GEOPHYSICS, VOL. 90, NO. 1 (JANUARY-FEBRUARY 2025); P. F1–F10, 9 FIGS.
10.1190/GEO2024-0041.1

CIGVis: An open-source Python tool for the real-time interactive visualization of multidimensional geophysical data

Jintao Li[1], Yunzhi Shi[2], and Xinming Wu[1]

ABSTRACT

As Python's role in processing and interpreting geophysical data expands, the need for a Python-based tool tailored for visualizing geophysical data has become increasingly critical. In response, the Computational Interpretation Group Visualization tool (CIGVis) — a fully open-source Python tool optimized for researchers and licensed under the Massachusetts Institute of Technology License — has been developed. It specializes in the real-time interactive visualization of multidimensional geophysical data, including volumetric data of geophysical properties, meshes of geologic objects (e.g., faults, horizons, and geobodies), points and tubes of well-log data. CIGVis enables users to interact with the data through operations such as rotation, panning, magnifying, and dragging slices. It also supports a multicanvas functionality, allowing a simultaneous visualization across multiple subcanvases with a unified camera perspective. Its ease of use allows for effective visualization with just a few lines of code across all major operating systems and extends to desktop and Jupyter environments, facilitating code execution in various settings. The core functionalities of CIGVis are exemplified using standard geophysical data sets, such as the F3 data set.

INTRODUCTION

In the field of geophysics, data visualization is not merely a means of data representation but a key to deeply understanding the subsurface. Lynch (2008, 2023) regards data visualization as a third type of resolution, beyond temporal resolution (the ability of a seismic wavelet to resolve thin reflections) and spatial resolution (the capacity of a seismic wavelet to resolve closely spaced geologic details). Neglecting the role of visualization could detrimentally affect the understanding of geophysical data.

Geophysical data often encompass multiple dimensions, obtained through various collection instruments, methods, and processing techniques. These include 1D well logs, 2D seismic sections, seismic horizons, 3D seismic surveys, and 4D time-lapse seismic data, among others. Each data type provides unique insights into the subsurface. Converting these data into intuitive visuals is crucial for geologic interpretation and resource assessment. Visualizing these data typically requires 3D visualization technologies, enabling scientists and engineers to view and interpret the data in a way that reveals their true form in the subsurface.

Over the past few decades, numerous commercial software programs have been developed for the interactive visualization of geophysical data. These tools allow for in-software data analysis, interpretation, and identification of geologic structures, such as faults, horizons, and channels. Well-established software such as Schlumberger's Petrel (Schlumberger, 2024), Halliburton's Landmark DecisionSpace (Halliburton, 2024), and GeoSoftware's HampsonRussell Software (GeoSoftware, 2024) have evolved over the years, offering comprehensive functionalities for a wide range of tasks. However, their expensive licensing fees and proprietary restrictions can be prohibitive for young researchers and academic institutions, and their complexity often introduces a steep learning curve.

With the rise of artificial intelligence, machine learning and deep learning are being widely used in geophysics for many tasks, such as data processing (Yu et al., 2019; Li et al., 2021), seismic interpretation (Wu et al., 2019; Li et al., 2023), and inversion (Das et al., 2019). Therefore, Python, a premier language in deep learning, has significantly advanced the processing, analysis, interpretation, and inversion of geophysical data. However, for many practitioners, there is a significant gap between performing computations using deep-learning frameworks and visualizing the results with industrial software. This gap arises because the outputs of deep-learning

Peer-reviewed code related to this article can be found at this paper's Supplemental Materials link.
Manuscript received by the Editor 31 January 2024; revised manuscript received 28 September 2024; published ahead of production 15 October 2024; published online 19 December 2024.
[1]School of Earth and Space Sciences, University of Science and Technology of China, Hefei, China and Mengcheng National Geophysical Observatory, University of Science and Technology of China, Hefei, China. E-mail: lijintao@mail.ustc.edu.cn; xinmwu@ustc.edu.cn (corresponding author).
[2]Amazon, Austin, Texas, USA. E-mail: shiyunzh@amazon.com.

frameworks are typically in the form of tensors or arrays, whereas industrial software requires input in specific file formats. The conversion between tensor formats and industrial standard data formats necessitates reorganizing the tensor data, adding special headers, and writing them into a particular format. Conversely, extracting data from industrial software involves removing the headers and converting the data back into tensors. For example, a 3D seismic data set often needs to be converted to the SEG-Y format to be supported and visualized by industrial software. From the user's point of view, this conversion process can be cumbersome, requiring conversion codes and consuming a substantial amount of time. As more workflows shift toward Python, there is a growing need for a Python-based geophysical data visualization tool, eliminating the additional conversion steps.

Several Python visualization tools exist, such as Matplotlib (Hunter, 2007) — a static and foundational plotting library; VisPy (Campagnola et al., 2015) — a high-performance real-time multidimensional visualization library; visualization toolkit (VTK) (Kitware, 2024) — an open-source software system for image processing, 3D graphics, volume rendering, and visualization; PyVista (Sullivan and Kaszynski, 2019) — a VTK-powered user-friendly package; and Plotly (Inc, 2024) — an interactive, open-source, and browser-based graphing library for Python. However, these tools either lack real-time interactive 3D visualization capabilities or are not tailored for geophysical data. For instance, Matplotlib is suitable for static 2D plotting, but it lacks the capability for real-time interactive 3D plotting. Libraries such as PyVista and Plotly offer basic real-time interactive features such as rotation and magnifying. However, they fall short of providing the ability for real-time interaction with 2D slices, which is a fundamental requirement in geophysics. This capability is essential for conveniently and quickly examining the internal features of 3D data. VisPy supports the ability to interact with 2D slices in real time (VTK might support this ability), but implementing this functionality requires a substantial amount of additional customization. Their basic functionalities do not support these features out of the box, and this capability is not mentioned in their documentation.

Although the aforementioned Python packages are robust and versatile foundational libraries designed for a wide range of applications, their generality can sometimes come at the cost of specialization. Using these general-purpose tools effectively in geophysics requires significant adaptation and a significant amount of additional code to meet the specific needs of geophysical research. For example, implementing the ability to move 2D slices in real time interactively, visualizing well logs, integrating well logs with 3D seismic data, overlaying various data sets (such as seismic data, faults, and impedance), and handling the unique coordinate system used in geophysics (wherein depth increases downward rather than upward) demand extensive customization. Seismic Canvas (Shi, 2024) is a Python tool specifically designed for the interactive visualization of 3D seismic data; however, it offers a relatively limited set of functionalities. In other words, existing tools do not offer out-of-the-box solutions for geophysical applications. In addition, their interfaces are not specifically tailored for geophysical data, which may result in some unique requirements that are not met with geophysical data.

In this paper, we introduce the Computational Interpretation Group Visualization tool (CIGVis), which is a Python tool developed by the Computational Interpretation Group (CIG), a research group at the University of Science and Technology of China, for the real-time interactive visualization of multidimensional geophysical data. Designed primarily for researchers, CIGVis enables the rapid visualization of multidimensional geophysical data with just a few lines of code (out-of-the-box application). This encompasses data such as 3D seismic data, horizons, faults, well logs, and geologic bodies, thereby allowing researchers to concentrate more on the analysis and interpretation of geophysical data. A key strength of CIGVis is its interactive features, which allow users to engage dynamically with the data through operations such as rotation, panning, magnifying, and dragging slices. In addition, CIGVis supports multicanvas functionality, allowing for simultaneous visualization on multiple subcanvases with a unified camera perspective. This capability is crucial for comparative analysis between different data sets.

Built on Seismic Canvas, CIGVis retains the real-time slice-dragging feature and significantly enhances and expands its functionality. The underlying architecture of CIGVis is based on libraries such as Vispy, Plotly, and Matplotlib. Rather than surpassing the existing Python visualization tools, our work focuses on adapting these foundational tools to serve geophysical data visualization needs better. We leverage the existing capabilities to make them more suitable for geophysical data visualization scenarios. We have integrated these libraries and written a considerable amount of custom code to cater to geophysical needs across various environments.

CIGVis has been optimized for compatibility across major operating systems and supports desktop and Jupyter environments, ensuring wide accessibility and flexibility for researchers. As a fully open-source software, CIGVis adopts the Massachusetts Institute of Technology License (MIT license; please see the "Data materials and availability" section). CIGVis is continuously evolving, with regular updates to its source code, documentation, and demos to cater to the expanding needs of users.

DESIGN PHILOSOPHY AND FRAMEWORK

CIGVis is a software specifically crafted for the visualization of geophysical data and tailored for researchers. Its architecture and features are shown in Figure 1. CIGVis's design philosophy embraces simplicity and real-time interactive 3D visualization while ensuring that the software maintains excellent extensibility and the freedom of open-source code. Extensibility means users can leverage lower-level libraries to create data formats that are not natively supported by CIGVis and integrate them with its existing features for seamless visualization.

Simplicity is reflected in the ease of installation and the user-friendly interface of CIGVis. Leveraging Python's cross-platform capabilities, CIGVis can be seamlessly installed on any major operating system using pip, eliminating the need for downloading the source code and navigating complex compilation steps. Moreover, CIGVis offers compatibility with Jupyter environments, although with some functionality constraints compared with a desktop environment. This ease of use empowers researchers to efficiently visualize 3D data with just a few lines of code.

Developed on the foundations of Vispy, Plotly, and Matplotlib, CIGVis offers flexible tools for visualizing diverse geophysical data and allows users to adapt visualizations to specific research needs. It provides advanced application programming interfaces (APIs) for geophysical data visualization out of the box, allowing researchers to focus on the data's characteristics without getting bogged down in technicalities. CIGVis supports real-time interactive visualization of 3D data, such as 3D seismic and geomagnetic data, enabling users to freely move slices within the data to observe internal features

DOI:10.1190/geo2024-0041.1

deeply. This means that even if CIGVis does not cover certain functionalities, users familiar with these two libraries can extend them as needed. Because we are more familiar with VisPy and Plotly, we chose these libraries as the foundation for our 3D visualization work. However, PyVista can also achieve 3D interactive visualization within Jupyter notebooks, though it, similar to Plotly, only supports basic operations such as rotation and magnifying. In addition, we think that Plotly's 3D visualization capabilities are more detailed and versatile compared with PyVista. Plotly offers more flexibility and aesthetic appeal in axis visualization, and it also has a larger community (15.5k stars on GitHub versus 2.4k for PyVista). On desktop environments, we opted for VisPy as our foundational library because PyVista's high level of encapsulation, although user friendly, makes it more complex to decompose, customize, and implement its real-time interactive 2D slice-dragging ability. Furthermore, CIGVis is governed by the MIT license, an open-source software license that grants users wide-ranging freedoms to use, modify, and distribute.

CIGVis is composed of several modules, each serving a specific purpose. The Utils module provides essential functions for handling geophysical data, including reading and writing various data formats, processing coordinate systems, and providing common functionalities. The 2D canvas module, based on Matplotlib, is specifically designed for visualizing data on a 2D canvas in research. The 3D canvas is the core of CIGVis, encapsulating high-level APIs for visualizing geophysical data. The Colormap module offers commonly used colormaps in geophysics, which are found in commercial software but not always included in Matplotlib. It also provides functions for users to adjust the colormaps as needed. The graphical user interface (GUI) module provides some limited but functional simple GUIs, as complex GUIs are not our development goal. Specifically, these functional features allow for the quick and convenient visualization of 3D geophysical data, along with the ability to easily adjust data slices and camera perspectives, all without requiring any coding.

The 3D canvas can handle various geophysical data, including seismic and geomagnetic data, horizons, faults, well logs, and 3D geologic bodies such as salt, channels, and paleokarst, and it supports an overlay display (e.g., showcasing seismic data with their relative geologic time). Beyond visualization, users can perform multiple interactive operations in the 3D canvas, such as rotation, movement, magnification, slice dragging, and multicanvas displays. The multicanvas feature allows users to divide a single canvas into multiple subcanvases. Each subcanvas can load data independently while sharing a unified camera perspective, which facilitates the comparison between different data sets.

CORE FEATURES

The core feature of CIGVis lies in its 3D canvas functionality. Users can visualize various types of geophysical data through simple code on this canvas. As shown in Figure 2, we demonstrate CIGVis's key features using the F3 data set. Notably, in the 3D canvas, all elements (called nodes) are independent of each other. By progressively adding new elements to the canvas, we illustrate the powerful capabilities of CIGVis.

3D volume

CIGVis represents a 3D volume by displaying multiple 2D slices in different orientations. Figure 2a shows an example of a 3D volume represented by multiple 2D slices in the *x*-, *y*-, and *z*-directions. The basic real-time interactions with the 3D canvas, such as rotation, panning, and magnifying, can be performed with the mouse and keyboard. Moreover, each slice is presented as an image, allowing for rapid rendering. Users can move any 2D slice in real time by clicking and dragging the mouse, conveniently observing data at any point in 3D space. The basic interactions with the tool can be seen in the GIF image provided in supplemental S1. A concise code example for quickly visualizing 3D data is shown in the following:

```
# A demo for quickly visualizing 3D data sets (e.g., seismic data)


import numpy as np
import cigvis


# d is a 3D numpy array
d = np.load('demo.dat', np.float32).reshape(256, 256, 256)


# create slices represent the 3D data set
node = cigvis.create_slices(d)
# plot, visualize the data set
cigvis.plot3D(node)
```

Geologic bodies

A precise depiction of 3D geologic bodies is crucial for subsurface resource exploration. For instance, channels and paleokarst are excellent hydrocarbon reservoirs, and identifying the boundaries of salt is key for velocity modeling and seismic structural analysis. Typically, artificial intelligence (AI) identification generates a

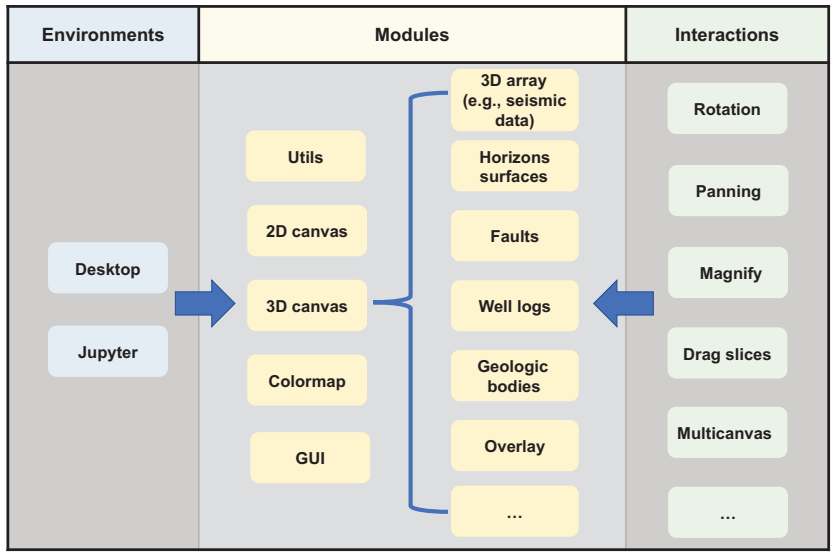

Figure 1. The design framework of CIGVis.

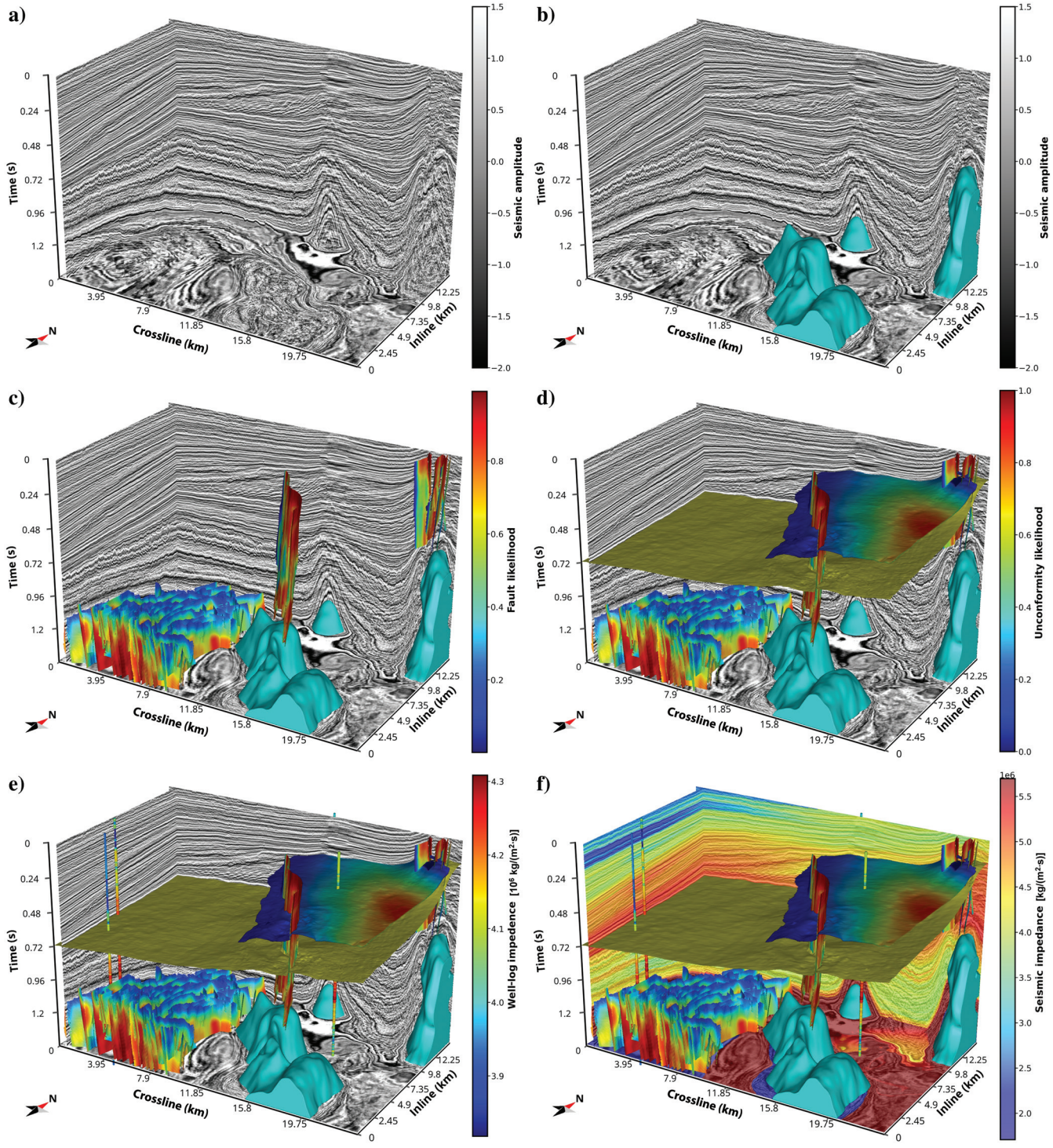

Figure 2. Demonstrating the core features of CIGVis using the F3 data set. We sequentially add geophysical data to a 3D canvas, showcasing the (a) 3D seismic data, (b) salt body, (c) fault surface with colors representing the fault likelihood, (d) horizons in various colors (with colors indicating unconformity likelihood for the upper one), (e) well logs with colors representing the impedance, and (f) an overlay of seismic data with their impedance.

DOI:10.1190/geo2024-0041.1

binary or probabilistic 3D array. CIGVis uses the marching cubes algorithm (Lorensen and Cline, 1987) to extract geologic body boundaries and displays them using a closed mesh constructed with triangular grids. Figure 2b shows a 3D representation of a salt body in the F3 data set.

Faults

Faults are vital pathways for hydrocarbon migration and play a crucial role in oil and gas exploration and natural hazards assessment. CIGVis conveniently visualizes fault planes, not only displaying their location but also superimposing the fault attributes (the fault likelihood [Hale, 2012] in this case and its colors representing the probability of the fault) in an intuitive manner, as shown in Figure 2c. In addition, the results of AI fault identification, which are often binary or discrete 3D arrays, can be overlaid on seismic data for visualization (see the "Overlay" and "Colormap" sections).

Horizons

Horizons are geologically significant underground interfaces, often considered to represent the same geologic period. Visualizing horizons is one of the most basic and vital requirements in geophysical visualization. CIGVis can process depth matrices of the same size as seismic horizons as well as handle 3D spatial coordinates, whether ordered or not. Users can display depth or seismic amplitude data on horizon surfaces and customize the colors for each point. Figure 2d shows two horizons: the lower horizon displays a full spatial distribution in yellow, whereas the upper horizon is an unconformity surface with colors representing its likelihood (Wu and Hale, 2015) (see the "Well logs" and "Jupyter" sections).

Well logs

A well log typically includes several curves, such as gamma, velocity, and density. As shown in Figures 2e and 3, CIGVis visualizes well logs using tubes with varying sizes and colors, where the tube's attributes represent the relative magnitude of the curve values. Alongside the tube that indicates the density through its color, additional curves such as velocity, gamma, and impedance are depicted as mesh surfaces attached to the tube, with the mesh's color and width varying based on the respective curve magnitudes. Three distinct surrounding mesh layers represent velocity, gamma, and impedance. CIGVis provides the capability to display vertical wells (Figure 3a) and deviated wells (Figure 3b) in conjunction with seismic data. The magnified view highlights the well logs for clearer observation, which removes the coordinate grid. Moreover, when the seismic slice is adjusted to the well's position, users can easily compare the seismic data with the well-log curves. In addition, CIGVis supports the representation of well logs as lines in 3D space (see the "Well logs" and "Jupyter" sections). For more detailed comparisons, a GIF in supplemental S1 shows different styles of well-log visualizations.

Overlay

Overlaying two or more types of 3D data helps us understand the correspondence between different data sets and verify the consistency of the results with reality. CIGVis easily achieves the overlay display of various data types. For example, Figure 2f shows the effect of overlaying 3D seismic data with its impedance. This display method not only retains the basic information of the seismic data, such as the horizons, but also clearly examines the accuracy of the impedance through the contrast with the foreground color. In addition, geologic formations such as faults and channels can be overlaid on seismic images to analyze the distribution of these geologic structures (see Figures 4c and "Overlay" and "Colormap" sections).

The following is a sample code for quickly achieving the result shown in Figure 2f:

```
# Example to visualize Figure 2f


# import Python libraries
import numpy as np
import cigvis
from cigvis import colormap



# Read geophysical data sets
#...
# seis, salt, intp, unc: 3D numpy arrays, means seismic, salt, impedance, likelihood
# hz2, unc2: 2D numpy arrays, means horizon, unconformity
# log_position, log_values: logs' trajectory and curve values (impedance, density, ...)
```

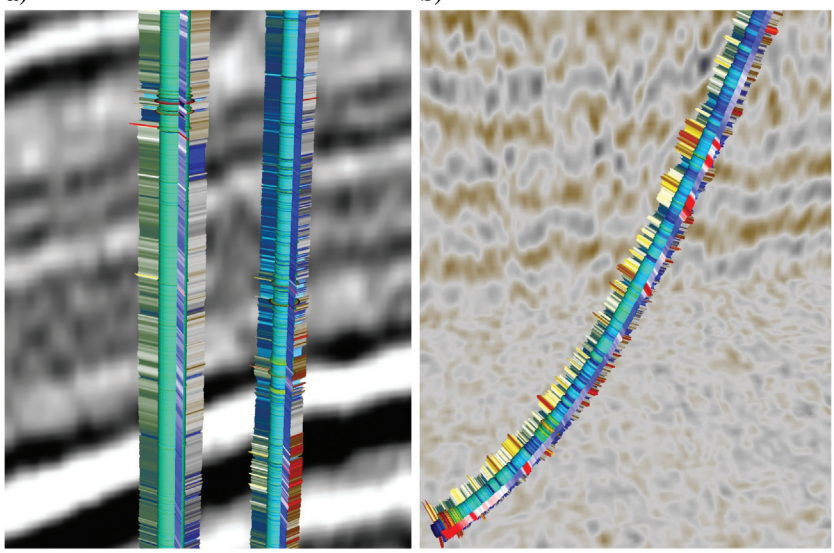

Figure 3. Display of the well logs in CIGVis, combining (a) vertical and (b) deviated wells with seismic data. The image is magnified to highlight the well logs, so the coordinates are omitted. The tube represents the well-log trajectory, with color indicating density and surrounding mesh surfaces showing velocity, gamma, and impedance. The size and color of the tube reflect the magnitude of each curve value. Dragging a seismic slice to the well location allows for easy comparison between the well log and seismic data.

```
  # set colormap and alpha for impedance data
(foreground image)
  fg_cmap = colormap.set_alpha('jet', 0.6)
  # create slices for a 3D seismic data set
  nodes = cigvis.create_slices(seis)
  # create salt body in cyan color
  nodes += cigvis.create_bodys(salt, 0.0, 0.0,
color = 'cyan')
  # horizons, hz2 is point positions, set as
yellow
  nodes    +=    cigvis.create_surfaces([hz2],
color = 'yellow')
  # unconformity, set color as likelihood, unc is
a 3D array
  nodes    +=    cigvis.create_surfaces([unc2],
volume = unc, value_type = 'amp')
  # log_position is a (N, 3) array, log_value is a
(N, m) value array of logs
  nodes  +=  cigvis.create_well_logs(log_posi-
tion, log_values)
  # create fault faces
  nodes  +=  cigvis.create_fault_skin(root  +
'skins/')
  # overlay seismic data (background) and its
impedance (foreground)
  nodes    =    cigvis.add_mask(nodes,    intp,
fg_cmap = fg_cmap)
  # add axes
  nodes    +=    cigvis.create_axis(seis.shape,
mode = 'axis', north_direction = [0.9, 0.03],
intervals = [0.025, 0.025, 0.004])
  # add colorbar for the foreground impedance
data set
  nodes    +=    cigvis.create_colorbar_from_
nodes(nodes, "Impedance", select = "mask")
  # plot, visualize them
  cigvis.plot3D(nodes)
```

Multiple canvases

A distinctive feature of CIGVis is its support for multiple canvases. This feature allows users to divide a single canvas into multiple subcanvases, each hosting independent nodes. Each subcanvas is independent, and any interactive operations can be applied to individual subcanvases. As shown in Figure 4, the 3D canvas is divided into four subcanvases: Figure 4a shows the 3D seismic data, Figure 4b shows the impedance data, Figure 4c shows the grayscale seismic data overlaid with the channel data, and Figure 4d simultaneously shows the seismic and channel body data. The data sets are sourced from cigChannel (Wang et al., 2024). A key functionality in this setup is the unified camera perspective across all the canvases. This means any action, such as rotation, panning, or magnifying, applied to one canvas affects all the others in a synchronized manner. For instance, if a slice is dragged in a specific direction in Figure 4a (indicated by the red arrow), the corresponding slices in other subcanvases will also move synchronously. This multicanvas display is particularly valuable for conducting a comparative analysis between different data sets, offering an efficient way to compare the various results and explore relationships between different types of data. A GIF image in supplemental S1 shows the dynamic effects of this feature.

In addition, this functionality might be suited for analyzing 4D geophysical models in relation to monitoring problems. As shown in Figure 5, we take CO_2 monitoring as an example to illustrate how to use CIGVis for displaying the 4D seismic monitoring of CO_2. We refer to the work of Sheng et al. (2023) to demonstrate the distribution of CO_2 plumes in the subsurface over different years. By linking various subcanvases, we can intuitively understand the temporal flow and changes in underground CO_2 plumes. A supplemental S1 GIF image dynamically shows this capability.

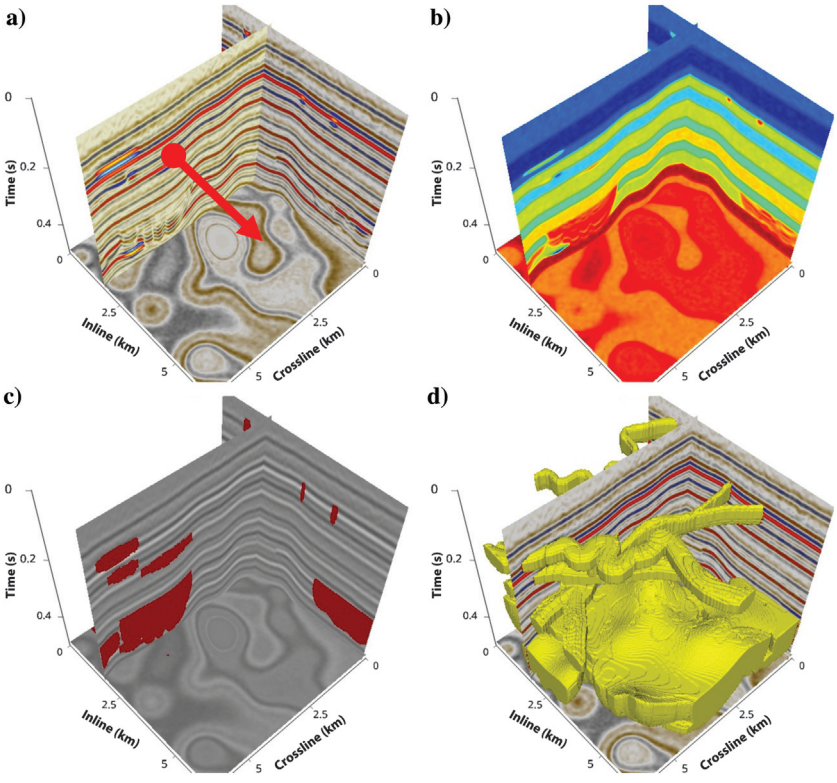

Figure 4. The multiple canvas feature of CIGVis. A single canvas can be divided into multiple subcanvases, each independently displaying different data sets from cigChannel (Wang et al., 2024): (a) seismic data, (b) impedance, (c) seismic data with channel labels (using an overlay), and (d) seismic data with the channel body. When a seismic slice is dragged along the red arrow in (a), the corresponding slices in other subcanvases move in a similar manner under a unified camera perspective. This can facilitate the comparative analysis of different data sets and results.

Jupyter

The 3D canvas of CIGVis offers two types of APIs to support desktop and Jupyter environments. Given that Jupyter is a web-based tool, it has certain performance limitations, such as

the ability to drag slices in real time and rendering speed. In the Jupyter environment, CIGVis is based on Plotly, and its rendering results may differ from those in the desktop environment without a reduction in quality. In fact, some users might prefer the rendering effects in the Jupyter environment. Figure 6 shows the visualization result of the F3 data in the Jupyter environment, where well-log data are displayed as lines. Although the desktop and Jupyter environments share a set of function signatures, CIGVis automatically calls the corresponding code execution based on the environment. However, there may be slight differences in the function parameters (especially those related to the colorbar) between these two environments. In the future, we aim to improve the functionalities in the Jupyter environment and strive for a unified API across both environments.

Colormap

The Colormap module of CIGVis not only provides various colormaps that can be used as extensions to Matplotlib but also implements conversion functions between Matplotlib, VisPy, and Plotly. Moreover, the module defines some practical color processing functions, such as color blending, custom colormaps, handling discrete colormaps (often related to the colorbar), and transparency settings. Using these functions, one can create rich and varied 3D visualization effects. For example, as shown in Figure 7, CIGVis achieves overlaying relative geologic time (RGT) with seismic data, partial RGT with seismic data, faults with seismic data, and a comprehensive overlay of the partial RGT and faults with seismic data by appropriately setting the transparency.

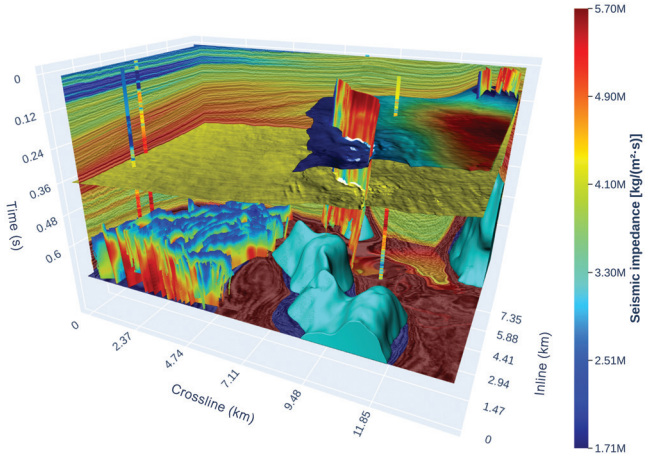

Figure 6. Visualization of the F3 data set using the Jupyter environment API. The meaning of the colors is consistent with Figure 2e. This example demonstrates CIGVis's capability within the Jupyter environment, showing that it can achieve functionality similar to the desktop version, although there are some differences in the features and performance.

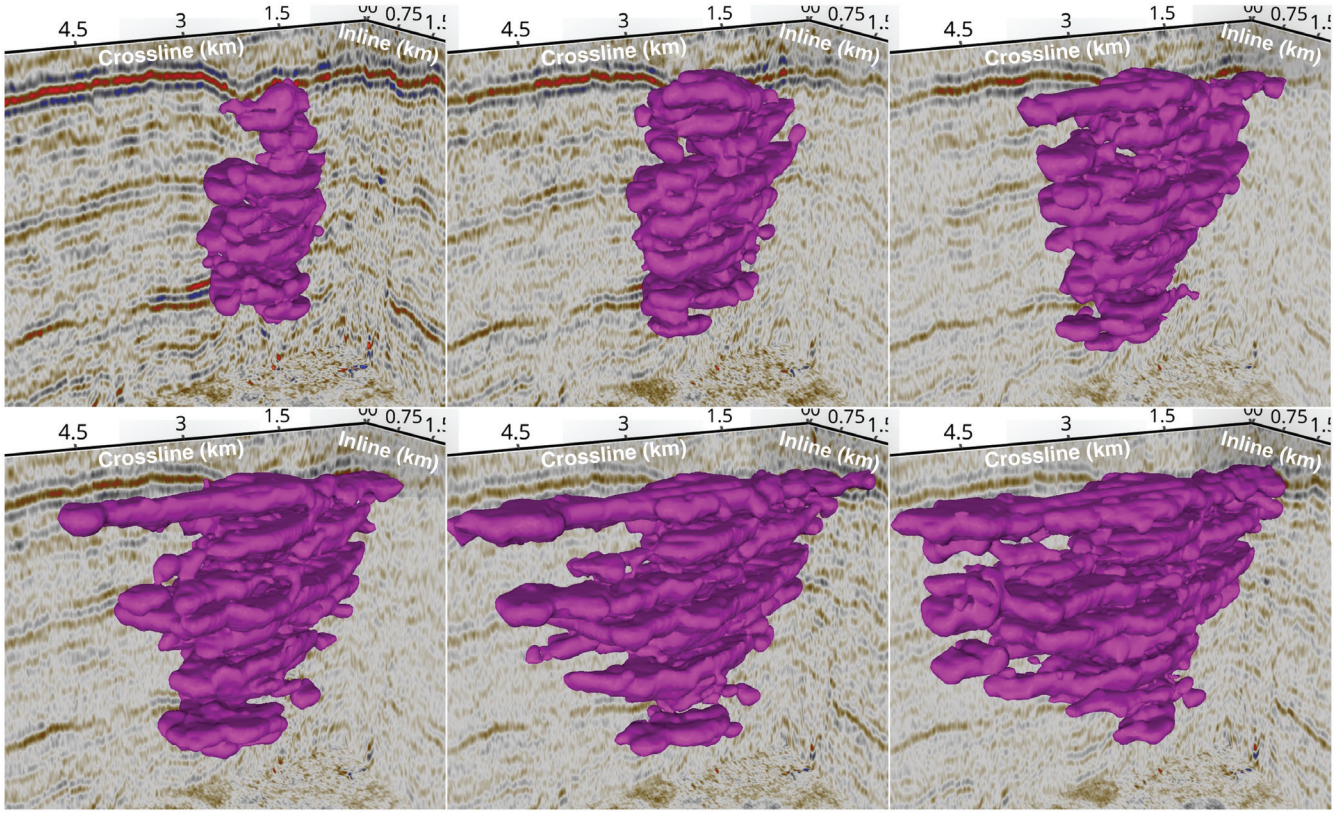

Figure 5. An example using CIGVis to display the distribution of CO_2 plumes over different years, based on the work of Sheng et al. (2023). A GIF image dynamically shows this capability in supplemental S1. This demonstrates how this functionality might be suited for analyzing 4D geophysical models in relation to monitoring problems.

Data format and large data sets

As a Python tool focused on visualization, CIGVis primarily accepts numerical Python (NumPy) (Harris et al., 2020) arrays or lists of NumPy arrays as input. To accommodate the fact that many users may have data in SEG-Y format, the open-source community offers several excellent tools for converting SEG-Y to NumPy arrays. These tools include cigsegy (Li, 2024), segyio (Equinor, 2024), seg-sak (Hallam, 2024), and others. Users can choose from these options to meet their specific needs (based on speed considerations, we recommend cigsegy and segyio). The combination of CIGVis and any of these SEGY conversion tools can cater to the requirements of most users. In addition, open volume data store (OpenVDS) is an emerging efficient data format particularly suitable for visualizing large data sets (such as seismic data exceeding 50 GB). CIGVis provides an interface similar to NumPy, capable of calling OpenVDS to read the data and simulate NumPy's slicing behavior. Tests on a MacBook Air 2022 with 8 GB of memory have shown that large data sets, specifically 81 GB in the OpenVDS format (which would be approximately 108 GB in the SEG-Y format), can be visualized rapidly using CIGVis, typically within 10 s, as shown in the supplemental S1 GIF image. Overall, the visualization of these three types of data only differs slightly during the reading stage. An example of how to read different data formats in CIGVis is shown in the following:

```
# The example of how to read a 3D data set in different common formats


# import Python libraries
import numpy as np
import cigvis
from cigsegy import SegyNP # or segyio, segysak

# Example of 3D data in binary format
shape = (660, 781, 1001)
d = np.fromfile('demo.dat', np.float32).reshape(shape)


# Example of 3D data in SEG-Y format
# Any of the SEG-Y conversion tools is OK
d = SegyNP('demo.segy', iline = 5, xline = 21)


# Example of 3D data in VDS format
d = cigvis.io.VDSReader('demo.vds')
# showing the 3D data set: d
nodes = cigvis.create_slices(d)
cigvis.plot3D(nodes)
```

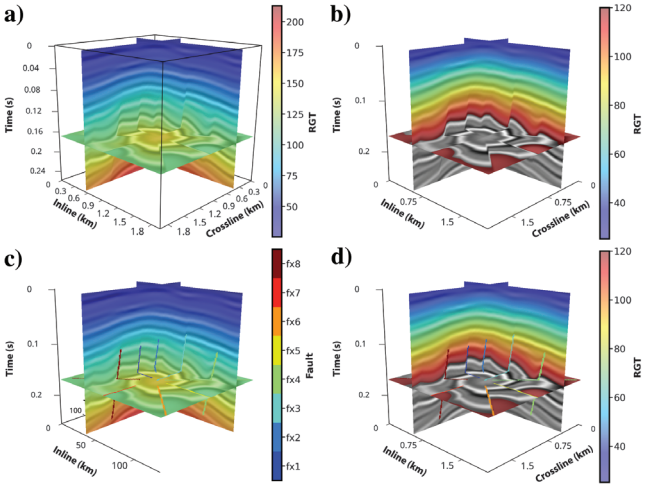

Figure 7. The CIGVis allows for diverse visual effects by adjusting the colormap module. (a) Seismic data displayed in conjunction with RGT; (b) adjusting the colormap upper limit to mask RGT values greater than 120, which typically correspond to areas of less interest; (c) overlaying fault visualization results based on (a) with each fault represented in a different color; and (d) overlaying fault visualization results based on (b). In addition, CIGVis supports various styles for coordinate axes and unit displays, which can be easily switched through parameters.

2D canvas

The 2D canvas module in CIGVis is a simple implementation using Matplotlib, featuring several functions for visualizing geophysical data. This module was added to ensure the completeness of CIGVis's visualization capabilities and to offer convenience and reference for geophysicists who are new to using Matplotlib. The 2D canvas meets the common visualization needs in geophysical research and publications. It can be used in conjunction with Matplotlib to create multiple subplots, supporting the display of

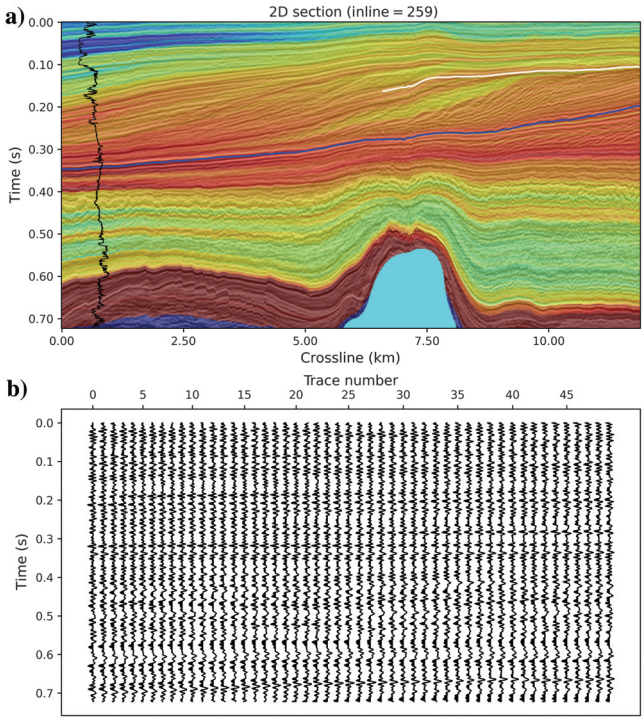

Figure 8. A demo of the 2D canvas module. (a) The inline 259 section of the F3 data set corresponding to a cross section from Figure 2e. It contains an overlay of seismic and impedance, with the white line indicating an unconformity, the blue line representing horizons, the black curve depicting the impedance well log, and the shape of the salt body highlighted in cyan. (b) The first 50 traces of the inline 259 section.

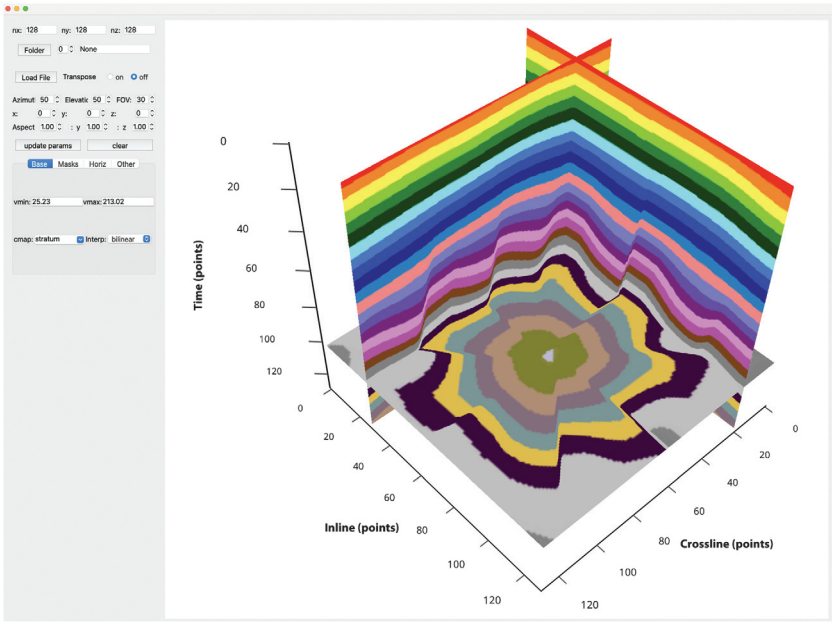

Figure 9. A simple GUI interface in CIGVis displaying a synthetic RGT volume.

common 2D images and 1D signals found in geophysics. Figure 8a shows the seismic image of the F3 data set at inline 259, with blue and white lines indicating the yellow horizon and unconformity surface from Figure 2f, respectively. The black curve on the left represents a well log at trace 33. Figure 8b shows the first 50 traces in the inline 259 section, a display style frequently seen in geophysical research.

GUI support

Although developing a professional GUI is not the primary goal of CIGVis (we focus more on simplicity and practicality for researchers), we still offer some basic GUI functionalities, and Figure 9 shows an example of the GUI. These simple GUIs provide limited features, such as allowing users to quickly visualize 3D data by dragging and dropping files. These functional aspects specifically include the ability to quickly and conveniently visualize 3D geophysical data as well as the capability to adjust slices and perspectives interactively. Users can perform these tasks without needing to write any code, allowing for efficient data exploration and visualization in real time, which is especially useful for users who are not familiar with programming or need a fast visualization solution.

CONCLUSION

This paper presents CIGVis, an open-source Python tool tailored for visualizing multidimensional geophysical data. Addressing a growing need within the geophysics community, CIGVis excels in real-time interactive visualization, making multidimensional data analyses more intuitive and accessible. Designed for wide accessibility, CIGVis is compatible with major operating systems and supports desktop and Jupyter environments

CIGVis's interactive features, including rotation, panning, magnifying, and dragging slices, allow users to interact deeply with various geophysical data types, such as 3D seismic data, faults, horizons, and well logs. Its multicanvas functionality enables simultaneous visualization across multiple subcanvases under a unified camera perspective. This is particularly valuable for comparative analysis, offering enhanced insights into complex geologic data sets.

Future development efforts will focus on enhancing the coordinate system to support the true geodetic coordinates and improving compatibility with diverse data formats, particularly those exported from commercial software such as Petrel. Minor patches aimed at refining visualization outcomes and improving the user experience will also be included. In addition, an interactive AI visualization toolset is being developed, allowing users to provide real-time feedback to AI models, such as refining fault interpretation results through direct interaction.

The code and some GIFs for all the figures in this paper can be found at https://cigvis.readthedocs.io/en/latest/gallery/index.html.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (grant no. 42374127) and the China Postdoctoral Science Foundation (grant no. 2024M753147) provided by Y. Li. We also thank the USTC Supercomputing Center for computational resources. We are grateful to the reviewers, especially editor-in-chief A. Guitton and associate editor M. Ravasi, for their valuable comments, and we appreciate the feedback from J. Yang, Z. Liu, and C. Dai.

DATA AND MATERIALS AVAILABILITY

As a fully open-source software, CIGVis adopts the Massachusetts Institute of Technology License (MIT license). Its source code is available on GitHub at https://github.com/JintaoLee-Roger/cigvis, and we also maintain detailed documentation at https://cigvis.readthedocs.io/en/latest/, where users can find a wealth of demos. The code and some GIFs for all the figures in this paper can be found at https://cigvis.readthedocs.io/en/latest/gallery/index.html.

REFERENCES

Campagnola, L., A. Klein, E. Larson, C. Rossant, and N. P. Rougier, 2015, VisPy: Harnessing the GPU for fast, high-level visualization: Proceedings of the 14th Python in Science Conference.

Das, V., A. Pollack, U. Wollner, and T. Mukerji, 2019, Convolutional neural network for seismic impedance inversion: Geophysics, **84**, no. 6, R869–R880, doi: 10.1190/geo2018-0838.1.

Equinor, 2024, SegyIO: Fast python library for SEG-Y files, https://github.com/equinor/segyio, accessed 30 May 2024.

GeoSoftware, 2024, Hampson-Russell-seismic reservoir characterization software, https://www.geosoftware.com/hampsonrussell, accessed 10 January 2024.

Hale, D., 2012, Fault surfaces and fault throws from 3D seismic images: 82nd Annual International Meeting, SEG, Expanded Abstracts, doi: 10.1190/segam2012-0734.1.

Hallam, A., 2024, SEGY-SAK, https://trhallam.github.io/segysak/, accessed 30 May 2024.

Halliburton, 2024, Landmark software solutions, https://www.halliburton.com/en/software, accessed 10 January 2024.

Harris, C. R., K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, and R. Kern, 2020, Array programming with NumPy: Nature, **585**, 357–362, doi: 10.1038/s41586-020-2649-2.

Hunter, J. D., 2007, Matplotlib: A 2D graphics environment: Computing in Science & Engineering, **9**, 90–95, doi: 10.1109/MCSE.2007.55.
Inc, 2024, Plotly: Interactive, scientific data visualization in your web browser, https://plotly.com/about-us/, accessed 10 January 2024.
Kitware, 2024, Visualization toolkit (VTK), https://vtk.org, accessed 10 January 2024.
Li, J., 2024, CIGSEGY: A tool for exchanging data between SEG-Y format and NumPy array inside Python environment, https://github.com/JintaoLee-Roger/cigsegy, accessed 10 January 2024.
Li, J., X. Wu, and Z. Hu, 2021, Deep learning for simultaneous seismic image superresolution and denoising: IEEE Transactions on Geoscience and Remote Sensing, **60**, 1–11, doi: 10.1109/TGRS.2021.3057857.
Li, J., X. Wu, Y. Ye, C. Yang, Z. Hu, X. Sun, and T. Zhao, 2023, Unsupervised contrastive learning for seismic facies characterization: Geophysics, **88**, no. 1, WA81–WA89, doi: 10.1190/geo2022-0148.1.
Lorensen, W. E., and H. E. Cline, 1987, Marching cubes: A high resolution 3D surface construction algorithm: ACM SIGGRAPH Computer Graphics, **21**, 163–169, doi: 10.1145/37402.37422.
Lynch, S., 2008, More than meets the eye — A study in seismic visualization: Ph.D. thesis, University of Calgary.
Lynch, S., 2023, High visual resolution interpretation: The case for virtual seismic reality: The Leading Edge, **42**, 541–549, doi: 10.1190/tle42080541.1.
Schlumberger, 2024, Petrel E & P software platform, https://www.software.slb.com, accessed 10 January 2024.
Sheng, H., X. Wu, X. Sun, and L. Wu, 2023, Deep learning for characterizing CO_2 migration in time-lapse seismic images: Fuel, **336**, 126806, doi: 10.1016/j.fuel.2022.126806.
Shi, Y., 2024, Seismic-canvas: Interactive 3D seismic visualization tool, https://github.com/yunzhishi/seismic-canvas, accessed 10 January 2024.
Sullivan, B., 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, doi: 10.21105/joss.01450.
Wang, G., X. Wu, and W. Zhang, 2024, cigChannel: A massive-scale 3D seismic dataset with labeled paleochannels for advancing deep learning in seismic interpretation: Earth System Science Data Discussions, **2024**, 1–27, doi: 10.5194/essd-2024-131.
Wu, X., and D. Hale, 2015, 3D seismic image processing for unconformities: Geophysics, **80**, no. 2, IM35–IM44, doi: 10.1190/geo2014-0323.1.
Wu, X., L. Liang, Y. Shi, and S. Fomel, 2019, FaultSeg3D: Using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation: Geophysics, **84**, no. 3, IM35–IM45, doi: 10.1190/geo2018-0646.1.
Yu, S., J. Ma, and W. Wang, 2019, Deep learning for denoising: Geophysics, **84**, no. 6, V333–V350, doi: 10.1190/geo2018-0668.1.

Biographies and photographs of the authors are not available.