

pyParaOcean: A Visualization plugin in Paraview for Oceanographers

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
Master of Technology
IN
Faculty of Engineering

BY
Boda Vijay Kumar



Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

June, 2022

Declaration of Originality

I, **Name**, with SR No. **SR-No** hereby declare that the material presented in the thesis titled

pyParaOcean: A Visualization plugin in Paraview for Oceanographers

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2020-2022**.

With my signature, I certify that:

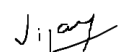
- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: June 29, 2022


Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Prof. Vijay Natarajan


Advisor Signature

© Boda Vijay Kumar
June, 2022
All rights reserved

DEDICATED TO

The Student Community

who can use and reuse this template to glory

Acknowledgements

First of all, I would like to express my gratitude to Prof. Vijay Natarajan of the Visualization and Graphics Lab (VGL), under the Department of Computer Science and Automation, IISc Bangalore, for his early involvement in this journey and for providing the necessary guidance and support at every stage. He offered to mentor and directed me with grace and enthusiasm. His persistent questions about every aspect of my work and our frequent conversations have helped me become a better researcher.

I am very appreciative of the great support I received from my friend Upkar Singh, of the VGL Lab. The progress of my work picked up an incredible pace, thanks partly to the countless discussions and knowledge-sharing sessions I had with him.

I would finally like to express my gratitude to my friends and all the loved ones who were always willing to provide a patient ear when I needed it and constant encouragement to bring this work to the finish line.

Abstract

Ocean scientists working with various simulation models and observational datasets rely on interactive visualizations to help them analyze. Scientists face many challenges in effectively communicating the data, especially when dealing with multivariate data. The reason is that the dataset can become large and multidimensional since Ocean data is multivariate. It can have scalar fields such as salinity, temperature, pressure, etc., vector fields like velocity through space and time, and the availability of various simulation models. There are many visualization tools available for visualizing the data. Since those tools are not built for specific tasks relevant to oceanography, the usage of the tool for domain experts and scientists will be challenging. This project aims to build a plugin in ParaView to create visualizations of the multivariate, temporal oceanography data and help gain insights.

Contents

Acknowledgements	i
Abstract	ii
Contents	iii
List of Figures	vi
List of Tables	viii
1 Introduction	1
2 Related Work	2
3 Problem Definition	3
4 Paraview	4
5 Datasets	6
5.1 Red Sea	6
5.2 Bay Of Bengal	6
6 Terminology	7
6.1 Vector Field Visualization	7
6.1.1 Line Integral Convolution	7
6.1.2 Particle Tracing	8
6.1.3 Streamlines and Pathlines	8
6.2 Volume rendering	9
6.3 Parallel Coordinates	9

7	pyParaOcean: Ocean Data visualization tool	10
8	Use Cases	13
8.1	RED SEA DATASET	13
8.1.1	Volume Rendering	13
8.1.2	Particle Tracer Uniform	15
8.1.3	Particle Tracer Vortex	16
8.1.4	Interactive Particle Path	17
8.1.5	Data Analysis	19
8.1.6	Streamlines	20
8.2	BAY OF BENGAL DATASET	22
8.2.1	Volume Rendering	22
8.2.2	Particle Tracer Uniform	22
8.2.3	Particle Tracer Vortex	23
8.2.4	Interactive Particle Path	24
8.2.5	Data Analysis	24
8.2.6	Salinity Tracking using Front	26
9	Performance	27
10	Credit authorship contribution statement	28
11	User Manual	29
11.1	Installation	29
11.2	Load	29
11.3	Usage	30
11.3.1	AnalysisFilter	30
11.3.2	InteractiveAnalysisFilter	32
11.3.3	Uniform Seeds Filter	32
11.3.4	Vortex Seeds Filter	33
11.3.5	Track Graph Filter	34
11.3.6	Interactive Track Path Filter	34
11.3.7	Interactive Particle Path	34
11.3.8	Delete Downstream	35
11.4	Remote Connection	35
11.5	How to Create a Filter in Paraview?	36

Bibliography

38

List of Figures

4.1	Paraview Architecture in Standalone mode	5
4.2	Paraview Architecture in client-server mode	5
6.1	LIC of Red Sea Dataset	8
6.2	Difference of Streamlines and Pathlines in a flow	9
7.1	Architecture of the pyParaOcean	10
7.2	Snapshot of the ParaView	11
8.1	Selecting the Volume Representation	13
8.2	Volume Rendering of Salinity	14
8.3	Volume rendering of Temperature	14
8.4	Selecting the Particle Tracer Uniform filter	15
8.5	Visualisation of Particle Tracer Uniform Filter	16
8.6	Selecting the Particle Tracer Vortex filter	17
8.7	Visualisation of Particle Tracer Vortex Filter	18
8.8	Selecting the Interactive Particle Path filter	18
8.9	Visualisation of Interactive Particle Path Filter	19
8.10	Selecting the Analysis filter	20
8.11	Visualisation of Analysis filter	21
8.12	Visualisation of Streamlines filter	21
8.13	Volume Rendering of Salinity	22
8.14	Visualisation of Particle Tracer Uniform Filter	23
8.15	Visualisation of Particle Tracer Vortex Filter	24
8.16	Visualisation of Interactive particle Path Filter	25
8.17	Data Analysis Filter for BoB dataset	25
8.18	Track graphs of Salnity	26

LIST OF FIGURES

11.1 Snapshot of Plugin Manager with pyParaOcean filter loaded	30
11.2 Snapshot of Analysis Filter for Bob data set at coordinates (85.5,5)	31
11.3 Selection of variables for Parallel Coordinates	31
11.4 Snapshot of Analysis Filter for Bob data set at coordinates (85.5,5)	32
11.5 Snapshot of Uniform Seeds Filter for Bob data set	33
11.6 Flow chart of UniformSeeds Filter	33
11.7 Visualisation of Interactive particle Path Filter	34

List of Tables

Chapter 1

Introduction

Oceanographers and domain experts deal with various data analysis tasks that help them understand ocean data. The data generation process can either use observational sensors or simulation models. So for a better understanding of the data, oceanographers have to use data sets from different simulation models and analyze the available features. Many application areas, such as tourism, fisheries, weather prediction, etc., depends on the output of the simulation models of the ocean. The challenging task is ocean data is multivariate and has a temporal domain; as a result, it will be large. There are many visualization tools that help in understanding the data, but oceanographers work with diverse tasks and need a tool specific to the ocean domain. A survey by Tominski et al.[13] has concluded that even with the availability of many advanced tools for visualization, oceanographers rarely used those. The aim of this project is to build a tool that is specific to ocean data sets, i.e. Spatio-temporal, multivariate data.

Chapter 2

Related Work

There are many tools that are built based on the requirements of oceanographers.

Vimtex[6] is a visual analytics system of coordinated, linked views to study multivariate geology datasets to understand temporal patterns and behaviour of different chemical species.

Afzal et al. [7] built the RedSeaAtlas, a web-based visual analytics tool that shows the wind dataset and associated attributes. It supports interactive charts; when clicked at the location, it shows detailed information.

(Nobre et al.)[8] OceanPaths supports interactive visual analysis of multivariate oceanography datasets by defining pathways along the currents and enabling Spatio-temporal analysis of variations in water properties.

(Wael H. Ali et al.)[4] SeaVizKit is an interactive multiscale visualization tool implemented in D3.js that allows users to view different ocean variables and different depths and timestamps.

(Sullivan)[11] PVGeo is an open-source Python package for geoscientific visualization in VTK and ParaView. It is ready to use standard python library, and it is powered by VTK (Schroeder et al., 2006)[9] and all the functionalities are provided as the plugins for ParaView.

Chapter 3

Problem Definition

The oceanographers and people from research, solving problems related to climate change, tourism planning, identifying fishing zones, and studying the oceans' behaviour want to analyze and visualize data interactively. Tools like ParaView are helpful in visualizing the data, and It is also interactive. Direct volume rendering is easy in ParaView, but it doesn't fulfil the requirements of oceanographers. One has to create a pipeline of filters in order to achieve some task. It can be challenging work for them. Based on existing literature, I have identified the requirements of domain experts and implemented them as features for the plugin. There is a need for such a system as all the features are not available in the integrated environment.

- Interactive Volume Rendering.
- Feature Selection and Explore scalars through time and space.
- Identifying patterns in the dataset, Comparison of multiple scalar fields(eg. salinity, temperature etc.)
- Understanding the transport of the particles over time with different seeding strategies
- Understanding the flow/ velocity field using streamlines, LIC
- Parallel coordinates for comparing, identifying patterns, anomalies in the data
- Interactive visualization of particle pathlines using parallel coordinates
- Front and Skeleton Features Based Methods for Tracking Salinity Propagation in the Ocean
- Plots of scalar values over Depth in multiple views.

Chapter 4

Paraview

Paraview^[3] is an Open source visualisation software. It has many built-in visualisation filters, and it supports creating custom filters using the programmable filter in python, and XML, which makes ParaView more flexible with any tasks. It has a custom transfer function in which one can edit, import, and export.

In ParaView, a pipeline module can do one of two things: produce data(source) or handle input data(filter). The module may utilise a mathematical model to generate data. Data processing comprises changing incoming data into a new output using specified operations. Many users' needs are met by ParaView's wide variety of readers, data sources, and filters. ParaView provides a mechanism for adding new modules via plugins in scenarios where the available collection does not meet your user needs. The plugins are created by utilising the APIs of VTK. Those plugins can be implemented in python and later import the sources/filters created in the form of plugins. Users or Developers have access to Python packages such as NumPy, which provide various numeric operations helpful for data transformation because the scripts are ordinary Python scripts. The pipeline of filters can be converted into a python script and saved as Macro inside ParaView.

ParaView is designed as a three-tier client-server architecture. Data Server, Render Server, Client. The data server is responsible for all data-related tasks like reading, writing, filtering etc. Render Server is for rendering. The client is for visualisation. The client manages the creation, execution, and destruction of objects on the servers, but it does not contain the data. Figure4.1 is the architecture of standalone mode. The client, data server, and render Server merge into a single serial application in standalone mode. When the user runs the ParaView, it instantly connects to a built-in server, allowing the user to use all of ParaView's features.

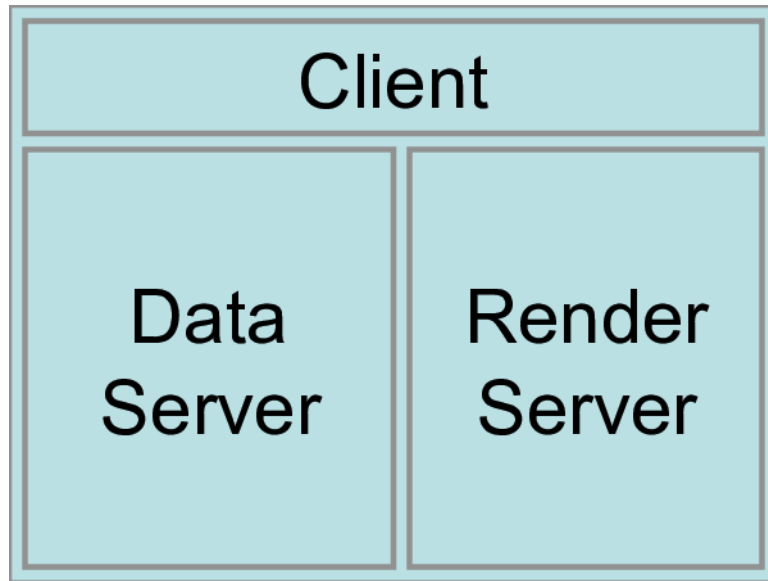


Figure 4.1: Paraview Architecture in Standalone mode

Figure 4.2 is architecture of client-server mode. User runs the *pserver* software on a parallel machine and connects it to the *paraview* client application in client-server mode. Because the *pserver* includes both a data server and a render server, it handles both data processing and rendering. Data flow over this connection is minimised since the client and server are connected via a socket, which is thought to be a somewhat slow means of communication. Paraview also supports multiprocessing so that users can use supercomputing resources when needed.

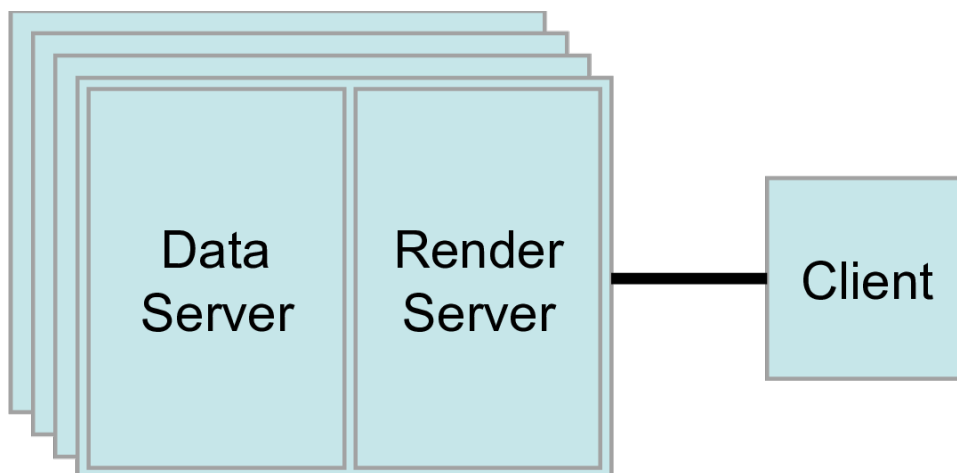


Figure 4.2: Paraview Architecture in client-server mode

Chapter 5

Datasets

5.1 Red Sea

Red Sea data[14] is taken from SciVis Contest2020. The dataset consists of 50 time-dependent ensemble members with three-dimensional scalar and velocity fields. The data representation is on the regular grid of dimensions 500x500x50, with 60-time steps covering a whole month of simulation time. Ensembles are the outputs of the simulated models with different parameters and initial conditions, and they vary drastically even with a slight change of parameters. Data set are the forecasts of MITgcm setups configured for the 30°E - 50°E and 10°N- 30°N covering the complete Red Sea. They are implemented in Cartesian coordinates with a horizontal resolution of 0.04°x 0.04°(4km) and 50 vertical layers, with a surface spacing of 4m and a bottom spacing of 300m. The dataset format is NETCDF, and each ensemble member is of size 30 GB(uncompressed).

5.2 Bay Of Bengal

The Bay of Bengal dataset is reanalysis data from the Nucleus for European Modelling of the Ocean (NEMO) repository [12], with a daily resolution of 122-time steps for June 2016 to September 2016. The data is in NetCDF format, with a 1/12°latitude-longitude resolution. Salinity measurements are available at 50 vertical levels, ranging from 1m near the surface to 450m towards the sea floor, including 22 samples in the upper 100m. The Bay of Bengal, a geographical region confined by longitudes 75°E and 96°E, latitudes 5°S to 30°N, and 200 m depth, is extracted from this data.

Chapter 6

Terminology

6.1 Vector Field Visualization

Vector fields are the representation of the flow. Many scientific simulations include vector data in addition to scalar variables. These variables are two or more independent scalar fields that describe a 2D or 3D vector array when combined. Vector fields have both magnitude and direction.

6.1.1 Line Integral Convolution

LIC [5] is one of the popular vector visualization methods that shows the actual structure of the vector field. Working behind LIC takes a vector field on the cartesian grid and a white noise texture as input and outputs the dense visualization of the flow. The texture is locally convoluted along the streamlines to generate the output. The figure shows the LIC view of the red sea dataset. Transfer function applied according to the velocity magnitude. Figure 6.1 shows the Surface LIC of the Red Sea data set. The rainbow desaturated transfer function is used on the LIC to focus on the higher velocity magnitudes. Observations are that the white circled regions showing red indicates the higher speed. The swirling part is called eddies.

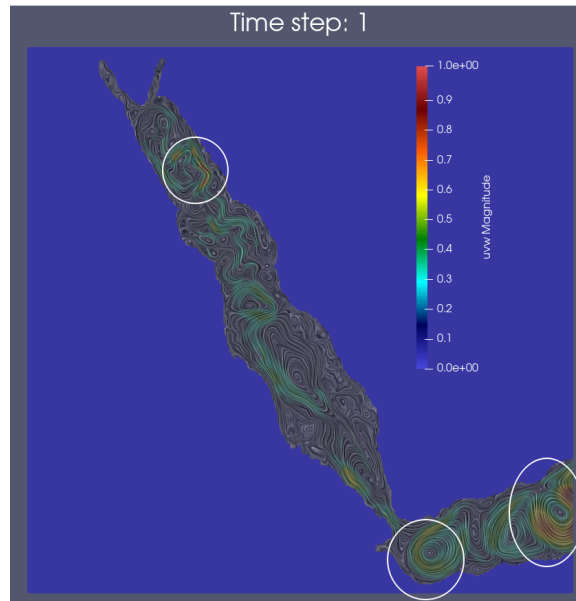


Figure 6.1: LIC of Red Sea Dataset

6.1.2 Particle Tracing

Particle tracing is a visualization technique that tracks the particle's position from one time step to another. In terms of implementation, we will use the Particle tracer filter in Paraview. We have to choose the seed positions and create points around each seed. Those points act as particles.

6.1.3 Streamlines and Pathlines

Streamlines are another powerful flow visualization technique. It is a line everywhere tangent to the velocity vector. They are the curves that connect the positions of particles in the vector field over a single time step. They are helpful when we want to find the direction of flow of a particle at a single time step. Paraview supports Streamline filter on a vector field. Before applying this filter, we need to select the seed position from which streamlines evolve. Challenges with this are they can easily create clutter in the visualization Pathlines are the lines that track actual path of the particle. ParticlePath filter in Paraview is applied on the velocity vector. Figure 6.2[1] shows the difference between the streamlines and Pathlines in a flow.

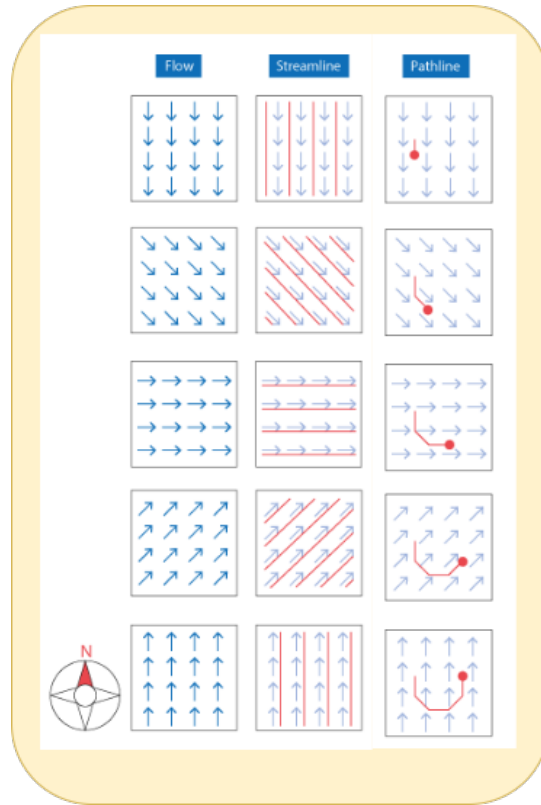


Figure 6.2: Difference of Streamlines and Pathlines in a flow

6.2 Volume rendering

Volume rendering is a classic scalar visualization technique that visualizes an entire 3D data set by rendering all data points with a transfer function based on colour and opacity. It requires the type of data to be structured or unstructured, but it works for different file types. It is available as a representation in Paraview. Paraview uses projected tetrahedra as a default Volume rendering algorithm. It also supports z sweep and ray casting. For the structured grid, it renders very fastly.

6.3 Parallel Coordinates

Parallel coordinates are the common way of analyzing and visualizing the higher-dimensional datasets. For an n-dimensional dataset, n- evenly spaced vertical lines are drawn. Each line represents one axis of data and is connected to the other. The most important feature is brushing(selecting the subset) of the features. Scatter Plots are very informative in bi-variate cases, but when data is multivariate, they won't help much. Parallel Coordinates are one of the best ways to visualize multivariate features.

Chapter 7

pyParaOcean: Ocean Data visualization tool

This project aims to help oceanographers and other users create visualizations and gain insights using relevant filters in ParaView. Depending on the requirement and size of the dataset, one can use different modes, either standalone or client-server. The plugin is implemented in python and uses the vtk module for the backend. Figure 7.1 represents the architecture of the tool.

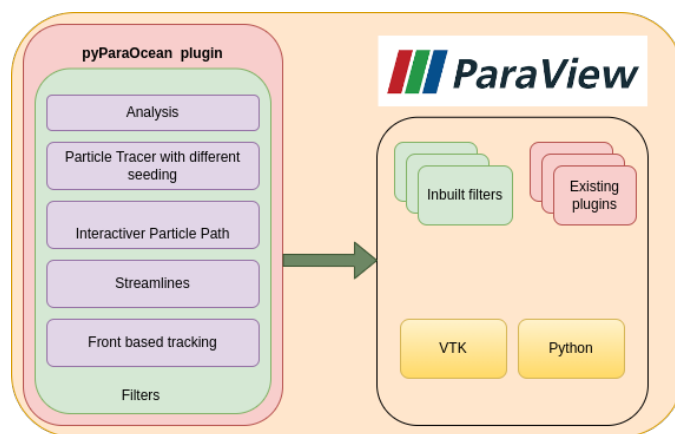


Figure 7.1: Architecture of the pyParaOcean

An snapshot of the ParaView with Bay of Bengal Dataset loaded is shown in Figure 7.2

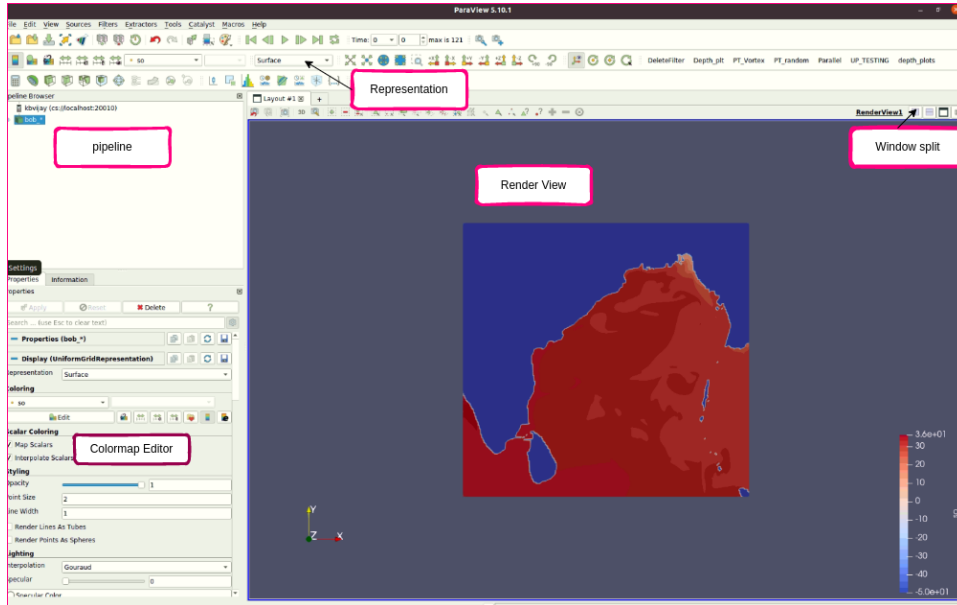


Figure 7.2: Snapshot of the ParaView

Most ocean data sets are available in NETCDF, but the paraview is built on top of VTK. It is preferable to have the format of data be in VTK format. The range of latitude and longitude is bounded, but the depth(z-axis) is not as it is measured in meters. So user cannot view complete data in the view, so transform the view by scaling down the z-axis. The tool supports various kinds of datasets. In order to understand the multivariate characteristics of the ocean data, there is a menu for colourmap, where the user can select desired variables and see the rendering in the RenderView. The Time panel is used for animation and viewing the data for the next time stamps. This helps the oceanographers to investigate the scalar fields in time and space. Paraview efficiently renders the 3D multivariate ocean data with zoom and multi-window capabilities. The interactive hovering of Points feature in paraview helps the user hover points over the region and get the scalar information at that location.

The features implemented in the plugin are Analysis Filter, ParticleTracerUniform Filter, ParticleTracerVortex Filter, Interactive Particle Path Filter, Spherical Gradient Filter, and Salinity Visualization with front features.

Oceanographers are interested in vertical sections of the dataset, i.e. depth slices, as they reveal the most important insights of the features like finding out low and high salinity over the desired location. The *Analysis filter* has three interactive views: slice view, line plot view, and parallel coordinates view. With the help of this filter, the user can view the vertical section of the dataset with a proper transfer function, and a line plot shows the curve, which gives some insights, and parallel coordinates are very helpful in order to select the particular range

of values and observe them in all the other views.

To understand the behaviour of particles over time, strategies like Streamlines and Particle Tracers are used. Seeding plays a crucial role for those. With the *ParticleTracerUniform* and *ParticleTracerVortex Filter* users can observe the impact of vortices with Vortex seeding in particle transport compared to Uniform seeding. Analyzing the path of the particles from selected regions is very useful for understanding the transport of custom range scalar values. Using Parallel Coordinates, the user can brush the custom range of scalars and apply an Interactive particle path filter for this task. The tracks formed by the font-based features describe the movement of the High Salinity core. The tracking methods discussed in the paper [10] are suitable for application to other regions as well for tracking water masses.

Chapter 8

Use Cases

8.1 RED SEA DATASET

8.1.1 Volume Rendering

Load the data using file menu in ParaView and select the desired scalar for colouring and volume in Representation tab. The User Interface of Paraview is shown in

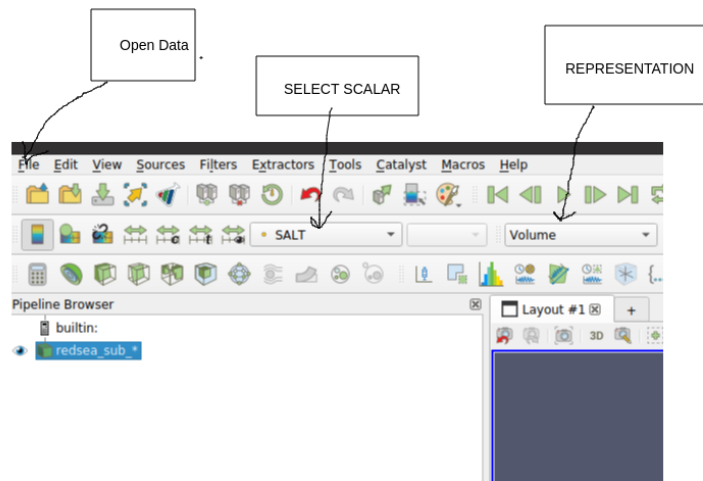


Figure 8.1: Selecting the Volume Representation

Figure 8.2 shows the Volume rendering of salinity. The maximum value for this is approximately 42 PSU which is located in the top end of the red sea, and the Middle of the Red Sea has a range of 36 to 40. and the minimum is 35 PSU which is near the Gulf of Eden(GOE). This is because there is a continuous flow of salt from GOE until the end of the sea, where there is no flow, so we have high salinity.

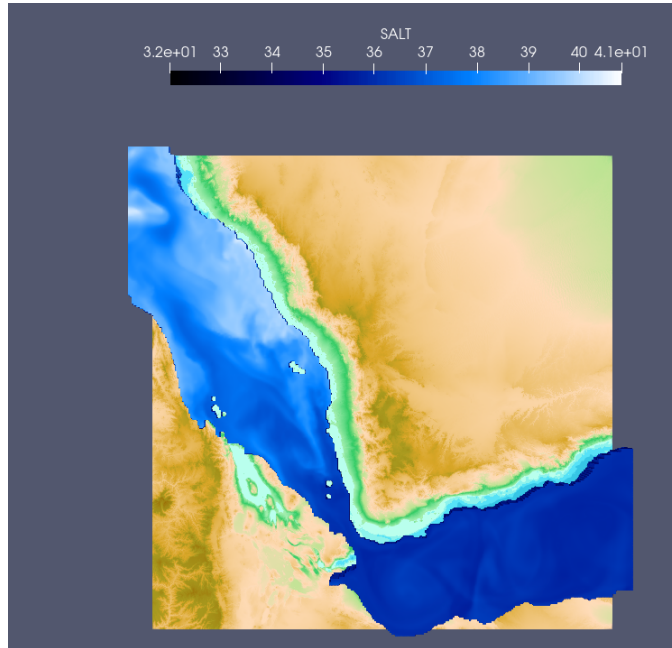


Figure 8.2: Volume Rendering of Salinity

Figure 8.3 shows the Volume rendering of Temperature. The surface of the Red Sea is hotter as compared to the bottom, and also, the middle of the red sea has maximum Temperature (more red in colour).

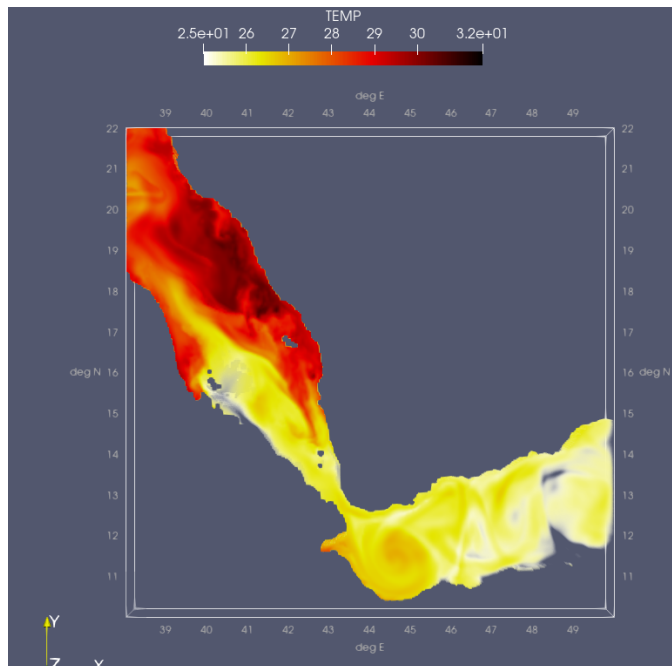


Figure 8.3: Volume rendering of Temperature

8.1.2 Particle Tracer Uniform

Requirement: For the dataset with velocity fields and temporal fields, the user wants to analyse the transport of particles over time

Architecture: Particle Tracer is a filter available in the Paraview. We need to give a seed source as the particles in order to apply this filter to the dataset. Seed, the particles are uniformly distributed all over the region.

Working: Load the data in ParaView using File and Open the dataset. Go to Filters and select the ParticleTracer UniformSeed for Uniform seeding strategy. The procedure of the filter selection is shown in the Figure 8.4

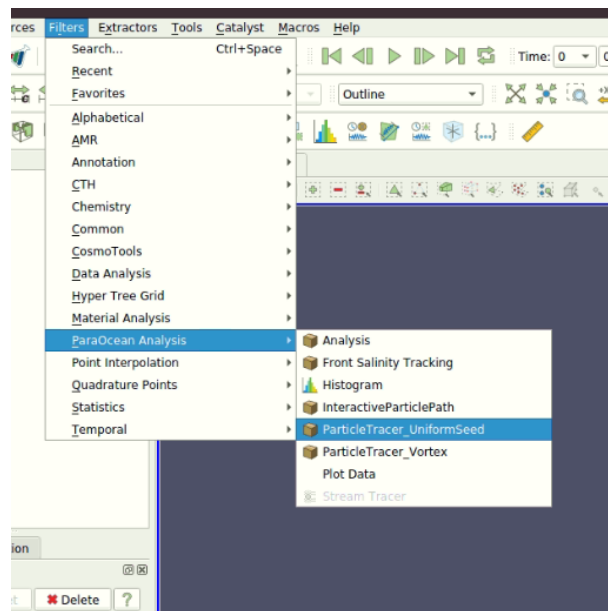


Figure 8.4: Selecting the Particle Tracer Uniform filter

The seeding strategy is very crucial for particle tracking. After applying the filter, the output visualisation of this filter is shown in Figure 8.5a For the first time step. I have shown the outputs of the time steps 1,9,41,60 in Figures 8.5a,8.5b,8.5c,8.5d which are randomly selected to sea transport of particles over time. The cyan colour represents the low salinity values, and the violet colour represents the high salinity values. So the low salinity values come out of the Red Sea and at the bottom of the Red Sea, whereas high salinity particles go inside and at the surface of the Red Sea.

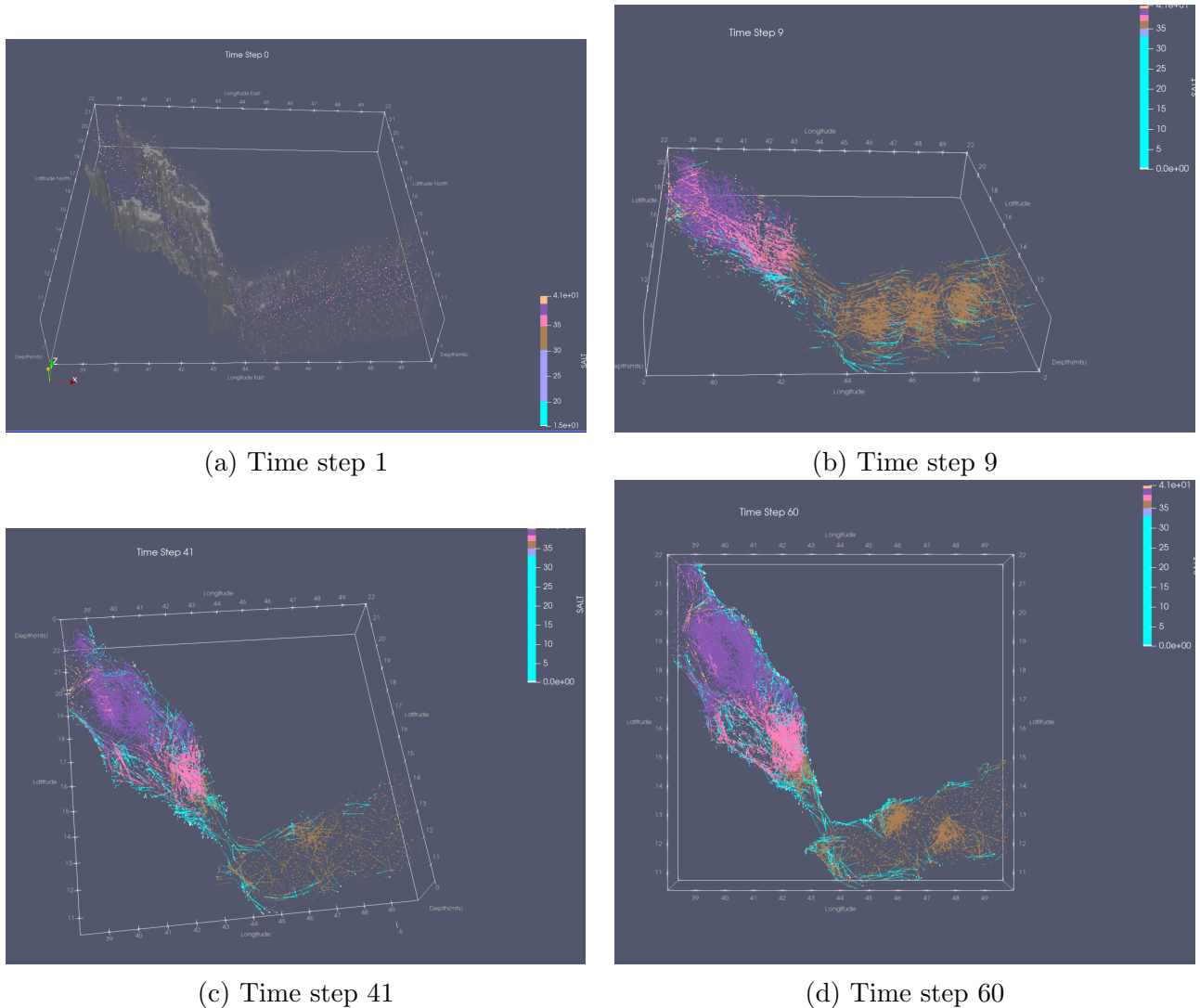


Figure 8.5: Visualisation of Particle Tracer Uniform Filter

8.1.3 Particle Tracer Vortex

Requirement: For the dataset with velocity fields and temporal fields, the user wants to analyse the transport of particles over time

Architecture: Particle Tracer is a filter available in the Paraview. We need to give a seed source as the particles in order to apply this filter to the dataset. Seed the particles in the places where vorticity is high and alongside Seed, some particles uniformly distributed all over the region for good visualisation of particles

Working: Load the data in paraview using File and Open the dataset. Go to Filters and select the ParticleTracer Vortex for vorticity-based seeding . The procedure of the filter selection is

shown in the Figure 8.6

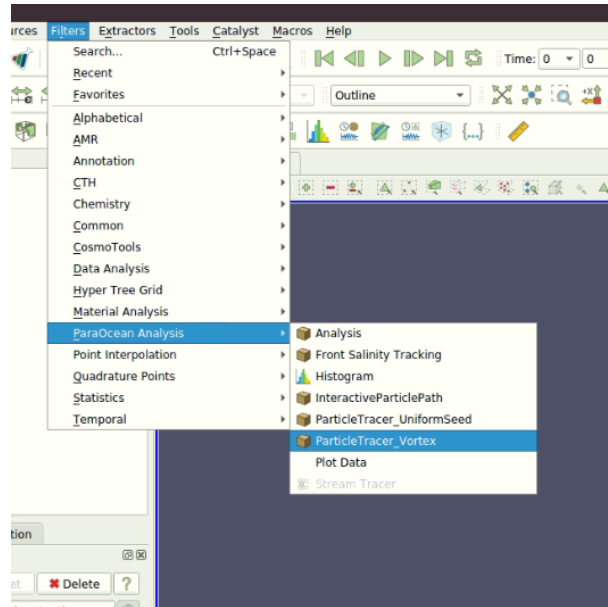


Figure 8.6: Selecting the Particle Tracer Vortex filter

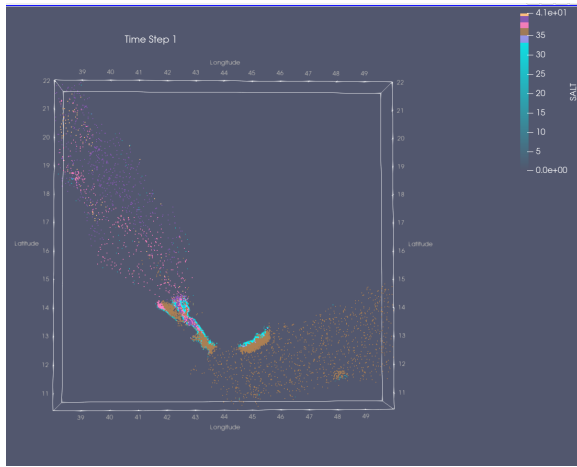
After applying the filter, the output visualisation of this filter is shown in Figure 8.7a For the first time step. I have shown the outputs of the time steps 1,9,41,60 in Figures 8.7a,8.7b,8.7c,8.7d which are randomly selected to sea transport of particles over time. The cyan colour represents the low salinity values, and the violet colour represents the high salinity values. So the low salinity values come out of the red sea and at the bottom of the red sea, whereas high salinity particles go inside and at the surface of the red sea. Using this seeding strategy, we can see how eddies affect the transport of the particles.

8.1.4 Interactive Particle Path

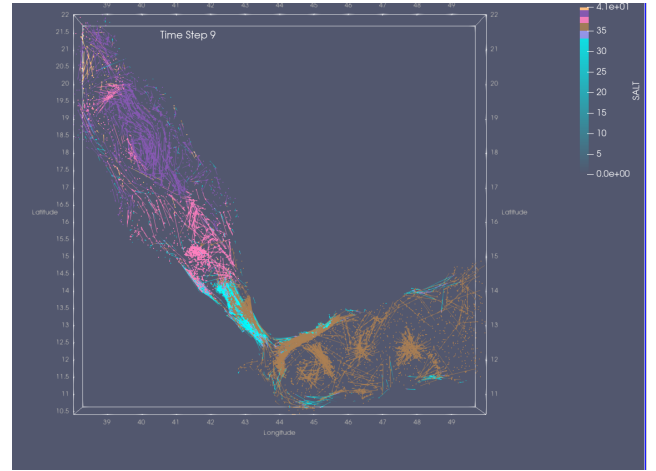
Requirement: For the dataset with velocity fields and temporal fields, the user wants to analyse the pathlines of particles over time

Architecture: Particle Tracer is a filter available in the Paraview. We need to give a seed source as the particles in order to apply this filter to the dataset. Parallel Coordinates for selecting the seed points

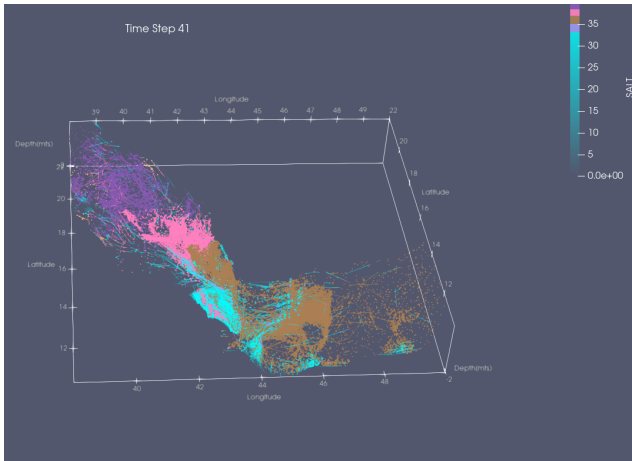
Working: Load the data in paraview using File and Open the dataset. Open Parallel Coordinates(PC) view alongside the render view and select the range of points using PC. Go to Filters and select the Interactive Particle Path . The procedure of the filter selection is shown in the Figure 8.8



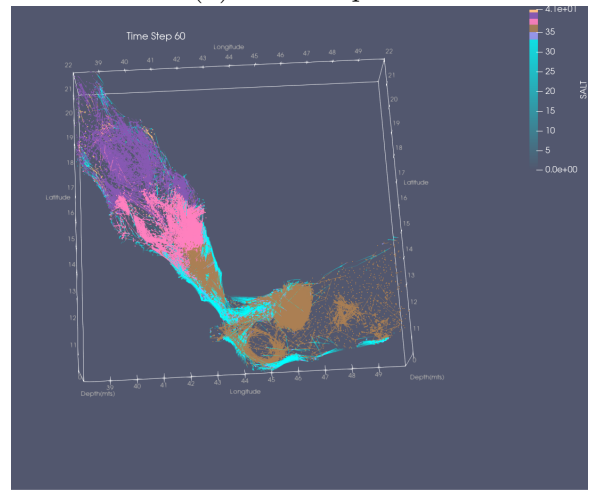
(a) Time step 1



(b) Time step 11



(c) Time step 41



(d) Time step 60

Figure 8.7: Visualisation of Particle Tracer Vortex Filter

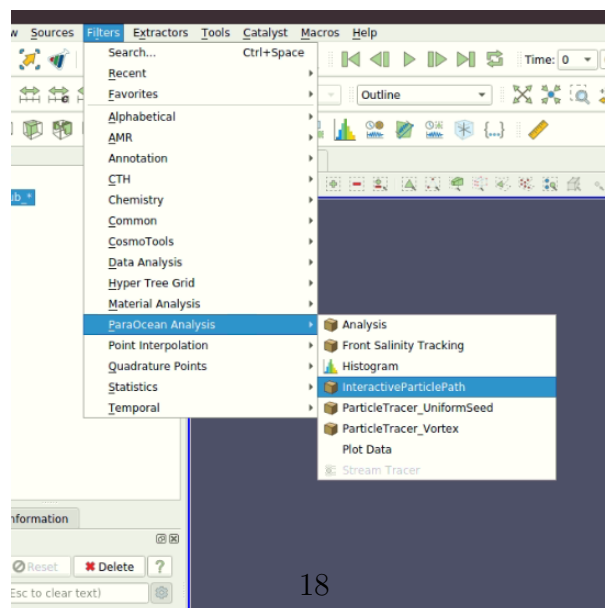


Figure 8.8: Selecting the Interactive Particle Path filter

After applying the filter, the output visualisation of this filter is shown in Figure 8.9. Figure 8.9a shows the selection of particles using parallel coordinates, and Figure 8.9b shows the pathlines of the selected particles. Since I have selected high saline particles, the violet colour pathlines go towards the red sea. Users can interactively select the particles and apply the filter to get the pathlines.

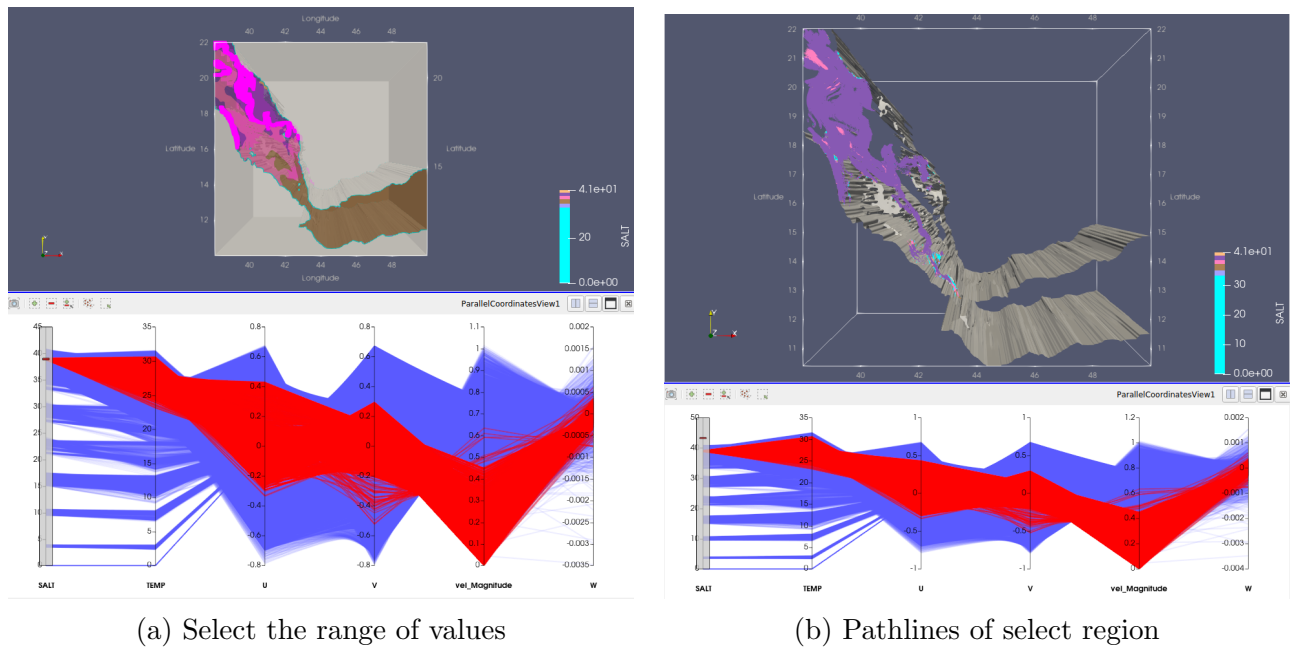


Figure 8.9: Visualisation of Interactive Particle Path Filter

8.1.5 Data Analysis

Requirement: For the dataset with scalar fields(salt, Temperature etc..), the user wants to analyse those scalar fields at desired location point over depth.

Architecture: The depth Slice view helps in doing such analysis. It contains three windows/views in the dashboard. Render View, Line Plot View, Parallel Coordinates. Render View has a slice of the data. Line Plot View has plots of the scalars, and Parallel Coordinates has a selection of multiple variables with the desired range of values.

Working: Load the data in paraview using File and Open the dataset.Go to Filters and select the Analysis . The procedure of the filter selection is shown in the Figure 8.10

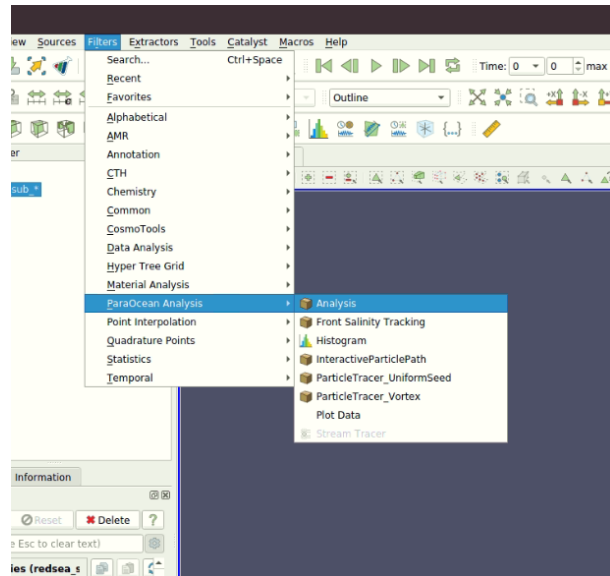


Figure 8.10: Selecting the Analysis filter

After applying the filter, the user needs to give the desired longitude and latitude value to investigate the interest. Figure 8.11 shows the output of the filter. It contains the line plots of salinity and Temperature over depth for the selected lat-long. And Parallel Coordinates are useful when the user wants to see the locations of some range of scalar values. The pink colour is the selected region. The Salinity and Temperature values are decreasing across the depth.

8.1.6 Streamlines

Requirement: For the dataset with scalar fields(salt, Temperature etc.), the user wants to analyse the behaviour of the velocity field

Architecture: Direct usage of streamlines. A high Vortex seeding strategy is used.

Working: Load the data in ParaView using File and Open the dataset. Go to Filters and select the Streamlines.

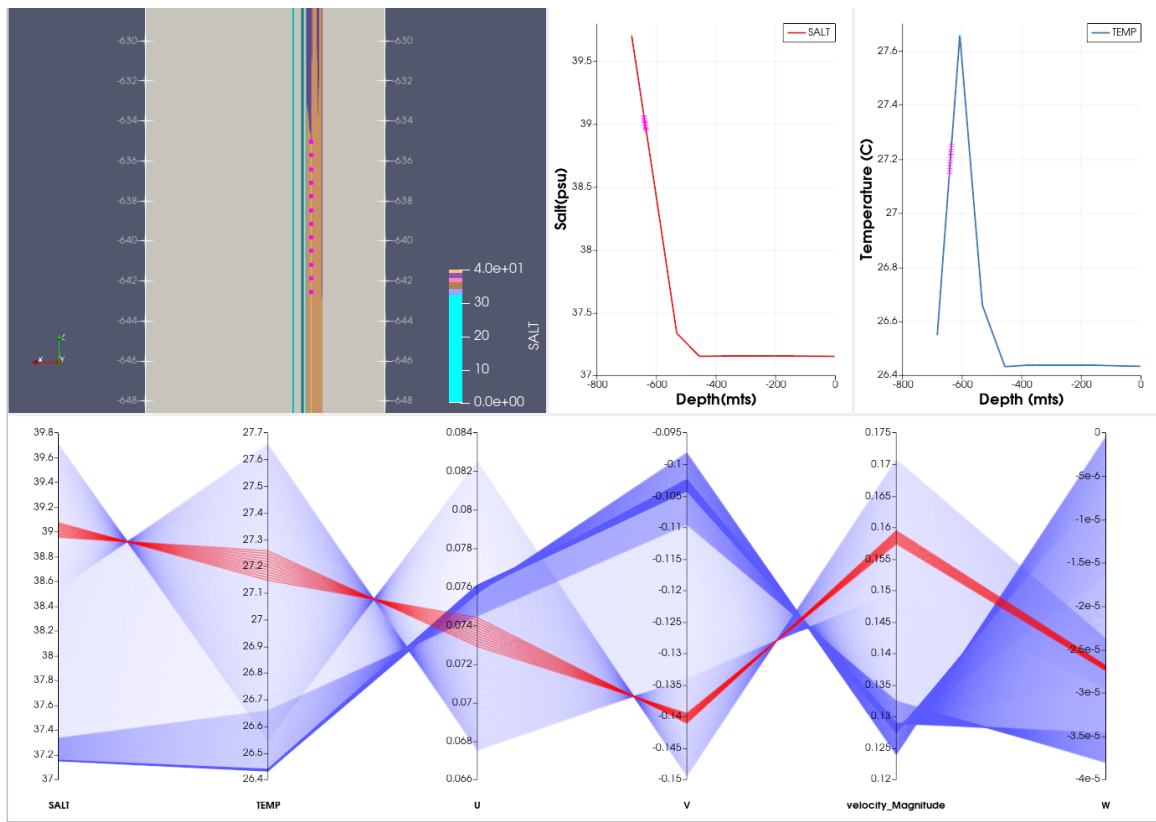


Figure 8.11: Visualisation of Analysis filter

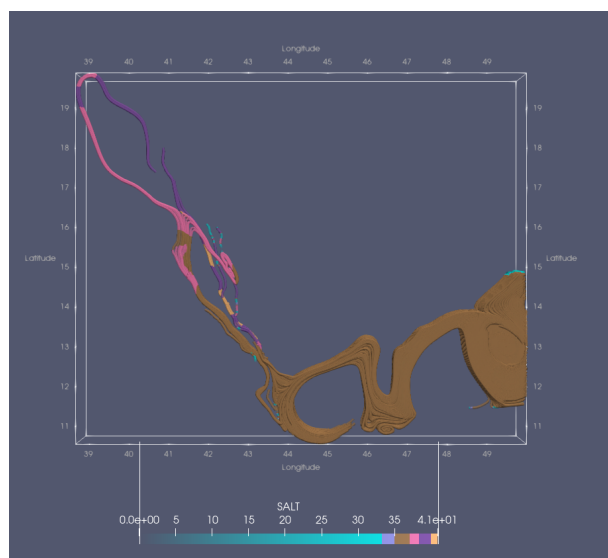


Figure 8.12: Visualisation of Streamlines filter

After applying the filter, we get the streamlines. Figure 8.12 shows the output of the filter.

Since the seeding strategy used is high vortex regions, we can see that streamlines are very from start to end and also, from the transfer function, we can see that those are streams of high salinity regions.

8.2 BAY OF BENGAL DATASET

This section has the same filters applied to the Bay Of Bengal Datasets. So the requirements, architecture and working are the same as the above. The transfer function is applied to salinity for all the filters.

8.2.1 Volume Rendering

The selection of the filters is the same as the Red Sea dataset. The Volume rendering of salinity is shown in Figure 8.13. It shows that High saline regions travel towards the south of the Bay of Bengal. Most of the BoB is coloured in violet coloured. The high saline regions are at the shores.

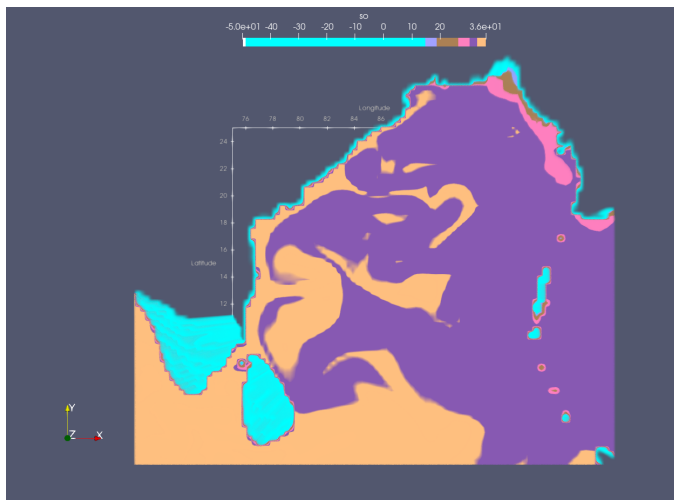
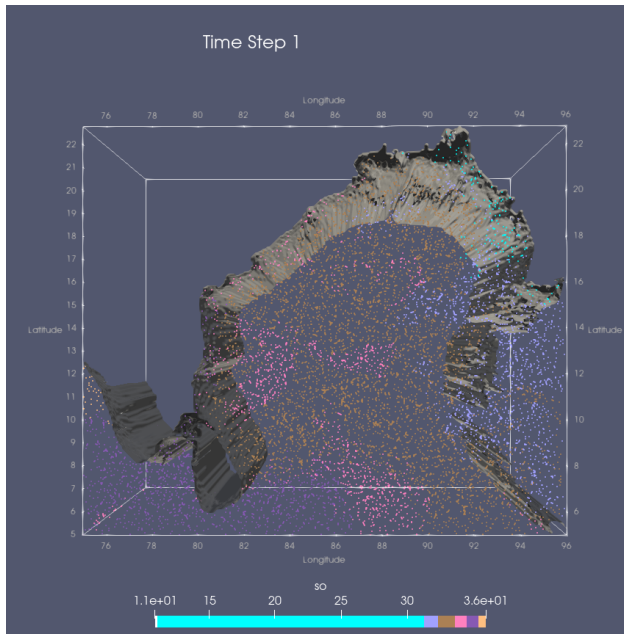


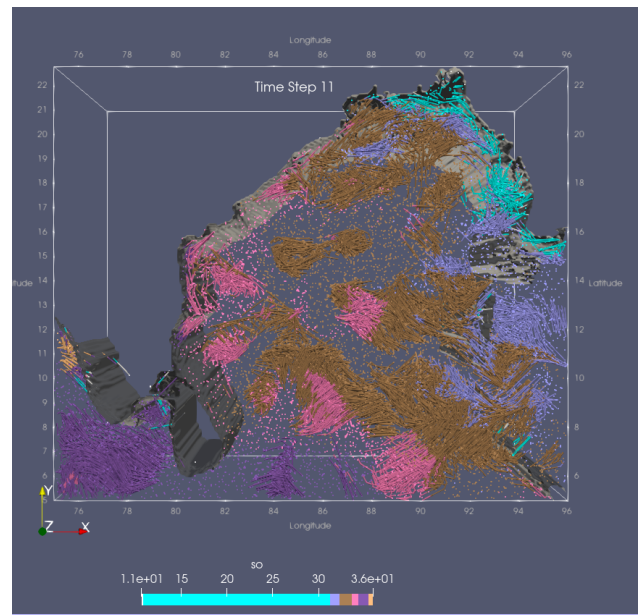
Figure 8.13: Volume Rendering of Salinity

8.2.2 Particle Tracer Uniform

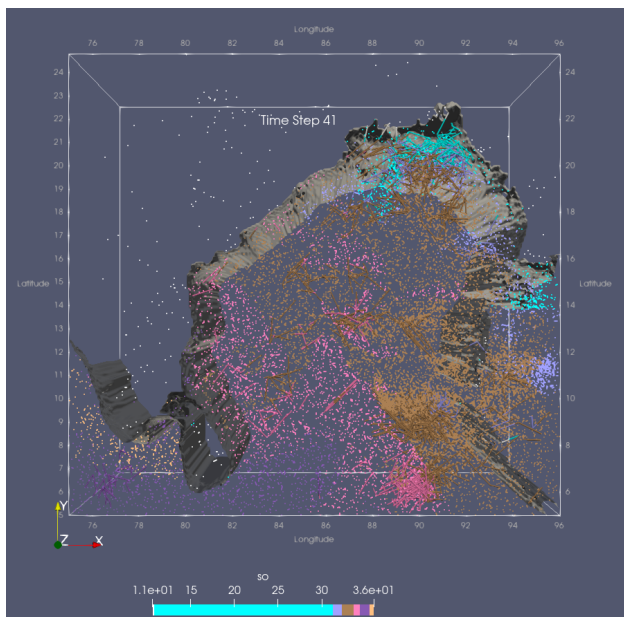
Figure 8.14 shows the output visualization of particle tracer uniform filter for the time steps 1,11,41,122. Using this uniform seeding strategy, we can see that high saline particles move towards the Bay of Bengal. The pink colour shows that salinity is dispersed into clusters and results in the high saline becoming less concentrated, and then the brown colour shows the consistent decrease in the salinity concentration. The initial particles go completely disappear, which tells that salinity concentration over the Bay of Bengal reduces over time.



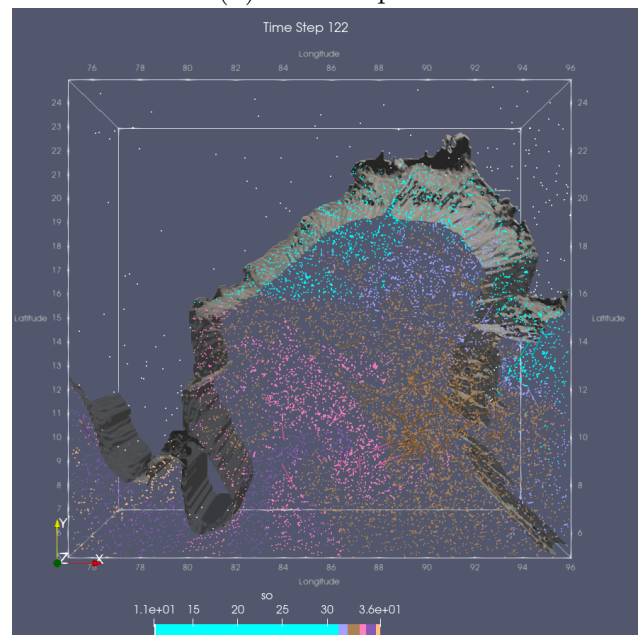
(a) Time step 1



(b) Time step 11



(c) Time step 41



(d) Time step 122

Figure 8.14: Visualisation of Particle Tracer Uniform Filter

8.2.3 Particle Tracer Vortex

Figure 8.15 shows the output visualization of particle tracer vortex filter for the time steps 1,11. Using this vortex seeding strategy, we can see that high saline particles move towards the Bay

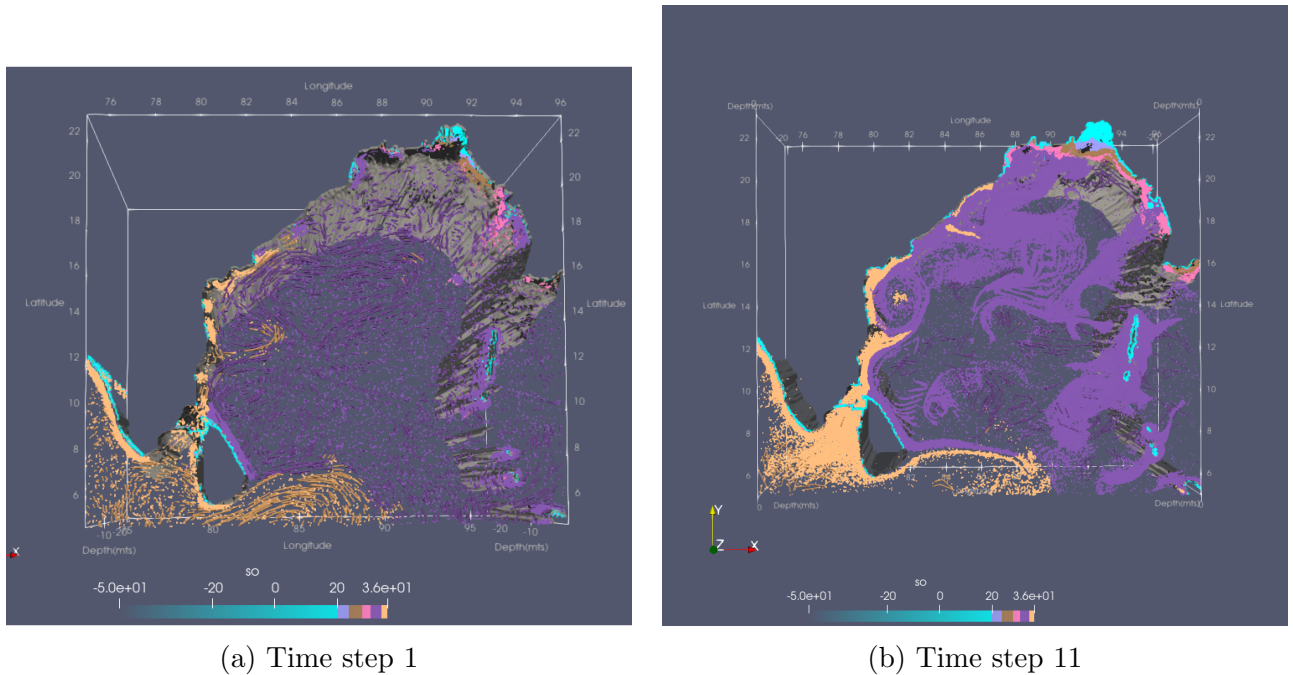


Figure 8.15: Visualisation of Particle Tracer Vortex Filter

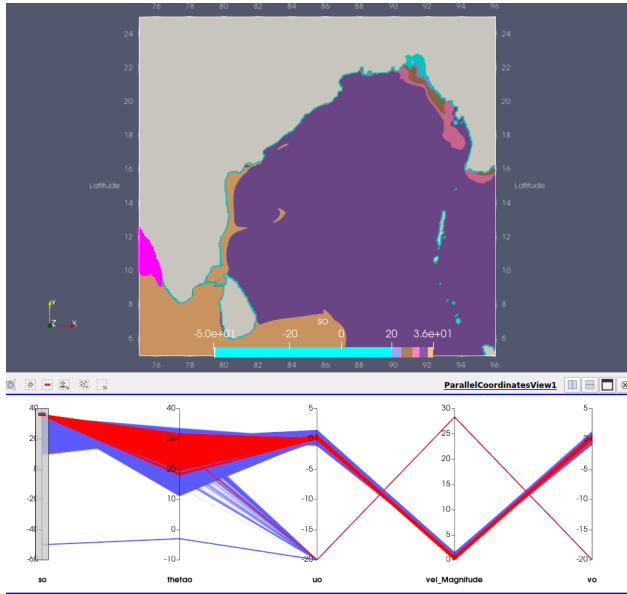
of Bengal. We can see the change in colour to violet and swirling motions in Figure 8.15b. This shows that high vorticity regions affect the movement of particles and all the vortex regions have less salinity. This helps the researchers to study the behavior of velocity fields.

8.2.4 Interactive Particle Path

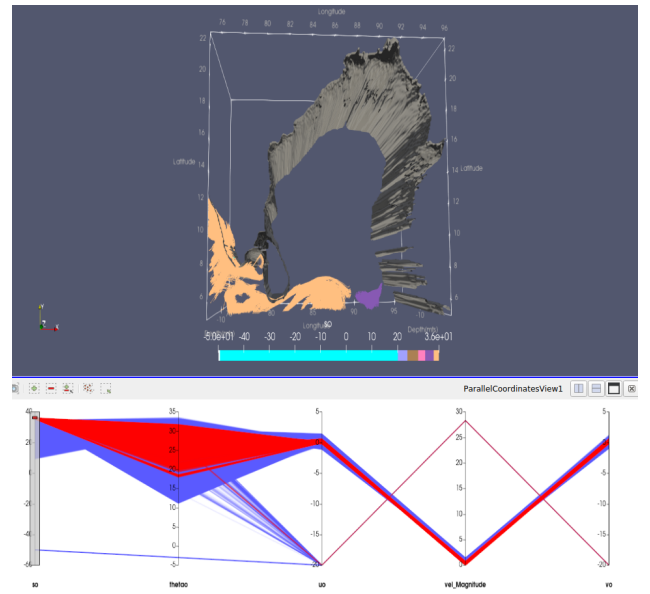
After applying the filter, the output visualisation of this filter is shown in Figure 11.7. Figure 8.16a shows the selection of particles using parallel coordinates, and Figure 8.16b shows the pathlines of the selected particles. Since I have selected high saline particles, the pathlines go towards the Bay of Bengal and eventually result in a decrease in concentration. Users can interactively select the particles and apply the filter to get the pathlines.

8.2.5 Data Analysis

After applying the filter, the user needs to give the desired longitude and latitude value to investigate the interest. Figure 8.17 shows the output of the filter. It contains the line plots of salinity and theta over depth for the selected lat-long. And Parallel Coordinates are useful when the user wants to see the locations of some range of scalar values. The pink colour is the selected region. The Salinity values are decreasing across the depth, whereas the theta is increasing over depth.



(a) Select the range of values



(b) Pathlines of select region

Figure 8.16: Visualisation of Interactive particle Path Filter

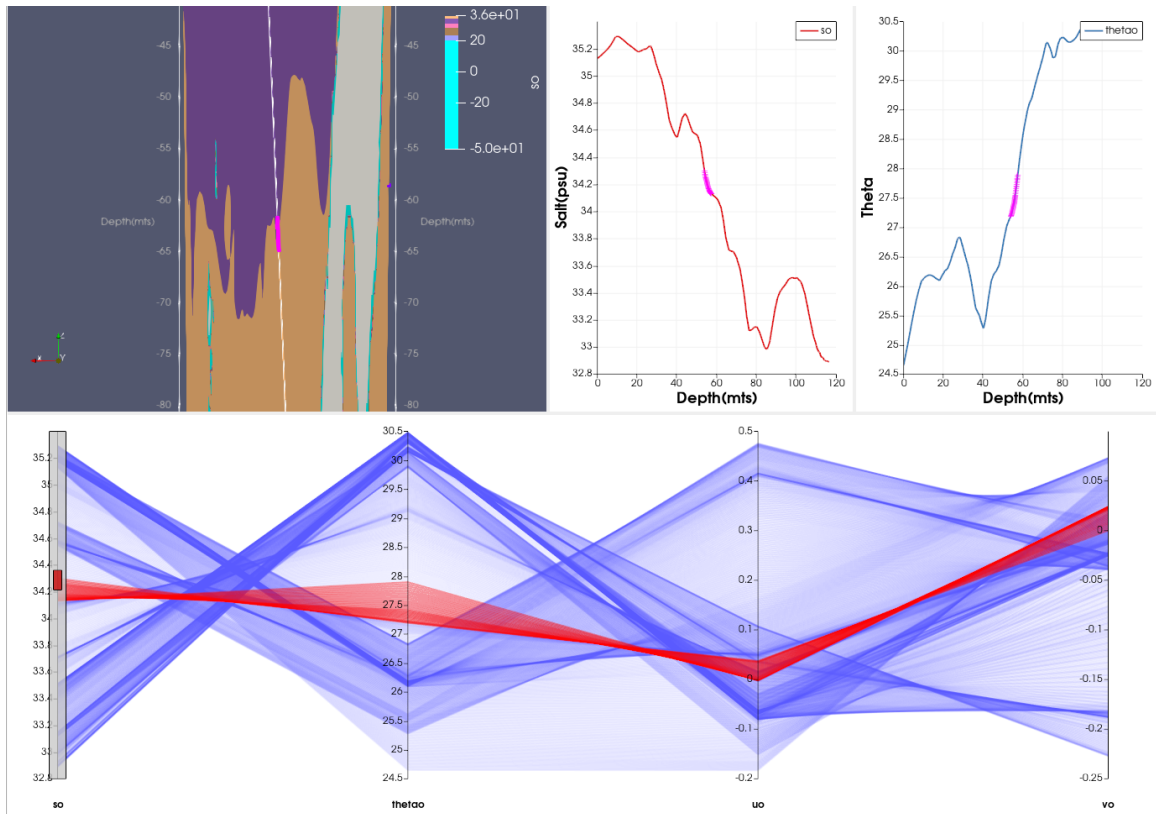


Figure 8.17: Data Analysis Filter for BoB dataset

8.2.6 Salinity Tracking using Front

Requirement: For the dataset with a salinity scalar field, the 3D velocity vector field and have to be temporal.

Architecture: Salinity tracking using Front Features[10].

Working: Load the data in paraview using File and Open the dataset. Go to Filters and select the Front Salinity Tracking Filter.

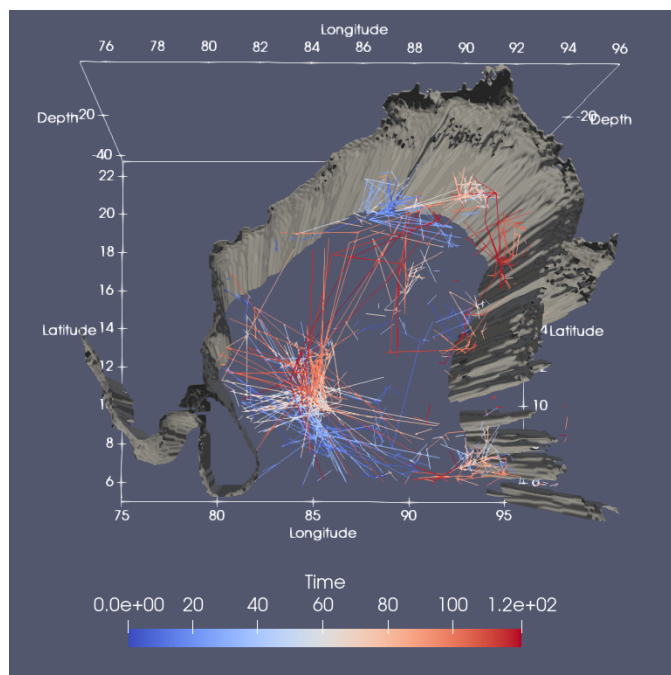


Figure 8.18: Track graphs of Salinity

The Bay of Bengal dataset only has two-dimensional vector fields. So in order to compute front features, we also need vertical velocity. So as a processing step, we have to estimate the vertical velocity. The paper [10] has a procedure for computing the Front Features. The whole algorithm is implemented as a filter for the plugin in the paraview. Figure 8.18 shows the track graph computed by the filter. The track graph represents the movement of the High salinity core. Each path in the graph represents the tracks of individual fronts[10].

Chapter 9

Performance

The Front features calculation for high salinity takes approximately 15 minutes. All other filters take around 2-3 secs on average. It has been observed in the timer-log functionality of ParaView. In this paper, we presented pyParaOcean, a Visualization framework and a plugin for ocean datasets. All the filters are implemented in python. It supports a client-server mode for faster computation and utilization of resources. All the filters are easy to use and can handle large datasets. In future work, we will implement the various temporal supported features like statistics over time and improve the filters' performance and make more user-interactive filters.

Chapter 10

Credit authorship contribution statement

Vijay Kumar:All pyParaOcean implementations.

Upkar Singh:Methodology - front computation code

Vijay Natarajan: Conceptualization of this study, Methodology, Writing - Review and Editing.

Chapter 11

User Manual

11.1 Installation

The package is currently supported in Linux-based systems. All the codes for the plugin are available in the bitbucket repository.

The link is: https://bitbucket.org/vgl_iisc/ocean-visualization-dashboard. To install the plugin, download the files from the repository and extract them into the local machine. To satisfy all the requirements, we need to install all the required packages. Navigate to the plugin directory in the terminal and run the command :

```
1 bash MakeFile.sh
2
```

11.2 Load

After installing the package, Open Paraview

```
1 Paraview->Tools->Manage Plugins
2 Click the Load plugin
3 Select the pyParaOcean.py file from the package and tick the Autoload option
4 .
```

All the filters and macros will be loaded into the ParaView It is shown in Figure 11.1.

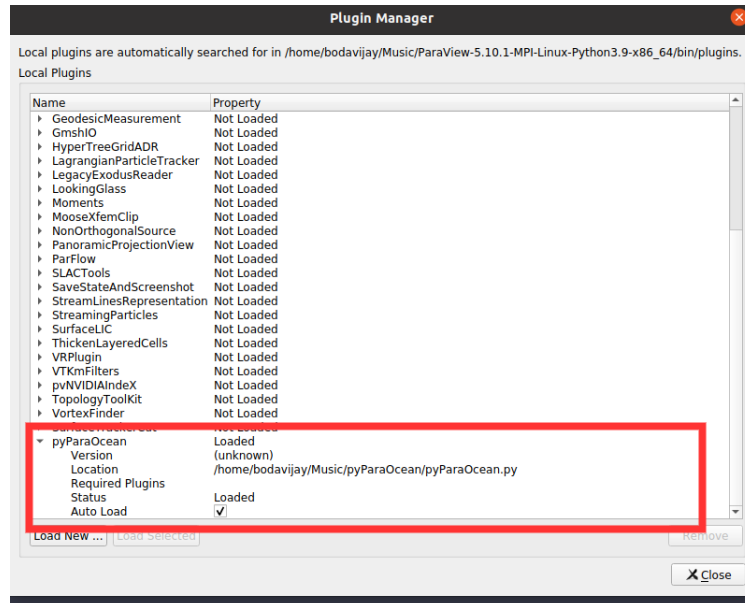


Figure 11.1: Snapshot of Plugin Manager with pyParaOcean filter loaded

11.3 Usage

There are five filters and two macros available in the pyParaOcean.

```

1 Filters :
2     AnalysisFilter
3     InteractiveAnalysisFilter
4     UniformSeedsFilter
5     VortexSeedsFilter
6     TrackGraphFilter
7     Interactive Track Path Filter
8 Macros :
9     Interactive Particle Path
10    Delete Downstream

```

All these files are flexible, as in, users can load individual files as plugins and use them.

11.3.1 AnalysisFilter

The aim of the Filter is to analyze the scalar fields over depth for a given coordinate. It has three views: render view, chart view, and parallel coordinates. In the properties section, we need to set the coordinates on which we do the analysis. The default value it has is (85.5,5). It is highlighted in a red rectangular box. The snapshot of the view is shown in figure [11.2](#)

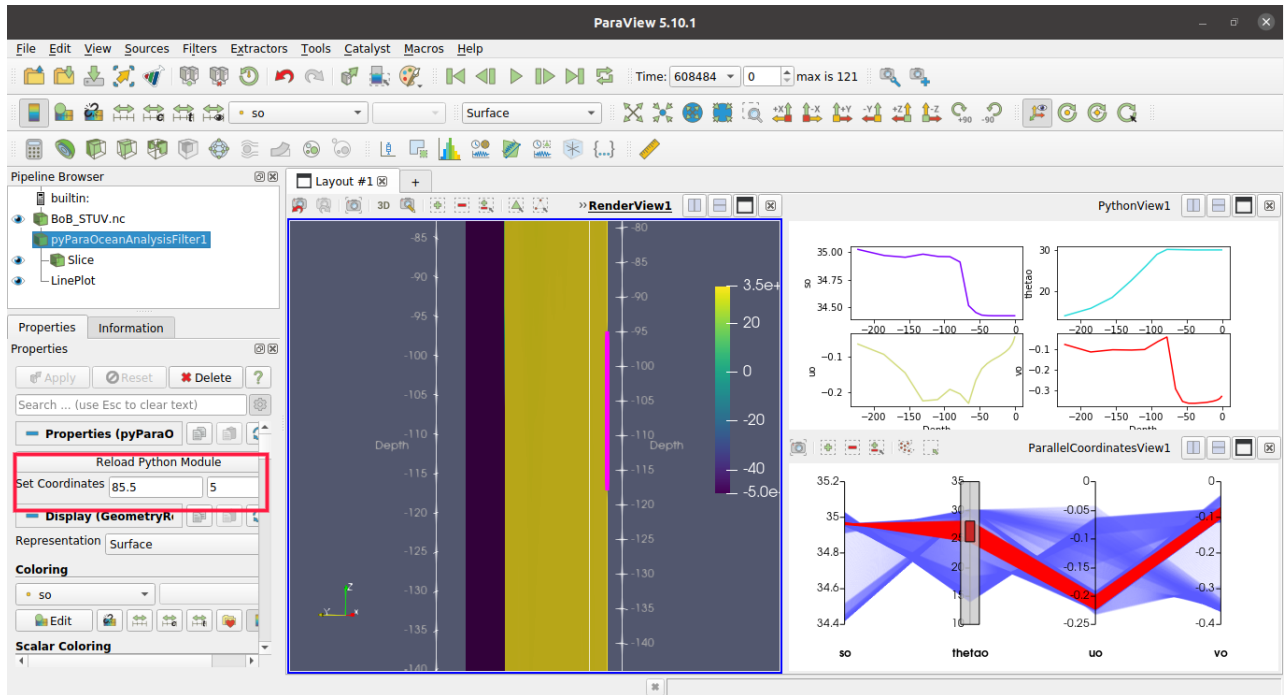


Figure 11.2: Snapshot of Analysis Filter for Bob data set at coordinates (85.5,5)

Parallel Coordinates are used for interactive purposes. The user can brush the range of values which will be reflected in the render view. The selection range is shown in pink colour in the render view and red colour in parallel coordinates. In the figure 11.2, the parallel coordinates view has only four scalars selected, but ParaView has the functionality to modify the scalar visibility in the properties section shown in Figure 11.3.

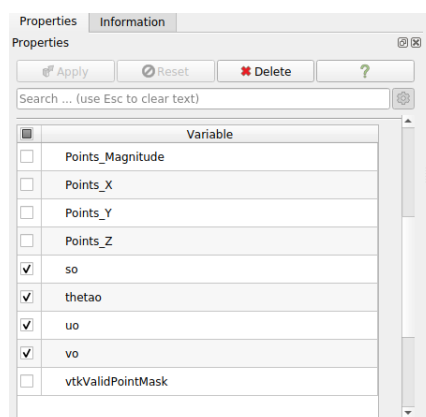


Figure 11.3: Selection of variables for Parallel Coordinates

11.3.2 InteractiveAnalysisFilter

It is the same as the Analysis Filter, but the Line charts are interactive. We utilize the functionality of LineChart View in ParaView. In this Filter, all the plots are overlaid on each other. The snapshot of the view is shown in figure 11.4

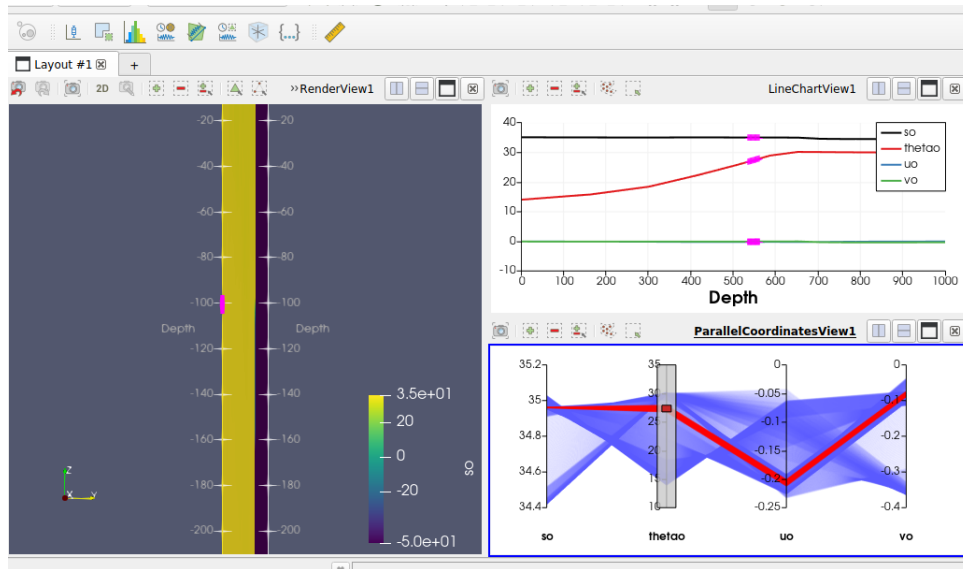


Figure 11.4: Snapshot of Analysis Filter for Bob data set at coordinates (85.5,5)

11.3.3 Uniform Seeds Filter

Seeding strategy is very important for the flow algorithms like particle tracer, streamlines etc. So to better understand the transport of particles, we employ two seeding strategies. One is Uniform, and the other is Vortex based.

This section explains the Uniform seeding strategy.

The Filter is for understanding the velocity vector and transport of the particles. The data set should have a velocity vector and must be temporal. After selecting this Filter, we can see two input text boxes in the properties section. It is highlighted in a green rectangular box.

The first property is Mask Points. It is the resolution of points that the user wants to set as particles all over the data set. The default value is 100, which means from every 100 points, one point is selected.

The second property is velocity vector. The user needs to specify the velocity vector of the data set with comma separated values. The default value is (U, V, W) , but user needs to modify it according to the data set. It also supports 2 dimensional vector, for example, the BoB data set has (uo, vo) vector field. So it doesn't have to be three dimensional vector. It is shown in

figure 11.5

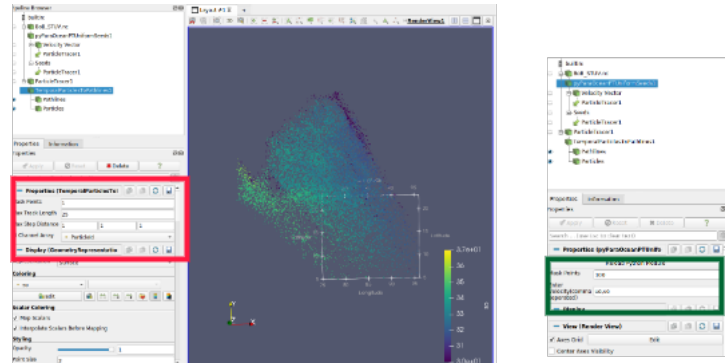


Figure 11.5: Snapshot of Uniform Seeds Filter for Bob data set

After applying the Filter, we get the two outputs one is the Velocity vector, and the other is Seeds. So we can apply any flow algorithms that need seeds as the source. Such ParaView available filters are Particle Tracer, Streamlines with custom source, Particle Path etc. For the case of Particle Tracing, we apply the *Particle Tracer* Filter in ParaView and After that *Temporal Particles To Path lines* Filter to get the Path lines and transport of the particles. We need to set the ID channel array as Particle Id shown in the red rectangular box. The flowchart of usage of this Filter is shown in Figure 11.6

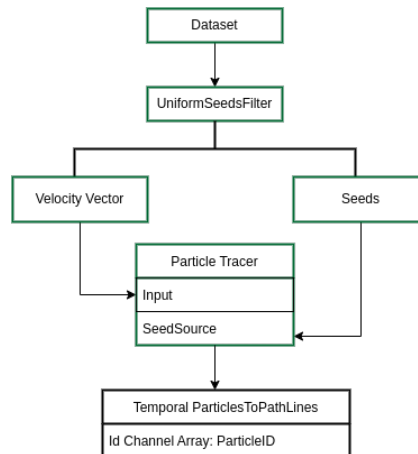


Figure 11.6: Flow chart of UniformSeeds Filter

11.3.4 Vortex Seeds Filter

This section explains the Vortex-based seeding strategy. For creating such seeds, we compute the magnitude of the vorticity and then take the range of $(\delta * \max(\text{vorticity magnitude}))$,

$\max(\text{vorticity magnitude})$), where $\delta \in (0, 1)$. All the requirements and the flow chart are the same as the Uniform Seeds Filter.

11.3.5 Track Graph Filter

This filter shows the computed Track Graph.

- 1 1. Open the Track Graph
- 2 2. Apply the Track Graph filter, it stores z coordinate as a scalar.

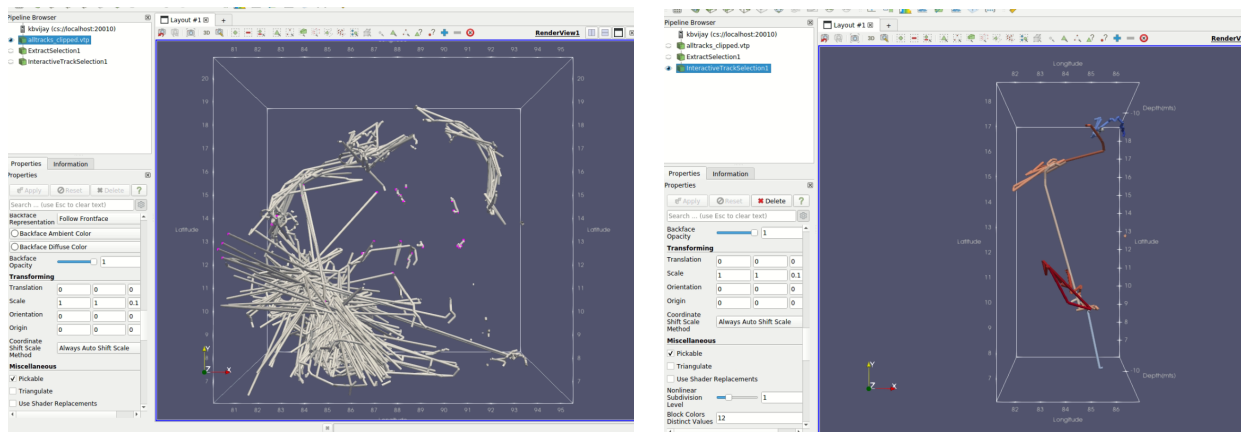
It is shown in 8.18

11.3.6 Interactive Track Path Filter

This filter is applied on the pre-computed track graphs. This helps in generating the representative tracks that describes the overall HSC movement with a focus on regions selected by the oceanographer[10].

- 1 1. Open the Track Graph
- 2 2. Select the points with polygon. It is shown in pink colour
- 3 3. Apply Extract Selection Filter
- 4 4. Apply InteractiveTrackPath Filter.

Figure 11.7a and 11.7b shows the snapshot of the filter usage. The



(a) Select Points with polygon

(b) Tracks of the selected points

Figure 11.7: Visualisation of Interactive particle Path Filter

11.3.7 Interactive Particle Path

It is imported as a macro rather than a filter to make it interactive. The procedure to use this macro is

1. Apply PythonCalculator Filter on the dataset to create the velocity vector. For example, In the Bay Of Bengal data set, we have u_0, v_0 together that become the velocity vector. So type `make_vector(u0,v0)` in this context.
2. Open ParallelCoordinates view layout alongside RenderView with Python Calculator as Active Source.
3. Select points using parallel coordinates and click on the Interactive Particle Path macro.

A snapshot of the particle path macro usage is shown in Figure [8.16a](#) and [8.16b](#)

11.3.8 Delete Downstream

ParaView can only delete the bottom-most source in a pipeline. The deletion task becomes more hectic when we have many filters, sources etc., in the pipeline browser. So to overcome that problem, we have created a macro that deletes all the downstream filters for a selected active source. The usage of this Filter is

1. Select the desired source that is to be deleted.
2. Click the DeleteDownStream macro

11.4 Remote Connection

When the dataset is very large, there will be a need to utilize good computing resources. ParaView can be operated in client-server mode. This section explains the procedure for using such architecture in ParaView. The official <https://docs.paraview.org/en/latest/ReferenceManual/parallelDataVisualization.html> has all the information regarding the usage of the remote server.

A simple usage of client-server mode is:

On the remote side

```
1 1.Connect to the server
2   ssh -L c_port:localhost:s_port username@host_ip_address
3   where c_port = client port, s_port = serverport
```

2.Download and Extract headless machine paraview by running the command:

```
curl -L "https://www.paraview.org/paraview-downloads/download.php?submit=Download&version=v5.10&type=binary&os=Linux&downloadFile=ParaView-5.10.1-egl-MPI-Linux-Python3.9-x86_64.tar.gz" -o ParaView.tar.gz"
```

3. Navigate to the bin folder of ParaView

```
1 Execute the command:
2   ./pvserver --server-port:s_port
```

```

3     Please notice the s_port convention
4 We will see the following output
5     Waiting for client...
6     Connection URL: cs://server_name:20010
7     Accepting connection(s): server_name:20010

```

In the client-side:

Open Paraview and click the connect button and Add the server

```

1 Username: name
2 ServerType: client/server
3 host: localhost
4 port: c_port
5 Click the configure button and select mode as manual and Finish.

```

Now we can use the client-server mode of ParaView by clicking the connect button and selecting the created server.

The below message is displayed on the terminal to reveal that client is connected.

```

1     Waiting for client...
2     Connection URL: cs://server_name:20010
3     Accepting connection(s): server_name:20010
4     Client connected.

```

While using client-server architecture, one should load the plugin both in client and server.

11.5 How to Create a Filter in Paraview?

This section is motivated from official paraview documentation [2]. Below code is a simple paraview filter that takes a dataset as input and returns the distance from origin to each point as output.

```

1 #modules to import
2 import vtk
3 import numpy as np
4 from paraview.util.vtkAlgorithm import *
5 from vtkmodules.numpy_interface import algorithms as algs
6 from vtkmodules.vtkCommonDataModel import vtkDataSet, vtkPolyData
7 from vtkmodules.util.vtkAlgorithm import VTKPythonAlgorithmBase
8 from vtkmodules.numpy_interface import dataset_adapter as dsa
9 @smproxy.filter()
10 @smproperty.input(name="Input")
11 class OriginDistanceFilter(VTKPythonAlgorithmBase):
12     def __init__(self):

```

```

13     # nInputPorts = number of inputs the filter can take
14     # nOutputPorts = number of Outputs filter generates
15     # outputType = Specify the type of output
16     super().__init__(nInputPorts=1, nOutputPorts=1, outputType="
vtkDataSet")
17     def RequestDataObject(self, request, inInfo, outInfo):
18         inData = self.GetInputData(inInfo, 0, 0)
19         outData = self.GetOutputData(outInfo, 0)
20         assert inData is not None
21         if outData is None or (not outData.IsA(inData.GetClassName())):
22             outData = inData.NewInstance()
23             outInfo.GetInformationObject(0).Set(outData.DATA_OBJECT(),
outData)
24         return super().RequestDataObject(request, inInfo, outInfo)
25     def RequestData(self, request, inInfo, outInfo):
26         #Accessind the input data objects
27         input0 = dsa.WrapDataObject(vtkDataSet.GetData(inInfo[0]))
28         #Retrieving the output object
29         output = dsa.WrapDataObject(vtkDataSet.GetData(outInfo))
30         #Get number of point sin data set
31         numPoints = input0.GetNumberOfPoints()
32         distance = []
33         for i in range(0, numPoints):
34             #get coordinate of each point
35             coord = input0.GetPoint(i)
36             x, y, z = coord[:3]
37             #compute distance
38             dist = np.sqrt(x ** 2 + y ** 2 + z ** 2)
39             distance.append(dist)
40         distance = np.array(distance)
41         #ShallowCopy copies all the input variables to the output
42         output.ShallowCopy(input0.VTKObject)
43         #append the computed distance to the output
44         output.PointData.append(distance, "Distance")
45         return 1

```

One can create such filters as per their requirements.

Bibliography

- [1] *Basic Course of Thermo-Fluid Analysis*. 8
- [2] *Paraview Documentation*. 36
- [3] J. Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large data visualization. *Visualization Handbook*, 01 2005. 4
- [4] Wael H. Ali, Mohamad H. Mirhi, Abhinav Gupta, Chinmay S. Kulkarni, Corbin Foucart, Manan M. Doshi, Deepak N. Subramani, Chris Mirabito, Patrick J. Haley, and Pierre F. J. Lermusiaux. Seavizkit: Interactive maps for ocean visualization. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–10, 2019. doi: 10.23919/OCEANS40490.2019.8962794. 2
- [5] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, page 263–270, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897916018. doi: 10.1145/166117.166151. URL <https://doi.org/10.1145/166117.166151>. 7
- [6] Aritra Dasgupta, R. Kosara, and Luke Gosink. Vimtex: A visualization interface for multivariate, time-varying, geological data exploration. *Computer Graphics Forum*, 34, 06 2015. doi: 10.1111/cgf.12646. 2
- [7] Sohaib Ghani, Shehzad Afzal, and Ibrahim Hoteit. Redseaatlas: A visual analytics tool for spatio-temporal multivariate data of the red sea. 06 2019. doi: 10.2312/envirvis.20191101. 2
- [8] Carolina Nobre and Alexander Lex. Oceanpaths: Visualizing multivariate oceanography data. In *Proceedings of the Eurographics Conference on Visualization (EuroVis '15) - Short Papers*. The Eurographics Association, 2015. doi: 10.2312/eurovisshort.20151124. 2

BIBLIOGRAPHY

- [9] William Schroeder, K. Martin, and William Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics*. 01 2006. [2](#)
- [10] Upkar Singh, T.M. Dhipu, P.N. Vinayachandran, and Vijay Natarajan. Front and skeleton features based methods for tracking salinity propagation in the ocean. *Computers and Geosciences*, 159:104993, 2022. ISSN 0098-3004. doi: <https://doi.org/10.1016/j.cageo.2021.104993>. URL <https://www.sciencedirect.com/science/article/pii/S0098300421002764>. [12](#), [26](#), [34](#)
- [11] C. Sullivan and Whitney Trainor-Guitton. Pvgeo: an open-source python package for geoscientific visualization in vtk and paraview. *Journal of Open Source Software*, 4:1451, 06 2019. doi: [10.21105/joss.01451](https://doi.org/10.21105/joss.01451). [2](#)
- [12] NEMO System Team. *NEMO ocean engine*. [6](#)
- [13] Christian Tominski, Jonathan F. Donges, and Thomas Nocke. Information visualization in climate research. *2011 15th International Conference on Information Visualisation*, pages 298–305, 2011. [1](#)
- [14] Habib Toye, Peng Zhan, Ganesh Gopalakrishnan, AdityaR Kartadikaria, Huang Huang, Omar Knio, and Ibrahim Hoteit. Ensemble data assimilation in the red sea: sensitivity to ensemble selection and atmospheric forcing. *Ocean Dynamics*, 67:1–19, 05 2017. doi: [10.1007/s10236-017-1064-1](https://doi.org/10.1007/s10236-017-1064-1). [6](#)