compression, data resampling, and data partitioning, the original data can be effectively processed, significantly reducing the amount of data required to be transmitted for a single data rendering.
- A particle-based dynamic fine-flow method was employed to develop a process for visualizing ocean dynamics. By mapping particle length, width, head and tail size to velocity and map viewport scaling level, the ocean dynamics data can be expressed with clarity and beauty while also improving the efficiency of data rendering.
- A global ocean dynamics visualization has been implemented using WebGL. Compared to Canvas, using the WebGL framework for rendering ocean dynamics data has greatly improved the efficiency of data rendering.

The rest of this article is organized as follows. Section II provides an overview of the system architecture, including the technical process, standardization rules, and rendering process. Section III describes a series of experiments conducted based on this architecture, demonstrating the effectiveness and superior performance of the visualization rendering. Finally, Section IV presents the conclusion.

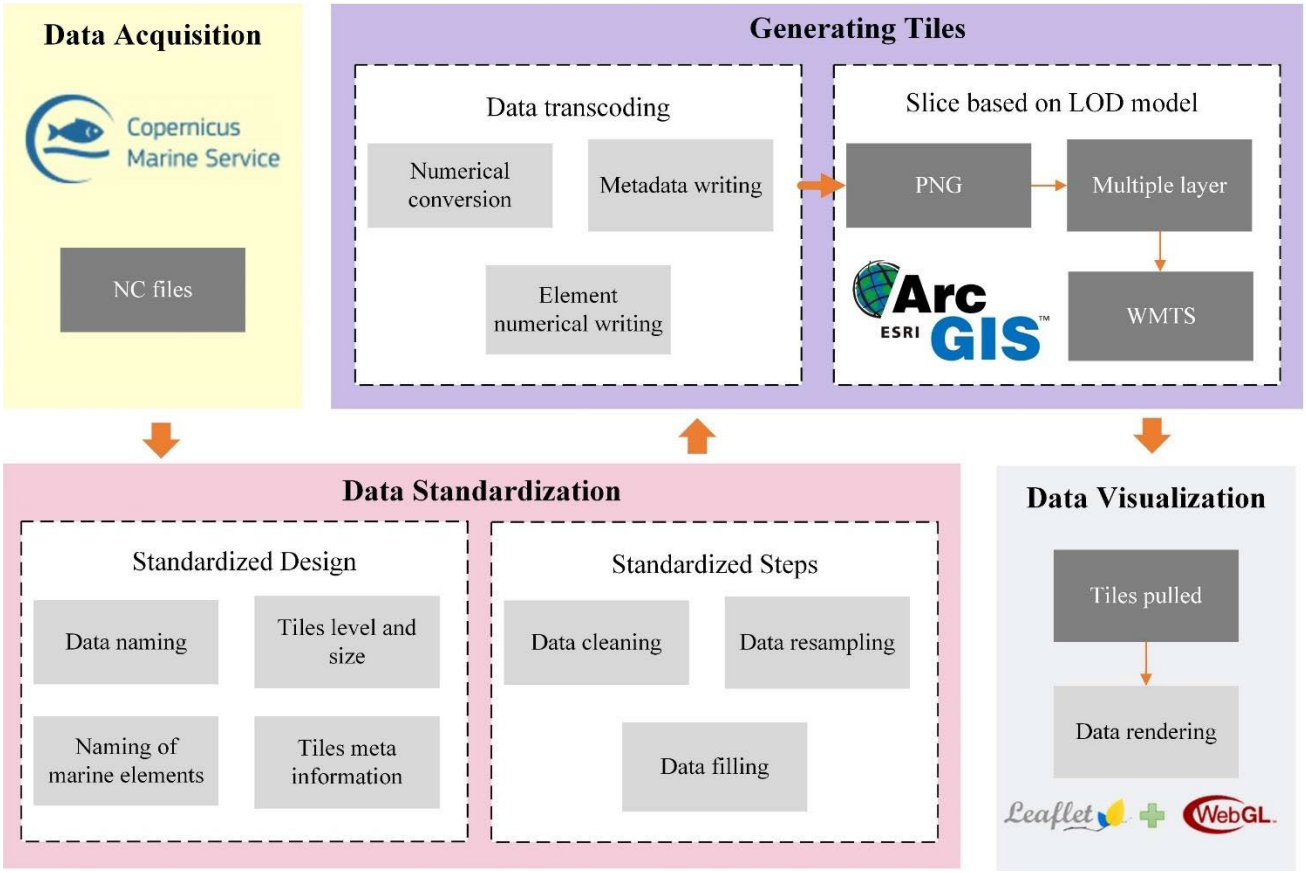


**FIGURE 1.** Technical flow chart.

## II. ARCHITECTURE DESIGN

This paper introduces a multi-source data processing and publication standard to provide a universal data rendering method compatible with various oceanic dynamic data sources. Each type of oceanic dynamic data was resampled at four LOD according to the standardization rules. The process achieves numerical conversion of the data, metadata writing, and oceanic element value writing and facilitates slice service publication by utilizing a web server. Finally, a marine data rendering mode is designed based on the target's and oceanic data's characteristics. This model is utilized to render vector marine data, following the technical process illustrated in FIGURE 1.

### A. DATA STANDARDIZATION

This paper presents a comprehensive set of data standards encompassing data naming conventions, tile hierarchy and size, and tile metadata to ensure seamless data transmission and visualization of oceanographic data from different sources. The proposed data naming conventions establish a consistent organization scheme based on the data's source, type, time, and depth. The following rules govern the naming conventions:

$$\begin{aligned}&(\text{Data Type}) - (\text{Data Source})\\ &\quad - (\text{YYYYmmDDHHMMSS}) - (\text{Depth})\end{aligned} \tag{1}$$

where: YYYY represents the year, mm represents the month, DD represents the day, HH represents the hour, MM represents the minute, and SS represents the second.

The process of data standardization also encompasses data filling and data resampling [26]. Data filling comprises two aspects: filling missing data within the entire original dataset by assigning the mean value of the $3 \times 3$ window surrounding the missing value, and filling data boundaries when they fall outside the range of longitude $[-180, 180]$ and latitude $[-90, 90]$. Data resampling involves extracting low-resolution images from high-resolution remote sensing images. In this study, the data was resampled into four levels of spatial resolution: 0.5°, 0.25°, 0.1°, and 0.05°, corresponding to levels 0, 1, 2, and 3, respectively. Bilinear interpolation was employed as the interpolation algorithm, which computes the value of the target point by interpolating the four neighboring points of the original point. The specific formula used is as follows:

$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \tag{2}$$

$$f(R_i) = \frac{x_2 - x}{x_2 - x_1} f(Q_{1i}) + \frac{x - x_1}{x_2 - x_1} f(Q_{2i}) \tag{3}$$

The formula can be expressed as follows: P represents the pixel coordinate value in the target image; Ri represents the coordinate value of the i-th interpolation in the x direction;

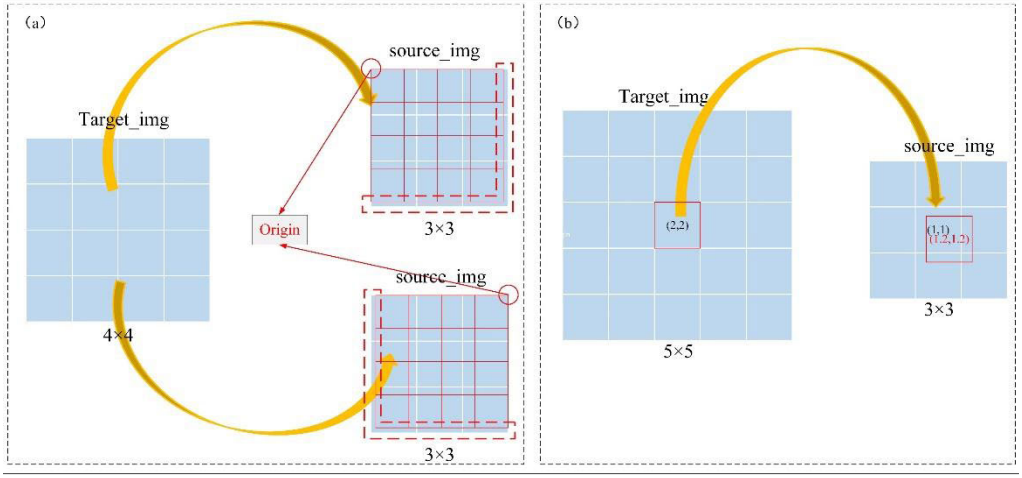


**FIGURE 2.** Schematic diagram for addressing the problem of resampling.

$Q_{1i}$ and $Q_{2i}$ represent the pixel coordinate values of the four original images; (x, y) is the pixel coordinate of the target image; $(x_1, y_1)$, $(x_1, y_2)$, $(x_2, y_1)$, and $(x_2, y_2)$ correspond to the coordinates of $Q_{11}$, $Q_{12}$, $Q_{21}$, and $Q_{22}$, respectively.

However, there are two issues with the resampling process using the above formula. Firstly, FIGURE 2a illustrates the resampling results with different coordinate system origins at the top left and top right corners, indicating that inconsistent results are produced based on the selected coordinate system origin. Secondly, FIGURE 2b shows that the geometric centre of the original image is (1,1) while that of the target image is (2,2). According to the corresponding relationship, the position of the geometric centre of the target image corresponds to (1.2,1.2) in the original image, indicating that the centre point of the target image is offset relative to the geometric centre point of the original image. This causes the overall image position to shift and causes a loss of relative positional information due to all participating points being shifted down and to the right. Therefore, it is necessary to optimize the bi-linear interpolation formula by aligning the geometric centre points to solve the issues mentioned above. The final optimized bi-linear interpolation formula is as follows:

$$\text{src\_x} = (\text{des\_x} + 0.5) \times \frac{\text{src\_w}}{\text{des\_w}} - 0.5 \tag{4}$$

$$\text{src\_y} = (\text{des\_y} + 0.5) \times \frac{\text{src\_h}}{\text{des\_h}} - 0.5 \tag{5}$$

The formula can be expressed as follows: des_x represents the x-coordinate value of a pixel in the target image, while des_y represents its y-coordinate value. src_w refers to the width of the source image, and des_w refers to the width of the target image. Similarly, src_h refers to the height of the source image, and des_h refers to the height of the target image.

### B. DATA SLICING

After the data resampling, the next step is data slicing, which involves metadata writing and value conversion. For the slicing level and size design, a four-layer pyramid structure is employed to store the ocean feature data values at different levels. The spatial resolutions of these four levels are 0.5°, 0.25°, 0.1°, and 0.05°, corresponding to levels 0, 1, 2, and 3, respectively. Each level covers a spatial range of longitude [−180, 180] and latitude [−90, 90]. Adhering to the principle of equal dimensions, the size of each tile is set as 375 × 370, as illustrated in FIGURE 3. The first five rows of each tile are reserved for metadata, as indicated in TABLE 1. Additionally, there is a 5-element overlap area surrounding each tile. Consequently, the size of the original data tile is 360 × 360.

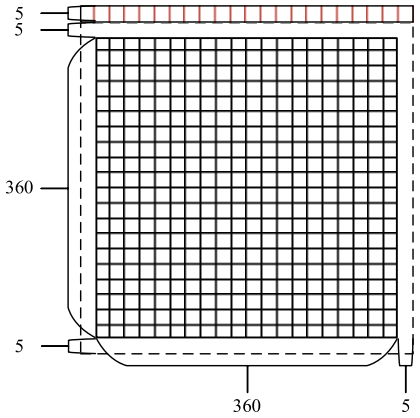


**FIGURE 3.** Schematic diagram of tile picture size.

**TABLE 1.** Meta information to be written.

| Index | Metadata Name | Write Line Number | Band |
|---|---|---|---|
| 1 | Marine data type | First row | R |
| 2 | Upper-left longitude | First row | G |
| 3 | Upper-left latitude | First row | B |
| 4 | Spatial resolution | Second row | R |
| 5 | Number of tile rows | Second row | G |
| 6 | Number of tile columns | Second row | B |
| 7 | Invalid value | Third row | R |
| 8 | Maximum value | Third row | G |
| 9 | Minimum value | Third row | B |

In this study, the oceanographic data, initially in floating-point format, is converted to PNG format using the ocean data model constructed. This conversion involves a numerical transformation process. A linear transformation is employed to map the data values onto the range of 0-255, which corresponds to the integer pixel values in the PNG format. The linear transformation formula used in this process is as follows:

$$\text{encodeValue} = \text{ceil}\left(\frac{\text{realValue} - \text{min}}{\text{max} - \text{min}} \times 255\right) \quad (6)$$

$$\text{decodeValue} = \left(\frac{\text{encodeValue}}{255} \times (\text{max} - \text{min}) + \text{min}\right) \quad (7)$$

The formula can be expressed as follows: realValue represents the actual numerical value of oceanographic data, while min and max respectively denote the minimum and maximum values of the oceanographic data in the current tile.

After completing the abovementioned steps, the original data transforms from a floating-point type to an integer type, and the decoded metadata is stored as character-based information. For ocean dynamics data, such as ocean currents, winds, and waves, two data values are simultaneously present, necessitating the recording of both values. For instance, the actual ocean currents consist of u and v components, where u represents the eastward current value, and v represents the northward current value. Following the numerical conversion method employed in this study, u is obtained and recorded in the R band, the converted v is recorded in the G band, the B band is used to record the sign, and the alpha band is used to record the value of dir $= u^2+v^2$. The recording rules for each byte (8 bits) in the B band are as follows: the first and second bits represent the signs of the u and v data, respectively. A value of 1 for the first bit indicates a positive sign for u, while a value of 1 for the second bit indicates a positive sign for v.

Following the conversion of image formats, the data processing LOD model is used to process the sliced images. The images are then organized by depth, level, latitude direction index, and longitude direction index based on the Web Server. Finally, the service publication of the ocean environmental data layer is completed, as shown in FIGURE 4.

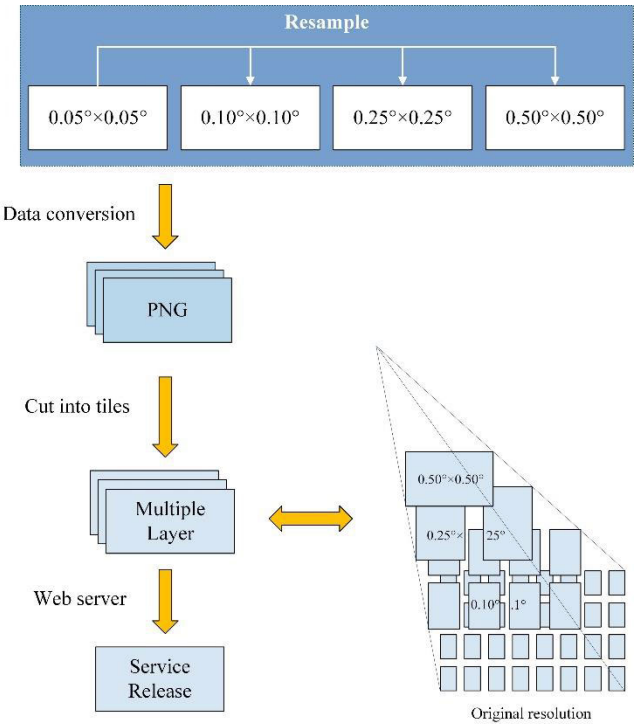


**FIGURE 4.** Schematic diagram of data processing LOD model.

### C. OCEAN DATA VISUALIZATION

In order to accommodate the differences between the data tiles in this paper and the standard WMTS (Web Map Tile Service) service tiles, a logical process for data retrieval within the map service framework needs to be established. Considering the characteristics of data slicing explained earlier, and the visualization requirements of oceanographic data, a class structure related to data retrieval and rendering is designed, consisting primarily of three groups: Tile, Layer, and Render. The Tile group represents the sliced oceanographic data, the Layer group represents the oceanographic data layers, and the Render group handles data rendering. Scalar represents scalar oceanographic data within these groups, and Vector represents vector oceanographic data, as illustrated in FIGURE 5. For this paper, we are only focusing on the visualization of vector data.

By converting the standard map service display level to the four LODs level, the appropriate data zoom level was determined based on the map window's current zoom level. Next, the corresponding set of tiles in the current display range of the window and enqueue all the computed tile sets for loading. The frontend application maintains a separate cache queue and iterates through the loading queue. Web Pull operation for each tile was conducted in the loading queue, if not in the cache queue. Once a tile finishes loading, the frontend saves a logical window (WindowField) that aligns with the map view range. The values within this logical window are obtained by sampling the loaded tiles using the sampling logic depicted in FIGURE 6.

The next step involves retrieving the corresponding tile images for dynamic ocean data and constructing the logical window. This process generates the WindowField, which includes the u, v, and dir data of the target ocean dynamic data. The dir data is represented using a colour table, and a spatial transformation is applied to the values within the WindowField to map them to a predefined colour gradient. Then, a dynamic particle animation is created, which response to re-rendering functions triggered by zooming or panning of the map. This process enables the reconstruction of the WindowField and the rendering of pixels within the viewport, ultimately achieving the visualization of dynamic ocean data.

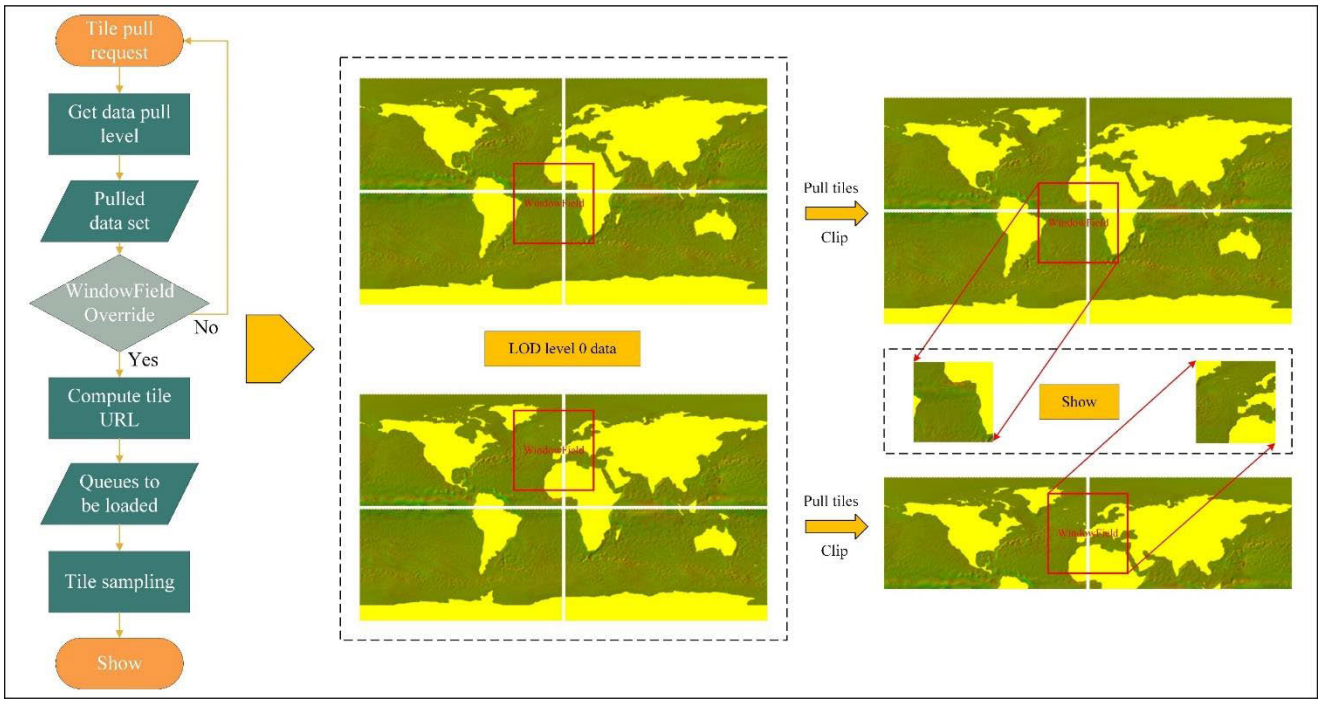


**FIGURE 5.** Tiles pull the logic diagram.

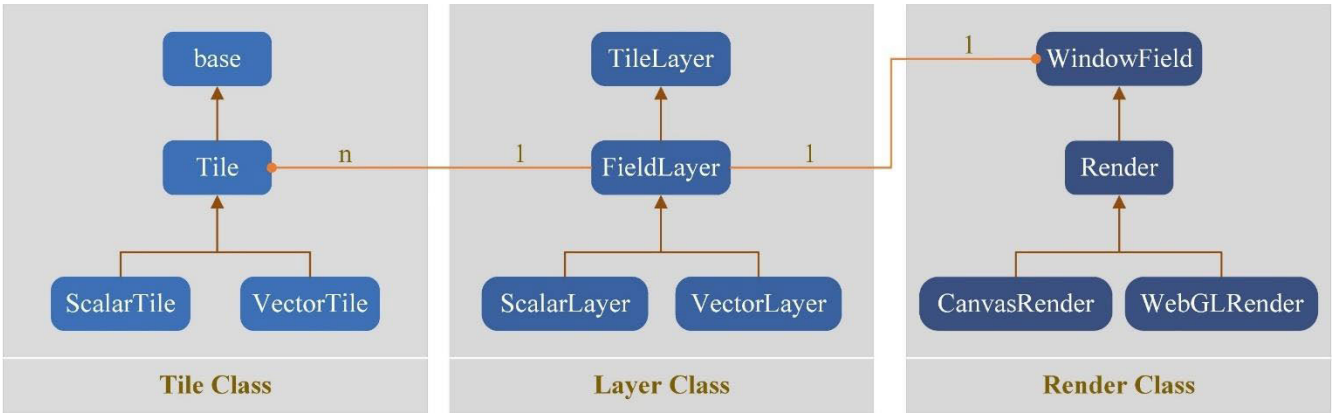


**FIGURE 6.** Class creation and inheritance design diagram.

When it comes to visualization, the adoption of a particle-based dynamic streamline method for rendering vector data offers advantages over glyph and vector line methods in representing the continuity of vector fields and capturing essential features like ocean eddies [35]. This method involves mapping the length, width, and head/tail size of the particle streamlines to the velocity and map viewport zoom level. It provides clear expression of the direction and orientation of the vector data by defining the shapes of the particles based on different speeds and levels. FIGURE 7 illustrates this approach, where the x-axis represents the length of the same particle at different speeds across various map viewport zoom levels, and the y-axis represents its width. This method enhances the visualization of vector data by effectively conveying its characteristics and capturing the dynamic nature of ocean phenomena.

In order to create a dynamic particle animation, it is necessary to store both the current position and the next position for each particle in terms of spatial position. The positions of the particles can be continuously updated based on the frame rate of the frontend rendering. When a particle updates to its next position (posnext), the coordinates of the subsequent position relative to posnext can be calculated. The calculation formula for determining the coordinates of the subsequent position is as follows:

$$pos^{x}_{next} = speed \times u + pos^{x} \tag{8}$$

$$pos^{y}_{next} = speed \times v + pos^{y} \tag{9}$$

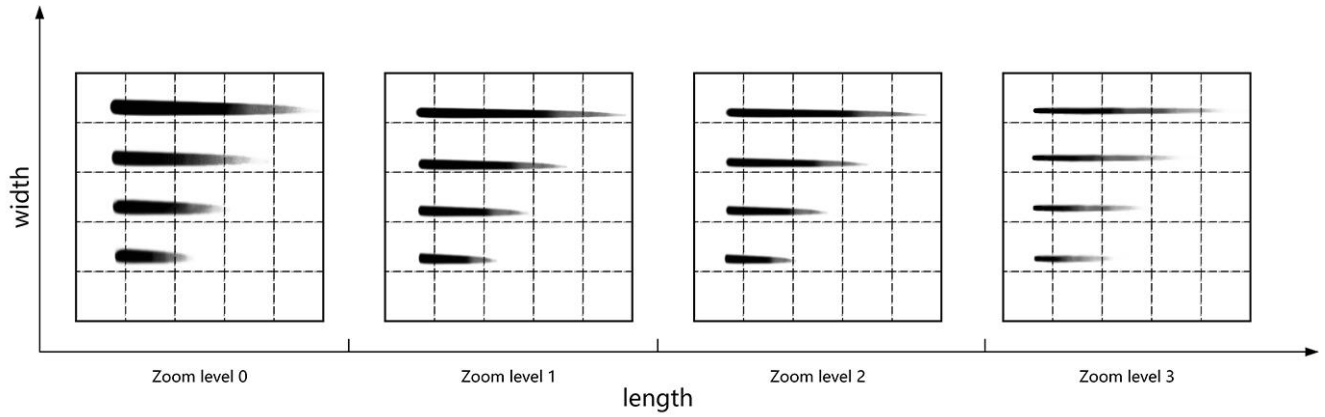


**FIGURE 7.** Description of particle shape.

The formula can be expressed as follows: speed is an adjustable parameter, u represents the eastward ocean current value, v represents the northward ocean current value, posx represents the current x coordinate, and posy represents the current y coordinate.

## III. EXPERIMENT

This paper uses the commonly used ocean data source, the Copernicus Marine Environment Monitoring Service (CMEMS), to address the differences in data sources and workload issues. CMEMS relies on the European Ocean Data Producer Network to establish ocean data products based on the latest scientific knowledge provided by the network. It provides about 160 different products for observation and model outputs, covering ocean physics (temperature, salinity, sea level, ocean currents, waves) and sea ice (concentration, thickness, drift) [36], [37].

Taking CMEMS's ocean current data as an example, we processed the data following the methodology outlined in this paper. As a result, we obtained the global ocean current data at LOD 2 for June 20, 2022, represented as a pyramid model with a resolution of 0.25° × 0.25°, as illustrated in FIGURE 8.

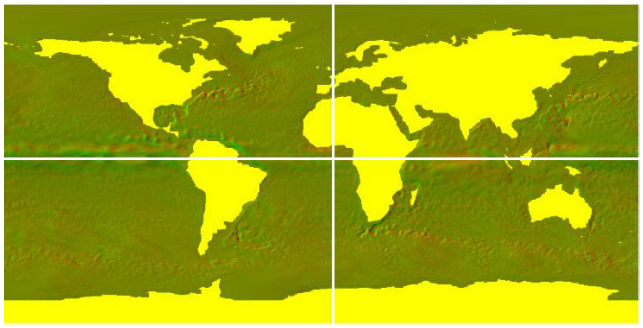

**FIGURE 8.** Global ocean current data at layer 2 of the LOD model.

A series of experiments were conducted to evaluate the effectiveness of the proposed Web-based dynamic visualization and rendering architecture presented in this paper—the experiments aimed to assess the system's performance and visual quality using vector field data. The implementation of the experiments was based on the Leaflet framework. Detailed information about the development and running environment is listed in TABLE 2.

**TABLE 2.** Development environment.

| Hardware | Software |
|---|---|
| CPU: Intel(R) Core(TM) i5-7300HQ | OS: Windows 10 |
| RAM:16GB | Browser: Chrome 11 |
| GPU:NVIDIA GeForce GTX 1050 | Tool: VSCode |

### A. RENDERING EFFECT OF OCEAN DATA

Web-based rendering of ocean data involves several challenges: the browser's viewport is incapable of displaying a large amount of map data and details concurrently; due to the enormous data volume, communication cannot be accomplished at once during the rendering process on the Web side; page reflow, resource loading blocking and other issues that induce page lag. Therefore, this paper aims to enhance the rendering performance of ocean data by employing LOD data processing pyramid model to reconstruct ocean environment dynamic data.

The visualization of the ocean dynamic field is based on the concept of a logical window. The logical window represents a portion of the ocean surface that is displayed on the screen. It acts as a viewport through which the ocean data is rendered and visualized. The logical window is constructed using the WindowField subclass, which inherits from the Field class. The WindowField contains the necessary information to define the boundaries and resolution of the logical window, allowing for precise rendering and visualization of the ocean dynamic data.

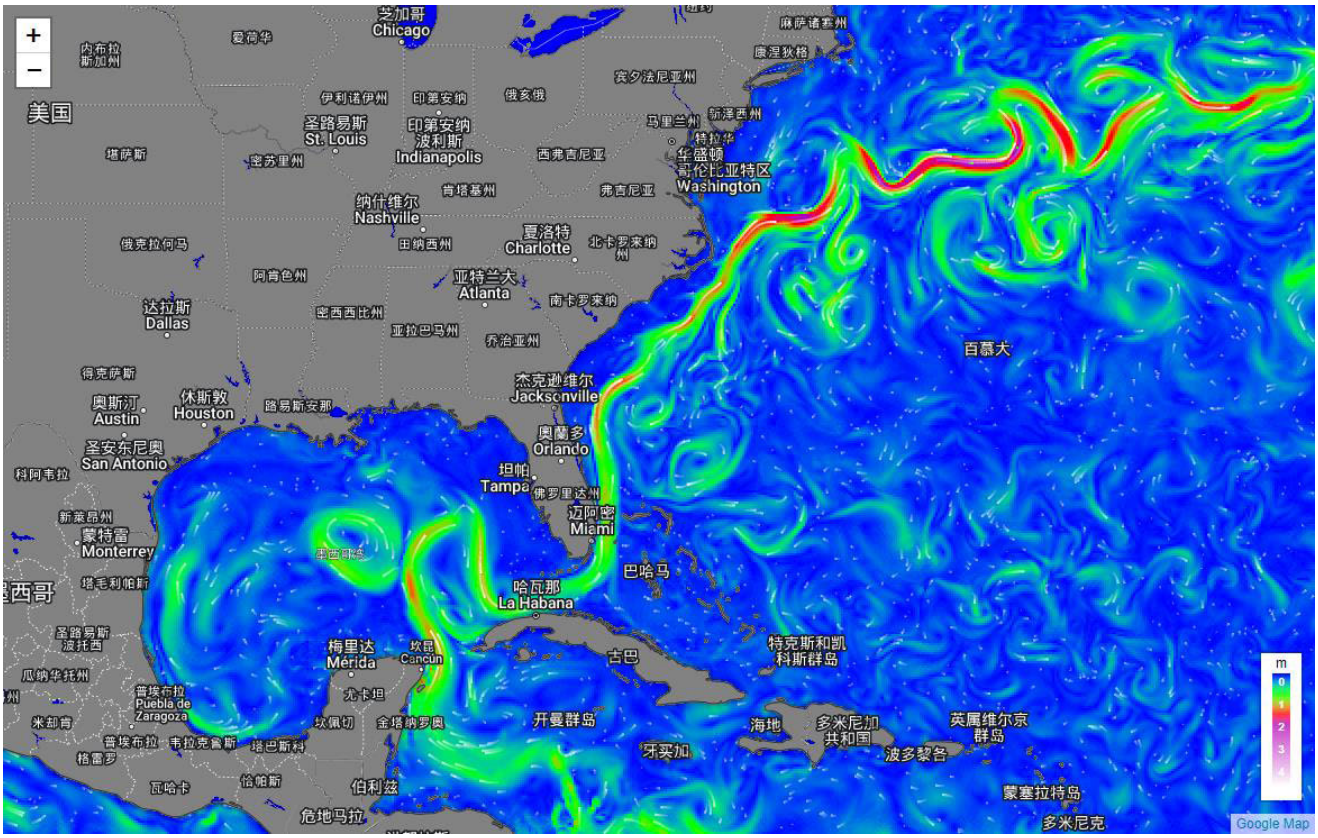


**FIGURE 9.** Dynamic rendering of ocean currents at zoom level 4.

The dynamic rendering of ocean currents is shown in FIGUREs 9, 10, and 11, where FIGURE 9 displays a view at zoom level 4, and FIGURE 10 and 11 show views at zoom level 7. The three images depict several common Atlantic ocean currents features, such as centres, attracting focus, and repelling focus [35]. FIGURE 9 presents more map content, but with lower spatial resolution and less detail in ocean current features, it could be more effective for analyzing the variation patterns of the ocean dynamic field. In contrast, FIGUREs 10 and 11 display less map content, but as the zoom level increases, the spatial resolution also improves, allowing for more evident observation of the details in ocean current features. Additionally, the ocean current data is rendered using a particle-based dynamic streamline method, which is visually appealing and conveys the direction, orientation, speed, and characteristics of ocean currents, effectively overcoming the limitations of traditional visualization methods [5].

### B. RENDERING TIME CONSUMPTION

To fully validate the performance of the proposed rendering framework (WebGL version) in this paper, a Canvas version was developed as a control, and the time-consuming of building the logical window (FT), frames per second (FPS), and the memory consumption during rendering were used as benchmarks to evaluate rendering efficiency. In the iterative rendering process, the particle number, particle disappearance transparency, particle motion speed factor, particle loss rate, and particle size were adjusted to influence rendering efficiency at different viewport levels to achieve testing objectives. The appropriate values of rendering parameters were selected based on experience, as shown in TABLE 3.

In this paper, the vector field of ocean current data was retrieved to perform the test for constructing the logical window, and the FT consumed during this process was used as an indicator of rendering performance. The experiments were conducted at different zoom levels, and each group performed 20 motion tests. The FTs recorded and analyzed in both the WebGL version and the Canvas version.The results of the experiments are presented in FIGURE 12, which illustrates the FT at different zoom levels of the WebGL and Canvas versions.

The average time taken by the WebGL version to construct 2D logical windows was recorded as 54.32 ms, 53.17 ms, 50.06 ms, and 49.51 ms at the respective zoom levels. The average time taken decreases as the map zooms in, indicating improved efficiency. The line fluctuation also decreases, suggesting increased stability in rendering. In contrast, as the map zoomed in, the Canvas version exhibited higher average time values and a lack of clear trends. The average time to construct 2D logical windows was 83.61 ms, 83.76 ms, 78.58 ms, and 92.72 ms at the respective

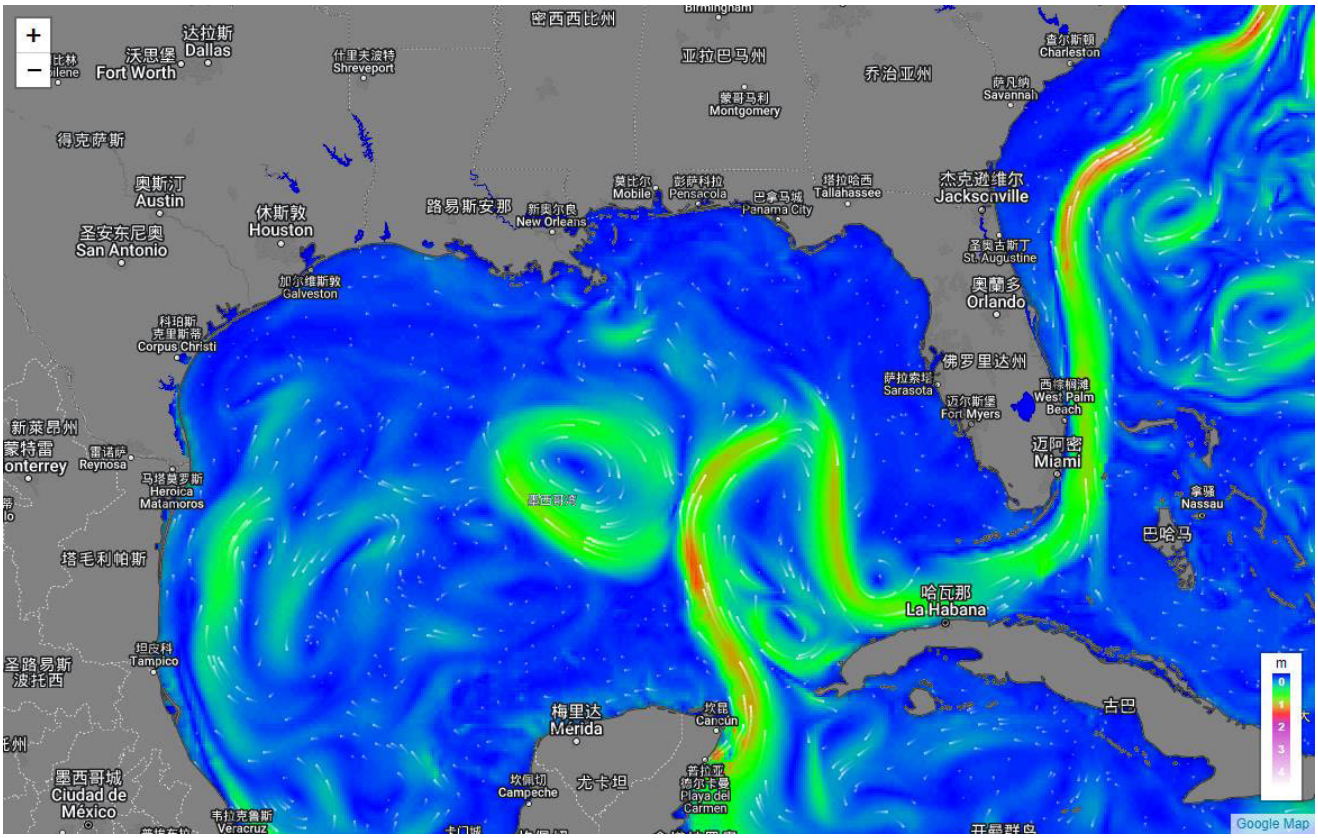


**FIGURE 10.** Ocean currents characteristics 1 at zoom level 7.

zoom levels. The line fluctuation remained high, indicating less stability in rendering.

Comparing the WebGL and Canvas versions, the WebGL version showed a 63.56% increase in performance when rendering dynamic data. By analyzing the line trends, it can be concluded that the WebGL version is more stable across all scaling levels. These results highlight the effectiveness of the proposed architecture in overcoming the limitations of slow rendering for massive vector data using traditional methods.

### C. PERFORMANCE AND STABILITY

FPS refers to the frames per second, which indicates the animation's refresh rate and smoothness. It serves as a performance metric for evaluating the dynamic rendering architecture. The frame rendering statistics were monitored in the Chrome 11 browser console during the experiments. After the page stabilized, the logical window was continuously moved, simulating user interaction and transitioning from a stable state to a construction state. Each group contains ten movement tests, and the FPS values during logical window construction were recorded at different movement frequencies. The movement frequencies were set at 1s, 0.75s, 0.5s, and 0.25s. The results of the experiments are depicted in FIGURE 13, showcasing the FPS values obtained during logical window construction at each movement frequency.

In FIGURE 13, the upper part displays the results of the WebGL-based, while the lower part shows the results of the Canvas-based. The coordinate frame contains three lines representing the mean FPS values, and the vertical distance between the upper and lower lines represents the deviation. The scatter plot represents the FPS value obtained for each group. The FPS of the WebGL-based exhibited a fluctuation range of [52.80, 59.50]. The mean FPS was 57.45fps, and the page zoom operations had minimal impact. However, with increased movement frequency, the FPS values showed a slight downward trend, indicating decreased performance stability. On the other hand, the FPS of the Canvas-based fluctuated between the range of [29.4, 59.50]. The mean FPS was 48.74fps, and the page zoom operation significantly impacted the performance. The Canvas version showed a lower FPS than the WebGL version. The WebGL-based demonstrated better performance and stability, with a higher average FPS and less impact from the map zoom operation. The result indicates that WebGL-based rendering architecture outperforms the Canvas-based regarding dynamic rendering efficiency.

In addition, the experiments revealed that as the movement frequency increased, the FPS and stability decreased. Comparing the WebGL-based with the Canvas-based, the WebGL-based exhibited an overall increase in the mean FPS by 18%, and the mean FPS in all zoom levels exceeded 55 fps, indicating good performance. In contrast, the Canvas-based

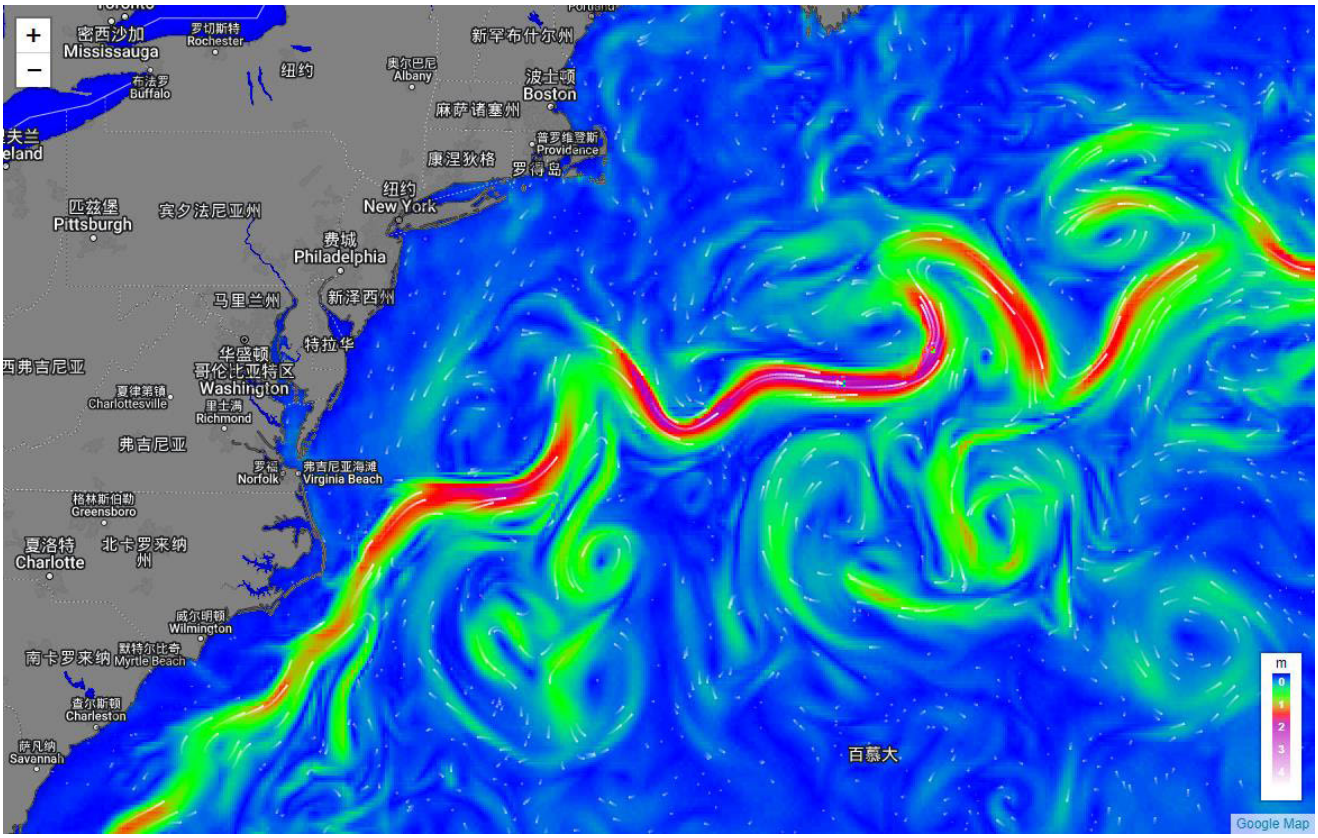


**FIGURE 11.** Ocean currents characteristics 2 at zoom level 7.

**TABLE 3.** Scale levels and rendering parameters.

| zoomLevel | numParticles | fadeOpacity | speedFactor | dropRate | dropRateBump | pointSize |
|---|---|---|---|---|---|---|
| 5 | 7056 | 0.995 | 7 | 0.003 | 0.01 | 2 |
| 6 | 7056 | 0.96 | 7 | 0.003 | 0.01 | 2 |
| 7 | 7056 | 0.96 | 6 | 0.003 | 0.01 | 2 |
| 8 | 7056 | 0.96 | 6 | 0.003 | 0.01 | 2 |

showed a difference of 30.10 fps between the maximum and minimum FPS, whereas the WebGL-based had a difference of only 6.70 fps. This significant reduction in this FPS difference indicates a relative increase of 77.74% in overall stability for the WebGL-based. These results highlight the proposed architecture's effectiveness, leveraging the GPU's parallel rendering capabilities and employing a frustum culling strategy. The system designed using this architecture achieves smooth animations and fast dynamic rendering of massive ocean data on the web.

### D. RENDERING DATA MEMORY CONSUMPTION

The memory consumption comparison compares the browser's memory usage when using the entire ocean data as the request source to when using the LOD-based image as the request source. Based on performance records in the browser console, we analyzed memory usage during the logical window construction of web pages at different resolutions. The tested image resolutions were $0.5° \times 0.5°$, $0.25° \times 0.25°$, and $0.1° \times 0.1°$. FIGURE 14 shows the position of the test viewport.

When we directly render the global ocean dynamics data as the data source, the memory usage of the three image resolutions during webpage loading is 7.02MB, 24.04MB, and 256.42MB, respectively. Since JavaScript has an automatic garbage collection mechanism, local variables are marked for collection when they are no longer needed, freeing up memory when the function ends. This causes the memory usage to gradually decrease and stabilize as the dynamic effects of the webpage are completed. Based on this mechanism, we can infer that the memory usage during the rendering process will reach a peak, with peak values of 8.21MB, 35.31MB, and 331.23MB for image resolutions of $0.5° \times 0.5°$, $0.25° \times 0.25°$, and $0.1° \times 0.1°$, respectively. The memory usage then

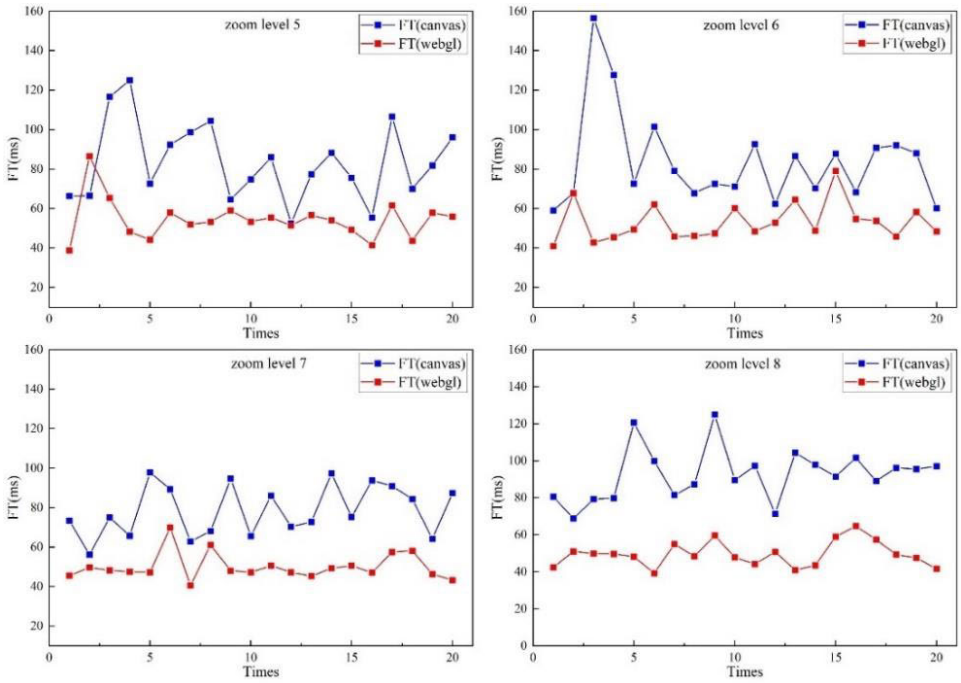


**FIGURE 12.** Time-consuming for building logical window.

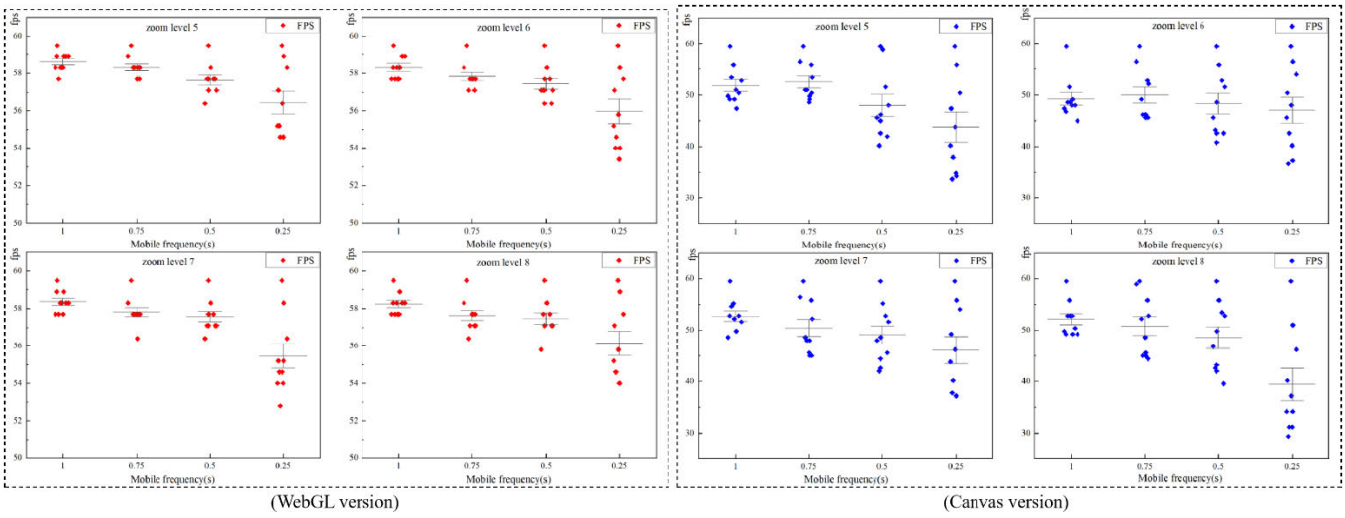


**FIGURE 13.** FPS value during logical window construction.

decreases and stabilizes in the range of [6.38, 6.71], [29.78, 30.92], and [306.32, 307.83]. If the memory usage continues to increase during the rendering process, it indicates a memory leak. The results show that rendering high-resolution image data not only causes webpage lag and even crashes due to excessive consumption of browser memory resources, but also leads to loss of accuracy if low-resolution image data is used, resulting in missing details.

By switching the data source to the LOD model constructed based on the method proposed in this paper and testing it with the same viewport, FIGUREs 15a, 15b, and 15c show the memory consumption during the rendering process. In contrast, the memory consumption for an image resolution of 0.5° × 0.5° remained consistent because that level was not sliced. However, at image resolutions of 0.25° × 0.25° and 0.1° × 0.1°, the peak memory consumption decreased by 17.30MB and 240.92MB, respectively, which is approximately half and one-fourth of the memory consumption of traditional methods (theoretically, the performance can reach a maximum of 1/4 and 1/16, respectively, where the numerator represents the number of tiles pulled and the denominator represents the total number of tiles), as shown

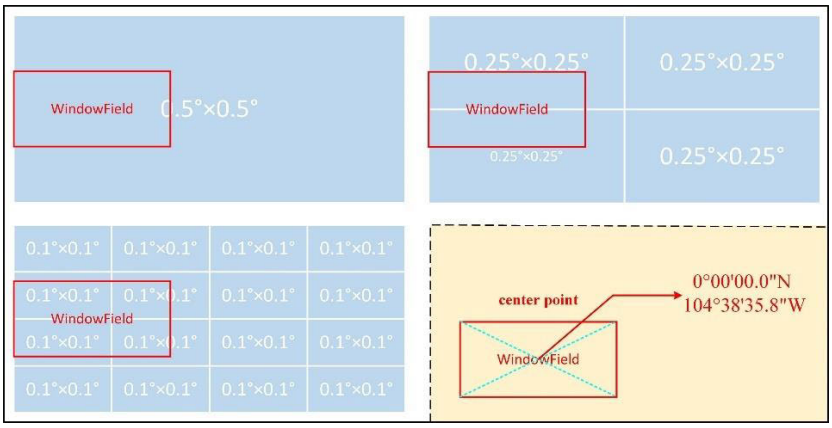


**FIGURE 14.** Illustration of the test window.

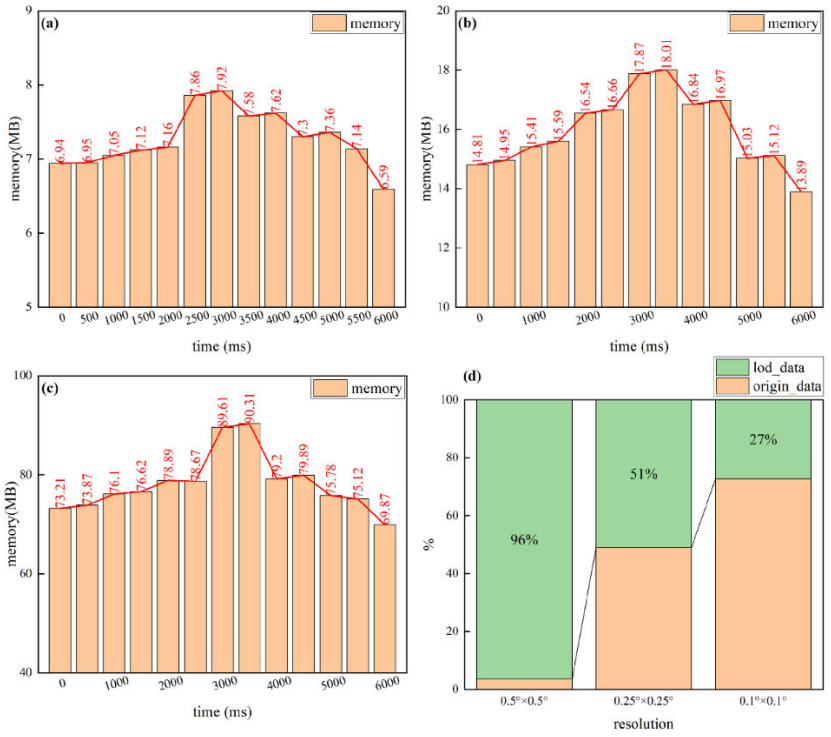


**FIGURE 15.** Loading performance renderings.

in FIGURE 15d. Therefore, it can be concluded that the method proposed in this paper significantly reduces the size of transmitted data in a limited viewport, effectively improving the realism and rendering speed of the scene.

## IV. CONCLUSION

To reduce the size of ocean data transmitted during visualization in a limited viewport and achieve effective ocean data visualization, this paper proposes a generic web-based rendering architecture for ocean dynamic factors. This architecture is utilized to realize dynamic rendering of particle movement trajectories and global ocean current data mapping, and the rendering performance is analyzed based on FT, FPS, and the memory consumption of data transmission. Compared with traditional methods, the proposed approach can significantly reduce the amount of ocean data transmitted during the rendering process, effectively addressing the slow rendering and poor visual effects of massive ocean data visualization in some systems, and overcoming the limitations of Canvas to achieve fast dynamic rendering of massive ocean data on the Web. However, there are still some limitations in this research. Although the standardized data satisfies the timeliness and correctness requirements of ocean environmental factor data system architecture, which solves

the problems of multi-source and heterogeneous redundant ocean data, single data format, and inconsistent data spatial resolution to some extent, there is still room for improvement in the resampling algorithm used in the process. Moreover, while the efficiency of drawing and rendering is greatly improved by directly obtaining relevant data through metadata when calling slices, the conversion of data format back and forth during this process causes some loss of accuracy in the processed data. Therefore, further research is needed to explore these issues in more depth in future studies.

## REFERENCES

[1] D. Olbers, J. Willebrand, and C. Eden, *Ocean Dynamics*. Springer, 2012.
[2] W. H. Ali, M. H. Mirhi, A. Gupta, C. S. Kulkarni, C. Foucart, M. M. Doshi, D. N. Subramani, C. Mirabito, P. J. Haley, and P. F. J. Lermusiaux, "SeaVizKit: Interactive maps for ocean visualization," in *Proc. OCEANS MTS/IEEE*, Seattle, WA, USA, Oct. 2019, pp. 1–10.
[3] Y. Wang, F. Li, B. Zhang, and X. Li, "Development of a component-based interactive visualization system for the analysis of ocean data," *Big Earth Data*, vol. 6, no. 2, pp. 219–235, Apr. 2022.
[4] M. Amani, S. Mehravar, R. M. Asiyabi, A. Moghimi, A. Ghorbanian, S. A. Ahmadi, H. Ebrahimy, S. H. A. Moghaddam, A. Naboureh, B. Ranjgar, F. Mohseni, M. E. Nazari, S. Mahdavi, S. M. Mirmazloumi, S. Ojaghi, and S. Jin, "Ocean remote sensing techniques and applications: A review (Part II)," *Water*, vol. 14, no. 21, p. 3401, Oct. 2022.
[5] Q. Shi, B. Ai, Y. Wen, W. Feng, C. Yang, and H. Zhu, "Particle system-based multi-hierarchy dynamic visualization of ocean current data," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 10, p. 667, Oct. 2021.
[6] S. Fulton and J. Fulton, *HTML5 Canvas: Native Interactivity and Animation for the web*. Sebastopol, CA, USA: O'Reilly Media, 2013.
[7] B. Danchilla, *Beginning WebGL for HTML5*. Berkeley, CA, USA: Apress, 2012.
[8] L. Liu, D. Silver, and K. Bemis, "Visualizing three-dimensional ocean eddies in web browsers," *IEEE Access*, vol. 7, pp. 44734–44747, 2019.
[9] J. A. James, T. Moh, and C. A. Edwards, "Web-based visualization of marine environmental data: Performance analysis of a MatPlotLib implementation," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, Oct. 2016, pp. 288–293.
[10] M. De-Lie, G. Peng-Fei, and W. Mi, "On realization of visualization system for global ocean simulation," in *Proc. Int. Conf. Audio, Lang. Image Process.*, Nov. 2010, pp. 1446–1450.
[11] C. A. Gutwin, M. Lippold, and T. N. Graham, "Real-time groupware in the browser: Testing the performance of web-based networking," in *Proc. ACM Conf. Comput. Supported Cooperat. Work (CSCW)*, Mar. 2011, pp. 167–176.
[12] M. E. Portman, "Visualization for planning and management of oceans and coasts," *Ocean Coastal Manag.*, vol. 98, pp. 176–185, Sep. 2014.
[13] T. Rossby, "Visualizing and quantifying oceanic motion," *Annu. Rev. Mar. Sci.*, vol. 8, no. 1, pp. 35–57, Jan. 2016.
[14] S. Martin, *An Introduction to Ocean Remote Sensing*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[15] K. Sayood, *Introduction to Data Compression*. Newnes, 2012.
[16] M. S. Mahmud, J. Z. Huang, S. Salloum, T. Z. Emara, and K. Sadatdiynov, "A survey of data partitioning and sampling methods to support big data analysis," *Big Data Mining Anal.*, vol. 3, no. 2, pp. 85–101, Jun. 2020.
[17] S.-X. Nan, X.-F. Feng, Y.-F. Wu, and H. Zhang, "Remote sensing image compression and encryption based on block compressive sensing and 2D-LCCCM," *Nonlinear Dyn.*, vol. 108, no. 3, pp. 2705–2729, May 2022.
[18] C. Hu, Y. Pu, F. Yang, R. Zhao, A. Alrawais, and T. Xiang, "Secure and efficient data collection and storage of IoT in smart ocean," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9980–9994, Oct. 2020.
[19] C. Shi, J. Zhang, and Y. Zhang, "A novel vision-based adaptive scanning for the compression of remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1336–1348, Mar. 2016.
[20] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner, *Level of Detail for 3D Graphics*. San Mateo, CA, USA: Morgan Kaufmann, 2003.
[21] M. Zárate, G. Braun, M. Lewis, and P. Fillottrani, "Observational/hydrographic data of the South Atlantic Ocean published as LOD," *Semantic Web*, vol. 13, no. 2, pp. 133–145, 2022.
[22] K. Xu and X. Xie, "Real time rendering technology of ocean data based on improved LOD model," in *Proc. Artif. Intell. Complex Syst. Conf. (AICS)*, Aug. 2020, pp. 68–72.
[23] G. K. Vallis, *Essentials of Atmospheric and Oceanic Dynamics*. Cambridge, U.K.: Cambridge Univ. Press, 2019.
[24] J. J. Van Wijk, "Spot noise texture synthesis for data visualization," in *Proc. 18th Annu. Conf. Comput. Graph. Interact. Techn.* New York, NY, USA: ACM, 1991, pp. 309–318.
[25] B. Cabral and L. C. Leedom, "Imaging vector fields using line integral convolution," in *Proc. 20th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, Sep. 1993, pp. 263–270.
[26] H. Ma, P. Wang, Y. Wen, and C. Yang, "Research on key technologies of 3D visualization of marine environmental field," *J. Phys., Conf. Ser.*, vol. 2006, no. 1, Aug. 2021, Art. no. 012042.
[27] T. Lv, J. Fu, and B. Li, "Design and application of multi-dimensional visualization system for large-scale ocean data," *ISPRS Int. J. Geo-Inf.*, vol. 11, no. 9, p. 491, Sep. 2022.
[28] R. Qin, B. Feng, Z. Xu, Y. Zhou, L. Liu, and Y. Li, "Web-based 3D visualization framework for time-varying and large-volume oceanic forecasting data using open-source technologies," *Environ. Model. Softw.*, vol. 135, Jan. 2021, Art. no. 104908.
[29] B. Lawson and R. Sharp, *Introducing HTML5*. Indianapolis, IN, USA: New Riders, 2013.
[30] M. Bajammal and A. Mesbah, "Web canvas testing through visual inference," in *Proc. IEEE 11th Int. Conf. Softw. Test., Verification Validation (ICST)*, Apr. 2018, pp. 193–203.
[31] M. Aristizabal, J. Congote, A. Segura, A. Moreno, H. Arregui, and O. Ruiz, "Hardware accelerated web visualization of vector fields—Case study in oceanic currents," in *Proc. Int. Conf. Comput. Vis. Theory Appl. (IVAPP)*. 2012, pp. 759–763.
[32] L. Andersson and O. Lenschow, "Optimizing a GPU-Accelerated Particle System for WebGL 1.0 using extension," Dept. Comput. Sci. Eng., Chalmers Univ. Technol., Univ. Gothenburg, 2021.
[33] V. Klemas, "Remote sensing of coastal and ocean currents: An overview," *J. Coastal Res.*, vol. 28, no. 3, pp. 576–586, May 2012.
[34] A. Yao, L. Wang, J. Li, X. Xia, X. Jin, and N. Jing, "2D/3D visualization of large-scale wind field based on WebGL," in *Proc. Int. Conf. Aviation Saf. Inf. Technol.*, Oct. 2020, pp. 269–274.
[35] K. Wu, Z. Liu, S. Zhang, and R. J. Moorhead, "Topology-aware evenly spaced streamline placement," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 5, pp. 791–801, Sep. 2010.
[36] P. Y. Le Traon et al., "From observation to information and users: The Copernicus Marine Service perspective," *Frontiers Mar. Sci.*, vol. 6, p. 234, May 2019.
[37] H. Xi, S. N. Losa, A. Mangin, M. A. Soppa, P. Garnesson, J. Demaria, Y. Liu, O. H. F. d'Andon, and A. Bracher, "Global retrieval of phytoplankton functional types based on empirical orthogonal functions using CMEMS GlobColour merged products and further extension to OLCI data," *Remote Sens. Environ.*, vol. 240, Apr. 2020, Art. no. 111704.

**DONGLIN FAN** received the B.S. degree in geographical information systems (GIS) from Guizhou University, Guiyang, China, in 2011, and the M.S. degree in cartography and geographic information engineering from the China University of Geosciences, Wuhan, China, in 2014. He is currently pursuing the Ph.D. degree in geological resources and geological engineering with the Guilin University of Technology, Guilin, China.

From 2016 to 2023, he was a Lecturer of GIS with the Guilin University of Technology. His current research interests include the development and visualization of ocean data fields and retrieval of ocean color using artificial intelligence.

**TIANLONG LIANG** was born in Liuzhou, Guangxi, China, in 2000. He received the bachelor's degree in geographic information science from the Guilin University of Technology, in 2022, where he is currently pursuing the master's degree in resources and environment. His current research interests include ocean optical remote sensing, ocean chlorophyll inversion, and related areas.

**HONGCHANG HE** received the Ph.D. degree in remote sensing and geographical information systems (GIS) from the University of Fribourg, Switzerland, in 2000. He was a Postdoctoral Researcher with the Canada Centre for Remote Sensing (CCRS), Canada, and later became a Professor with the Hangzhou Dianzi University, China. Currently, he is a Professor with the Guilin University of Technology, China, where he develops algorithms for retrieval of chlorophyll, sea surface temperature, and salty in coastal waters using artificial intelligence. His research interest includes fishing ground prediction.

**MENGYUAN GUO** was born in Mianyang, Sichuan, China, in 1997. She received the bachelor's degree in art from the Sichuan Film and Television College, in 2019. She is currently pursuing the master's degree in digital media art with the Guilin University of Technology. Her current research interest includes emotional design and related fields.

**MENGHUI WANG** was born in Cangzhou, Hebei, China, in 1996. She received the bachelor's degree in surveying and mapping engineering from the City College, Hebei University of Technology, in 2019. She is currently pursuing the master's degree in surveying and mapping science and technology with the Guilin University of Technology. Her current research interests include ocean optical remote sensing, atmospheric correction for 2-class water, and related areas.