The Journal of Open Source Software

Ratio Imaging Analyzer (RIA): A Lightweight, Standalone Python Tool for Portable Ratiometric Fluorescence Analysis

Kui Wang [1¶]

1 Center for Excellence in Brain Science and Intelligence Technology (Institute of Neuroscience), Chinese Academy of Sciences, 320 Yue Yang Road, Shanghai, 200031 P.R.China ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Ratiometric fluorescence imaging stands as a cornerstone technique in modern quantitative biology. By measuring the ratio of fluorescence intensities at two distinct wavelengths, this method renders measurements independent of sensor concentration, optical path length, and uneven illumination, making it the "gold standard" for quantifying dynamic intracellular events (Tao et al., 2023). Its application spans from monitoring ion dynamics (e.g., Calcium, pH) to tracking essential metabolites (e.g., Tryptophan, ATP) using an expanding toolkit of genetically encoded biosensors.

Ratio Imaging Analyzer (RIA) is a lightweight, open-source desktop application designed to streamline the processing of such ratiometric imaging data. Unlike complex image processing libraries that require scripting skills, RIA bridges the gap between raw data and biological insight through a user-friendly Graphical User Interface (GUI). It empowers researchers to perform motion correction, dynamic background subtraction, interactive thresholding, and real-time Region of Interest (ROI) analysis on standard personal computers, facilitating rapid hypothesis testing and data exploration.

The Journal of Open Source Software

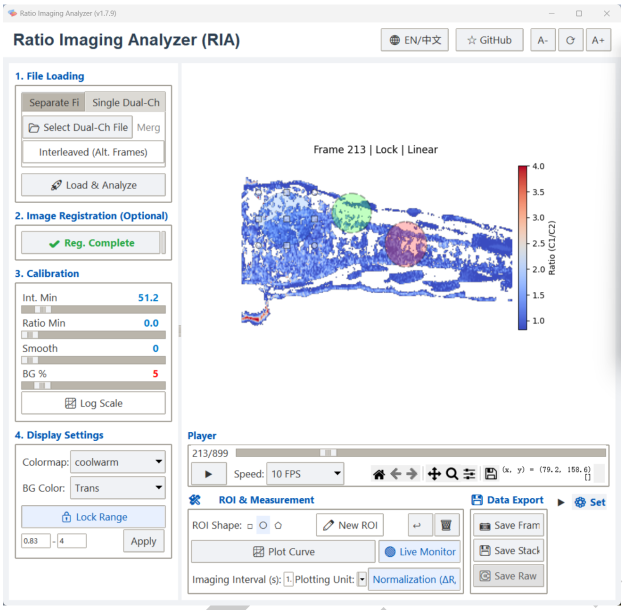

Figure 1: The main user interface of RIA. The layout consists of three main sections: the left panel for data loading, image registration, and parameter calibration; the central canvas showing the generated pseudocolor ratiometric image with a colorbar; and the bottom dashboard for time-lapse playback, ROI management, and data export.

Statement of Need

Despite the widespread adoption of ratiometric sensors, quantitative analysis of time-lapse data remains a bottleneck. A significant gap exists for user-friendly tools tailored to "wet-lab" experimental biologists who lack programming expertise.

While commercial software packages (e.g., MetaFluor) are powerful, they are often tied to acquisition workstations via hardware dongles, limiting accessibility. Conversely, open-source alternatives have struggled to provide a modern, integrated experience. For instance, legacy ImageJ plugins like *Color Ratio Plus* are largely deprecated and difficult to access. Critically, these older tools often lack essential dynamic features required for modern analysis, such as adjustable Look-Up Tables (LUTs), real-time background subtraction, and motion correction.

RIA addresses these challenges by offering a lightweight, **"all-in-one"** standalone executable. It streamlines the entire workflow—from loading raw TIFF stacks to generating publication-quality ratiometric figures or movies—without requiring complex plugin installations or script assembly. Specifically, RIA:

1. **Eliminates Technical Barriers**: Researchers can process data on standard personal laptops (Windows) without setting up Python environments.
2. **Integrates Essential Tools**: Unlike piecemeal solutions, RIA bundles motion correction (ECC), interactive thresholding, tunable visualization, and real-time ROI plotting into a single interface.
3. **Enhances Efficiency**: Utilizing vectorized operations from `NumPy` (Harris et al., 2020) and the C++ backend of `OpenCV` (Bradski, 2000), RIA processes large datasets instantly, ensuring the smooth operation of the softwave, and enabling rapid tests of hypothesis. # Implementation

RIA is developed in Python 3, utilizing `tkinter` for a native, dependency-minimal Graphical User Interface (GUI). The software architecture separates the UI logic from the core processing engine to ensure responsiveness. Recent updates have focused on minimizing the software footprint (~73 MB) and maximizing processing speed.

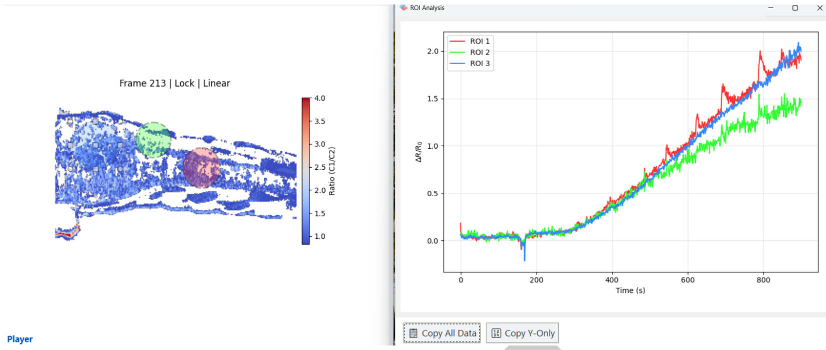

Figure 2: Interactive analysis workflow. Selecting a Region of Interest (ROI) on the image (left) triggers the instant calculation and plotting of the mean ratio over time (right).

Key technical features include:

- **High-Performance Image Processing**: To achieve real-time performance on consumer-grade CPUs, RIA leverages `OpenCV-headless` (Bradski, 2000). This replaces heavier dependencies like `SciPy`, significantly reducing application size and startup time.
- **Motion Correction**: RIA integrates the Enhanced Correlation Coefficient (ECC) algorithm, offering both high speed and correction accuracy, to automatically align image stacks and compensate for sample drift during long-term imaging sessions. Importantly, we validated this approach using real in vivo imaging datasets, demonstrating its robustness and reliability under practical experimental conditions.
- **Normalized Convolution for Smoothing**: Standard Gaussian blurring can introduce artifacts at image boundaries containing `NaN` values (masked background). RIA implements a custom **Normalized Convolution** algorithm using OpenCV primitives. This approach computes the weighted average of valid pixels only, preventing the propagation of `NaN` values and preserving data integrity at cellular edges.
- **Interactive Visualization**: The plotting engine, powered by `Matplotlib` (Hunter, 2007), features a threaded observer pattern. This allows users to draw and drag ROIs on the video stream with instant updates to the time-course trace, facilitating rapid identification of physiological events.
- **Memory Optimization**: Large TIFF stacks are handled using memory-efficient IO strategies, maintaining data in `uint16` format until calculation to minimize RAM usage.

Acknowledgements

We acknowledge the open-source community for maintaining the foundational libraries that make this tool possible, specifically NumPy, Matplotlib, and OpenCV.

References

Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*, *25*, 120–125.

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., & others. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Tao, R., Wang, K., Chen, T., & others. (2023). A genetically encoded ratiometric indicator for tryptophan. *Cell Discovery*, *9*(1), 106. https://doi.org/10.1038/s41421-023-00608-1