`ImageLogViewer`: An open-source solution for exploring images from micro-resistivity and ultrasonic boreholes

Rodrigo Piva[1], Matheus A. Cruz[1], Paola Braga[1], Rodrigo Dias[1], Paulo Siqueira[1,3], Willian A. Trevizan[2], Candida M. de Jesus[2], André M. Souza[1,4], Rodrigo Salvador[3], Leandro Fernandes[3], Flávia C. Bernardini[3], Elaine P. M. de Sousa[4], Daniel de Oliveira[3], Marcos Bedo[3]

[1]Wikki Soluções em Engenharia
Parque Tecnológico – Rio de Janeiro/RJ – Brazil

{rodrigo.piva, matheus.altomare, paola.braga}@wikki.com.br

{rodrigo.dias, paulo.costa, andre.souza}@wikki.com.br

[2]Centro de Pesquisas Leopoldo Américo Miguez de Mello
Petróleo Brasileiro S.A – Rio de Janeiro/RJ – Brazil

{williantrevizan, candida.jesus}@petrobras.com.br

[3]Institute of Computing
Fluminense Federal University – Niterói/RJ – Brazil

{laffernandes, fcbernardini, salvador, danielcmo, marcosbedo}@ic.uff.br

[4]Institute of Mathematics and Computer Science
University of São Paulo – São Carlos/SP – Brazil

{andre.moreira.souza@usp, parros@icmc.usp}.br

Abstract. *Borehole image logs are critical for characterizing reservoirs in the petroleum industry. Exploring this data is challenging because image logs are extensive in size and demand multiple measurements to be juxtaposed side-by-side, such as resistivity and acoustic data. Despite its importance, no open-source applications exist to analyze such data. In this demonstration, we present* `ImageLogViewer`[1]*, a Python-based tool for visualizing image logs. It can simultaneously display acoustic and dynamic/static micro-resistivity data with multiple display modes and custom windowing to facilitate the analysis of structures, e.g., fractures and cavities. It also integrates image processing, machine, and deep learning models for classifying and segmenting regions of interest.*

1. Introduction

Borehole image logs are a valuable source of geological information that plays a critical, prospective role in the petroleum industry. Such logs provide fine-grained sedimentary and structural information essential for geologists to assess the properties of subsurface rock formations accurately [Brekke et al. 2017, Lai et al. 2018]. Researchers can gain valuable insights by analyzing the images obtained from borehole image logs, such as orientation, size, shape, and distribution of geological features, including faults, bedding

[1]Source code available at `https://github.com/Rodrigo-P/ImageLogViewer`

planes, and other structural elements. Additionally, borehole image logs can reveal depositional environments within rocks and other sedimentary structures, which are not otherwise visible [Brekke et al. 2017, Nian et al. 2017]. For instance, researchers can use these logs to study the history of deformation and tectonic activity that has shaped the subsurface formations. Therefore, accurate interpretation of borehole image logs is critical to petroleum extraction as they provide clues to understand the geological processes.

Accordingly, image log scans are the initial and baseline tools for exploring subsurface data in scientific and industrial fields. Many commercial data viewers[2] have gained popularity by offering advanced features that meet the specific requirements of geoscientists and engineers. However, such applications are proprietary, and open-source alternatives are non-existent. While commercial solutions provide robust support and advanced features, their closed-source nature limits customization and integration with other tools. This demonstration fits the need to balance proprietary offerings and the benefits of open-source solutions to promote collaboration within the scientific community.

We highlight interpreting borehole image logs without proper visualization tools imposes a series of limitations, as distinguishing important textures and structures from the vast amount of data with varying levels of quality is nearly impossible with simple file exploration. Moreover, experts rely on sophisticated analysis techniques and previous expertise to differentiate between formations, faults, and fractures.

Several potential methods can be used to enhance image visualization in the context of image logs. One such technique, commonly used in medical image processing, is image windowing [Gonzalez and Wintz 1977, Kamalian et al. 2016]. This technique re-quantifies the pixels' intensity to a smaller interval based on user-provided *center* and *width* parameters, thus facilitating the identification of relevant regions and details[3]. Furthermore, integrating machine learning techniques to classify and segment relevant structures automatically could assist researchers in their analyses. In addition to visualization, users can use Machine Learning (ML) and Deep Learning (DL) techniques to automatically classify and segment relevant structures. Integrating such methods in an image log viewer could help researchers quickly annotate large datasets.

In this demonstration, we present `ImageLogViewer`, an open-source tool for visualizing different image logs. Our tool includes an interface for visualizing acoustic and resistivity logs. It has preset windowing options to highlight nodules and cavities, and windowing parameters can be dynamically adjusted. `ImageLogViewer` includes functionalities such as geological texture categorization and machine-learning-based cavity detection. In addition to a default setup, `ImageLogViewer` allows the integration of models for classifying and segmenting structures in image tiles.

2. Development Details

`ImageLogViewer` [1] is a cross-platform desktop application designed to facilitate the analysis of borehole image logs. It consists of interconnected modules – See Figure 1. Leveraging Python *tkinter* and *CustomTkinter* packages, the application's Graphical User Interface (GUI) efficiently manages the retrieval and visualization of user-provided image

[2]Examples are ALT's WellCAD, Baker Hughes' STAR Imager Service, Geoactive's Subsurface Interpretation Software, SLB's Techlog Wellbore Imaging, and GeoLog from U.S. Geological Survey (USGS).

[3]The borehole image logs in this demonstration contain mono-color pixels in a 2^{16} scale.

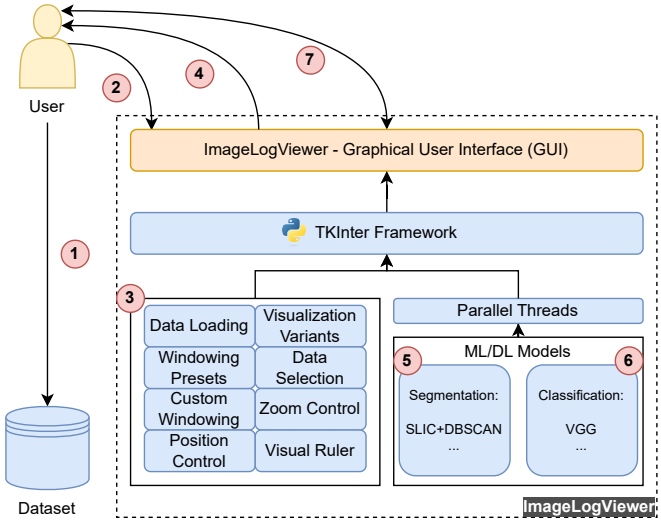

Figure 1. Overview of `ImageLogViewer`'s architecture.

log datasets – Figure 1 (1–3). Alongside *tkinter* and *CustomTkinter*, other dependencies for the application's functionality are: *NumPy OpenCVPandas*and *Matplotlib*. Additional ML and DL dependencies are *PyTorch scikit-learn* and *scikit-image*. After the application fully loads the image log data, it uses *tkinter*'s modules to display image representations to the user with the selected visualization parameters – Figure 1(4).

As an advanced feature, the user can process data with segmentation and classification models – Figure 1(5–6). The user can provide these models as additional Python scripts and pre-trained model parameters (besides `ImageLogViewer` default), whereas `ImageLogViewer` displays the results to the user – Figure 1 (7).

Demonstration data source[1]**.** Image logs are typically classified data, so we developed an approach for generating synthetic resistivity data for this demonstration. The primary technique for synthetic data generation we used is a 2D variation of the Perlin noise [Perlin 1985]. This noise generator technique allowed us to create textures with a frequency parameter that controls the variation speed in the output structures. Accordingly, we combined different noise generation stages to replicate image log tiles with a brecciated texture (Figure 2).

The process involved generating images with Perlin noise and then quantizing them within three colors to characterize general structures. Then, we combined multiple textures with diverse frequency parameters to generate image sub-structures with different sizes. Next, we quantized the images with five colors and applied Gaussian blur to reduce the presence of borders in the generated data. Finally, we added vertical bands to replicate the missing data between sensor pads, similar to the actual data.

3. Demonstration

Upon accessing `ImageLogViewer`, users can visualize the main GUI panel (Figure 3). The system provides a range of functionalities for efficient analysis of acoustic and

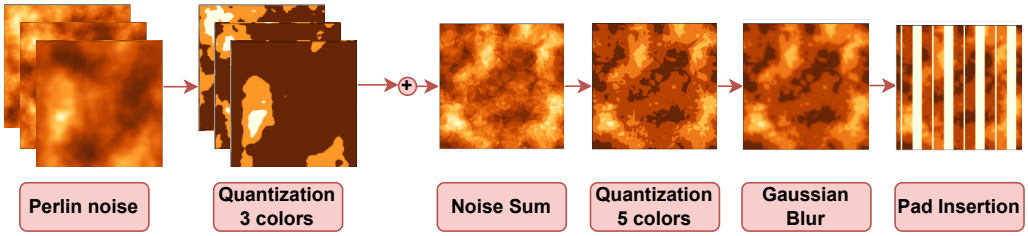

Figure 2. The generation of synthetic image log data with brecciated texture.

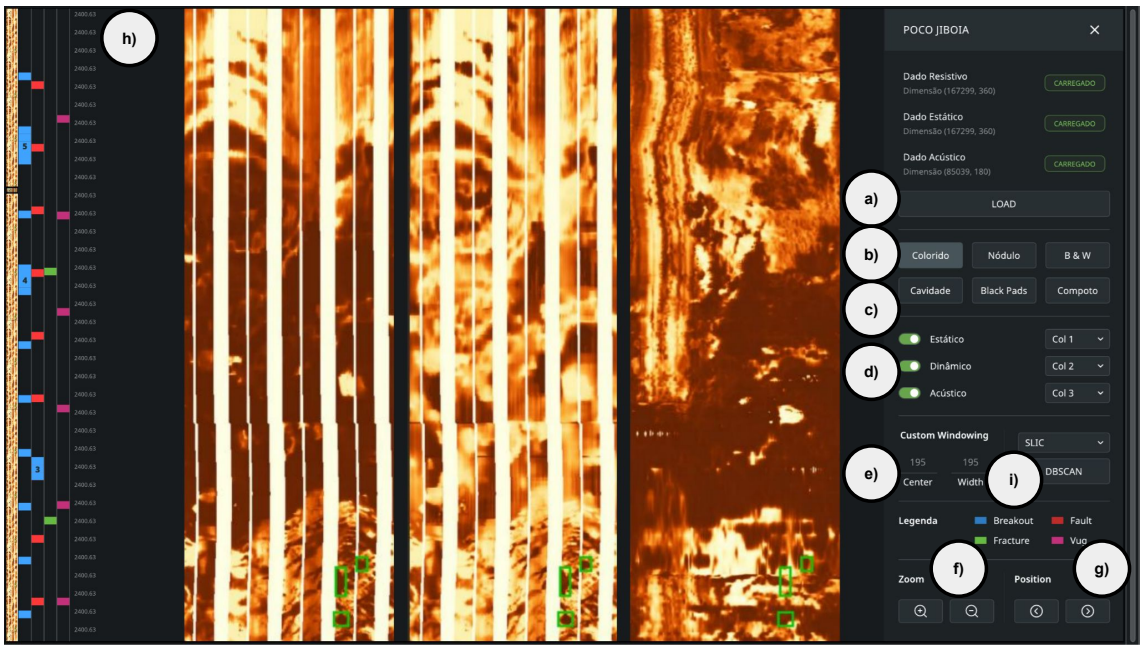

Figure 3. `ImageLogViewer`'s main interface.

static/dynamic resistivity image logs. Users can (a) load their data sets for analysis. A borehole image log usually includes around 432MB of data, which may vary depending on the borehole's depth. Multiple visualization variants (b) are available to optimize contrast and aid analysis, allowing users to select the most suitable representation for their data – Figure 4 (1–3). Predefined windowing presets (c) can adjust display settings to highlight nodules and cavities – Figure 4 (4–6).

Additionally, users can selectively visualize individual types of loaded logs (d) or customize windowing parameters (e) to tailor visualization to their needs, adjusting center and width parameters as required. Zoom (f) and position (g) controls enable detailed examination and navigation within the logs, allowing users to explore broad trends and fine-scale features. Furthermore, a visual ruler tool (h) aids navigation and provides borehole depth information from the original loaded files.

Lastly, `ImageLogViewer` facilitates the execution of ML models (i) for further analysis. In this last functionality, advanced users may define an ML model inside a Python script, which `ImageLogViewer` will call using the currently loaded tales as *PyTorch* tensors. As of this publication, we have support for general ML models (such as those from *scikit-learn* and deep learning models with *PyTorch*. For

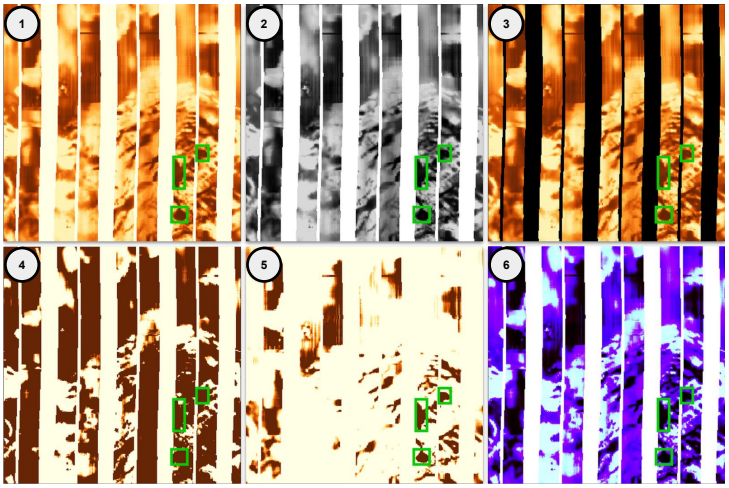

Figure 4. `ImageLogViewer`'s visualization with the following variants: (1) Original, annotated log; (2) Black and White (B&W); (3) Black Pads (for resistivity data); (4) Windowing preset for nodules; (5) Windowing preset for cavities; (6) Composite windowing preset for nodules and cavities.

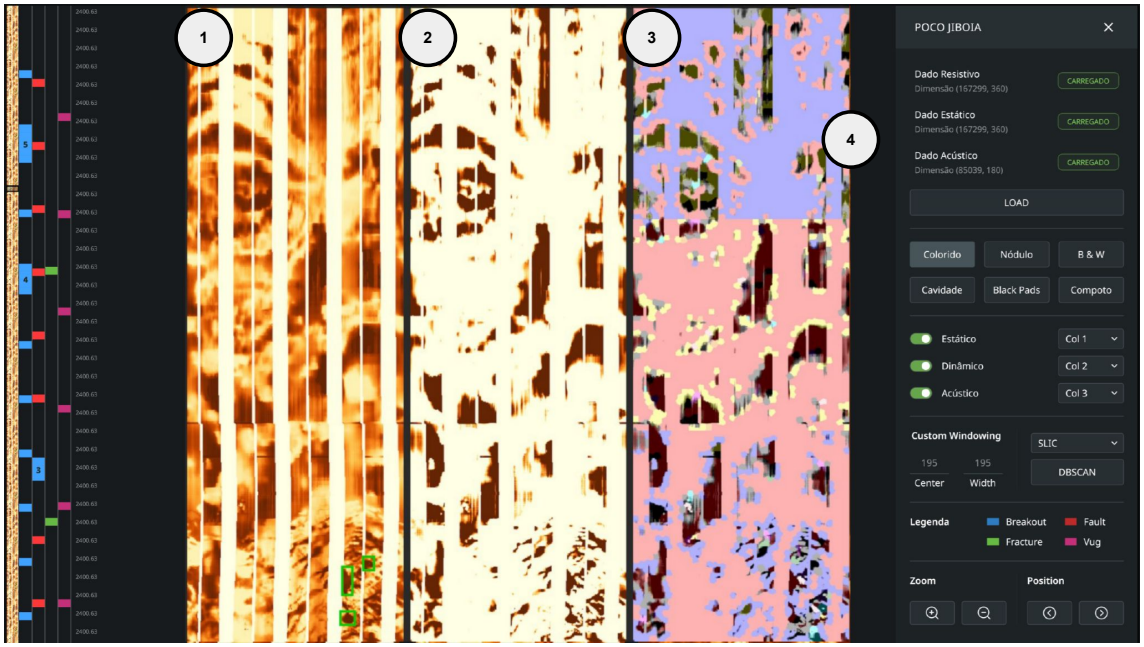

Figure 5. An example of `ImageLogViewer` segmentation with the original image (1), the image with custom windowing (2), VGG-16-based texture classification (background colors in (3)), and the SLIC+DBSCAN for cavity detection (4).

classification, the model's output should be the probabilities of each class. On the other hand, for segmentation, the output should be a matrix of labels with the same shape as the input image. `ImageLogViewer` preset parameters are *(i)* a fine-tuned VGG-16 [Simonyan and Zisserman 2014] for geological texture classification and *(ii)* the SLIC [Achanta et al. 2012] superpixel-based construction methods with texture features clustered by DBSCAN [Schubert et al. 2017] for cavity detection. An example of segmentation with Simple Linear Iterative Clustering (SLIC) superpixels and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is shown in Figure 5.

4. Conclusions

In this demonstration, we presented `ImageLogViewer`, an open-source alternative to commercial, license-based solutions for visualizing micro-resistivity and borehole acoustic image log data. The application includes several features built to assist geologists in analyzing the structures and textures found within the data. We highlight the solution can be easily extended by researchers adding their ML models inside `ImageLogViewer`, allowing the visualization of the outputs produced by their models.

Limitations and Future Work. `ImageLogViewer` has some limitations, such as the performance of classification and segmentation functionalities are tied to their language-specific library implementations, *i.e.*, computationally expensive models may take several seconds to execute. Therefore, it is essential to consider using techniques to enhance the efficiency of these models' implementations, including multiprocessing and caching. Additionally, the current `ImageLogViewer` version does not include functions for annotating image logs. In future work, adding user-based annotation functionalities would significantly extend the application's usability.

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

Brekke, H., MacEachern, J. A., Roenitz, T., and Dashtgard, S. E. (2017). The use of microresistivity image logs for facies interpretations: An example in point-bar deposits of the McMurray Formation, Alberta, Canada. *AAPG Bulletin*, 101(5):655–682.

Gonzalez, R. and Wintz, P. (1977). *Digital Image Processing*. Advanced book program: Addison-Wesley. Addison-Wesley Publishing Company, Advanced Book Program.

Kamalian, S., Lev, M. H., and Gupta, R. (2016). Chapter 1 - Computed tomography imaging and angiography – principles. In Masdeu, J. C. and González, R. G., editors, *Handbook of Clinical Neurology*, volume 135 of *Neuroimaging Part I*, pages 3–20. Elsevier.

Lai, J., Wang, G., Wang, S., Cao, J., Li, M., Pang, X., Han, C., Fan, X., Yang, L., He, Z., and Qin, Z. (2018). A review on the applications of image logs in structural analysis and sedimentary characterization. *Marine and Petroleum Geology*, 95:139–166.

Nian, T., Wang, G., and Song, H. (2017). Open tensile fractures at depth in anticlines: A case study in the Tarim basin, NW China. *Terra Nova*, 29(3):183–190.

Perlin, K. (1985). An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, page 287–296, New York, NY, USA. Association for Computing Machinery.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems*, 42(3):19:1–19:21.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.