

An interactive visualization tool for the exploration and analysis of multivariate ocean data

Preetha K. G.¹, Saritha S.¹, Jishnu Jeevan², Chinnu Sachidanandan¹, P. A. Maheswaran³

¹Department of Computer Science and Engineering, Rajagiri School of Engineering and Technology, Kochi, India

²Euro-Mediterranean Center on Climate Change (CMCC), Lecce, Italy

³DRDO, Naval Physical Oceanographic Laboratory, Kochi, India

Article Info

Article history:

Received Jul 17, 2024

Revised Aug 2, 2024

Accepted Aug 5, 2024

Keywords:

Multivariate

Ocean data

Overlay

Rendering

Visualization

ABSTRACT

Ocean data exhibits great heterogeneity from variances in measuring methods, formats, and quality, making it extremely complicated and diverse due to a variety of data kinds, sources, and study elements. A few examples of data sources are satellites, buoys, ships, self-driving cars, and distant systems. The processing of data is made more challenging by the significant regional and temporal variations in oceanic characteristics including temperature, salinity, and currents. This work presents an interactive tool for multivariate ocean parameter visualisation, specifically overlays, based on Python. In ocean data visualisation, overlays are extra visual layers or data points that are layered to improve comprehension over a basic map. Based on the available data and the visualisation goals, these overlays are chosen and blended. Users can customise overlays with this tool, which also supports formatting, 2D and 3D visualisation, and data preparation. In order to reduce artefacts, it uses kriging interpolation for 3D visualisation and a modified version of the ray casting algorithm for representing octree data. By integrating overlays like as bathymetry, currents, temperature, and marine life, users can produce visually appealing and comprehensive depictions of ocean data. This method provides a thorough grasp of intricate marine processes by making it easier to see patterns, trends, and abnormalities in the data.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Preetha K. G.

Department of Computer Science and Engineering, Rajagiri School of Engineering and Technology

Rajagiri Valley P O, Kochi, Kerala, India

Email: preetha_kg@rajagiritech.edu.in

1. INTRODUCTION

The spatial and temporal visualization of ocean features is critical for many oceanography researchers. They use in situ data and programming models to extract insights and vital information about the world ocean. Visualization of ocean data/parameters conveys the inherent patterns, trends, and anomalies in the ocean. Oceanographers often prefer using maps with colorful markers for the visualization and analysis of ocean data. However, mapping oceanographic measurements to be effective is challenging because (i) there are diverse array of parameters, such as temperature, salinity, and oxygen content, at a wide range of depth and time scales and (ii) the interaction of the multiple parameters needs to be captured in the visualization [1], [2]. Researchers utilize approaches that allow them to display several ocean metrics on one plot or chart at the same time to make sense of these complicated observations [3], [4]. This can help identify patterns and interactions between variables that would not be obvious if each variable is evaluated separately.

These approaches can visualize temperature, salinity, currents, nutrients, and chlorophyll concentration, to name a few oceanographic factors. By visualizing many variables at the same time, it is possible to acquire a more comprehensive understanding of ocean dynamics and how different parameters interact with one another. By merging these multiple visualizations, researchers can acquire a deeper understanding of the intricate relationships between different ocean characteristics and how they affect marine ecosystems and climate trends [5], [6].

The main goal of the study is to create visual overlays of multivariate ocean data. This assists oceanographers in better understanding ocean parameters like temperature, salinity, density, ocean currents, and their relationships. The focus of this multivariate analysis is on how these parameters are distributed and how they influence each other. This paper presents interactive visualizations of multivariate analysis using Python data visualization libraries. Users can benefit from this by better understanding complex data sets, identifying patterns and trends, and making data-driven decisions. The paper also has a contributing component in 3D visualization [7], [8] in which the study makes use of the octree-based method for data representation and henceforth the ray casting algorithm is modified to make use of kriging interpolation to achieve the visualization with fewer artifacts.

The remainder of the paper is structured as follows: section 2 provides a detailed design description of the proposed methodology. Section 3 demonstrates the visualization results and their interpretations. Conclusions and future work are drawn in section 4.

2. METHOD

This section provides an overview of the framework, describes the data types supported by the same, and the architecture of the framework, followed by the algorithms used in this work. The architecture of the proposed framework is shown in Figure 1. The entire architecture is divided into four layers (i) data layer, (ii) data management layer, (iii) data rendering layer, and (iv) visualization layer. The work starts from the initial data layer, which stores the heterogeneous, multivariate ocean data. The data management layer handles the pre-processing and format compatibility of the diverse dataset. The data rendering layer focuses on the implementation of the visualization algorithm in either of the two modes, namely (i) conventional CPU mode of implementation through which the basic plots like location maps, profile plots, scatter plots, colormaps, vector plots are implemented based on the Python script language in an interactive mode; and (ii) GPU mode of implementation where the algorithms are modified to decipher the data patterns more accurately. Finally, the visualization layer provides the mapping in univariate as well as multivariate modes. The improvement in this layer is achieved in this framework by modifying the ray casting algorithm and using kriging interpolation to reduce the artifacts during the visualization in 3D mode.

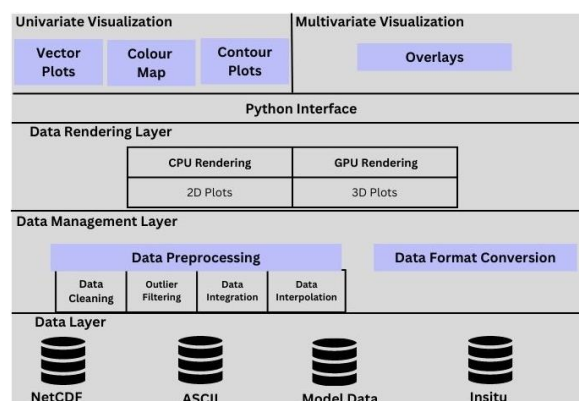


Figure 1. Architecture diagram of the proposed framework

2.1. Data format

The framework is designed to work with all commonly used data formats. A brief overview of the data sets used in the present study is given below. The data sets mentioned are accessible from www.cen.uni-hamburg.de and www.copernicus.eu.in. The ocean reanalysis/analysis system4 (ORAS4) employs version 3.0 of the Nucleus for European Modelling of the Ocean (NEMO) [9] in the ORCA1 horizontal discretization, with a 1°-horizontal resolution in the extra-tropics and a refined meridional resolution in the

tropics. ORAS4 has 42 vertical levels, with the first 18 corresponding to the upper 200 m, and it assimilates temperature, salinity, and sea surface height anomaly data, but not velocity observations. World Ocean Atlas (WOA) data [10] is also used in the study. The model is forced by atmospheric-derived daily surface fluxes from ERA40 [11], ERA-Interim [12], and the ECMWF [13] operational archive for different periods from 1957 to 2015. The study utilized $1^\circ \times 1^\circ$ monthly gridded temperature and salinity data from 1979 to 1990 and ERA-Interim data for 10-meter wind components to analyze wind characteristics.

2.2. Data management layer

The data management layer manages data pre-processing, validation, and conversion into the appropriate format. Date pre-processing: this layer plays an important role in the architecture. Pre-processing helps to ensure that ocean data is accurate, consistent, and suitable for analysis. Data cleaning is used to remove or correct any errors or inconsistencies in raw data. Errors can occur as a result of instrument failure, human error, or other factors. Filtering outliers, correcting missing or invalid data, and checking for duplicate data are all examples of data-cleaning methods. Data integration is the process of combining data from various sources or sensors. This can be accomplished by combining data with similar properties or by aligning data based on time or location. Interpolation is the process of estimating missing data points from nearby data points. When there are gaps in the data or missing values, interpolation is frequently required.

Format conversion: different formats (and also different coordinate systems) pose a challenge in data visualization. The common data formats which are supported by the proposed tool include NetCDF, ASCII, Binary and HDF. NetCDF is the most widely used data format. NetCDF is machine-independent, and is self-descriptive in nature. Because it is simple to access subsets of a dataset, NetCDF data are easily scalable. To process NetCDF files quickly and effectively, it is usually desirable to create a subset of data from the enormous file. The tool is capable of converting the unstructured measured data to structured data.

2.3. Data rendering engine

In this layer, the oceanographic data is rendered to generate 2D and 3D images or animation from a 3D scene. The rendering pipeline has a sequence of stages like geometric processing, rasterization, shading, texturing and blending to transform the oceanographic data to a 2D/3D image or animation. The 2D images are rendered with the support of vector field extraction from data followed by line rendering and surface rendering. When 3D images are rendered, volume rendering and is surface rendering schemes are adopted. Rendering is a computationally intensive process and, in this work, the same is implemented using the central processing unit (CPU) or the graphics processing unit (GPU).

In the CPU implementation part, the gridded data management scheme is adopted. For instance, in oceanographic data analysis, temperature readings are gathered at different depths and locations using sensors or satellites. These data points can be interpolated and rendered using finite difference methods [14] or ray casting algorithm [15]. The resulting visualizations can be used to better understand the thermal structure of the ocean and help to identify regions of heat exchange between the ocean and the atmosphere. In CPU rendering, the computer performs calculations in a serial fashion, executing one instruction at a time. This is the sole reason behind its accuracy as well as the slow rendering speed.

In the GPU implementation part, the gridded data is managed using an octree data structure. Octree [16] is a data structure that can be used to represent volumetric data such as ocean temperature, salinity and currents. Octrees are tree-based data structures that recursively divide a 3D space into eight equal-sized subspaces, or octants. The octree structure allows efficient compression of data, by storing only the most significant data values in the parent nodes, while the more detailed data values are stored in the child nodes. Each octree node can be defined by its coordinates in 3D space, represented as a vector (x, y, z) . The coordinates of the root node are usually defined as $(0, 0, 0)$, and the coordinates of each child node can be calculated using the (1):

$$child_nodes = parent_nodes + size/2 * (2 * index_child - 1) \quad (1)$$

where $index_child$ is an integer between 1 and 8 representing the index of the child node, and $size$ is the length of the side of the parent node. The representation of an octree in tree form is given in Figure 2.

To construct an octree, the 3D space is recursively subdivided into smaller cubes until the desired level of detail is reached. The subdivision algorithm typically works by checking whether each cube contains any data points, and if so, dividing it into eight smaller cubes. The subdivision can be performed using the following steps:

- i) Check if the current cube contains any data points.
- ii) If the cube is too small or contains no data points, mark it as a leaf node and stop
- iii) Otherwise, divide the cube into eight smaller cubes.
- iv) Recursively apply the subdivision algorithm to each of the smaller cubes.

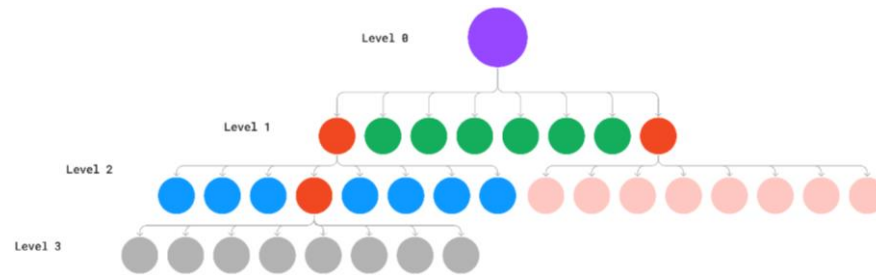


Figure 2. Mesh representation of an octree

For example, to represent sea surface temperature (SST) using an octree, we can divide the ocean surface into octants and assign a temperature value to each octant. The octree can be built recursively by dividing each octant into eight smaller octants until a desired level of detail is achieved. One approach to representing SST using an octree is to use a multi-resolution representation, where each level of the octree represents a different level of detail. At the highest level of the octree, each octant represents a large area of the ocean surface with a low level of detail, and at lower levels of the octree, each octant represents a smaller area with a higher level of detail. To store the temperature values in the octree, a hash table can be used to store the value associated with each octant. Octrees are thus useful for level-of-detail rendering, where different levels of detail can be displayed based on the distance from the observer. The level of detail is determined by the depth of the octree. This can be particularly useful for interactive visualization of large oceanographic datasets. The octree traversal traces the ray and its collision with the scene. The rendering speed in GPU implementation is further improved using sorted sibling traversal [17] by the ray as described in the following steps.

- 1) Start at the root
- 2) If the ray intersects the root bounds, check the children
 - If the child is a non-leaf, find the nearest child closest to the ray origin, by comparing with basis x , y , and z planes
 - i. To determine which of the parent node's three axis planes the ray has collided with, look for the next closest sibling.
 - ii. The entry point of the beam into the next closest sibling will be revealed by the closest plane that is struck.
 - iii. If no plane is struck, then the ray is headed out of the node and will not hit any other children of the current parent.
- 3) Else, halt the entire process.

2.4. Python interface

Python has a wide variety of packages that can be used to visualize ocean parameters. Depending on the particular requirements, one or more of these libraries may be useful. The key visualization tools in the suggested framework for data analysis are Matplotlib and Plotly. A well-known Python data visualisation toolkit called Matplotlib provides several tools for making plots, charts, and graphs. It may show a variety of ocean parameters, including temperature, salinity, sea level, and currents. A Python module called Plotly enables us to make interactive, web-based visualizations. Making interactive maps, scatter plots, line plots, and other visualizations is possible with it.

Python Qt is used as the graphical user interface (GUI) for interactive visualization. Creating high-quality, native-looking apps with a consistent user interface and behaviour across platforms is straightforward with Qt's wide range of tools and modules. Python programmers can use the PyQt or PySide libraries to access the Qt framework. PySide is an alternate set of bindings that is only partially compatible with PyQt, which is a set of Python bindings for the Qt framework. The Qt framework includes essential elements such as Qt Core, Qt GUI, and Qt Widgets. Qt Core provides non-GUI features like event management, data storage, networking, and threading. Qt GUI offers tools for building graphical user interfaces, including widgets, layouts, and event handling, while Qt widgets contains reusable UI elements like menus, buttons, and text editors, along with layout managers for organizing widgets.

2.5. Visualization layer

The objective of visualization layer is to provide univariate and multivariate visualization of the oceanographic dataset. The univariate visualization techniques are available in different methods like (a) colormaps, (b) vector plots, (c) contour plots, (d) stick plot, (e) streamline plots, and (f) rose plots.

The matplotlib library in Python includes the ‘cmocean’ module, which contains colormaps tailored to specific oceanographic variables, such as thermal, haline, ice, oxy, and algae. For example, the ‘algae’ colormap in shades of green represents chlorophyll, while the ‘thermal’ colormap transitions from dark blue to yellow for temperature values. These colormaps are preferred over conventional ones like ‘rainbow’ and ‘jet’ for better cognition in oceanographic data visualization. For visualizing oceanographic data with magnitude and direction, such as wind and current, vector plots, stick plots, and streamline plots are used. The ‘quiver’ function in Python helps plot vector and stick representations, and the ‘streamplot’ function provides streamline plots illustrating the velocity field’s density and magnitude. Additionally, the ‘bar_polar’ function in matplotlib can plot rose charts to visualize wind speed and direction distributions at a given location.

While rendering contour plots in 2D and 3D visualization, the rendering speed suffers as there are not sufficient points available as samples in the spatial area considered. Thus, the oceanographic data is characterized of non-uniform section interval (eg-depths at non-uniform intervals in Argo float). Hence, to speed up the ray-casting algorithm for volume rendering, appropriate spatial interpolation technique is to be adopted. The spatial interpolation method chosen is kriging [18]. Kriging uses a linear combination of the observed values in the adjacent nodes of the octree, with weights determined by their spatial correlation, given as (2).

$$Z(x) = \sum \lambda_i Z(x_i) \quad (2)$$

where $Z(x)$ is the estimated value of the variable at the target location x , λ_i is the weight assigned to the i -th observation at location x_i , and the summation is taken over all the observations within the specified neighborhood around the target location. The weights are determined by a variogram that describes the spatial correlation between the variable at different locations. The selection of the variogram to be used in kriging for a particular variable, say, SST data is influenced by a number of variables, including the spatial scale of the SST variability, the sampling density of the data, and the underlying physical processes that produce the SST variability. In this implementation, anisotropic variogram is chosen, as the SST data varies in different directions [19], [20]. The anisotropic variogram is expressed as (3).

$$\gamma(h) = 1/2 * E[(Z(x) - Z(x+h))^2] \quad (3)$$

where h is the lag distance, $Z(x)$ is the variable of interest at location x , and E [21] denotes the expected value operator. The lag distance, h , can be replaced by a lag vector $h=(h_1, h_2, h_3)$ and the covariance between two locations is a function of the lag vector and the orientation of the anisotropy. A rotation matrix R can be used to transform the lag vector, h , into a new coordinate system as (4).

$$\gamma(h) = 1/2 * E[(Z(x) - Z(x+Rh))^2] \quad (4)$$

where Rh is the lag vector in the new coordinate system.

Thus, with kriging, the sampling step of ray casting algorithm results in less artifacts. The visualization layer also provides multivariate visualization of the oceanographic data, which helps to understand the causal relationship between the different parameters under consideration. The multivariate visualization [22], [23] provides the simultaneous visualization of multiple variables. For example, the foundation layer of a multivariate map might be the SST, with additional layers showing bathymetry, ocean currents, and chlorophyll concentration. This would give a complete picture of where h is the lag distance, $Z(x)$ is the variable of interest at location x , and the ecological and physical characteristics of the ocean in a particular area. An extensive perspective of the vertical structure of the ocean can be obtained by utilising a vertical profile map or a contour plot, where the various characteristics are stacked on top of one another [24], [25].

3. RESULTS AND DISCUSSION

Python is used to implement the system since it has an adequate data processing package. Python modules like Matplotlib, Basemap, and Numpy are essential components of the program. Users have the option to select from a variety of plot types for a single variable study, including contour plots, vector plots, stream line plots, and color maps. Plots can be customized in a variety of ways, including location and color selections. The framework offers alternatives for multivariate visualization by overlaying the many fundamental plots. The initial launch page of the framework which provides the data converter is given in Figure 3. The interface of the system provides options to import data into the framework and provides an

interface for data management. The framework is tested on the system with specification of 8 GB RAM and a graphics card of NVIDIA GeForce GTX 1650 4 GB GDDR6 on an i5-11300H, 3.1 to 4.4 GHz processor.

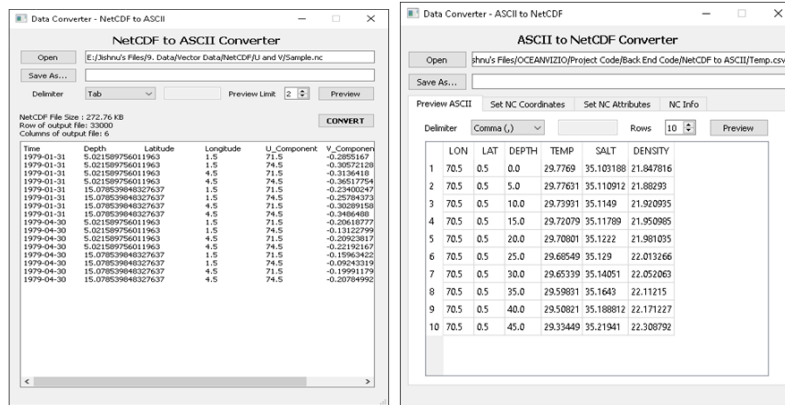


Figure 3. Data converter in data management layer

3.1. Basic plots

The basic plots implemented in this framework include profile plots, vector plots, color maps, and streamline plots. Users are provided with different options to customize the plots. Sample examples of customization frames for plots are displayed in Figure 4. The vector plot interface includes options for providing, time and depth index, as well as location specifications. The arrow settings provide the user options to customize the scale, shaft and head width and head length, and head axis length for the vector plot. The color setting options are also provided for the vector plot. Similarly, the contour plot offers customization in terms of time and depth indices, as well as location preferences. Users can select the relevant parameter and specify the parameter’s range through the contour plot interface. The contour lines can be customized in terms of color, style, and thickness. The color map also provides similar choices along with options to extend the color bar range according to the user’s preference. A sample plot generated by the data rendering engine in the proposed framework, utilizing the CPU implementation, is depicted in Figure 5. The basic 2D plots thus provide a continuous spatial understanding of the parameter under study. Additionally, a sample 3D plot for the variable under investigation is presented in Figure 6, which is implemented using the octree concept mentioned earlier. Users can enhance the visualization smoothness using the interaction techniques provided by the Python interface.

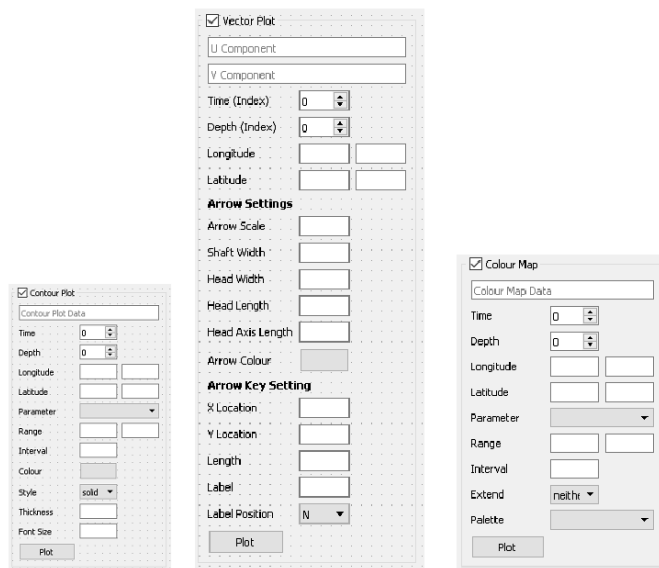


Figure 4. Customization options for basic plots

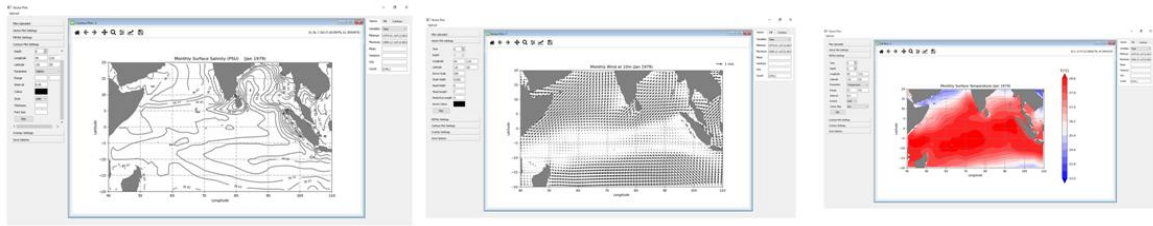


Figure 5. Basic 2D plots-contour plot, vector plot, and colour map

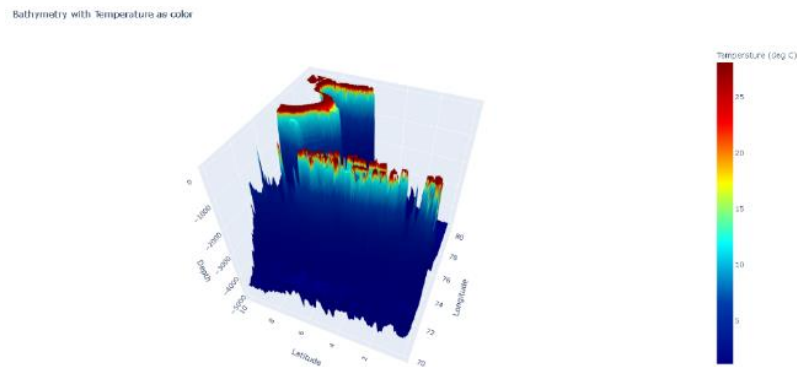


Figure 6. 3D plot for temperature

3.2. Overlay plots

Multiple data sets are displayed on one graph in an overlay plot, enabling direct comparison and the detection of trends, patterns, and correlations between variables. With the use of overlay plots, numerous datasets or variables can be compared visually at once. The charts aid in examining the relationships between various factors. We can see correlations, dependencies, or causal links by overlaying two or more basic graphs. Figure 7 represents the example overlay plots. Figures 7(a) and 7(b) displays sample graphs of wind overlaid on temperature anomaly and salinity overlaid on temperature.

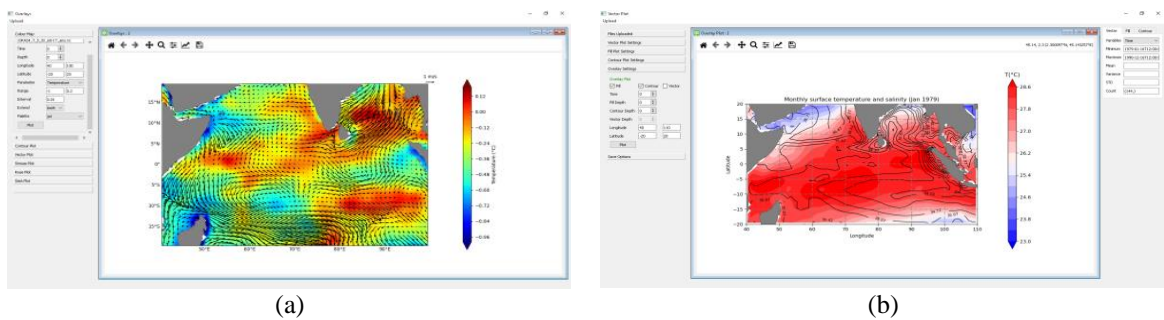


Figure 7. Examples of overlay (a) wind overlaid on temperature anomaly and (b) salinity overlaid on temperature

4. CONCLUSION

The main challenge in analyzing large volumes of ocean data is its inherent complexity. To address this, an interactive and adaptable multi-scale, multivariate visualization system is essential. This study introduces a novel Python-based interactive tool designed for visualizing multivariate data in both 2D and 3D formats. The tool supports various data formats and allows for single-variable visualizations like vector plots and color maps, as well as multivariate visualizations. It processes and refines data using multiple pre-processing techniques and accommodates octree representation and kriging interpolation for improved 3D plots. Overlay plots enable comparisons and the identification of patterns, trends, and correlations among diverse variables, making the system adaptable for visualizing different types of scientific data.

This study presents an interactive application for multivariate visualisation of ocean characteristics, using advanced 3D visualisation techniques and customisable overlays, based on Python. The technology makes it easier to identify patterns, trends, and anomalies, integrates a variety of data sources, and facilitates thorough data processing—all of which contribute to a better knowledge of marine processes. By adding real-time data integration and analysis to the tool's capabilities, future research can build on these discoveries and provide more dynamic and instantaneous insights into ocean conditions.




ACKNOWLEDGEMENTS

This work is financially assisted by Naval Research Board (NRB) of Defence Research and Development Organization (DRDO), Ministry of Defence, Government of India. The authors are thankful to the scientists in Naval Physical and Oceanographic Laboratory (NPOL), Kochi, for their research guidance. Authors are thankful to copernicus.eu consortium for providing the sea level and geostrophic data.




REFERENCES

- [1] C. Xie, M. Li, H. Wang, and J. Dong, "A survey on visual analysis of ocean data," *Visual Informatics*, vol. 3, no. 3, pp. 113–128, Sep. 2019, doi: 10.1016/j.visinf.2019.08.001.
- [2] S. Liu *et al.*, "An intelligent modeling framework to optimize the spatial layout of ocean moored buoy observing networks," *Frontiers in Marine Science*, vol. 10, Apr. 2023, doi: 10.3389/fmars.2023.1134418.
- [3] P. C. Wong and R. D. Bergeron, "30 years of multidimensional multivariate visualization," *Scientific Visualization Overviews Methodologies and Techniques*, no. 2, pp. 3–33, 1997.
- [4] G. Chen *et al.*, "Toward digital twin of the ocean: from digitalization to cloning," *Intelligent Marine Technology and Systems*, vol. 1, no. 1, p. 3, Sep. 2023, doi: 10.1007/s44295-023-00003-2.
- [5] O. Abdusattarov, "Environmental monitoring geospatial data visualization methods," *Prospects of Development of Science and Education*, vol. 1, no. 10, pp. 115–120, 2023.
- [6] Q. Lin *et al.*, "Characterization of lacustrine harmful algal blooms using multiple biomarkers: historical processes, driving synergy, and ecological shifts," *Water Research*, vol. 235, p. 119916, May 2023, doi: 10.1016/j.watres.2023.119916.
- [7] C. Li and C. Zheng, "High-efficiency method for 3D visualization of marine environmental information," *Earth Science Informatics*, vol. 16, no. 1, pp. 367–377, Mar. 2023, doi: 10.1007/s12145-023-00946-4.
- [8] J. Li, T. Huang, P. Hu, W. Cui, and S. Cheng, "A visual analytics framework for ocean scalar volume data," *Ocean-Land-Atmosphere Research*, vol. 2, p. 17, Jan. 2023, doi: 10.34133/olar.0014.
- [9] G. Madec, "NEMO ocean engine." France, Institut Pierre-Simon Laplace (IPSL), p. 300, 2008, [Online]. Available: <https://nora.nerc.ac.uk/id/eprint/164324/>.
- [10] J. I. Antonov, R. A. Locarnini, T. P. Boyer, A. V. Mishonov, and H. E. Garcia, "World ocean atlas 2005. Vol. 2, Salinity," NOAA Atlas NESDIS 62, U.S. Government Printing Office, 2006.
- [11] S. M. Uppala, P. W. Källberg, A. J. Simmons, U. Andrae, V. D. C. Bechtold, and Coauthors, "The ERA-40 re-analysis," *Quarterly Journal of the Royal Meteorological Society*, vol. 131, pp. 2961–3012, 2005.
- [12] D. P. Dee *et al.*, "The ERA-Interim reanalysis: configuration and performance of the data assimilation system," *Quarterly Journal of the Royal Meteorological Society*, vol. 137, no. 656, pp. 553–597, 2011, doi: 10.1002/qj.828.
- [13] "ECMWF operational archive." European Centre for Medium-Range Weather Forecasts, Accessed: Jun. 01, 2024. [Online]. Available: <https://apps.ecmwf.int/archive-catalogue/?class=od>.
- [14] K. Mattsson and M. H. Carpenter, "Stable and accurate interpolat ion operators for high-order multiblock finite difference methods," *SIAM Journal on Scientific Computing*, vol. 32, no. 4, pp. 2298–2320, 2010, doi: 10.1137/090750068.
- [15] C. T. Suva and J. S. B. Mitchell, "The lazy sweep ray casting algorithm for rendering irregular grids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 142–157, 1997, doi: 10.1109/2945.597797.
- [16] K. Yamaguchi, T. L. Kunii, K. Fujimura, and H. Toriya, "Octree-related data structures and algorithms," *IEEE Computer Graphics and Applications*, vol. 4, no. 1, pp. 53–59, 1984, doi: 10.1109/MCG.1984.275901.
- [17] C.-F. Chang, G. Bishop, and A. Lastra, "LDI tree: a hierarchical representation for image-based rendering," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*, 1999, pp. 291–298, doi: 10.1145/311535.311571.
- [18] M. A. Oliver and R. Webster, "Kriging: a method of interpolation for geographical information systems," *International Journal of Geographical Information Systems*, vol. 4, no. 3, pp. 313–332, 1990, doi: 10.1080/02693799008941549.
- [19] Y. Yang, Z. Zhu, X. Shen, L. Jiang, and T. Li, "The influences of atlantic sea surface temperature anomalies on the ENSO-independent interannual variability of East Asian summer monsoon rainfall," *Journal of Climate*, vol. 36, no. 2, pp. 677–692, 2023, doi: 10.1175/JCLI-D-22-0061.1.
- [20] T. Guinaldo, A. Voldoire, R. Waldman, S. Saux Picart, and H. Roquet, "Response of the sea surface temperature to heatwaves during the France 2022 meteorological summer," *Ocean Science*, vol. 19, no. 3, pp. 629–647, 2023, doi: 10.5194/os-19-629-2023.
- [21] D. E. Myers, C. L. Begovich, T. R. Butz, and V. E. Kane, "Variogram models for regional groundwater geochemical data," *Journal of the International Association for Mathematical Geology*, vol. 14, no. 6, pp. 629–644, 1982, doi: 10.1007/BF01033884.
- [22] 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," *Environmental Modelling & Software*, vol. 135, p. 104908, Jan. 2021, doi: 10.1016/j.envsoft.2020.104908.
- [23] W. H. Ali *et al.*, "SeaVizKit: interactive maps for ocean visualization," in *Oceans 2019 MTS/IEEE Seattle*, Oct. 2019, pp. 1–10, doi: 10.23919/OCEANS40490.2019.8962794.
- [24] X. Han, J. Liu, B. Tan, and L. Duan, "Design and implementation of smart ocean visualization system based on extended reality technology," *Journal of Web Engineering*, vol. 20, no. 2, pp. 557–574, 2021, doi: 10.13052/jwe1540-9589.20215.
- [25] C. Gan, W.-H. Cao, K.-Z. Liu, and M. Wu, "Spatial estimation for 3D formation drillability field: a new modeling framework," *Journal of Natural Gas Science and Engineering*, vol. 84, p. 103628, Dec. 2020, doi: 10.1016/j.jngse.2020.103628.




BIOGRAPHIES OF AUTHORS

Dr. Preetha K. G.    completed her Ph.D. in Mobile Ad hoc Networks from Cochin University of Science and Technology in 2018. She is currently working as Professor in the Department of Computer Science and Engineering at Rajagiri School of Engineering and Technology in Kerala, India. She has around 22 years of academic experience. Her research interests include data analytics, machine learning, deep learning, mobile computing, wireless networks, and ad-hoc networks. She can be contacted at email: preetha_kg@rajagiritech.edu.in.






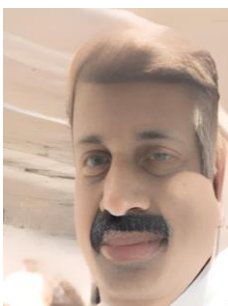
Dr. Saritha S.    received the Ph.D. Degree in Computer Science and Engineering from Cochin University of Science and Engineering, Kerala, India. She is currently serving as a Professor at Rajagiri School of Engineering and Technology (Autonomous) in Kerala, India. She has also contributed numerous journal articles, conference papers, book chapters, and patents, reflecting the innovative contributions to different fields in computer science and engineering. She can be contacted at email: saritha_s@rajagiritech.edu.in.






Mr. Jishnu Jeevan    completed his B. Tech in Computer Science and Engineering from The Albertian Institute of Science and Technology, Kerala, India in 2018. He completed his M.Tech in Computer and Information Science from the Department of Computer Science, CUSAT, in 2021. Currently he is working as Junior Research Associate, Euro-Mediterranean Center on Climate Change (CMCC), Italy. He can be contacted at email: jishnujeevan@gmail.com.



Dr. Chinnu Sachidanandan    is an oceanographer and marine scientist from India. She has completed her graduate studies in oceanography from Cochin University of Science and Technology (CUSAT) and went on to pursue doctoral research in marine sciences at the prestigious CSIR-National Institute of Oceanography. Currently working as a research fellow in Rajagiri School of Engineering and Technology, in Kerala, India. Her research expertise lies in the field of oceanography and marine sciences, with a focus on the physical processes that shape oceans. She can be contacted at email: chinnu.sachi7@gmail.com.



Dr. P. A. Maheswaran    obtained his Masters and Ph.D. in Oceanography from Cochin University of Science and Technology, Kochi, Kerala. He is currently working as a Scientist at DRDO-Naval Physical and Oceanographic Laboratory, Kochi. His research areas include, Mixed layer dynamics, thermohaline structure, sonar oceanography. He can be contacted at email: maheswaran.npol@gmail.com.