Pycro-Manager: open-source software for customized and reproducible microscope control

To the Editor — Cutting-edge innovations in biological microscopy are increasingly blurring the line between data acquisition and data analysis. Computational microscopy and machine-learning-based methods take this paradigm to an extreme, often producing raw measurements that are not human-interpretable without postprocessing[1–3]. Furthermore, new data-adaptive imaging methods rely on data processing during acquisition to actively control various hardware settings of the microscope.

Testing new ideas and applying them for biological discovery is often impeded by a lack of control software that is capable of meeting demands for speed and performance, integrating new and diverse types of hardware, providing the flexibility to adapt in real time to the data being captured, and providing user-friendly programming interfaces. As a result, researchers often end up developing custom software that works only with specific instruments, using closed-source and/or proprietary programming languages. In addition to slowing development, this lack of a consistent software ecosystem creates barriers to reproduction and replication of new techniques. These include the increased likelihood of bugs in code that is not reused and tested in different contexts, the burden on new users of understanding new code bases, and the extra work of disentangling high-level code that is not properly abstracted from the underlying hardware. In many fields the Python programming language — in particular, the NumPy and SciPy ecosystem[4,5] — has emerged as a common framework for reproducing research results that rely on complex, multilayer scientific workflows in a portable, scriptable form[6] that is accessible to researchers with various levels of programming experience.

The central challenge of such a consolidated approach in microscopy is the diversity of hardware used in different types of microscopes, ranging from custom-built components on optical tables to turnkey commercial systems. μManager[7,8] is an essential tool for controlling microscopes, thanks to an extensive library of 'device adapters' for controlling different types

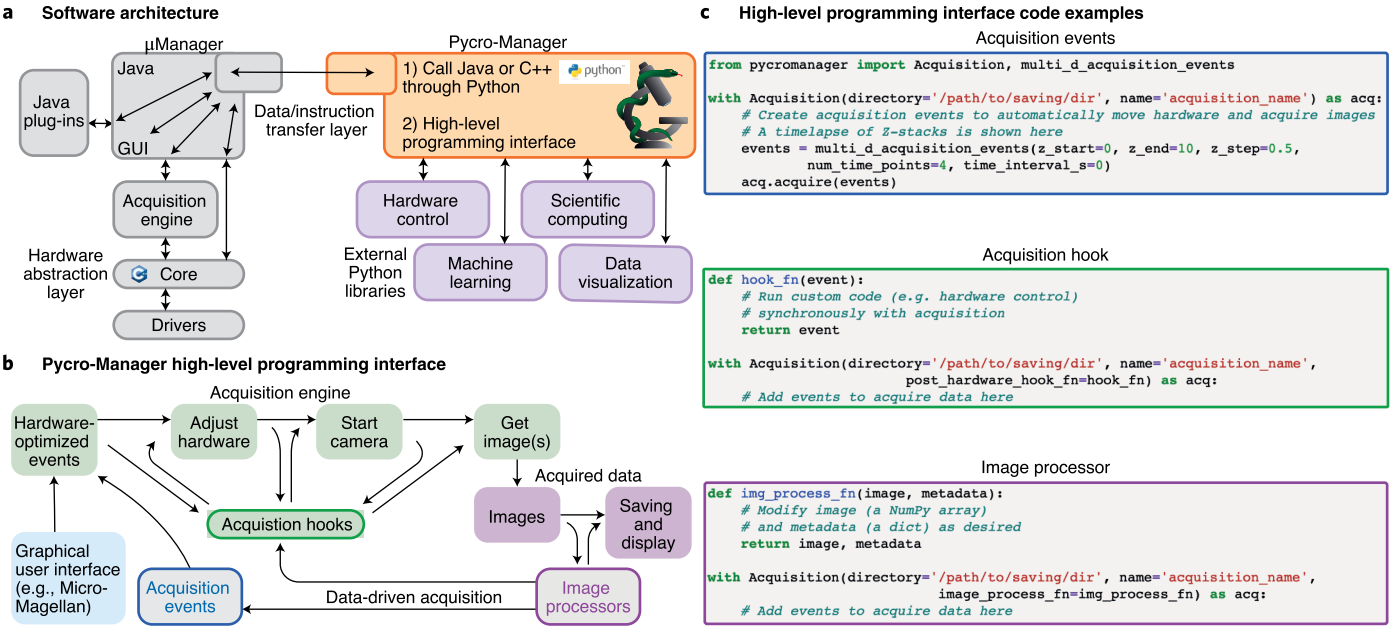

Fig. 1 | Pycro-Manager. a, Software architecture overview. Grey: the existing parts of μManager provide generic microscope control abstracted from specific hardware, a graphical user interface (GUI), a Java plug-in interface, and an acquisition engine that automates various aspects of data collection. Orange: Pycro-Manager enables access to these components through Python over a network-compatible transport layer, as well as a concise, high-level programming interface for acquiring data. Purple: these provide integration of data acquisition with Python libraries for hardware control, data visualization, scientific computing and so forth. **b**, Pycro-Manager's high-level programming interface. The data acquisition process in Pycro-Manager starts with a source of acquisition events (blue), from either programming or a GUI. These events are passed to the acquisition engine (green), which optimizes them to take advantage of hardware triggering where available, sends instructions to hardware and acquires images. The resulting images are then saved and displayed in the GUI (magenta). The three main abstractions of the Pycro-Manager high-level programming interface (acquisition events, acquisition hooks, and image processors) enable fine-grained control and customization of this process. **c**, Code examples. Code snippets for implementing acquisition events (blue), acquisition hooks (green) and image processors (magenta). ("Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation, used with permission from the Foundation. C++ logo from http://isocpp.org/about).

of hardware, from cameras to complete microscopes, with a single programming interface. Community contributions of device adapters, plug-ins and scripts provide hundreds of developer-years of microscopy automation.

Despite the power of these libraries, which are written in C++ and Java, they are often difficult to integrate with the latest developments in computer vision and scientific computing, which most readily interface with NumPy. This not only increases the difficulty of developing techniques that rely on both customized data capture and data analysis, but also hinders the dissemination and adoption of these techniques by fragmenting them across multiple tools and programming languages and making them harder to understand and test in new contexts.

Many groups are developing microscopy control software that is written mostly or entirely in Python, including, Python Microscope[9], Python Microscopy Environment (http://www.python-microscopy.org/) and Tormenta[10] (a complete list is at https://doi.org/10.5281/zenodo.4433237). However, these tools are currently limited to specific types of microscopy (for example, super-resolution) and/or support a relatively small number of hardware devices. Thus, despite the potential benefits of such a design, they face an enormous task to match µManager's flexibility and device support.

To address these needs, we present here Pycro-Manager. Pycro-Manager is built on a translation layer that converts Java objects, functions and data into language-agnostic messages that are reconstituted as Python objects or functions and as NumPy arrays (Fig. 1a and Supplementary Information). As a result, the existing capabilities of µManager can be called as if they had been written in Python, without forcing users to learn Java, forcing Java developers to learn Python, or abandoning of the relative strengths of either language (Supplementary Information).

In addition, Pycro-Manager provides a new library of high-level programmatic building blocks (in Python) for customizing data acquisition. These are designed to facilitate creation of complex data acquisitions with concise, readable code while maintaining the flexibility required for customization and the ability to handle multi-terabyte datasets acquired at speeds over 1 GB s^{-1} (Supplementary Information).

The high-level application programming interface (API) contains three main abstractions: acquisition events, acquisition hooks and image processors (Fig. 1b,c). These building blocks can be used individually or combined for more complex applications. Acquisition events enable customized instructions for how to adjust hardware when acquiring images and how to index those images along arbitrary axes (for example, time, *z*, etc.) for storage and display. For common microscopy workflows like time-lapses and *z*-stacks, events can be automatically generated by high-level functions (Fig. 1c). Alternatively, they can be created manually from nested Python lists and dictionaries to allow a greater degree of customization. Acquisition hooks enable the execution of arbitrary Python code concurrently at various stages of the data acquisition process. Image processors give access to the image data as soon as it is acquired, for modification or for use in data-driven feedback loops on the acquisition process.

The combination of Jupyter notebooks (https://jupyter.org/) — interactive documents that consolidate text, code, equations and results — and the Pycro-Manager high-level API provides a means to describe the full workflow of a research project, from data acquisition to analysis to results, thereby facilitating understanding, dissemination and reproduction of new microscopy technologies. For techniques that rely heavily on computation, these notebooks can be as valuable as (or even more valuable than) their corresponding research papers.

In the Supplementary Information and online documentation (https://pycro-manager.readthedocs.io/en/latest/#), we provide tutorials describing the basic features of Pycro-Manager, as well as Jupyter notebooks outlining sample applications in microscope alignment, light sheet microscopy, integrated sample processing, computational microscopy and machine learning, and the control of microscopes over networks.

The source code and documentation for Pycro-Manager can be found in the Supplementary Information, and future updates will be posted at https://github.com/micro-manager/pycro-manager and https://pycro-manager.readthedocs.io/en/latest/#.

Reporting summary

Further information on experimental design is available in the Nature Research Reporting Summary linked to this paper. ❐

Henry Pinkard[1,2,3,4] ✉, Nico Stuurman[5], Ivan E. Ivanov[6], Nicholas M. Anthony[7], Wei Ouyang[8], Bin Li[9], Bin Yang[6], Mark A. Tsuchida[9], Bryant Chhun[6], Grace Zhang[1], Ryan Mei[1], Michael Anderson[10,11], Douglas P. Shepherd[12], Ian Hunt-Isaak[13], Raymond L. Dunn[14], WiebkeJahr[15], SaulKato[16,17,18], LoïcA.Royer[6], Jay R. Thiagarajah[10,11], Kevin W. Eliceiri[9], Emma Lundberg[8], Shalin B. Mehta[6] and Laura Waller[1,3]

[1]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA, USA. [2]Computational Biology Graduate Group, University of California Berkeley, Berkeley, CA, USA. [3]Berkeley Institute for Data Science, University of California, Berkeley, Berkeley, CA, USA. [4]University of California San Francisco Bakar Computational Health Sciences Institute, San Francisco, CA, USA. [5]Howard Hughes Medical Institute and University of California, San Francisco, San Francisco, CA, USA. [6]Chan Zuckerberg Biohub, San Francisco, CA, USA. [7]Department of Biomedical Engineering, Northwestern University, Evanston, IL, USA. [8]Science for Life Laboratory, School of Engineering Sciences in Chemistry, Biotechnology and Health, KTH – Royal Institute of Technology, Stockholm, Sweden. [9]University of Wisconsin at Madison, Madison, WI, USA. [10]Division of Gastroenterology, Hepatology and Nutrition, Boston Children's Hospital, Boston, MA, USA. [11]Harvard Medical School, Boston, MA, USA. [12]Center for Biological Physics and Department of Physics, Arizona State University, Tempe, AZ, USA. [13]Applied Physics Graduate Program, John Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. [14]Neuroscience Graduate Program, Department of Neurology, Department of Cell and Tissue Biology, University of California, San Francisco, San Francisco, CA, USA. [15]Institute of Science and Technology Austria, Klosterneuburg, Austria. [16]Department of Neurology, University of California, San Francisco, San Francisco, CA, USA. [17]Weill Institute for Neuroscience, University of California, San Francisco, San Francisco, CA, USA. [18]Kavli Institute for Fundamental Neuroscience, University of California, San Francisco, San Francisco, CA, USA.
✉*e-mail: hbp@berkeley.edu*

Published online: 5 March 2021
https://doi.org/10.1038/s41592-021-01087-6

References
1. Boominathan, V. & Adams, J. K. *IEEE Signal Process. Mag.* **33**, 23–35 (2016).
2. Gustafsson, M. G. L. *J. Microsc.* **198**, 82–87 (2000).
3. Pavani, S. R. P. & Piestun, R. *Opt. Express* **16**, 22048–22057 (2008).
4. Harris, C. R. et al. *Nature* **585**, 357–362 (2020).
5. Virtanen, P. et al. *Nat. Methods* **17**, 261–272 (2020).
6. Poldrack, R. A., Gorgolewski, K. J. & Varoquaux, G. *Annu. Rev. Biomed. Data Sci.* **2**, 119–138 (2019).
7. Edelstein, A., Amodaj, N., Hoover, K., Vale, R. & Stuurman, N. *Curr. Protoc. Mol. Biol.* **92**, 1–17 (2010).
8. Edelstein, A. D. et al. *J. Biol. Methods* **1**, 10 (2014).
9. Miguel, D. et al. Preprint at *bioRxiv* (2021).
10. Barabas, F. M., Masullo, L. A. & Stefani, F. D. *Rev. Sci. Instrum.* **87**, 126103 (2016).

Acknowledgements

We thank S. van der Walt and K. Marchuk for discussion during development. This project was funded by Packard Fellowship and Chan Zuckerberg Biohub Investigator Awards to L.W.; STROBE: A NSF Science and Technology Center; an NSF Graduate Research Fellowship awarded to H.P.; a Berkeley Institute for Data Science/UCSF Bakar Computational Health Sciences Institute Fellowship awarded to H.P. with support from the Koret Foundation, the Gordon and Betty Moore Foundation, and the Alfred P. Sloan Foundation to the University of California, Berkeley. K.W.E., B.L. and M.T. were funded by the Chan Zuckerberg Initiative and NIH grant P41GM135019.

Author contributions

Initial concept: H.P. Early development and planning: H.P., N.S., L.W. Development of core libraries: H.P., N.S., I.E.I., N.M.A, M.T., B.C., I.H., S.B.M., L.W. Application development, testing, bug finding, and supervision: H.P., I.E.I., W.O., B.L., B.Y., G.Z., R.M., M.A.,D.P.S., W.J., R.D., S.K., L.A.R., J.R.T., K.W.E., E.L., S.B.M, L.W. Manuscript writing: H.P., N.S., I.E.I., S.M., L.W.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41592-021-01087-6.

Peer review information *Nature Methods* thanks Robert Haase and Dominic Waithe for their contribution to the peer review of this work.

nature research

Corresponding author(s): Henry Pinkard

Code and Software Submission Checklist

Prior to submitting your work to Nature Research, we strongly recommend that you ask at least one colleague who is unfamiliar with your software to install the tool(s), follow the instructions, and provide feedback. This process will help ensure that reviewers will also be able to run your software.

You must submit all required content as a single zip file prior to peer review or provide a link where editors and reviewers can access all required content.

Required content

- [x] Compiled standalone software and/or source code
- [x] A small (simulated or real) dataset to demo the software/code

A README file that includes:

1. System requirements
   - [x] All software dependencies and operating systems (including version numbers)
   - [x] Versions the software has been tested on
   - [x] Any required non-standard hardware
2. Installation guide
   - [x] Instructions
   - [x] Typical install time on a "normal" desktop computer
3. Demo
   - [x] Instructions to run on data
   - [x] Expected output
   - [x] Expected run time for demo on a "normal" desktop computer
4. Instructions for use
   - [x] How to run the software on your data
   - [ ] (OPTIONAL) Reproduction instructions

   We encourage you to include instructions for reproducing all the quantitative results in the manuscript.

Additional information

Describe your software's license for use. We strongly recommend using a license approved by the Open Source Initiative.

BSD 3-Clause "New" or "Revised" License

Provide a link to the code in an open source repository (when available).

https://github.com/micro-manager/pycro-manager

Your manuscript should include a complete, detailed description of the code's functionality (i.e. pseudocode).

Please indicate where this is found:

- [ ] Main text
- [ ] Methods section
- [x] Elsewhere (specify):

https://pycro-manager.readthedocs.io/en/latest/features.html

Examples of well-structured software packages

1. https://github.com/neurodata-papers/MGC
2. https://github.com/neurodata-papers/LOL
3. https://www.nature.com/nbt/journal/v34/n6/abs/nbt.3569.html#supplementary-information
4. https://www.nature.com/nature/journal/v548/n7669/full/nature23463.html#extended-data
   https://github.com/yasharhezaveh/Ensai
5. https://www.nature.com/nbt/journal/v34/n11/full/nbt.3685.html#supplementary-information
   https://github.com/IFIproteomics/LFQbench