



# A Scalable System for Visual Analysis of Ocean Data

Toshit Jain<sup>1</sup> , Upkar Singh<sup>1</sup>, Varun Singh<sup>1</sup>, Vijay Kumar Boda<sup>1</sup>, Ingrid Hotz<sup>2,1</sup>, Sathish S. Vadhiyar<sup>3</sup>, P. N. Vinayachandran<sup>4</sup>, Vijay Natarajan<sup>1,5</sup> 

<sup>1</sup>Department of Computer Science and Automation (CSA), Indian Institute of Science Bangalore, India

<sup>2</sup>Department of Science and Technology (ITN), Linköping University, Norrköping, Sweden

<sup>3</sup>Department of Computational and Data Sciences (CDS), Indian Institute of Science Bangalore, India

<sup>4</sup>Centre for Atmospheric and Oceanic Sciences (CAOS), Indian Institute of Science Bangalore, India

<sup>5</sup>Zuse Institute Berlin, Germany

---

## Abstract

Oceanographers rely on visual analysis to interpret model simulations, identify events and phenomena, and track dynamic ocean processes. The ever increasing resolution and complexity of ocean data due to its dynamic nature and multivariate relationships demands a scalable and adaptable visualization tool for interactive exploration. We introduce *pyParaOcean*, a scalable and interactive visualization system designed specifically for ocean data analysis. *pyParaOcean* offers specialized modules for common oceanographic analysis tasks, including eddy identification and salinity movement tracking. These modules seamlessly integrate with *ParaView* as filters, ensuring a user-friendly and easy-to-use system while leveraging the parallelization capabilities of *ParaView* and a plethora of inbuilt general-purpose visualization functionalities. The creation of an auxiliary dataset stored as a *Cinema* database helps address I/O and network bandwidth bottlenecks while supporting the generation of quick overview visualizations. We present a case study on the Bay of Bengal (BoB) to demonstrate the utility of the system and scaling studies to evaluate the efficiency of the system.

## CCS Concepts

• *Human-centered computing* → *Visualization techniques; Scientific visualization;*

---

## 1. Introduction

Oceanography refers to the study of the physical, biological, and chemical features and characteristics of the ocean. A comprehensive study of the oceans of the earth has implications for scientific understanding, environmental management, and societal well-being. It is crucial for predicting extreme events like hurricanes and tsunamis, enhancing our understanding of large-scale planetary processes such as global warming, and ensuring the sustainable management and preservation of ocean resources and marine ecosystems. With the advancements in collection and generation of ocean data [Ros89, FD06], there is a demand for scalable tools that support effective and interactive visualization of ocean data. Data in oceanography typically consists of multivariate 3D spatiotemporal fields. The fields are generated using simulations or available from satellite imagery, buoy sensors, or in-situ physical observations. Thanks to the explosion in the capability of modern computers and imaging technology, the size of ocean datasets is becoming larger and larger. As a result, visualizing these datasets at interactive speeds is a challenging problem. Further, the datasets are seldom stored locally. The storage on remote servers results in an additional time cost that is dependent on the bandwidth. The dynamic nature of the multiple scalar and vector fields representing physical quantities pose further challenges to the development

of efficient visualization methods. Among the various fields, ocean currents stand out as an important field of interest. Ocean current is a predominant factor in maintaining the heat equilibrium of the ocean-atmosphere system and in influencing the transport of minerals and salt.

Several complex structures such as eddies and surface fronts are studied in oceanography. Mesoscale eddies [Rob10], circular spiral-like oceanic current pattern that typically span tens to hundreds of kilometers in diameter and last for days to months, are a salient feature of the ocean and vital for ocean analysis [McW08, MJD\*99, BNBD\*07]. A *surface front* is the boundary between two or more volumes of water with distinct temperature or salinity characteristics. It is represented as a subset of the boundary of a temperature or salinity isovolume. Understanding its movement through 3D space and time is crucial for analyzing ocean dynamics and processes. There also exist some submesoscale features that play a role in the study of the movement of particles within the ocean. Submesoscale features are transient features that span over 0.1 to 10 km and can last between a few hours to a few days. The submesoscale features generate spatial and temporal variations in salinity, temperature, and density that create diverse habitats for different marine organisms, dictating their distribution and abundance in the ocean. Submesoscale currents also help in understanding and

predicting the movement and spread of particles in the ocean like nutrients and pollutants. One of the most prominent and useful sub-mesoscale features is called a *filament*, a thin and elongated isovolume of water most often found at the fringes and boundaries of an eddy or along the coast. Since the scale of these features is very small and transient, they are only visible in high-resolution ocean models with a resolution of about ~1 km.

This paper addresses the need for scalable solutions for 3D ocean visualization. It introduces pyParaOcean, a visual analysis system that leverages the power of the widely used open source visualization engine ParaView [AGL05] to enable scalable visualizations of data available from ocean models while supporting a multitude of tasks and functionalities that are specialized for oceanography.

### 1.1. Related work

Visualization in oceanography is a challenging area of research due to the rapidly increasing size, heterogeneity, and multivariate nature of the data, and the inherent complexity of ocean phenomena. The use of general purpose analysis and visualization software such as Matlab, Tecplot, AVS, and ParaView is prevalent in the community. However, oceanographers often use tools developed specifically for ocean data, such as Ferret [Fer23], pyFerret [pyF23], and Copernicus MyOcean [myO23]. These specialized tools are often developed as standalone solutions and further produce 2D views of the data.

Some software frameworks developed within the visualization community provide 2D and 3D data visualization capabilities. COVE [GSK\*08] is a collaborative ocean visualization environment that supports interactive analysis of ocean models over the web. RedSeaAtlas [AGT\*19] supports the selection of regions in a 2D map and provides exploratory views of winds, waves, tides, chlorophyll, etc. over the Red Sea. OceanPaths [NL15] is a multivariate data visualization tool that computes pathways tracing ocean currents and supports the plotting of different high-dimensional data along the pathways. This enables the study of correlations between different oceanographic features. [HBK\*23] developed a 3D eddy identification technique based on the sea surface height and the velocity field. Sea surface height and velocity profile have also been previously used for eddy detection [MAIS16].

An oceanographer's analysis workflow includes a few common tasks [GSK\*08] such as inspection of temperature and salinity distributions and vertical cross sections, compare recently measured salinity data against model data, inspect and analyze current vorticities and circulation based on flow data, and analyze extreme events. Isosurfaces and volume rendering are natural choices for visualization of 3D temperature and salinity distributions [DAN12, PBI04]. However, visualization of dynamically changing distributions is a challenge. VAPOR [LJP\*19] is one of the few tools that provides efficient 3D visualization for oceanography and atmospheric science applications. The VAPOR data collection (VDC) model supports interactive visual analysis of large data sizes on modern GPUs and commodity hardware.

Xie et al. [XLWD19] and Afzal et al. [AHG\*19] present surveys of visual analysis methods and tools developed for ocean data. Xie et al. classify the visual analysis tasks into four categories: study of

different environmental variables, ocean phenomena identification and tracking, discovery of patterns and correlations, and visualization of ensembles and uncertainty. Further, they identify opportunities and unexplored areas for future work including efficient and scalable methods for data processing and management, identification of features at multiple scales, and immersive platforms.

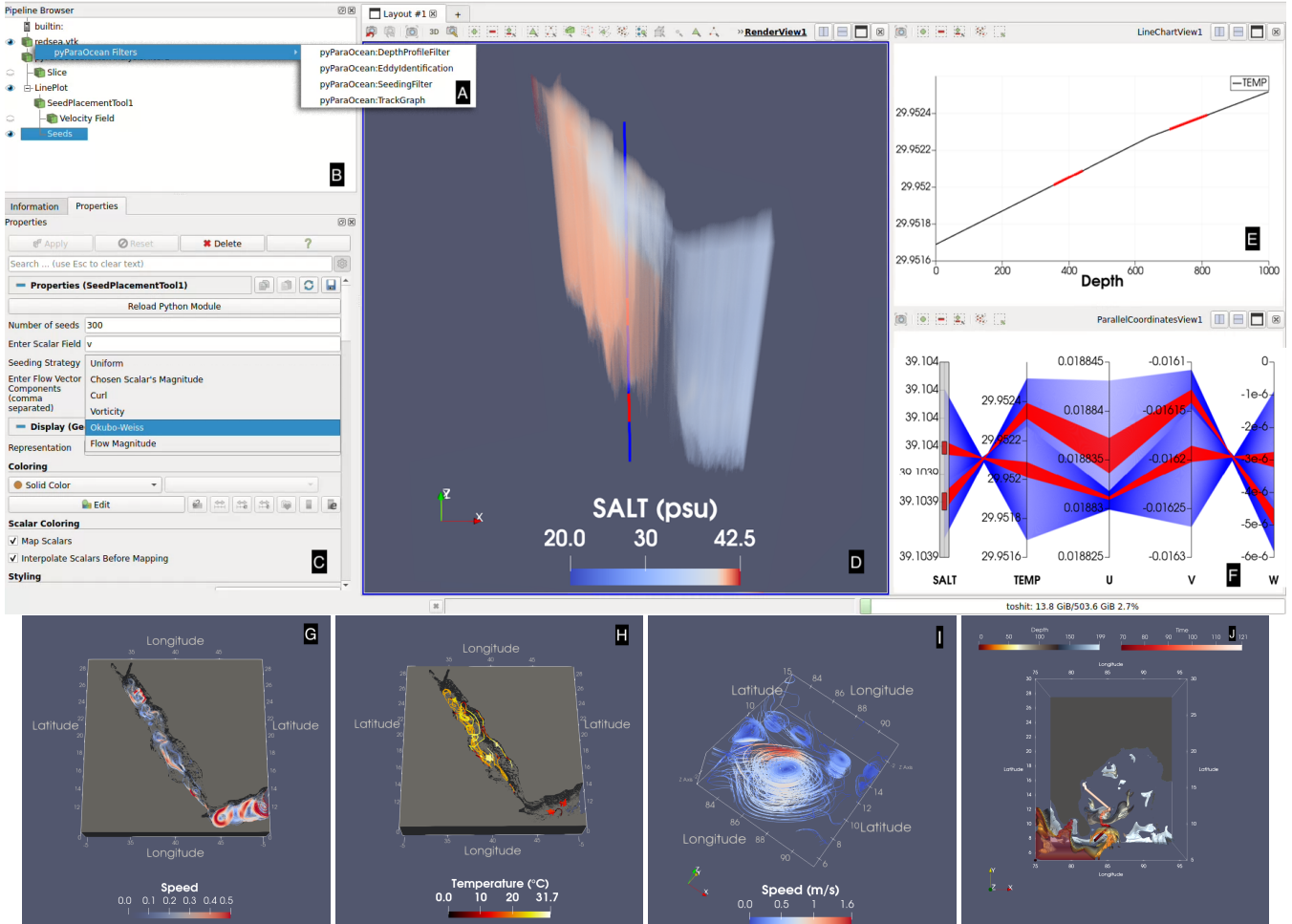
Singh et al. [SDVN22, SVN24] used geometric and topological descriptors to track high salinity water. The study showed that upon entering the Bay of Bengal (BoB), the high salinity water mass splits into three major directions and advances toward Visakhapatnam, the coast of the Andaman and Nicobar Islands, and the center of BoB. In this paper, we report an improved fast and parallel implementation of the salinity front tracking method for large data sizes to demonstrate that the approach is scalable.

### 1.2. Contributions

We present pyParaOcean, a scalable system for visual analysis of ocean data. The visualization capabilities of pyParaOcean are available via a seamless integration into ParaView [AGL05] using plugins. Further, the server-client architecture of ParaView is leveraged to scale the computations and visualizations to large data sizes. pyParaOcean also offers a Cinema Science database generator [AJO\*14] to enable quick analysis and overview generation of a dataset. Using a Cinema database helps to bypass the initial setup time needed for launching a ParaView server on multiple cores and distributing a dataset on those cores, when the goal of the user is to simply to generate an overview. An advantage of this approach is that it offers the flexibility to jump into ParaView for subsequent in-depth analysis and visualization. Main contributions of this paper include

- pyParaOcean: A scalable, extensible, and three-dimensional visual analysis system for ocean data that offers support for seeding strategies for fieldline computation, generating parallel coordinate and vertical section views, and for computing and visualizing eddies and fronts.
- A parallel implementation of an algorithm for front-based tracking of salinity movement [SDVN22].
- A detailed scaling study of the pyParaOcean modules for extracting and visualizing complex mesoscale structures, including eddies and surface fronts.
- A Cinema Science database generator that enables quick analysis and overview visualization via slices along time and depth.
- A case study on a high resolution dataset to demonstrate the need for a scalable ocean analysis system.

The case studies presented in this paper, performed in collaboration with an oceanographer, focus on the exploration of the salinity distribution within the BoB and demonstrate the utility of the proposed application-centric scalable visualization system. A preliminary version of this paper introduced pyParaOcean as a visual analysis tool [JSB\*23] and outlined its key functionalities and features. This paper presents a detailed description of the system, describes additional functionalities, new parallel implementations, a detailed scaling study, improvements in I/O via the generation of a Cinema Science database, and a new case study on a large dataset.



**Figure 1:** *pyParaOcean* functionality and user interface. (A) All *pyParaOcean* modules are implemented as *ParaView* filters. (B) *ParaView* pipeline browser shows the different datasets under study and the filters applied on them. (C) The seeding filter from *pyParaOcean* provides multiple options for tracing fieldlines. The figure illustrates the usage of various filters showcasing (D) salinity visualization using volume rendering, (E) interactive depth profile visualization, (F) multivariate data visualization using a parallel coordinates plot of all fields in the dataset, (G) flow visualization with streamlines, (H) interactively seeded pathline visualization, (I) eddy detection and visualization, and (J) tracking high salinity water movement with a surface front track.

## 2. Data

Oceanography involves the study of intricate temporal and spatial processes that encompass interactions between entities in various scales. The analysis spans from smaller scale features such as eddies and fronts to large-scale structures such as ocean basins and circulation patterns [XLWD19]. Ocean data often consists of a collection of time-varying scalar and vector-valued fields on 3D domains. The data is available from simulations, satellite imagery, buoy sensors, or in-situ observations. The large data sizes are due to the availability of high-performance computing and storage resources, increased sampling resolution, and a growing number of observables. The various fields are available as samples on a rectilinear grid, also called a latitude-longitude grid, in the NetCDF format [RD90]. While the description below is restricted to rectilinear grids, the filters in *pyParaOcean* extend to data available on

other grids that are supported by *ParaView*. All visualizations and analysis in the paper is performed on data over the Bay of Bengal region available from two sources, GLORYS12V1 and ROMS. Both are *reanalysis* datasets, which integrate numerical simulation models with observational inputs thereby ensuring spatiotemporal consistency.

**Global Ocean Physics Reanalysis (GLORYS12V1).** This dataset [Ser18] is a reanalysis product and provides multiple fields including salinity and horizontal velocities across latitude and longitude in NetCDF format. The salinity field is considered during the period June 2016 – September 2016 at daily resolution (122 time steps) on a 3D rectilinear grid, regularly sampled horizontally with a latitude-longitude resolution of  $1/12^\circ$  and irregularly sampled across depth at 50 levels. The data is clipped to the ge-

ographical region corresponding to the BoB, specified as bounds on longitude ( $75^{\circ}E$  to  $96^{\circ}E$ ) and latitude ( $5^{\circ}S$  to  $30^{\circ}N$ ), using the command line tool Climate Data Operators (CDO) [Sch19]. Further, the salinity field is considered up to a depth of 200 m and resampled at regular depth levels. The resampling computation is scheduled in parallel for each time step. High salinity water movement is observed only in relatively shallow waters [ASM\*17] and hence the restriction. The resulting NetCDF file is used for all further processing and analysis. We refer to this dataset as the GLO-RYS dataset henceforth. The scalability analysis of the front-based salinity movement tracking algorithm requires data at different resolutions. This is created by resampling the salinity field at regular intervals of depth, latitude, and longitude using linear interpolation. The samples are at depth levels 1 m apart up to 200 m depth, and at a latitude-longitude resolution of  $1/(12 \times r)^{\circ}$ ,  $r \in \mathbb{Z}$ . The resulting 3D regular grid data enables efficient volume processing and visualization.

**Bay of Bengal ROMS.** This dataset is generated from the well-established high-resolution Regional Ocean Modeling System (ROMS) [SM05]. ROMS is a free surface, three-dimensional primitive equation ocean circulation model based on non-linear sigma ( $\sigma$ ) coordinate of [SH94] and widely used for a diverse range of regional ocean applications. The model configured for the BoB basin stretches from  $77^{\circ}E$  to  $99^{\circ}E$  in the zonal direction and from  $4^{\circ}N$  to  $23^{\circ}N$  in the meridional direction with spatial resolution of  $1/96^{\circ}$  ( $\approx 1\text{km}$  in zonal direction). There are 40 vertical depth ( $\sigma$ ) levels. The vertical levels are allocated in such a way that the vertical resolution is highest in the top 150 m of the water column. For initial and boundary conditions, the HYbrid Coordinate Ocean Model (HYCOM)  $1/12^{\circ}$  daily reanalyses data are used [Ble02, Hal04, CHS\*07]. The model was initialized on January 1, 2012, and integrated till December 31, 2013, without any salinity relaxation. For presenting the scaling studies, the dataset is interpolated to the same 27 depth levels as the GLORYS dataset to present comparable results. Again, the scalability analysis of the front-based salinity movement tracking algorithm requires data at different resolutions. This is created by resampling the salinity field in all 240 time steps (daily resolution beginning June 2012). Again, the samples are at depth levels 1 m apart up to 200 m depth, and at a latitude-longitude resolution of  $1/(12 \times r)^{\circ}$ ,  $r \in \mathbb{Z}$ .

### 3. pyParaOcean: Design and architecture

In this section, we describe the system design and architecture of pyParaOcean, a tool built upon ParaView [AGL05] and designed to support visualization tasks in oceanography. ParaView is an open-source visualization software built upon VTK [SML06] that enables the creation of a visualization pipeline from a network of executable modules. Each module within ParaView can be considered as a functional unit, featuring various input and output ports. A module can perform one of three functions: data generation (zero input ports; one or more output ports), perform some computation or transformation on the incoming data (multiple input and output ports), or render and produce images or graphic primitives (no output ports). While ParaView is a versatile visualization tool equipped with an extensive array of readers, data sources and fil-

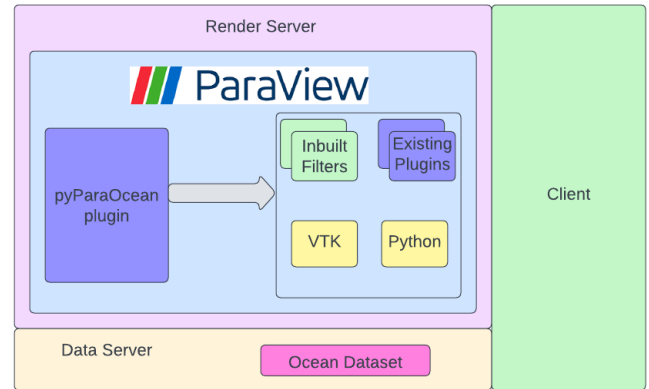


Figure 2: *pyParaOcean* system architecture.

ters, the sheer volume of available filters can become overwhelming and challenging to navigate, especially for experts in specific application domains. Additionally, ParaView offers the flexibility to incorporate new modules through python plugins, ensuring that users can adapt the software to their needs when the existing set of modules are inadequate.

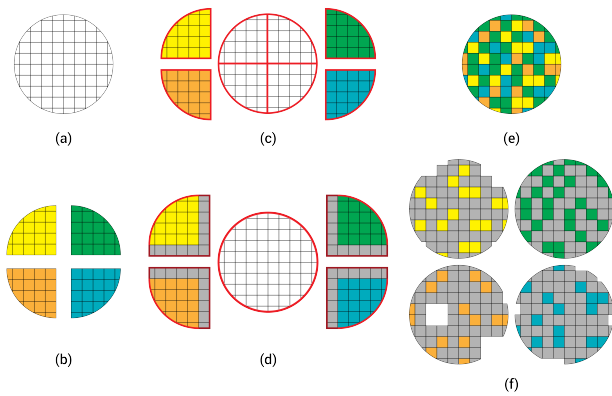
Figure 2 shows the architecture of pyParaOcean, which comprises of data parallel *filters* specifically designed to offer visualization capabilities for the interactive exploration of the three-dimensional ocean data. This design supports easy incorporation of new functionalities as filters in ParaView, the parallel execution of the data server and the render server, and interactive exploration through the client.

#### 3.1. Parallelism

ParaView provides support for parallel data processing, which is helpful for handling large and high-resolution datasets. ParaView can be deployed in a server-client manner. The server stores all data, plugins, and is responsible for rendering, reading, writing, and computing. It can be deployed on a cluster or a supercomputer in a parallel manner using OpenMPI. Moreover, the server can be split into a data server for handling all data processing tasks and a render server for handling all rendering tasks. Each server can be deployed on a different compute hardware. Both the data server and the render server are collectively referred to as *pvserver*. All interactions are handled at the client, which drives the process by creating the visualization pipeline and displaying the generated visualizations.

#### 3.2. Load balancing and ghost cells

ParaView partitions the data into a number of chunks that is equal to the number of parallel processes. Each partition is sent to a different process, and the processes compute the filter on the partition assigned to them independently. VTK and ParaView are designed to keep the communication between processes to a minimum and do not exchange data with each other. This is one of the reasons for the efficiency of ParaView in a distributed setting. Since data is not shared between processes, ParaView uses a concept called *ghost cells* to ensure that the filters produce the correct output. A ghost



**Figure 3:** (a) An example 2D mesh. (b) The mesh is partitioned for parallel processing. Four processes are created and each chunk is sent to a different process. Each process is represented by a unique color (yellow, green, blue, orange). The `vtkDataSetSurfaceFilter` filter is applied to the dataset to compute its boundary. (c) Output of the filter when there are no ghost cells. The filter incorrectly reports edges from the interior of the data as boundary. (d) Ghost cells are inserted on the common boundary between the individual partitions. The filter now reports the correct output, since all false positives are attached to the ghost cells, which are eliminated from the final output. (e) An alternate partitioning of the data into four processes. (f) After the addition of a ghost cell layer, it is apparent that this is an inefficient partitioning of the data since all processes work on almost the entire data, due to poor load balancing.

cell is a data item (say, a voxel in a 3D grid) that belongs to one process but is duplicated on another process corresponding to a spatial neighbor that shares a common boundary. These cells are read-only for the process. While the data is available to the process, the cell is “owned” by another process. Figure 3 provides an example to illustrate the idea. The partitioning of the data also dictates the load balancing and performance of a filter. Partitioning and load balancing assume high importance in the context of ocean data. If the data contains landmass, a region where no computation takes place, the process that holds the chunk of data with little or no ocean data will run idle and result in an inefficient load balancing.

#### 4. pyParaOcean: Functionalities

In this section, we list the functionalities and modules available within pyParaOcean. These modules for ocean data visualization are implemented as plugins and filters in ParaView. Figure 1 shows the output of the modules and the user interface of pyParaOcean.

##### 4.1. Seed placement and fieldlines visualization

Fieldlines, which encompass streamlines and pathlines, offer a comprehensive perspective of a 3D vector field. pyParaOcean provides a filter to choose from diverse seeding strategies for streamline and pathline computation. The seeds produced by this filter

serve as input for the customized source streamline integrator or particle tracer within ParaView (Figure 1(G,H)). Streamlines are a collection of integral curves aligned with the velocity vector at each spatial point, serving as instantaneous representations of flow patterns that help understand significant oceanographic occurrences like eddies, currents, and filaments. Pathlines trace the evolution of the velocity field over time, by depicting the trajectory of a massless virtual particle from a seed point at a specific time step. Pathlines are helpful in comprehending transport phenomena such as salinity advection and debris accumulation. Generating the pathlines is compute intensive compared to streamlines, and is heavily dependent on I/O speed.

The seeding filter controls the number of seeds and manner in which the domain is sampled for seed placement, see Figure 1 (C). Sampling can be achieved through (a) uniform distribution, (b) weighted based on flow speed, curl, vorticity, or the Okubo-Weiss criterion, or (c) weighted according to user defined scalar fields that are calculated earlier in the pipeline. Users have the flexibility to fine tune various line integration parameters and sampling preferences to minimize visual complexity, focus computations on specific areas of interest, maximize domain coverage, and emphasize interesting flow characteristics.

##### 4.2. Isovolumes and isosurfaces

Volume rendering is a natural choice for visualizing 3D scalar fields because it provides a quick overview of the distribution. The volume rendering filter in ParaView can be customized to visualize subvolumes of interest by choosing an interval within the range of the scalar field. For instance, a user may extract an isovolume containing the mean salinity / temperature value within the spatial region of interest, or an isovolume that captures high salinity water.

##### 4.3. Depth profile view

This filter enables the user to inspect a vertical column of the ocean, specified by a longitude and latitude pair. It drops a “needle” into the ocean and samples points along this vertical line at different values of depth. A linked parallel coordinates plot presents a depth profile of all scalar fields sampled along the vertical column. A line plot view of the chosen scalar field against increasing depth values is also displayed. Optionally, the scalar field mapped to a vertical slice at the chosen longitude is shown in the volume render window. The user can select and highlight a subset of points in the vertical column from the parallel coordinates plot and track them across time in all views. This is useful for studying vertical mass transport, especially upwelling or downwelling via Ekman transport [Sar13] in eddy centers, and to study the depression of isotherms indicating redistribution of heat [KNR\*07].

##### 4.4. Front tracking

Oceanographers frequently study water masses that are responsible for mass or heat transport. The water masses are often characterized by distinct temperature or salinity ranges. Singh et al. [SDVN22] developed novel representations of high salinity water using connected components of an isovolume boundary, called the *surface*

front. The front tracking filter computes the surface front, traces their movement over time, and generates a track graph summarizing the movement of all surface fronts. The filter uses the Python multiprocessing library for parallel processing and can therefore execute only on the cores present on the local workstation (client). In order to support larger data sizes, we have developed a stand-alone Python script that uses MPI to utilize all cores in a cluster to compute the track graph. A selected set of tracks derived from this graph can be displayed for visual analysis. Surface fronts have proven to be effective representations of high salinity water masses and have been utilized to trace the movement of high salinity water as it enters the BoB from the Arabian Sea.

#### 4.5. Eddy identification and visualization

The eddy identification filter in pyParaOcean is designed to be extensible and allows for multiple implementations. The current implementation of the filter focuses on mesoscale eddies [AMM17]. It uses the velocity field in individual time steps and does not require any derived fields. This 3D detection scheme can be applied in parallel across time steps and across depth slices since the vertical velocity is not used. The flow speed of the swirling fluid decreases radially inward towards the center of rotation. The filter inspects the local minima of the flow speed to identify potential eddy centers. Vertical velocity is ignored to discount the motion of upwelling or downwelling in vortex cores, thus enhancing the corresponding flow minima. Noise and less significant minima are removed by applying a topological persistence-driven simplification [TFL\*17]. Next, the method employs an approximation of the winding angle criterion [FFH21] by checking if the streamline crosses into all four quadrants of an XY plane that is centered at the minimum [GEP04]. This method is more effective in regions with relatively stationary eddy centers. Streamlines seeded near the core of an eddy form spirals or closed loops. The boundary of an eddy is determined using a binary search along the radial axes. The search helps locate the seed that is furthest from the eddy center but results in a spiral or nearly closed loop streamline. The filter displays all streamlines originating near the detected vortex core, and hence presents a 3D profile of the eddy.

There is potential for implementation of other existing methods for reliable eddy identification and visualization. For example, a vorticity based method [McW90] that identifies the eddy centers using vorticity extrema and calculates the eddy boundary by comparing the neighborhood vorticity values to the center, or one that uses a special Okubo-Weiss parameter based on shear and strain deformation and the vertical component of vorticity to measure rotation and hence identify potential eddies [Oku70]. The filter may be extended to support other eddy detection methods [MAIS16, FFH21] that may be selected via the interface.

#### 4.6. I/O and the NetCDF format

Ocean data is typically stored in the NetCDF format. NetCDF (Network Common Data Form) is a versatile file format and software library widely used in scientific research, particularly in areas such as atmospheric science, climate science, and specifically within oceanography. It is self-describing, which means that the file contains metadata, such as data attributes, dimensions, and variable

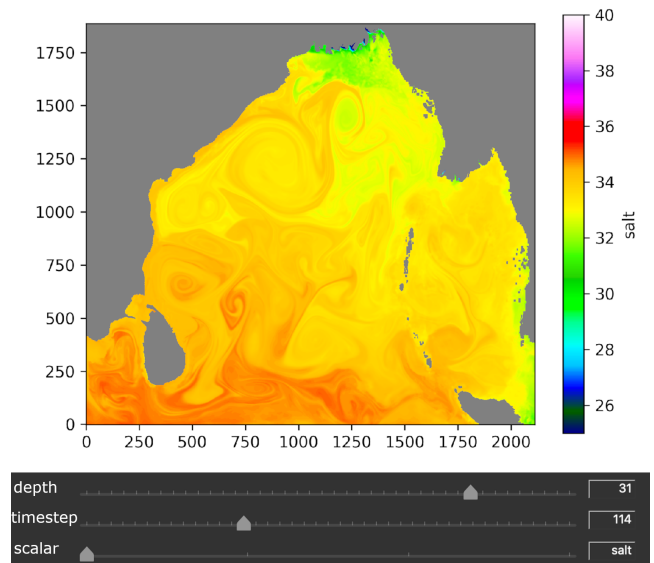
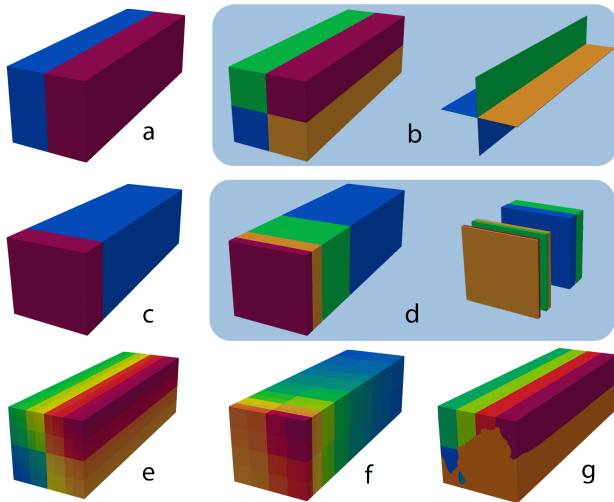


Figure 4: Cinema view with sliding toggles to scroll through the depth slices, time steps, and different scalar fields

specification that describe the structure and content of the data. It offers a hierarchical structure for complex data, is platform-independent, supports large datasets efficiently, and is accessible through various programming languages. Since this data format is self-describing, metadata handling can become an I/O bottleneck for the NetCDF format since the metadata is often stored in a single location within the file system. Beginning with version 4, NetCDF supports parallel I/O, built on top of parallel HDF5. There exist some libraries like PNetCDF [LLC\*03] that bring parallel I/O support to older versions of NetCDF. However, the efficiency of the parallel I/O continues to be heavily dictated by metadata handling and the file system being used.

#### 4.7. Cinema database generator

High-resolution datasets consume a lot of storage space and are impractical to store within the local workstation. The bottleneck of loading the dataset into memory is further amplified due to the additional task of retrieving the dataset from a remote location. The I/O is dependent on the bandwidth of the connection, the distance between the server and the local machine, and the I/O speeds of the storage drives in the server. To circumvent this, we generate an auxiliary dataset that is smaller than the original dataset by several orders of magnitude. One such approach for data reduction and storing data artifacts is the Cinema project, which stores visualizations in an image database for post hoc interactive visualization and exploration of the data. This is especially useful when working with an ocean dataset if the images are generated for every depth slice and every time step. Visualizations in oceanography have traditionally been restricted to 2D depth slices of the ocean. We generate high-resolution float images of the required time steps for every depth slice. This strategy preserves the submesoscale features.



**Figure 5:** Partitioning the spatial domain of the ROMS dataset for efficient visualization. Each block in the partition, represented using a unique color, is sent to a unique core that processes the data within the block independently. (a,c) partitioning into 2 blocks. (b,d) partitioning into 4 blocks (left) and the corresponding ghost cells (right). (e,f) partitioning the domain into 144 blocks. (g) partitioning the domain into 8 blocks. BoB is shown overlaid in solid orange, indicating that several blocks are restricted to land and hence are not assigned any computational task.

The cinema database helps the oceanographer swiftly scroll through the data, where the common practice is to study individual depth slices. We note that the cinema generator is flexible enough to allow the user to store the vertical slices instead in the cinema format, as necessary. An example can be seen in Figure 4, which shows different sliding toggles to scroll through different depth slices, timesteps, and various scalar fields like salinity, temperature, and velocity. pyParaOcean generates the image database using float images. These images are stored in the standard PNG format, where each pixel contains the corresponding value of the scalar data. This representation enables the user to visualize derived fields within the Cinema viewer directly. The dramatic reduction in I/O times, computation, and storage requirements make it possible for an oceanographer to swiftly scrub through the dataset and identify regions of interest without any friction.

## 5. Parallel and distributed computation

In this section, we discuss how pyParaOcean is set up for handling large high-resolution datasets. Distributed parallel computing is enabled in ParaView by launching the pserver on a remote cluster and connecting to it from the local client. The visualization pipeline is constructed on the client end, and all computation is handled by the remote server. In addition to the size of the dataset, the efficiency is also heavily dictated by how the dataset is distributed across the different cores.

### 5.1. Load balancing

Figure 5 illustrates different approaches to partitioning the spatial domain of the data. Each unique colored block of the partition is processed independently by a different core. The partitions in Figure 5(a,b,e) are generated automatically by ParaView into 2, 4, and 144 blocks, respectively. In contrast, the partitions in Figure 5(c,d,f) are the more efficient distribution of data that we propose for this dataset across 2, 4, and 144 blocks, respectively.

The efficiency of the pyParaOcean filters depends on achieving balanced data distribution across processing cores. Blocks containing only the landmass data points correspond to zero computational workload and create bottlenecks due to load imbalance. The ocean region solid orange (■) is overlaid on the grid in Figure 5(g) to help identify the blocks that contain ocean data points. The two green blocks (■) on the top left corner in Figure 5(g) do not overlap with the ocean data points. These blocks consist only of landmass data points, which correspond to NaN values, and do not contribute to the computation. Similarly, in Figure 5(b) the top-left green block (■) contains only landmass data points, while the other three cores perform more computations since they receive actionable ocean data points. Hence, Figure 5(d) is a better partition which assigns equal amount of data to all four cores. Further, the ocean data is distributed equally between the four cores, thereby eliminating the load imbalance. Similarly, the partition in Figure 5(c) is better than the one shown in Figure 5(a) because the blue block (■) contains fewer ocean data points as compared to the red block (■).

Filters such as the fieldline generator do not require interaction between data items in the vertical (depth) direction. In this case, partitioning the data along the depth slices offers a significant advantage. Since the computation is performed independently within a slice, the filter does not require to access information from neighboring slices. In contrast, partitioning along latitude or longitude results in a block that requires communication with neighboring blocks via ghost cells for streamline computation. Data is requested through ghost cells only on demand. So, the ghost cells in Figure 5(d) are not used to compute the output of the fieldline computation filter, whereas the ghost cells in Figure 5(b) are used and the filter requests data from them. So, even though the number of ghost cells in the former case is considerably larger, the partition remains computationally efficient because the ghost cells are not utilized. The ocean data has lower resolution along the depth direction. So, the partition has to eventually be along latitude and longitude. So, with increasing number of blocks, the computation times for the two partitioning strategies become similar. The similarity between the partitioning schemes is visible from Figures 5(e,f).

### 5.2. Parallel front tracking

A recently developed front-based method for tracking of salinity movement has helped document HSC propagation in the BoB [SDVN22]. This algorithm for front computation and tracking is now implemented in parallel across time steps and across individual depth slices, resulting in an efficient pyParaOcean module.

Front-based tracking proceeds in two major steps. First, extract

an isovolume ( $\geq 35$  psu) and compute the boundary curve as the intersection of the isovolume with each depth level. The boundary curve may consist of multiple components. Segment each connected component of the boundary curve into a north-facing segment and a south-facing segment. Next, connect such north-facing segments across all depth levels if they lie within a small neighborhood specified by a distance parameter  $n$ , resulting in surface fronts. In the second step, the surface front correspondence between time steps is identified using the parameter  $n$ . Two fronts correspond from consecutive time steps correspond to each other if they lie within a small spatial neighborhood specified by the distance parameter  $n$  or spatially overlap with each other. A track graph, whose vertices correspond to the surface fronts and whose directed arcs correspond to pairwise connections between surface fronts, is constructed to summarize all possible movements of the front.

In the first step of the method, all computations within a time step are independent of other time steps. Hence, they are executed in parallel. In the second step, computing arcs between two consecutive time steps requires that all surface fronts within the two time step are already computed. This necessitates a synchronization at the end of the first step. The neighborhood search and correspondence computation is sped up by representing the isovolume as a binary grid and utilizing simple matrix operations. The boundary of the isovolume is computed using a simple  $3 \times 3$  averaging filter, followed by a selection of voxels whose value lie strictly between 0 and 1. This boundary is further processed to extract the north-facing boundary, which is eventually stored again as a binary 3D grid (1s representing the boundary). Neighbor search for a given distance parameter  $n$  is made efficient by transforming it into a simple overlap problem. Each 1 is expanded into a  $n \times n \times 2$  neighborhood of 1s centered at the voxel and extending to the next time step. A subsequent connected component labeling step using a 26-neighbor connectivity labels individual connected components of the expanded grid of 1s. Multiplying the values in this grid with the corresponding values of the boundary grid produces the grid containing the surface fronts, where each voxel has a unique label that indicates the component of the surface front.

Each component of the surface front corresponds to a vertex of the track graph. We use an  $n \times n$  mask that consists of 1s for voxels in the circular neighborhood of radius  $n$  and 0s otherwise. Arcs of the track graph are computed using this circular binary mask of radius  $n$  in time step  $t + 1$ , centered at all voxels whose label is non-zero in time step  $t$ . For each such voxel with a non-zero label in time step  $t$ , we multiply the mask with the grid value at the voxel in time-step  $t + 1$ . All values outside the neighborhood are set to 0, resulting in a collection of unique non-zero labels. These labels are unique identifiers of surface front components, where each label represents the destination of the arc and the origin of the arc is the surface front containing the voxel from time step  $t$ . This implementation results in significant runtime improvements because several of the computations are transformed into matrix operations.

The method described above is executed in parallel for each time steps. The computation is parallelized over all available cores using MPI or the Python multiprocessing library. A process is responsible for a single time step  $t$ . The process computes surface fronts within time step  $t$ , and waits until surface fronts for time step  $t + 1$  are

computed before computing arcs between the two time steps. The total runtime is dominated by the longest running process, provided that a sufficient number of cores are available to independently handle each time step. Moreover, all the grids and maps are stored as NumPy arrays [HMvdW\*20] and all grid operations are performed using inbuilt NumPy methods. NumPy is faster at grid operations because it supports array processing without requiring to individually address each element. Further, NumPy uses machine-native data types as opposed to Python's object types and provides fast implementations in languages such as C and Fortran for each grid and matrix operation. Many of the NumPy operations also leverage multithreading. The culmination of all these factors results in fast computations.

## 6. Scaling studies

In this section, we present an experimental study of the scaling behavior of the pyParaOcean modules for computing and visualizing ocean structures.

### 6.1. Experimental setup

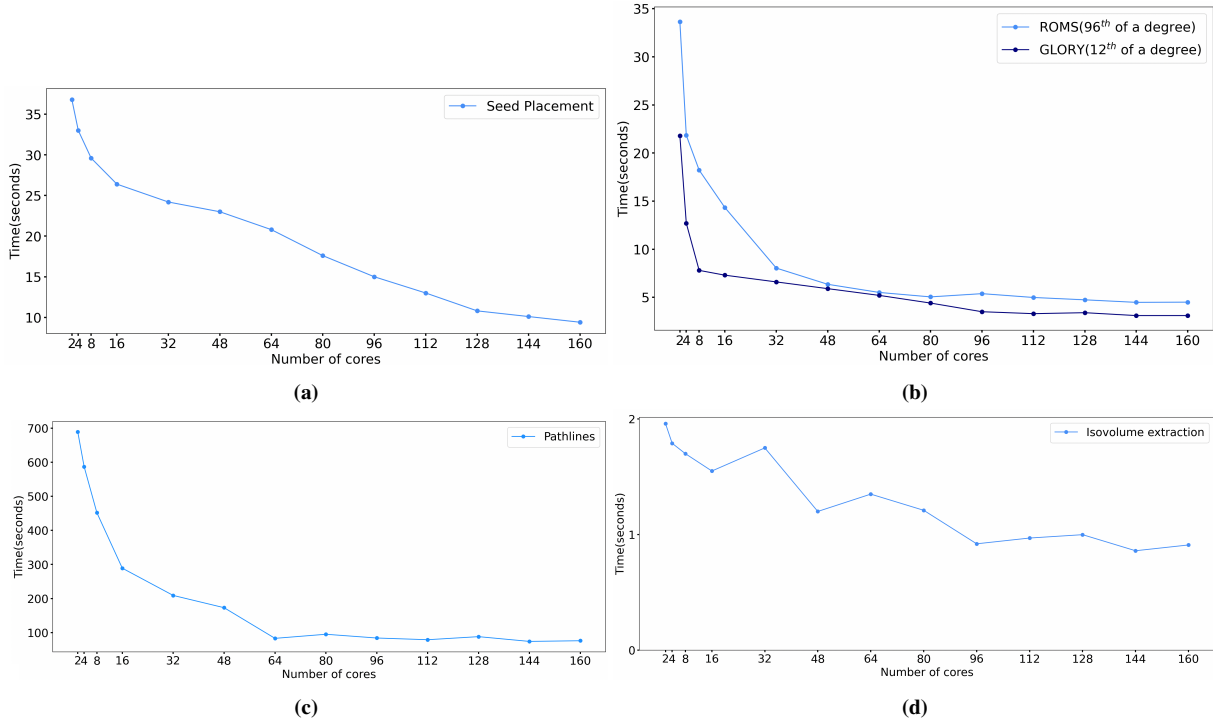
All scaling experiments are performed on a cluster consisting of eight nodes. Each node contains an AMD EPYC Processor with 32 cores and 256GB RAM. For executing the front-based salinity tracking, we use MPI to schedule processes on cores in all cluster nodes. All scripts are written in Python3.

### 6.2. pyParaOcean filters

The filters in pyParaOcean demonstrate excellent scalability with increasing number of processing cores. This allows for efficient handling of large and complex ocean datasets. Figure 6a shows the scaling behavior of the seed placement tool with increasing number of processing cores. We load the dataset using the default partitioning scheme of ParaView and execute the filter to place 1000 seeds weighted according to the vorticity field. This ensures that a larger number of seeds are placed in regions with high vorticity. The calculation of the vorticity field and subsequent placement of seeds is parallelized. We observe that the runtime reduces from 36s across two cores to approximately 9s across 160 cores. A similar trend is observed when we execute the streamline (Figure 6b) and the pathline (Figure 6c) filters. Both filters are executed with 500 seeds. Figure 6b shows the runtimes, both on the ROMS and the GLORYS datasets. We observe that the filter scales well, and saturates at approximately 80 cores, after which there is no significant speedup. A similar saturation is also observed in Figure 6c where the speedup for the particle tracer saturates after approximately 64 cores. The runtimes reported in Figure 6c are based on an experimental run that preloads 10 time steps into the memory. The particle tracer filter runs on multiple time steps and each time step in the ROMS dataset is stored as a different NetCDF file in a server. Therefore, to report the correct computation times that are not inflated due to I/O and the network bandwidth constraints, we preload these files into memory before beginning the computation. We discuss I/O in greater detail within Section 6.5.

The isovolume provides a useful overview but is computationally





**Figure 6:** Scaling behavior of *pyParaOcean* filters. (a) The seed placement filter applied on the ROMS dataset. The seeds are weighted according to vorticity. The scaling is linear, with a steeper drop in runtime up to 16 cores followed by a smaller slope. (b) The streamlines filter computed on 500 seeds. Runtimes drop initially for both ROMS and GLORYS datasets. After the initial steep drop, the plot saturates at 80 cores. (c) The pathline filter applied on 500 seeds, particles tracked over 10 time steps. After an initial sharp drop, runtimes saturate past 64 cores. (d) Isovolume extraction filter applied on the ROMS dataset. Runtimes steadily decrease but saturate at 96 cores. The fluctuations in the runtime may be attributed to the I/O times of the logger routine.

heavy due to the size of the data that is processed and produced as output. Figure 6d shows a general downward trend in the runtime with increasing number of cores. The total runtime of the filter is less than 2s. The fluctuations in the runtimes are likely due to wait times when multiple processes attempt to write into the same log file.

### 6.3. Parallel front tracking

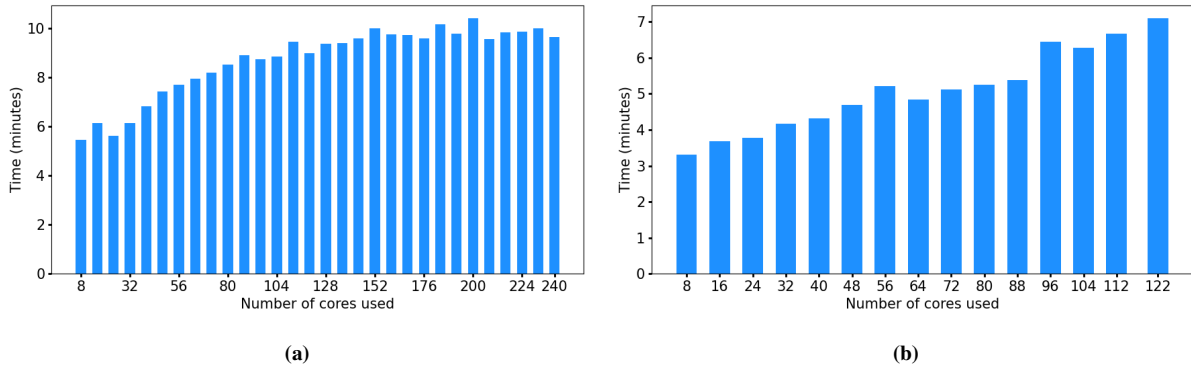
Section 5.2 described a parallel algorithm for front-based tracking of the HSC. In the ideal scenario, if the number of available cores is equal to the number of time steps and all cores are utilized then the run time should remain constant with increasing number of time steps. Providing a larger number of cores than the number of time steps is wasteful for the current implementation of the algorithm. In our first experiment, we evaluate this weak scaling behavior. We observe an increase in runtime even as the number of cores is increased in tandem with the number of time steps, as seen in Figures 7a and 7b. This increase is not large, is within expectations, and the curve seems to flatten as the number of time steps is increased further. This weak scaling behavior is indicative of the applicability of the implementation to larger time periods.

The front-based tracking algorithm is executed on the ROMS and GLORYS data at multiple spatial resolutions to study the runtime

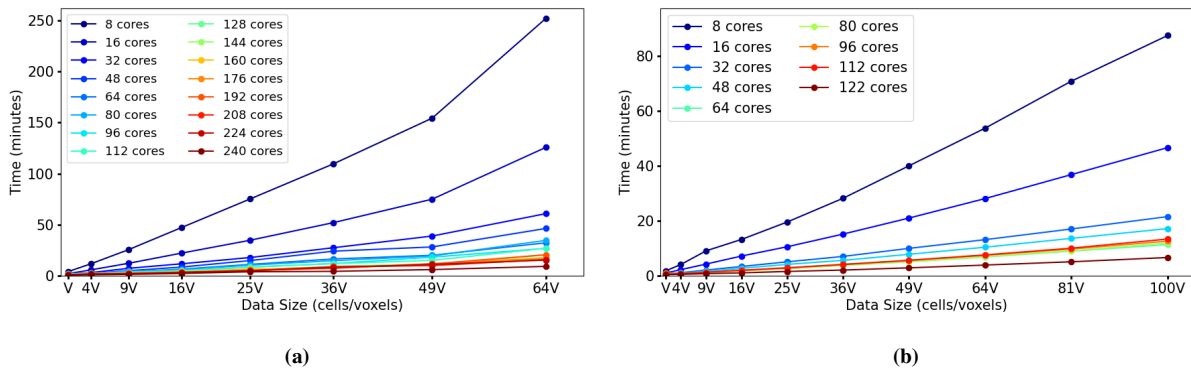
complexity in practice. Executing 240 time steps at a high resolution requires more RAM than available in our experimental setup. On average, it takes 9.6 and 7.1 minutes to process the largest volumes of ROMS ( $2113 \times 1825 \times 200 \times 240$  voxels) and GLORYS ( $2521 \times 3001 \times 200 \times 122$  cells/voxels), respectively.

The aim of the next experiment is to study the runtime complexity. A linear increase in resolution across latitude and longitude results in a quadratic increase in the total size of the data. In the ideal scenario, we expect the runtime to increase linearly with the data size, given that the computational power (number of cores) remains constant. Figures 8a and 8b show that our implementation exhibits close to linear scaling suggesting its applicability for larger datasets with predictable run times. Figure 8a shows a spike in runtime at a spatial resolution of 96<sup>th</sup> of a degree. We believe this sharp increase in runtime is due to the size of data approaching the upper limits of the available RAM. We restrict the experiments on ROMS to this spatial resolution due to the memory limitation.

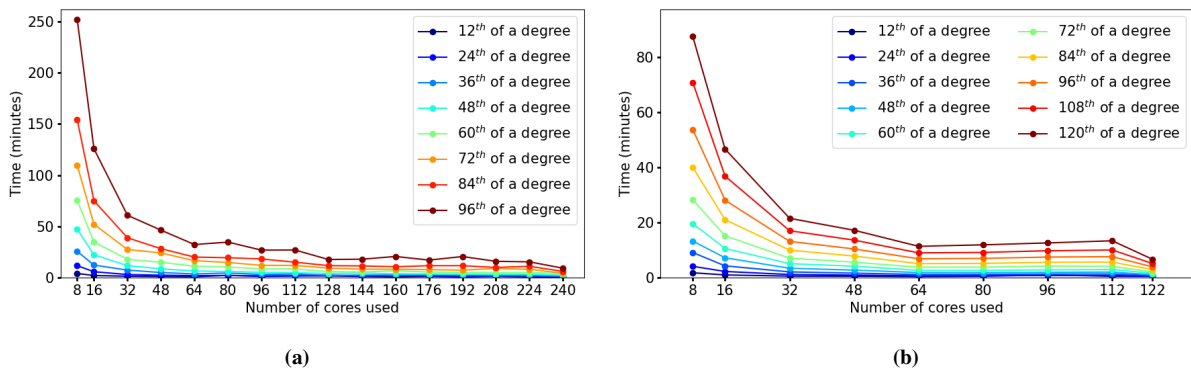
For a fixed data size, increasing the number of computational cores results in an improvement in run time as shown in Figures 9a and 9b. This strong scaling behavior is studied in our final experiment. The improvement in runtime flattens eventually as discussed earlier.



**Figure 7:** Weak scaling study. (a) ROMS data with a resolution of  $96^{\text{th}}$  of a degree. (b) GLORYS data with a resolution of  $120^{\text{th}}$  of a degree. In both datasets, we observe an increase in runtime as the number of cores and time steps increase. The curve appears to flatten towards the end for the ROMS dataset.



**Figure 8:** Runtime complexity. (a) Processing 240 time steps of increasing resolutions of the ROMS data using multiple cores. Let  $V$  denote the data size at a resolution of  $12^{\text{th}}$  of a degree, which corresponds to  $265 \times 229 \times 200$  voxels at each time step or  $265 \times 229 \times 200 \times 240$  voxels in total. (b) Processing 122 time steps of increasing resolution of the GLORYS data using multiple cores. Again,  $V$  denotes the data size at a resolution of  $12^{\text{th}}$  of a degree, which in this case equals  $253 \times 301 \times 200$  voxels at each time step or  $253 \times 301 \times 200 \times 122$  voxels in total. In both datasets, runtime increases linearly with data size. We observe a spike at  $96^{\text{th}}$  of a degree ( $64V$ ) for the ROMS dataset.



**Figure 9:** Strong scaling study. (a) Processing 240 time-steps of a fixed resolution ROMS data using an increasing the number of cores. (b) Processing 120 time steps of a fixed resolution GLORYS data. In both cases, runtime drop is steep initially followed by a eventual flattening of the curve.

#### 6.4. Effect of data distribution

The scalability of the filters in ParaView is highly dependent on how the data is distributed across the processing cores. Figure 10b shows the runtimes for the streamline filter with and without redistribution. The redistribution offers a significant advantage for smaller number of cores (2,4,8). The runtime improvement may be attributed to two reasons. Firstly, as we can see from Figures 5(a,b,g), the default partition skews the data distribution balance. Some nodes are assigned blocks corresponding to land, which does not correspond to any computation. Further, streamline computation requires information from neighboring blocks since the data is partitioned along latitude and longitude. This adds an overhead of requesting and computing additional information from the neighboring blocks in the form of ghost cells. Both drawbacks are eliminated by slicing the data along the depth dimension, as observed from Figures 5(c,d). First, the skew in the data distribution is eliminated by cutting along depth because each process is assigned an equal amount of land and ocean data. Second, streamline computation does not utilize vertical velocity. Generally, vertical velocities in the ocean are several order of magnitudes smaller than the horizontal velocities. The effect of vertical velocities is negligible for visualization of streamlines, which essentially represent the horizontal flow field. Hence, the processes do not require the ghost cells. The exceptions are eddies and small scale turbulence.

Figure 10a plots the number of ghost cells generated without redistributing the data (grey) and with data redistribution (blue). The black dashed line indicates the total number of voxels in the dataset. Even though the number of ghost cells is considerably higher with redistribution as compared to the default partition from ParaView, the runtimes with redistribution are lower. However, as the number of processing cores increases, the two partitioning schemes become similar, see Figures 5(e,f). Hence, the runtimes for both partitioning schemes also converge to similar numbers.

#### 6.5. I/O and the Cinema database

Figure 11 shows the average time taken to load a single time step into memory. NetCDF is a self-describing file format. Both metadata and the actual data are stored within the same file. A file reader is expected to read the metadata to understand the data format. A consequence of this feature is that when multiple processes are launched to execute a filter, all the processes attempt to read the common metadata. While the data is distributed among the processes and can be read with some degree of parallelism, reading the common metadata causes a serialization. Hence, as the number of cores increase, we observe an increase in the time taken to load the dataset into memory. Another factor that impacts the time to load a file is the network bandwidth. For large data sizes, the network bandwidth could become the bottleneck, overshadowing any improvements in the I/O speeds. So, it becomes impractical to load a time step to render a quick overview for the user.

In order to support the ability to provide a quick overview of the data, we propose the generation of a Cinema database [AJO\*14] for local storage. We used our Cinema generator to generate float images for 4 scalars namely salinity, temperature, velocities in  $x$  and  $y$  directions for a total of 100 time steps. This resulted in a

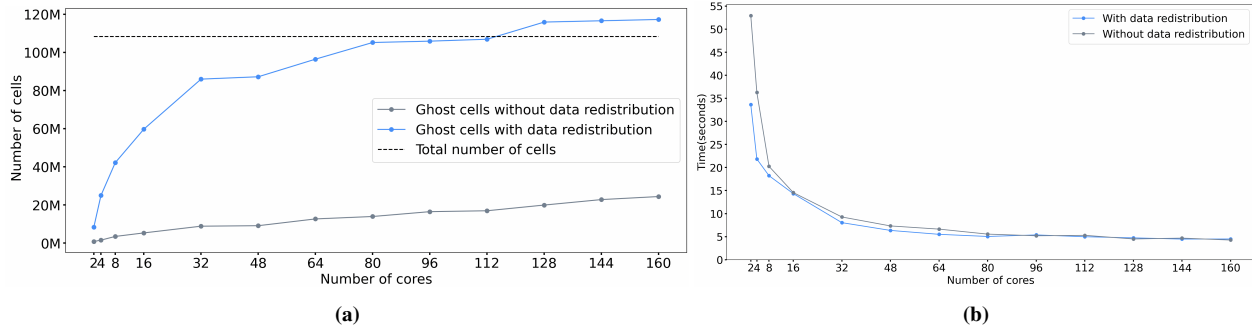
reduction in data size from approximately 750GB to 2.6GB. The Cinema database is 0.35% of the original but captures all attributes that are typically studied by an oceanographer to get an overview of the data. The generator can also be tuned to generate and store additional fields.

#### 7. Case study

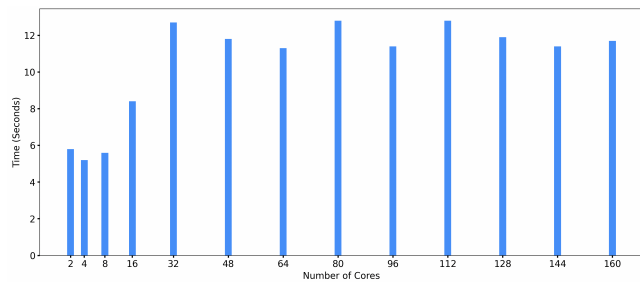
The ocean circulation in the BoB is complex owing to the large amount of fresh water that enters the northern part of the bay and the seasonally reversing monsoon wind forcing. A river plume flows equator ward along the northern part of the east coast of India but the currents are oriented in the opposite direction in the southern part of the bay. Figure 12 presents a rough schematic of the major currents and eddies in the bay during the monsoon season. The Summer Monsoon Current (SMC), a prominent feature of Indian ocean circulation, flows around Sri Lanka and into the BoB. In this section, we describe the use of pyParaOcean to study different phenomena in the BoB, particularly during the monsoon. This study demonstrates the utility of pyParaOcean in the study of ocean systems and is also of independent interest in terms of observations regarding the ocean structures in the BoB.

**Eddies.** As shown in Figure 12, a large anticyclonic eddy (AE) located to the right of the SMC and a cyclonic eddy known as the Sri Lanka Dome (SLD) to its left [VY98] are regular features in this region during summer. The AE has a diameter of about 500 km, located to the southeast off the coast of Sri Lanka, and is characterized by intense downwelling inside owing to its anticyclonic circulation. Vinayachandran et al. [VY98, RVBN19] proposed that the AE is formed by the interaction of the SMC and the incoming Rossby waves from Sumatra. The timeline of the appearance and disappearance of the AE was documented in later work [VCMN04]. We extract the eddy and visualize it using streamlines by applying the respective pyParaOcean filters with seeds placed near the vortex cores. The AE begins forming in June, develops into its circular shape in July, and weakens in August, as shown in Figure 13 and the accompanying video.

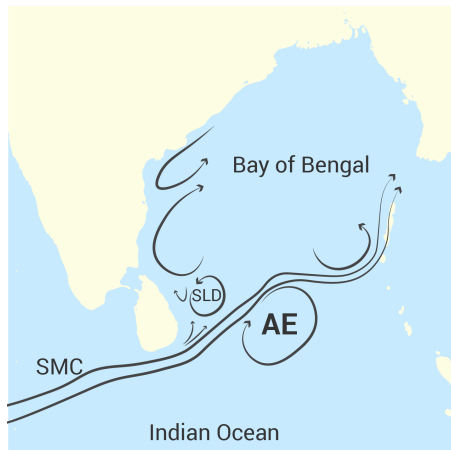
**Salinity transport.** The SMC carries high salinity water from the Arabian Sea into the BoB along its path. This supply of high salinity water is essential to maintain the salt balance of the bay. pyParaOcean serves as an efficient tool to analyze the effects of AE on the salt balance of the BoB. Streamlines and pathlines offer visualization of the circulation associated with the AE and its movement in the ocean. The fieldlines may be overlaid on a volume rendering of a scalar field to visualize the transport caused by the eddy. Figure 14 and the accompanying video show the streamlines overlaid on a salinity volume rendering at different time steps to show the role of the AE in transport of salt. The movement of high salinity water from the Arabian sea by the SMC into the BoB and its recirculation by the AE is well captured in this representation. Tracking surface fronts of high salinity water and highlighting the long-lived tracks helps capture an overview of significant salinity movement in the region. We observe a track in Figure 15 that moves towards the coast of India.



**Figure 10:** Effect of data distribution on scaling behavior. (a) A larger number of ghost cells are generated after data redistribution. (b) Streamlines are integrated with 500 seeds in the ROMS dataset. The filter is significantly faster for a smaller number of cores after redistribution, even when the number of ghost cells is large. But, runtimes are similar for larger number of blocks when the distributions become similar as shown in Figure 3.



**Figure 11:** I/O time increases with number of cores. Time taken to load a single time step in the NetCDF format from the ROMS dataset. The time to load a file into memory increases with number of cores and flattens after 32 cores.

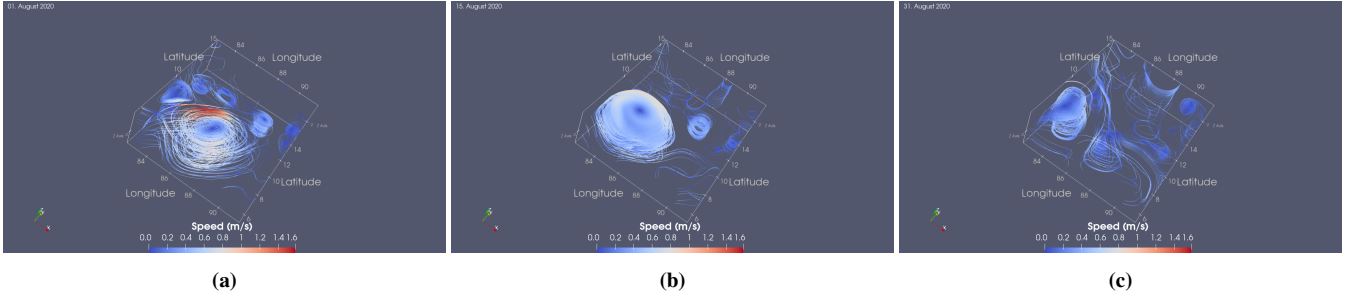


**Figure 12:** Currents and eddies in the BoB during the monsoon season, including the Summer Monsoon Current (SMC), the Sri Lanka Dome (SLD), and an anticyclonic eddy (AE).

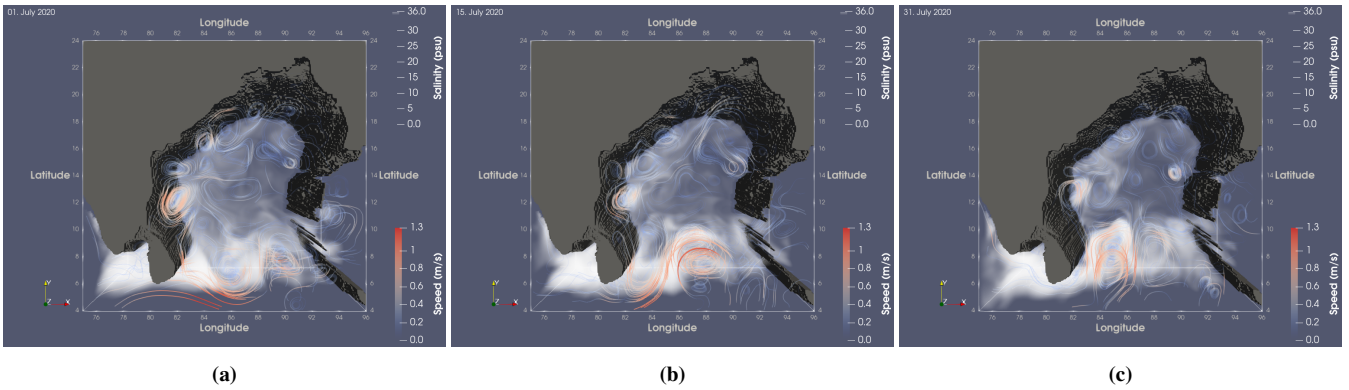
**Downwelling.** Figure 16 and the accompanying video show the use of the depth profile filter to visualize the depression of the 27°C isotherm by the AE. The anticyclonic nature of the eddy causes a downwelling inside the eddy and pushes the relatively warmer water downward. The parallel coordinates view shows changes in temperature, salinity, and speed in the water column caused by the arrival of the eddy at the point of interest.

**Filaments.** An ocean filament is a long narrow strand of moving water within the ocean. Filaments can transport heat, nutrients, and marine life across vast distances. They are formed by wind patterns, currents, and differences in water density. Filaments act like rivers in the ocean, transporting water with unique characteristics, such as temperature, salinity, nutrients, and marine life, across long distances. These elongated features can stretch for hundreds of kilometers, while their width is typically only in the order of 10 kilometers and last from a few days to weeks. Due to their small scale, these features are difficult to analyze in a low resolution dataset. We demonstrate the need for a high resolution dataset such as ROMS in Figure 17. Two isovolumes are rendered in Figure 17 - a low salinity (31.5 - 32.5 psu) isovolume is represented with a green colormap and a higher salinity (33.5 - 34.5 psu) isovolume is depicted by the blue-red colormap. Figure 17a shows the isovolumes in the ROMS dataset at a resolution of 96<sup>th</sup> of a degree, in which the filaments are legible. The same filaments are not as clearly visible at a lower resolution of 12<sup>th</sup> of a degree of the same dataset in Figure 17b. To demonstrate this further with examples, the three arrows point to (a) a green colored low-salinity filament that is intact in Figure 17a, but broken up in Figure 17b, (b) a red colored isovolume that loses its shape in the lower resolution dataset, and, (c) two eddies in dark blue color that lose their structure in the lower resolution dataset. Such filaments and fine features belong to a class of so-called submesoscale features in the ocean. The tools available in pyParaOcean support the efficient tracing of filaments and other submesoscale structures and hence facilitate further statistical study to determine their impact on the ocean environment.

We use the depth profile filter to analyze the behavior of a low salinity filament in the northern BoB. We drop a needle at 17.5° N, 88.5° E and analyze the behavior of salinity using an Eulerian



**Figure 13:** Dissipation of a large anticyclonic eddy in the BoB during August 2020. Streamlines with seeds near detected vortex cores are computed to show the evolution of eddy profiles in 3D.



**Figure 14:** The BoB between July 1, 2020 and July 31, 2020. Visualization of the flow using streamlines with uniform seeding and the  $\geq 35$  psu salinity isovolume. (a) July 1, 2020: The AE is forming around  $8^{\circ}\text{N}$  and  $90^{\circ}\text{E}$  with the SMC streamlines visible from  $78^{\circ}\text{E}$  to  $86^{\circ}\text{E}$ . (b) July 15, 2020: The AE,  $8^{\circ}\text{N}$  and  $87^{\circ}\text{E}$ , has matured into a circular shape and moves westward towards Sri Lanka. The  $\geq 35$  psu isovolume shows a recirculation of high salinity waters into the Bay by AE. (c) July 31, 2020: The AE,  $7^{\circ}\text{N}$  and  $84^{\circ}\text{E}$ , reaches the eastern coast of Sri Lanka where it begins dissipating.

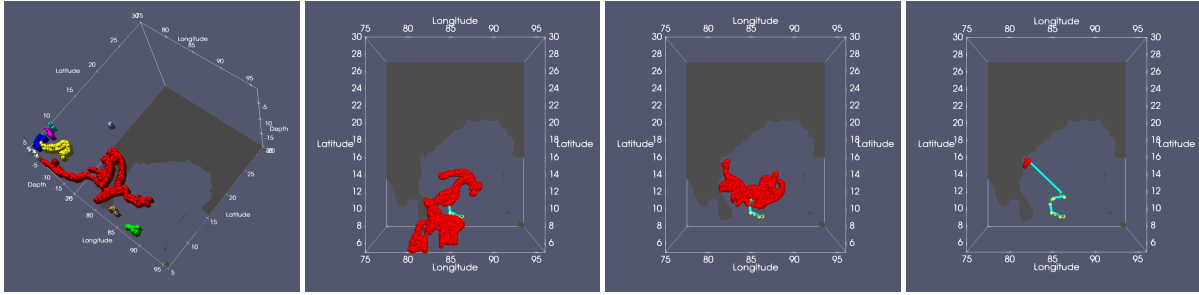
method. This involves the study of the salinity behavior on the needle across different time steps. We also display an isovolume that contains the low salinity filament, colored based on the salinity distribution. Figure 18a shows the behavior of salinity on April 13, 2012 in the salinity-depth plot. The salinity increases steadily at this time step as we go deeper into the ocean. This is due to the higher salinity water having more density. In the next time step at April 15, shown in Figure 18b, we observe that the salinity isovolume crosses the needle, resulting in a sudden drop in salinity in the depth plot. The filament affects the salinity of the water only within the top 200 m and we see the normal steady increase in salinity as we go deeper. The local drop in salinity can lead to the formation of barrier layers [VMRB02] and the shallowing of mixed layers affecting the air-sea interaction.

Next, we study the temperature distribution within this filament as it crosses the same needle. Figures 18c and 18d correspond to the April 13 and April 15 time steps, respectively. The temperature distribution is visualized using a volume rendering of the salinity isovolume and a depth plot. We observe from the temperature-depth plot that the temperature rises by  $0.5^{\circ}\text{C}$  when the isovolume crosses the needle. The extraction of the filament using one property (salinity) followed by its visualization in terms of the time evolution

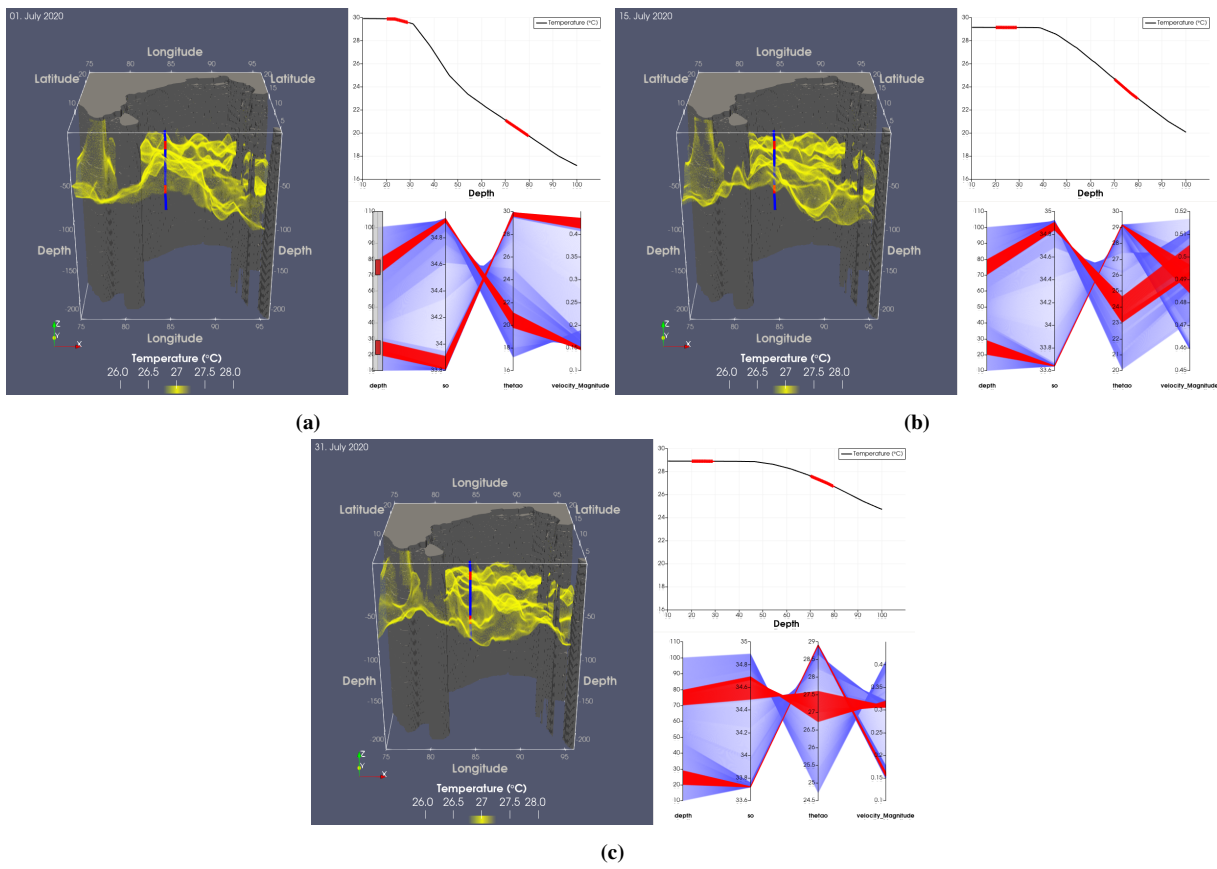
of the temperature field is naturally supported by pyParaOcean filters. This analysis throws some light on how the filament interacts with the surroundings. In this case, the salinity of the filament does not change as it moves, which implies that the mixing of the filament with the surrounding water is low. But, we observe from the volume rendering that the filament does not hold its temperature. The change in temperature over time can be attributed to the interaction of the filament with the atmosphere. This method can be extended to determine exchanges between coastal and open ocean waters, and can be helpful in tracking tracers like sediments and chlorophyll in the fresher water filaments or for tracking oil spills and microplastics to identify the impact of these tracers on the ocean. Quantification of the magnitude and spatial extent of such changes in physical parameters can impact predictions by ocean and atmospheric models.

## 8. Discussion

The scalability experiments and case study validate the utility and applicability of pyParaOcean to large datasets, with potential for broader use in geoscience applications. We now present user inputs and discuss extendibility of the system with respect to the system design and visualization methodology.



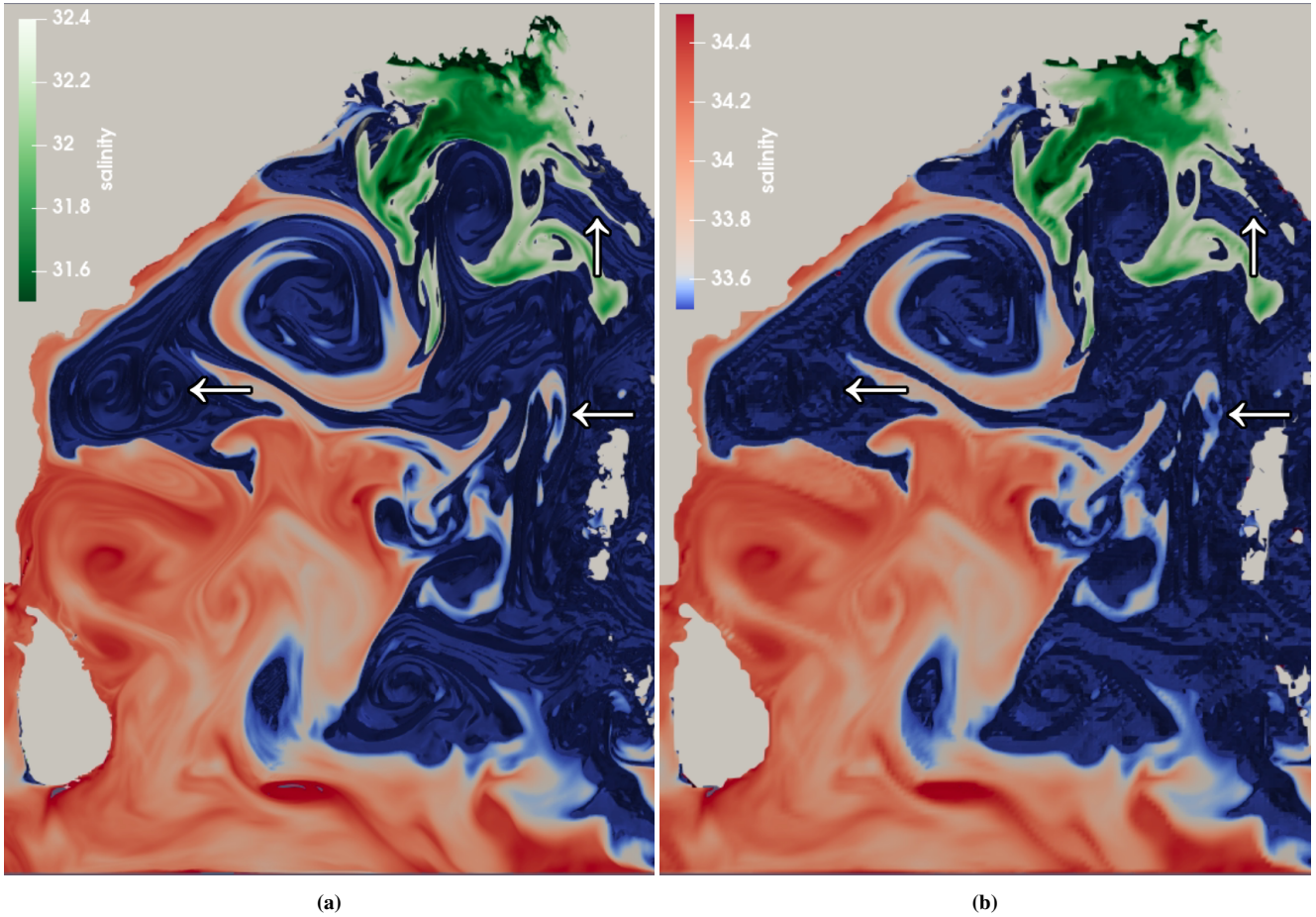
**Figure 15:** Visualizing movement of high salinity water via computation and tracking of surface fronts of high salinity isovolumes. (left) Surface fronts computed at one time step. (middle, right) One of the components of the surface front moves towards the east coast of India, near Visakhapatnam. The evolution of this surface front component is computed and visualized as a track.



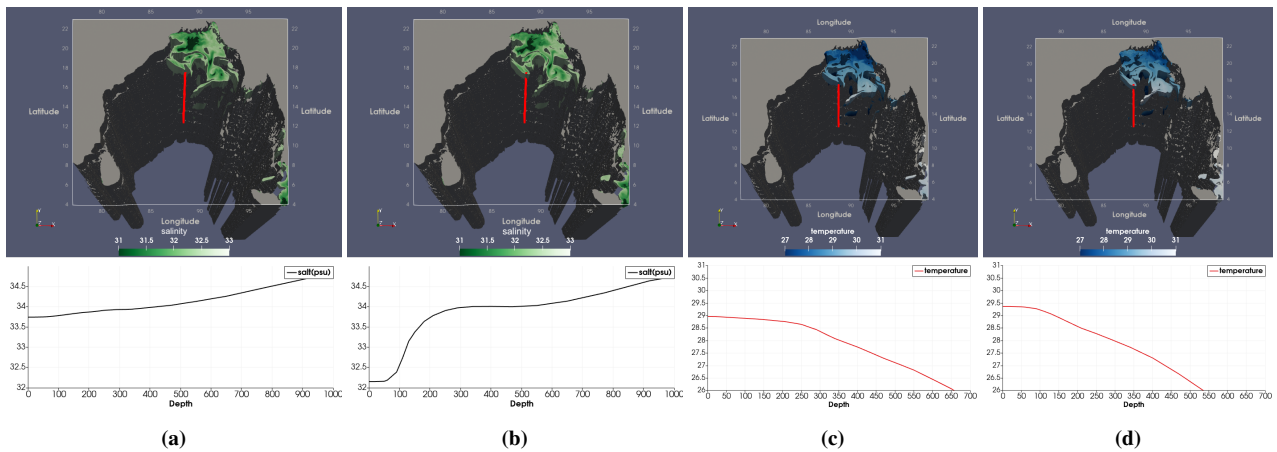
**Figure 16:** The depression of the 27° C isotherm (yellow) by the anticyclonic eddy in the BoB. A needle is dropped at 7° N, 84° E and the depth profile shows the temperature drop. The interactive parallel coordinates plot is used to brush-select 10 m intervals at depths of 25 m and 85 m. (a) July 1, 2020: The downwelling of the AE can be seen around 8° N and 90° E at the depth of 100 m. As it forms, the AE pushes the 27° isotherm down. (b) July 15, 2020: The AE, 8° N and 87° E, can be seen moving east with the depression of the isotherm and the depth profile of temperature begins to flatten near 29°C as the eddy moves closer to the needle. (c) July 31, 2020: The AE center, 7° N and 85° E, is very close to the needle and the depression in the isotherm has moved all the way to near the east coast of Sri Lanka.

**User experience.** This case study was conducted in collaboration with a senior oceanographer coauthor. Below are some comments from them regarding the significance of the study and the advan-

tages of using pyParaOcean for the visual analysis tasks – “Analyzing high-resolution ocean model outputs is a challenging task. In addition to the large volume of the time-dependent 3D dataset, the analysis task is further complicated when multiple variables have



**Figure 17:** The need for higher resolution data to extract certain ocean structures. (a) A low salinity isovolume (green colormap) and a higher salinity isovolume (blue-red colormap) extracted from the ROMS data at a resolution of 96<sup>th</sup> of a degree clearly depicts filaments and some eddies (white arrows) in the BoB. (b) Isovolume extracted from the data at a resolution of 12<sup>th</sup> of a degree. The filaments and eddies are broken up and the clarity of the structures are lost.



**Figure 18:** Visualizing the temporal behavior of a low salinity filament using the depth profile filter on April 13 and 15, 2012. (a,b) Needle placed at 17.5° N, 88.5° E. Plot of salinity vs. depth and analysis of the filament isovolume may help in the study of barrier layers. (c,d) Evolution of temperature distribution within the filament is studied using isovolume extraction, volume rendering, and the depth profile plot.

to be visualized and associations amongst them need to be examined. The combination of Cinema Viewer and the isovolume and depth profile representations provide a relatively handy approach to plow through the large volume of data. As an example, the inflow of large volume of fresh water from river systems such as Amazon and Ganga-Brahmaputra are well known. These rivers also bring along other substances such as sediments and nutrients. Understanding the distribution of these foreign matter is crucial for monitoring the health of the oceans and their impact on the marine system. In the case study, we have shown how temperature and salinity associated with such inputs are tracked and a preliminary assessment of their interaction with the surroundings can be made. A similar approach can be adopted for other passive and active tracers in the ocean and to assess the interaction between different species of tracers. The tools developed here can be used to investigate upwelling fronts or the transport along the periphery of eddies. The Cinema database is detailed enough to get meaningful insights from the data and compact enough to be carried in a USB flash drive.”

Anecdotal inputs from two oceanographers, including one coauthor, suggest that the installation of pyParaOcean is straightforward, and a mild learning curve is sufficient to understand and get acquainted with the visualization pipeline of ParaView. Notably, the interactivity of the system is a significant advantage. While our oceanographer collaborators typically use tools such as pyFerret for 2D analysis, they found the capabilities of pyParaOcean to be very useful and easy to use. In their typical workflow, identifying or studying a phenomenon involved analyzing individual time steps in a NetCDF viewer before importing them into Ferret. The Cinema viewer was an efficient alternative, offering a portable and lightweight solution for quick tasks while additionally supporting the detailed and interactive study of the time evolution of 3D fields. Beyond the functionalities provided by pyParaOcean, the built-in features of ParaView, such as extracting a subset, transfer function editing, calculator tools, and slicing the domain of a dataset proved to be incredibly useful in their analysis workflow.

**System Design.** We developed pyParaOcean as a plugin for the open source software ParaView, to capitalize on its robust scalability support through OpenMPI and advanced visualization capabilities. As demonstrated in the previous sections, this choice results in strong scaling performance and provides a flexible foundation for adaptation across various geoscience applications. While ParaView offers an extensive array of visualization techniques, it is not entirely self-sufficient; additional tools are necessary to bridge certain functionality gaps. For instance, we incorporate the Cinema Science generator as an auxiliary tool, enabling users to analyze the data and tasks at hand before launching the resource-intensive filters within ParaView.

Our implementation of pyParaOcean filters relies, to a large extent, on existing VTK implementations with the objective of computational efficiency and conformation to latest standards. For more complex computation, such as the eddy identification and visualization filter, we designed the filters such that they support multiple implementations catering to the data, the available scalar and vector fields and the task at hand; see subsection 4.5.

Another key factor for selecting ParaView is its server-client ar-

chitecture, which provides flexibility to independently deploy the client, the render server, and the data server. This modular architecture enables users to reliably scale to large datasets and efficiently distribute the compute resources at their disposal. Importantly, the workflow, visualization pipeline, and interface presented to the user remain consistent regardless of deployment configuration and thus ensures seamless user experience.

**Visualization Design.** The design of the visualization modules in pyParaOcean was guided by continuous feedback from our oceanographer coauthor to ensure they meet practical needs in ocean data analysis. The filters described above address many of the core tasks essential for exploring and analyzing ocean datasets. We envision pyParaOcean as an evolving system that will support additional tasks in future, possibly for other geoscience applications. Central to the plugin is its visualization pipeline, inherited from ParaView, which provides users with an intuitive GUI to layer multiple computational steps. The pipeline design abstracts much of the complexity behind these tasks, while also giving advanced users the option to engage with more sophisticated analyses using tools like the programmable filter and the Python shell. This extensibility enables contributions that further expand the capabilities of the system, such as from users towards oceanography or other geoscience applications. The seemingly trivial routine tasks such as data slicing, calculating derived scalars and vectors, and 3D rendering are already built into ParaView and can be easily included into the pipeline. Our visualization choices were further shaped by a commitment to maintaining interactivity even when the system is deployed across multiple cores in a cluster environment. pyParaOcean aims to keep users in control by continuously displaying computation output and enabling intuitive, seamless interaction with data throughout the analysis workflow.

## 9. Conclusions

This paper presented an interactive and scalable system, called pyParaOcean, for the visualization of 3D time-varying fields in oceanography. A detailed case study helped confirm the utility of pyParaOcean and a comprehensive scaling study demonstrated its applicability of its modules to large data sizes. In future, we plan to incorporate fast and efficient parallel implementations of algorithms for computing other ocean structures of interest and for tracking them across time. Identification and tracking of features represented by scalar fields is a recurring problem in different subfields of geoscience. The design approach and methods described in this paper may be extended to study datasets from related application domains such as atmospheric science and meteorology. pyParaOcean is available for download in the public domain for use by the community [pyP24].

## Acknowledgments

This research was funded by a grant from SERB, Govt. of India (CRG/2021/005278), partial support to PNV from National Supercomputing Mission, DST, Govt. of India, the Dr. Ram Kumar IISc Distinguished Visiting Chair Professorship in EECS, and a scholarship from MoE, Govt. of India. VN acknowledges support from



the Alexander von Humboldt Foundation, and Berlin MATH+ under the Visiting Scholar program. Part of this work was completed when VN was a guest Professor at the Zuse Institute Berlin.

**Data Availability.** The data that support the findings of this study are available from the corresponding author upon reasonable request.

**Conflict of Interest.** None.

## References

- [AGL05] AHRENS J., GEVECI B., LAW C.: Paraview: An end-user tool for large data visualization. *The visualization handbook 717* (2005). 2, 4
- [AGT\*19] AFZAL S., GHANI S., TISSINGTON G., LANGODAN S., DASARI H. P., RAITOS D. E., GITTINGS J. A., JAMIL T., SRINIVASAN M., HOTEIT I.: RedSeaAtlas: A visual analytics tool for spatio-temporal multivariate data of the red sea. In *EnvirVis: Workshop on Visualization in Environmental Sciences (EnvirVis2019)* (2019), pp. 25–32. 2
- [AHG\*19] AFZAL S., HITTAWA M. M., GHANI S., JAMIL T., KNIO O., HADWIGER M., HOTEIT I.: The state of the art in visual analysis approaches for ocean and atmospheric datasets. *Computer Graphics Forum 38*, 3 (2019), 881–907. 2
- [AJO\*14] AHRENS J., JOURDAIN S., O’LEARY P., PATCHETT J., ROGERS D. H., PETERSEN M.: An image-based approach to extreme scale in situ visualization and analysis. In *SC’14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2014), IEEE, pp. 424–434. 2, 11
- [AMM17] AMORES A., MELNICHENKO O., MAXIMENKO N.: Coherent mesoscale eddies in the North Atlantic subtropical gyre: 3-D structure and transport with application to the salinity maximum. *Journal of Geophysical Research: Oceans 122*, 1 (2017), 23–41. 6
- [ASM\*17] ANUTALIYA A., SEND U., MCCLEAN J. L., SPRINTALL J., RAINVILLE L., LEE C. M., JINADASA S. U. P., WALLCRAFT A. J., METZGER E. J.: An undercurrent off the east coast of sri lanka. *Ocean Science 13*, 6 (2017), 1035–1044. URL: <https://os.copernicus.org/articles/13/1035/2017/>, doi:10.5194/os-13-1035-2017. 4
- [BLE02] BLECK R.: An oceanic general circulation model framed in hybrid isopycnic-cartesian coordinates. *Ocean modelling 4*, 1 (2002), 55–88. 4
- [BNBD\*07] BENITEZ-NELSON C. R., BIDIGARE R. R., DICKEY T. D., LANDRY M. R., LEONARD C. L., BROWN S. L., NENCIOLI F., RII Y. M., MAITI K., BECKER J. W., ET AL.: Mesoscale eddies drive increased silica export in the subtropical pacific ocean. *Science 316*, 5827 (2007), 1017–1021. 1
- [CHS\*07] CHASSIGNET E. P., HURLBURT H. E., SMEDSTAD O. M., HALLIWELL G. R., HOGAN P. J., WALLCRAFT A. J., BARAILLE R., BLECK R.: The hycom (hybrid coordinate ocean model) data assimilative system. *Journal of Marine Systems 65*, 1–4 (2007), 60–83. 4
- [DAN12] DINESHA V., ADABALA N., NATARAJAN V.: Uncertainty visualization using HDR volume rendering. *The Visual Computer 28* (2012), 265–278. 2
- [FD06] FRASER S., DICKSON B.: Data mining geoscientific data sets using self organizing maps. *Mastering the Data Explosion in the Earth and Environmental Sciences, Extended Abstracts* (2006), 5–7. 1
- [Fer23] Ferret. <https://ferret.pmml.noaa.gov/Ferret/>, 2023. [Online; accessed 28-April-2023]. 2
- [FFH21] FRIEDERICI A., FALK M., HOTZ I.: A winding angle framework for tracking and exploring eddy transport in oceanic ensemble simulations. In *EnvirVis: Workshop on Visualization in Environmental Sciences (EnvirVis2021)* (2021). 6
- [GEP04] GUO D., EVANGELINOS C., PATRIKALAKIS N. M.: Flow feature extraction in oceanographic visualization. In *Proc. Computer Graphics International* (2004), pp. 162–173. doi:10.1109/CGI.2004.1309207. 6
- [GSK\*08] GROCHOW K., STOERMER M., KELLEY D., DELANEY J., LAZOWSKA E.: COVE: A visual environment for ocean observatory design. *Journal of Physics: Conference Series 125*, 1 (2008), 012092. 2
- [Hal04] HALLIWELL G. R.: Evaluation of vertical coordinate and vertical mixing algorithms in the hybrid-coordinate ocean model (hycom). *Ocean Modelling 7*, 3–4 (2004), 285–322. 4
- [HBK\*23] HUA W., BEMIS K., KANG D., OZER S., SILVER D.: A hybrid 3D eddy detection technique based on sea surface height and velocity field. In *Proc. Workshop on Visualisation in Environmental Sciences (EnvirVis)* (2023). doi:10.2312/envirvis.20231101. 2
- [HMvdW\*20] HARRIS C. R., MILLMAN K. J., VAN DER WALT S. J., GOMMERS R., VIRTANEN P., COUNAPEAU D., WIESER E., TAYLOR J., BERG S., SMITH N. J., KERN R., PICUS M., HOYER S., VAN KERKWIJK M. H., BRETT M., HALDANE A., DEL RÍO J. F., WIEBE M., PETERSON P., GÉRARD-MARCHANT P., SHEPPARD K., REDDY T., WECKESSER W., ABBASI H., GOHLKE C., OLIPHANT T. E.: Array programming with NumPy. *Nature 585*, 7825 (Sept. 2020), 357–362. URL: <https://doi.org/10.1038/s41586-020-2649-2>, doi:10.1038/s41586-020-2649-2. 8
- [JSB\*23] JAIN T., SINGH V., BODA V. K., SINGH U., HOTZ I., VINAYACHANDRAN P., NATARAJAN V.: pyParaOcean: A System for Visual Analysis of Ocean Data. In *Workshop on Visualisation in Environmental Sciences (EnvirVis)* (2023), Dutta S., Feige K., Rink K., Zeckzer D., (Eds.), The Eurographics Association. doi:10.2312/envirvis.20231100. 2
- [KNR\*07] KUMAR S. P., NUNCIO M., RAMAIAH N., SARDESAI S., NARVEKAR J., FERNANDES V., PAUL J. T.: Eddy-mediated biological productivity in the Bay of Bengal during fall and spring intermonsoons. *Deep Sea Research Part I: Oceanographic Research Papers 54*, 9 (2007), 1619–1640. 5
- [LJP\*19] LI S., JAROSZYNSKI S., PEARSE S., ORF L., CLYNE J.: Vapor: A visualization package tailored to analyze simulation data in earth system science. *Atmosphere 10*, 9 (2019), 488. 2
- [LLC\*03] LI J., LIAO W., CHOUDHARY A., ROSS R., THAKUR R., GROPP W., LATHAM R., SIEGEL A., GALLAGHER B., ZINGALE. M.: Parallel netCDF: A high-performance scientific I/O interface. In *Proc. ACM/IEEE Conference on Supercomputing* (2003), p. 39. doi:10.1109/SC.2003.10053. 6
- [MAIS16] MATSUOKA D., ARAKI F., INOUE Y., SASAKI H.: A new approach to ocean eddy detection, tracking, and event visualization—application to the northwest pacific ocean. *Procedia Computer Science 80* (2016), 1601–1611. 2, 6
- [McW90] MCWILLIAMS J. C.: The vortices of two-dimensional turbulence. *Journal of Fluid mechanics 219* (1990), 361–385. 6
- [McW08] MCWILLIAMS J. C.: The nature and consequences of oceanic eddies. *Ocean modeling in an eddying regime 177* (2008), 5–15. 1
- [MJD\*99] MCNEIL J., JANNASCH H., DICKEY T., MCGILLICUDDY D., BRZEZINSKI M., SAKAMOTO C.: New chemical, bio-optical and physical observations of upper ocean response to the passage of a mesoscale eddy off bermuda. *Journal of Geophysical Research: Oceans 104*, C7 (1999), 15537–15548. 1
- [myO23] Copernicus myOcean. <https://marine.copernicus.eu/access-data/ocean-visualisation-tools>, 2023. [Online; accessed 28-April-2023]. 2
- [NL15] NOBRE C., LEX A.: OceanPaths: Visualizing multivariate oceanography data. In *Proceedings of the Eurographics Conference on Visualization (EuroVis 2015) - Short Papers* (2015), The Eurographics Association. doi:10.2312/eurovisshort.20151124. 2
- [Oku70] OKUBO A.: Horizontal dispersion of floatable particles in the

- vicinity of velocity singularities such as convergences. *Deep sea research and oceanographic abstracts* 17, 3 (1970), 445–454. 6
- [PBI04] PARK S., BAJAJ C., IHM I.: Visualization of very large oceanography time-varying volume datasets. In *Computational Science-ICCS 2004: 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part II 4* (2004), Springer, pp. 419–426. 2
- [pyF23] pyferret. <https://ferret.pmel.noaa.gov/Ferret/>, 2023. [Online; accessed 28-April-2023]. 2
- [pyP24] pyParaOcean. [https://bitbucket.org/vgl\\_iisc/pyparaocean/](https://bitbucket.org/vgl_iisc/pyparaocean/), 2024. [Online; accessed 19-June-2024]. 16
- [RD90] REW R., DAVIS G.: Netcdf: an interface for scientific data access. *IEEE Computer Graphics and Applications* 10, 4 (1990), 76–82. doi:10.1109/38.56302. 3
- [Rob10] ROBINSON I. S.: Mesoscale ocean features: Eddies. *Discovering the Ocean from Space: The unique applications of satellite oceanography* (2010), 69–114. URL: [https://doi.org/10.1007/978-3-540-68322-3\\_3](https://doi.org/10.1007/978-3-540-68322-3_3). 1
- [Ros89] ROSENBLUM L. J.: Visualizing oceanographic data. *IEEE computer graphics and applications* 9, 3 (1989), 14–19. 1
- [RVBN19] RATH S., VINAYACHANDRAN P., BEHARA A., NEEMA C.: Dynamics of summer monsoon current around sri lanka. *Ocean Dynamics* 69 (2019), 1133–1154. 11
- [Sar13] SARMIENTO J. L.: Ocean biogeochemical dynamics. In *Ocean Biogeochemical Dynamics*. Princeton university press, 2013. 5
- [Sch19] SCHULZWEIDA U.: CDO user guide, 2019. URL: <https://doi.org/10.5281/zenodo.3539275>, doi:10.5281/zenodo.3539275. 4
- [SDVN22] SINGH U., DHIPU T. M., VINAYACHANDRAN P. N., NATARAJAN V.: Front and skeleton features based methods for tracking salinity propagation in the ocean. *Computers & Geosciences* 159 (2022), 104993. doi:<https://doi.org/10.1016/j.cageo.2021.104993>. 2, 5, 7
- [Ser18] SERVICE E. U.-C. M.: *GLORYS12V1 : Global Ocean Physics Reanalysis [Dataset]*. Mercator Ocean International, 2018. URL: <https://doi.org/10.48670/moi-00021>, doi:10.48670/moi-00021. 3
- [SH94] SONG Y., HAIDVOGEL D.: A semi-implicit ocean circulation model using a generalized topography-following coordinate system. *Journal of Computational Physics* 115, 1 (1994), 228–244. 4
- [SM05] SHCHEPETKIN A. F., MCWILLIAMS J. C.: The regional oceanic modeling system (roms): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean modelling* 9, 4 (2005), 347–404. 4
- [SML06] SCHROEDER W., MARTIN K., LORENSEN B.: *The Visualization Toolkit (4th ed.)*. Kitware, 2006. 4
- [SVN24] SINGH U., VINAYACHANDRAN P., NATARAJAN V.: Advection-based tracking and analysis of salinity movement in the indian ocean. *Computers & Geosciences* 182 (2024), 105493. 2
- [TFL\*17] TIERNY J., FAVELIER G., LEVINE J. A., GUEUNET C., MICHAUX M.: The Topology Toolkit. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2017), 832–842. 6
- [VCMN04] VINAYACHANDRAN P. N., CHAUHAN P., MOHAN M., NAYAK S.: Biological response of the sea around Sri Lanka to summer monsoon. *Geophysical Research Letters* 31, 1 (2004). URL: <https://doi.org/10.1029/2003gl018533>. 11
- [VMRB02] VINAYACHANDRAN P., MURTY V., RAMESH BABU V.: Observations of barrier layer formation in the bay of bengal during summer monsoon. *Journal of Geophysical Research: Oceans* 107, C12 (2002), SRF–19. 13
- [VY98] VINAYACHANDRAN P. N., YAMAGATA T.: Monsoon response of the sea around Sri Lanka: generation of thermal domes and anticyclonic vortices. *Journal of Physical Oceanography* 28, 10 (1998), 1946–1960. 11
- [XLWD19] XIE C., LI M., WANG H., DONG J.: A survey on visual analysis of ocean data. *Visual Informatics* 3, 3 (2019), 113–128. 2, 3