Submission code: RNV0V7GBNA

See this and other abstracts on our website: https://home.sbgf.org.br/Pages/resumos.php

A Python-Based Tool for 3D Visualization of Seismic Data: CimaView3D

Darlan Anschau (SENAI CIMATEC), Victor Koehne (SENAI CIMATEC), Daniel Revelo (SENAI CIMATEC), Diego Barrera (SENAI CIMATEC and INCT-GP), Caio Leão (SENAI CIMATEC), Laila Vergne (SENAI CIMATEC), Matheu Santos, Rodrigo Santos

A Python-Based Tool for 3D Visualization of Seismic Data: CimaView3D

Abstract

Open-source Python visualization tools play a vital role in enhancing customization and improving the user experience for geophysical data analysis. However, their effective implementation is often accompanied by challenges in High-Performance Computing (HPC), such as optimizing the utilization of computational resources, ensuring the compatibility of domain-specific workflows, and managing the increasing complexities associated with big data and artificial intelligence integration. Based on a comprehensive review of existing tools, we developed CimaView3D, an open-source visualization tool designed to streamline the visualization and analysis of 3D seismic data. Leveraging Python as the primary programming language and Jupyter as the development environment, CimaView3D is built upon CIGVis, a domain-specific visualization tool, and introduces advanced customization features and interactive plotting capabilities, previously unavailable. Key innovations include functionalities for adjusting data representation, incorporating subplots, generating cross-sections, placing markers, grouping elements, and customizing layouts. These features were developed extending Plotly-based components implemented in CIGVis for improvements. This paper discusses the development process, highlights the new functionalities, and presents results that demonstrate the tool's effectiveness in enhancing geophysical data visualization.

Introduction

Matplotlib for Python, Barrett et al. (2005), is a well-known visualization library that is neither optimized for 3D nor adapted for geophysical data. Despite the large amounts of data size, the execution time for seismic data tasks can currently be outperformed by other visualization libraries in the literature. Notwithstanding, the Jupyter environment is a *de facto* technology with numerous advantages for a broad range of users. It offers access to a powerful system of development, where the user can interact with code and outputs without the need of having to set complex client-server configurations.

After searching for functionalities in tools for 3D visualization and imaging, the following packages were selected and studied because of the relevant contributions to this work: Lexcube (Söchting et al., 2023) proved to be very limited for our objectives. Although it presents the data in a standard cube format and provides a time-lapse interaction, it does not have the elements we searched that are present in other packages; PyVista (Sullivan and Kaszynski, 2019) has backend implementations for Jupyter via the trame library. So, it is possible to create the client/server structure to render the Visualization ToolKit (VTK) used by PyVista, but this brings some complications with it and some functions are not yet compatible with Jupyter; and CIGVis (Li et al., 2025), on the other side, has an implementation that is compatible with the Plotly library. The Plotly library is well-established in the areas of statistics and data analysis, often serving as a replacement for Matplotlib. Furthermore, the CIGVis package focuses on the visualization of seismic data.

Having these features in mind and seeking characteristics such as reproducibility, standardization, customization, and optimization, as well as to facilitate the installation and usage, we implemented a new open-source visualization tool for the seismic data field over CIGVis, named CimaView3D.

Methodology

CIGVis is built over a Seismic Canvas (Kroeger, 2015) implementation, which is domain-centered, having specific functions for geophysical data, but has the disadvantage of lacking a way to standardize and customize settings both for individual and groups of components as we wanted. On one side, a new extension library for CIGVis named viserplot improves Jupyter support but needs further client-server configuration and does not satisfy our specifications for improvement. On the other side, CIGVis' Plotly implementation is a higher-level implementation that lacks customization which could be achieved with some modifications at the Plotly level. As stated by Frants and Ohman (2024), Plotly offers powerful programming tools for interactive data visualizations. However, its usage inside the CIGVis implementation does not satisfy the full features specified for CIGVis and neither does satisfy the needs for our customizations. We enabled these customization features through the functions, providing parameters to be passed to Plotly. In this sense, geophysicists can easily set preferences, rely on the patterns they define, and reuse the code in other circumstances.

To interpret plotted images, it is important to note that the plots as well as the data have to be reliable. The data in a plot must display the properties to explore whether it has different noises or shape alterations, whether it holds different aspects or color schemes, and so forth. So, to reproduce the plotting of an image it is desirable to have the same configurations the plotting was carrying when it was plotted. With these configurations, one can reproduce the same experiments and/or utilize the same plot settings for further analysis. Besides, holding the same configuration also allows the interaction with the data as a visualization.

Figure 1 displays two typical interactive plots in CimaView3D. To produce these images from the same seismic data, we passed values to the parameters that create the slices, including the subplots and the colormap schemes, and to the specific camera configuration parameters. After they were plotted from the cells inside Jupyter, we then exported the images holding their starting settings. As a result, the subplots enable comparing different colormap schemes and different cross-sections.

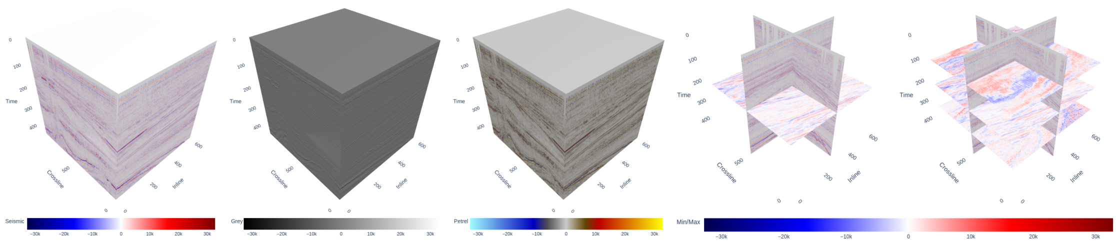

Figure 1: The three first subplots hold different colormap schemes displayed in the output window for comparisons; while the fourth and fifth subplots display different slices from the same seismic data.

A domain-specific tool must respect the field's technical specifications. Thus, to achieve a good user experience the tool should provide little effort in the domain of usage, because it would lay on the standards of the field. Notwithstanding, one should be able to customize the plotting settings, because one may want to standardize different needs and/or preferences in groups of settings, to be reused, or modify particular properties of the elements individually.

With CimaView3D, users can establish standardized settings that are readily available for use, ensuring consistency and efficiency in seismic data visualization. For example, whenever the user might want to use markers as well-defined objects, the markers can be grouped by setting a name that shares properties between these elements and is referred in the legends. One can also customize the components individually. For example, the user might want to change the color of some markers because they pertain to the same group, but must be highlighted.

CimaView3D also allows the utilization of tailored functions. For example, whenever a physicist has to plot the receptors above the data, at the surface, the physicist should create them as markers, one by one. In our approach, tailored functions can be programmed for this task, by passing the regular spacing values between these objects to: the function `surface_grid` to create a grid at the surface; and the function `surface_intersections` to get the positions where this grid intersects (independently of creating a grid). A ‘receptors’ group can then provide the properties that the markers share and the functions then automate the placing of the markers, positioning them at the grid.

Figure 2 shows two plots where markers are customized and used: for the first two subplots, the markers are set manually but grouped; while for the third and fourth subplots, a grid is automatically created and markers are automatically plotted in the place where ‘receptors’ should be.

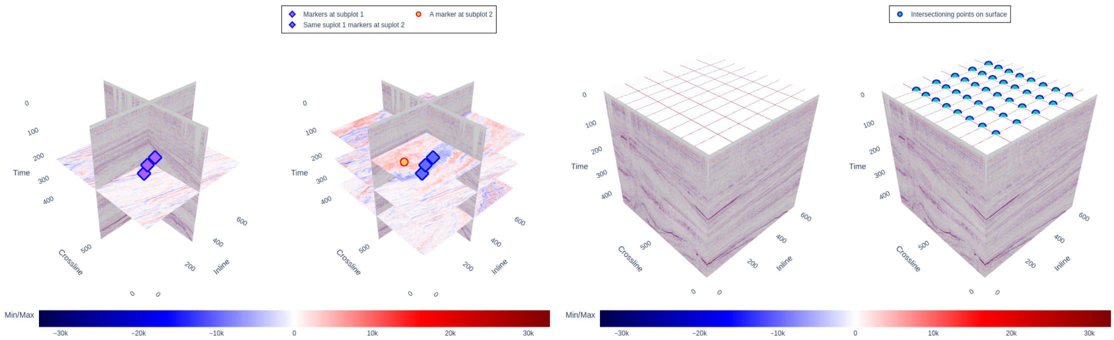

Figure 2: The two first subplots have different kinds of customized grouped markers, which are attached to the subplots; while the fourth subplot displays usage for the special function `surface_grid`, which adds a grid on the top of the data surface, and the fifth subplot displays the addition of markers at the intersections of the grid, through the function `surface_intersections`.

Results

We mapped some Plotly functions and CIGVIS functions to our implementation. Some elements have already been targeted to the modified to allow standardizations and customizations: data aspect (original data or cube); markers (a marker has a specific location where it is placed, and has many properties, that can be shared or not); annotations (an annotation has a specific location that can be visible and pointed in its surroundings, receiving some context, e.g. arrows can be the pointers, which can be grouped sharing the same properties, but having different angles individually); fonts and legends (for visibility); cross-sectioning (positioning slices); grid (background, axes, axes labels, axes numbering, axes labels, etc); subplots (for comparisons on the same data, but holding different components or settings); colormap bar (title, orientation, etc); and camera settings (as a dictionary `camera` or as its components `eye`, `center`, and `up`).

Some settings are standards reused by physicists that can be hard-coded in order to facilitate the user to follow patterns. It also prevents the user from writing code whenever it is better to choose between a group of well-defined settings. Figure 3 shows an example of code and plot in CimaView3D.

```
import CimaView3D as cima                          # import CimaView3D
# create slices
node = cima.create_slices(
    seismic_data, cmap='seismic',
    cbar_params={'title':"Seismic"})
# camera settings
center=dict(x=0,y=0,z=-.06)
eye=dict(x=1.32,y=1.32,z=1.32)
# start plot function
cima.plot3D( node, font_size=14,
    z_label="Depth",                               # start z axis settings
    z_autorange=True,
    z_tickvals=[0,50,100,150,200,300,400,500],
    y_tickvals=[0,200,400,600,800,1000],           # y axis
    eye=eye, center=center,                        # camera settings
    show_grid=True,                                # start grid settings
    x_bgcolor='rgba(255,0,0,0.3)',
    y_bgcolor='rgba(0,255,0,0.3)',
    z_bgcolor='rgba(0,0,255,0.3)')
```

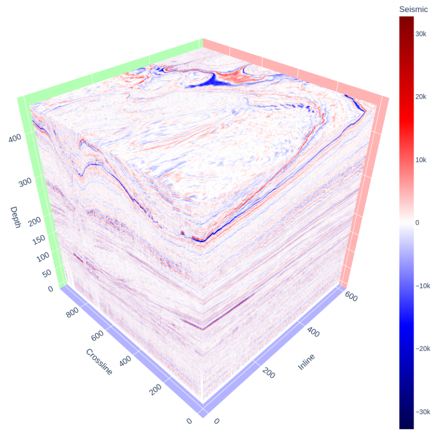

Figure 3: A functional code regarding some possibilities with CimaView3D in Jupyter and its output: importing the library to the scope of development, creating slices from a given `seismic_data` variable, configuring the positioning of the camera, and other customizations for the plot components.

Conclusions

In this work, we presented CimaView3D, a visualization tool that successfully balances standardization and customization for seismic data analysis. The tool addresses key challenges in the field, offering enhanced functionalities tailored to the needs of geophysicists. Planned enhancements include integrating predefined visualization patterns for common tasks and publishing the library on a Python package repository. Comprehensive resources, including installation instructions, a detailed tutorial, and the source code, are available on GitHub at https://github.com/cs2i-senai-cimatec/CimaView3D.

Acknowledgments

This research was conducted within the framework of the Marchenko project, with financial support from CENPES/Petrobras and CNPq through the INCT-GP. We would like to thank CENPES and SENAI CIMATEC for granting permission to publish this research.

References

Barrett, P., J. Hunter, J. T. Miller, J.-C. Hsu, and P. Greenfield, 2005, Matplotlib–a portable python plotting package: Astronomical data analysis software and systems XIV, 91.

Frants, M., and M. D. Ohman, 2024, Online visualization of geospatial cruise data with plotly python.: American Geophysical Union, Ocean Sciences Meeting, DO14B–2493.

Kroeger, G. C., 2015, Seismic canvas: Evolution as a data exploration and analysis tool: AGU Fall Meeting Abstracts, S33C–2793.

Li, J., Y. Shi, and X. Wu, 2025, Cigvis: An open-source python tool for the real-time interactive visualization of multidimensional geophysical data: Geophysics, **90**, F1–F10.

Söchting, M., M. D. Mahecha, D. Montero, and G. Scheuermann, 2023, Lexcube: Interactive visualization of large earth system data cubes: IEEE Computer Graphics and Applications, **44**, 25–37.

Sullivan, C., and A. Kaszynski, 2019, Pyvista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (vtk): Journal of Open Source Software, **4**, 1450.