microPublication BIOLOGY

4/2/2026 - *Open Access*

JuNkIE-CLImax: exploring multidimensional images in notebooks and the terminal

Rodrigo Fernandez-Gonzalez[1,2,3,4§], Raymond Hawkins[1,2]

[1]Institute of Biomedical Engineering, University of Toronto, Toronto, ON, CA

[2]Ted Rogers Centre for Heart Research, Translational Biology and Engineering Program, University of Toronto, Toronto, ON, CA

[3]Department of Cell and Systems Biology, University of Toronto, Toronto, ON, CA

[4]Developmental and Stem Cell Biology Program, Hospital for Sick Children, Toronto, ON, CA

[§]To whom correspondence should be addressed: rodrigo.fernandez.gonzalez@utoronto.ca

Abstract

Modern microscopy generates multidimensional images. Graphical user interfaces allow visualization of multidimensional images, but image exploration is significantly harder in text-based environments. These include Jupyter notebooks, popular for the distribution of data analysis pipelines; and the terminal, often the only user interface available when accessing images in remote servers. We developed JuNkIE-CLImax, a pair of multidimensional image explorers designed to work in Jupyter notebooks and the terminal, respectively. JuNkIE-CLImax are written in Python, using Rust extensions to optimize performance; they are platform agnostic; and they provide application programming interfaces that enable integration into complex image visualization and analysis pipelines.

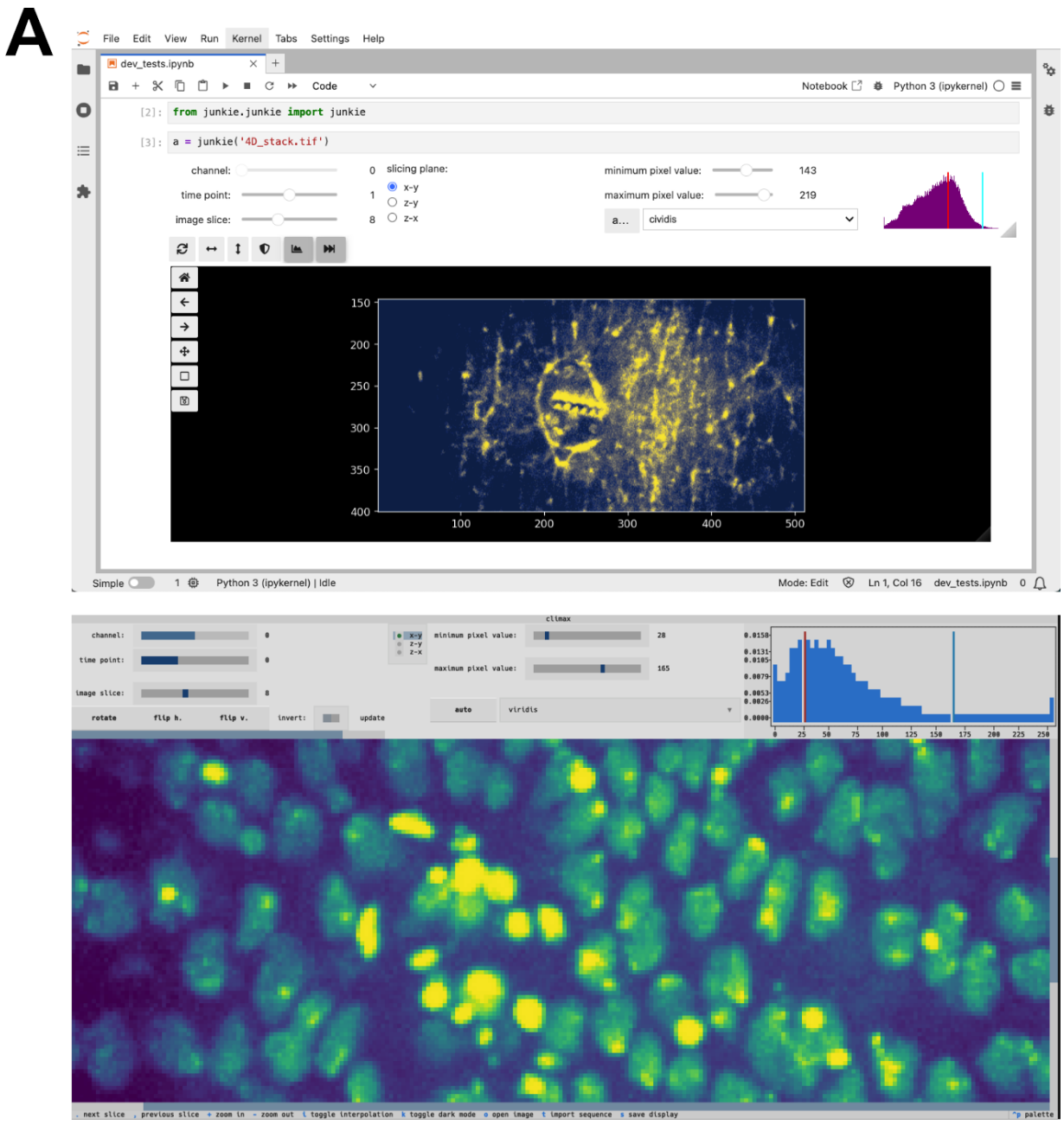

Figure 1. JuNkIE-CLImax: multidimensional image explorers for notebooks and the terminal:

(A) JuNkIE (top) and CLImax (bottom) user interfaces displaying multidimensional images in a notebook and the terminal, respectively. JuNkIE-CLImax provide widgets to browse different image dimensions; expose multiple colormaps, including scientific colormaps; and conduct simple image operations, including rotation, flipping, zooming, reslicing and contrast adjustment. **(B)** Character-based image rendering. Sample image (left) and corresponding *rich* Style

strings to render the image in the terminal (right), using a half-block character to display each pair of vertically-adjacent pixels. For instance, the Style parsed from "*rgb(255,0,0) on rgb(0,255,255)*", when associated with the half-block character, renders a red half-block on a cyan background, representing a cyan pixel in the top row and a red pixel in the bottom row, respectively. End-of-line characters ("\n") indicate the ends of pairs of rows.

Description

Quantitative image analysis of microscopy datasets is a central tool in modern cell and developmental biology (Meijering and Cappellen, 2007; Senft et al., 2023). The advent of machine learning methods has established Python as the *de facto* standard for the analysis of biomedical images (Jacquemet, 2021; Morgado et al., 2024). Jupyter notebooks (Granger and Pérez, 2021) provide a beginner-friendly environment that enables prototyping and distribution of complex image processing pipelines (von Chamier et al., 2021). Platforms such as Google Colab or DigitalOcean provide free access to high-end computing capacity to execute notebooks, thus "democratizing" the use of resources that would otherwise be too costly. Notebooks facilitate data exploration and increase the reproducibility of data processing pipelines (Kluyver et al., 2016; Pimentel et al., 2019; Samuel and Mietchen, 2024). The leading position of notebooks for data analysis is cemented by the recent availability of large language models within notebooks (Qiu, 2023), which further facilitates development, debugging and execution of image processing and quantification routines. Thus, notebooks are a central element in the analysis of biomedical image data.

Both biological and medical imaging generate multidimensional data. Confocal and light sheet microscopies, for instance, allow collection of three-dimensional stacks (X, Y, Z) of different fluorescent reporters (wavelength), over the duration of a biological process (time). Multimodality imaging, such as correlative confocal and electron microscopies (Casares-Arias et al., 2021), confocal and super-resolution (Xiang et al., 2018), or the combination of anatomical and functional data in medical imaging (Pichler et al., 2008), introduces a sixth imaging dimension. Processing and analysis of multidimensional image datasets often requires image exploration at many stages in which rapid visualization and browsing are key steps. Two-dimensional images can be easily displayed in notebooks, *e.g.* with *matplotlib* (Hunter, 2007). However, few tools exist to display, browse and interact with multidimensional images in a notebook (Haase, 2021), and some of the available options provide complex user interfaces and functionality that prevent light-weight, inline multidimensional image browsing (Sofroniew et al., 2025).

Microscopy images are generally archived in remote backup servers. The advent of cloud-based storage services has only increased this trend. Remote and cloud-based servers often do not provide a graphical user interface, but simply a terminal-based, command line interface. Terminals can display text-based information about the images (file name, size, date, etc.), but do not allow image visualization. The lack of tools for image display on the terminal makes it hard to select images either for further processing or to generate figures for publication.

Here we introduce JuNkIE-CLImax, a pair of open source tools (https://bitbucket.org/rfg_lab/junkie and https://bitbucket.org/rfg_lab/climax) for the display and interactive visualization of multidimensional microscopy images in environments traditionally considered to be document or text-based. JuNkIE is a JUpyter NotebooK Image Explorer that enables the opening, display and simple manipulation of multidimensional images in Jupyter notebooks. CLImax, a Command Line IMAge eXplorer, allows users to open, visualize and browse multidimensional images on the terminal, using character-based strategies for image rendering.

JuNkIE-CLImax: multidimensional image explorers

JuNkIE-CLImax can load multidimensional images in the multi-page TIFF format often found in microscopy (Besson et al., 2019), as well as images stored in the increasingly popular *zarr* array format (Miles, 2015). JuNkIE-CLImax can also import file sequences that represent different dimensions (*e.g.* time or wavelength) of a multidimensional image. Internally, images are stored using *numpy*, a package for numerical computation, and its *ndarray* data structure to represent multidimensional arrays (Harris et al., 2020). JuNkIE-CLImax display a two-dimensional view of a multidimensional image (Fig. 1A). Both packages provide sliders to select a specific Z-plane, time point and channel from the multidimensional image stored in memory, using both mouse and keyboard-based interactions. JuNkIE-CLImax continuously update the displayed image in real time in response to user input. To enable use in settings with reduced compute power (*e.g.* remote storage servers), it is possible to disable the continuous update of the display, such that updates only occur when the user interaction is completed (*e.g.* when the mouse button is released, but not while the mouse is dragged). XY, XZ or YZ sectioning planes can be selected. Additionally, JuNkIE-CLImax provide functionality for simple image manipulation. This includes rotation and flipping (both horizontal and vertical), colormap selection and inversion (including all the colormaps available in *matplotlib* (Crameri et al., 2020; Crameri, 2023)), and both manual and automated contrast adjustment.

JuNkIE

JuNkIE displays images using the *Figure* class in *matplotlib*. A *matplotlib Figure* can hold different plot elements, including images. The *matplotlib Figure* provides a toolbar with options to pan, zoom in and out, reset the view, or save a

snapshot (Fig. 1A, top).

JuNkIE provides a simple application programming interface (API). Using the JuNkIE API it is possible to open images from disk, by providing a file or folder path, or to view volumetric information already loaded as an *ndarray*. Substrings can be used to distinguish which files in an image sequence store specific dimensions (*e.g.* different image channels). Lastly, the colormap used to initially display an image, as well as the size of the figure can be determined with input parameters when invoking JuNkIE.

JuNkIE is distributed with a test suite using *pytest* (Krekel et al., 2004) and the *nbval* plugin for notebook-based testing (Cortes-Ortuno et al., 2016), with code coverage over 90%.

CLImax

To display images on the terminal, we created a character-based image renderer. The image renderer uses *rich*, a Python package for text formatting that allows the use of coloured characters in the terminal (McGugan, 2019; Burns, 2022). Our image renderer takes a two-dimensional *ndarray* as its input, and produces a sequence of *rich* Segments, each consisting of a character and some formatting (a *Style*), punctuated by end-of-line characters at the end of each image row (Fig. 1B). *rich Styles* determine the colours used to display a character and its background. Colours are defined by the corresponding red, green and blue (rgb) components, specified as a string. Initially, we rendered each pixel using a space character (' '), assigning the pixel value as the background colour and no foreground colour. Because characters on the terminal are rectangular (elongated along the vertical axis), we rendered each image column twice to obtain square pixels. As a consequence, image rendering was relatively slow, requiring 380±23 ms (mean±standard deviation) to render a 512x672 pixel image on an Apple M3 Max equipped with 48 GB of RAM.

To optimize image rendering, we first reduced the number of terminal characters necessary to display a pixel. We took advantage of the half-block character ('▄') to visualize pairs of vertically-adjacent pixels (same column, consecutive rows) with a single terminal character (Fig. 1B). For each pair of pixels, we rendered the bottom pixel by assigning its value as the foreground colour of a half-block character, and the top pixel by assigning its value as the background colour of the same half-block character (Fig. 1B). This approach allowed us to display each pixel with half a character, in contrast to the previous method, which required two characters per pixel. We further accelerated rendering by creating a *Style* cache that used a Python dictionary to store the *rich Styles* corresponding to parsed colour-strings. After these optimizations, rendering speed increased by 57% (163±13 ms/image) and memory consumption decreased by a factor of four.

To further speed up image display, we implemented the hot-loop of the renderer in Rust (Matsakis and Klock II, 2014). The Rust code implements a helper function with two inputs, a two-dimensional image, and a colour lookup table that contains the string representation of each of the colours available in the current colormap. The helper function then scans the image columns, two rows at a time, generating the *Style* string for the character that will display a pair of vertically-adjacent pixels. We parallelized the scan over the image columns using the Rust crate *rayon* (Matsakis and Stone, 2014) to create a parallel iterator. Overall, Rust-based image rendering was 11% faster (146±13 ms/image) than our fastest Python implementation.

We designed the CLImax user interface with *textual* (McGugan, 2021) (Fig. 1A, bottom). *textual* is a Python package for the development of user interfaces and applications for the terminal. *textual* provides different widgets (*e.g.* buttons, dropdown boxes, text labels, etc.). We created a new *ImagePanel* widget that displays an image using the character-based renderer described above. *textual* apps provide a command palette that includes options to change the theme used to display the application, save a screenshot, or show a help panel. In addition, CLImax provides options to open a new image, zoom in or out, display the image with bilinear interpolation (particularly useful when zooming in/out), or save the current display.

CLImax can be invoked from the command line with different options. Similar to the JuNkIE API, the CLImax command line parameters enable image selection and substring and colour map specification, as well as initial zoom level, particularly important given that image rendering in the terminal is computationally intensive, with a cost that increases with the number of pixels to be rendered.

CLImax is also tested with *pytest*, using the *asyncio* plugin (Seifert, 2015) to test asynchronous user interaction. CLImax tests provide code coverage greater than 90%.

Acknowledgements: We are grateful to Negar Balaghi, Alexandra Korolov and Willow Peterson for comments on the manuscript.

References

Besson Sb, Leigh R, Linkert M, Allan C, Burel JM, Carroll M, et al., Swedlow. 2019. Bringing Open Data to Whole Slide Imaging. Lecture Notes in Computer Science,Digital Pathology : 3-10. DOI: 10.1007/978-3-030-23937-4_1

Burns,D. (2022) rich-pixels, https://github.com/darrenburns/rich-pixels.

Casares-Arias J, Alonso MA, San Paulo l, González MaU. 2021. Correlative confocal and scanning electron microscopy of cultured cells without using dedicated equipment. STAR Protocols 2: 100727. DOI: 10.1016/j.xpro.2021.100727

von Chamier L, Laine RF, Jukkala J, Spahn C, Krentzel D, Nehme E, et al., Henriques. 2021. Democratising deep learning for microscopy with ZeroCostDL4Mic. Nature Communications 12: 10.1038/s41467-021-22518-0. DOI: 10.1038/s41467-021-22518-0

Cortes-Ortuno,D. *et al.* (2016) nbval, https://github.com/computationalmodelling/nbval.

Crameri,F. (2023) Scientific colour maps, https://zenodo.org/records/8409685. DOI: 10.5281/zenodo.8409685

Crameri F, Shephard GE, Heron PJ. 2020. The misuse of colour in science communication. Nature Communications 11: 10.1038/s41467-020-19160-7. DOI: 10.1038/s41467-020-19160-7

Granger BE, Perez F. 2021. Jupyter: Thinking and Storytelling With Code and Data. Computing in Science & Engineering 23: 7-14. DOI: 10.1109/MCSE.2021.3059263

Haase,R. (2021) stackview, https://github.com/haesleinhuepf/stackview.

Harris CR, Millman KJ, van der Walt SfJ, Gommers R, Virtanen P, Cournapeau D, et al., Oliphant. 2020. Array programming with NumPy. Nature 585: 357-362. DOI: 10.1038/s41586-020-2649-2

Hunter JD. 2007. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering 9: 90-95. DOI: 10.1109/MCSE.2007.55

Jacquemet G. 2021. Deep learning to analyse microscopy images. The Biochemist 43: 60-64. DOI: 10.1042/bio_2021_167

Kluyver Thomas , Ragan-Kelley Benjamin , Pérez Fernando , Granger Brian , Bussonnier Matthias , Frederic Jonathan , et al., Jupyter Development Team. 2016. Jupyter Notebooks - a publishing format for reproducible computational workflows. Positioning and Power in Academic Publishing: Players, Agents and Agendas : 10.3233/978-1-61499-649-1-87. DOI: 10.3233/978-1-61499-649-1-87

Krekel,H. *et al.* (2004) pytest, https://github.com/pytest-dev/pytest.

Matsakis,N.D. and Klock II,F.S. (2014) The rust language. In, *ACM SIGAda Ada Letters*. ACM, pp. 103–104.

Matsakis,N.D. and Stone,J. (2014) Rayon: A data parallelism library for Rust, https://github.com/rayon-rs/rayon/.

McGugan,W. (2019) rich, https://github.com/Textualize/rich.

McGugan,W. (2021) Textual, https://textual.textualize.io/.

Meijering E, Cappellen Gv. 2007. Quantitative Biological Image Analysis. Principles and Practice,Imaging Cellular and Molecular Biological Functions : 45-70. DOI: 10.1007/978-3-540-71331-9_2

Miles,A. (2015) zarr, https://github.com/zarr-developers/zarr-python.

Morgado L, Gómez-de-Mariscal E, Heil HS, Henriques R. 2024. The rise of data-driven microscopy powered by machine learning. Journal of Microscopy 295: 85-92. DOI: 10.1111/jmi.13282

Pichler BJ, Judenhofer MS, Pfannenberg C. 2008. Multimodal Imaging Approaches: PET/CT and PET/MRI. Handbook of Experimental Pharmacology,Molecular Imaging I : 109-132. DOI: 10.1007/978-3-540-72718-7_6

Pimentel JF, Murta L, Braganholo V, Freire J. 2019. A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks. 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR) : 507-517. DOI: 10.1109/MSR.2019.00077

Qiu,D.L. (2023) Jupyter AI, https://github.com/jupyterlab/jupyter-ai.

Samuel S, Mietchen D. 2024. Computational reproducibility of Jupyter notebooks from biomedical publications. GigaScience 13: 10.1093/gigascience/giad113. DOI: 10.1093/gigascience/giad113

Seifert,M. (2015) pytest-asyncio, https://github.com/pytest-dev/pytest-asyncio.

Senft RA, Diaz-Rohrer B, Colarusso P, Swift L, Jamali N, Jambor H, et al., Cimini. 2023. A biologist's guide to planning and performing quantitative bioimaging experiments. PLOS Biology 21: e3002167. DOI: 10.1371/journal.pbio.3002167

Sofroniew,N. *et al.* (2025) napari: a multi-dimensional image viewer for Python, https://zenodo.org/doi/10.5281/zenodo.3555620. DOI: 10.5281/zenodo.3555620

Xiang W, Roberti MJ, Hériché JK, Huet Sb, Alexander S, Ellenberg J. 2018. Correlative live and super-resolution imaging reveals the dynamic structure of replication domains. Journal of Cell Biology 217: 1973-1984. DOI: 10.1083/jcb.201709074

4/2/2026 - *Open Access*

Funding: This work was supported by the Natural Sciences and Engineering Research Council of Canada (418438-13), the Canadian Institutes of Health Research (186188), the Canada Foundation for Innovation (30279), the Translational Biology and Engineering Program of the Ted Rogers Centre for Heart Research., and the University of Toronto EMHSeed Program. RFG is the Canada Research Chair in Quantitative Cell Biology and Morphogenesis.

Conflicts of Interest: The authors declare that there are no conflicts of interest present.

Author Contributions: Rodrigo Fernandez-Gonzalez: conceptualization, funding acquisition, methodology, project administration, resources, software, supervision, validation, visualization, writing - original draft, writing - review editing, investigation. Raymond Hawkins: software, writing - review editing.

Reviewed By: Ignacio Arganda-Carreras

History: Received January 23, 2026 **Revision Received** March 11, 2026 **Accepted** March 27, 2026 **Published Online** April 2, 2026 **Indexed** April 16, 2026

Citation: Fernandez-Gonzalez R, Hawkins R. 2026. JuNkIE-CLImax: exploring multidimensional images in notebooks and the terminal. microPublication Biology. 10.17912/micropub.biology.002120