Enderscope.py: A library for computational imaging using the EnderScope automated microscope

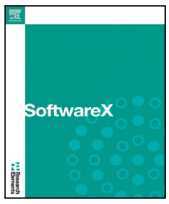
Sirine Gharbi, Emeric Poiraud, Hugo Le Guenno, Erwan Grandgirard, Charly Rousseau, Niamh Burke, Jerome Mutterer, David Rousseau

SoftwareX 31 (2025) 102210

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx



Original Software Publication

Enderscope.py: A library for computational imaging using the EnderScope automated microscope

Sirine Gharbi [a], Emeric Poiraud [a], Hugo Le Guenno [b], Erwan Grandgirard [c], Charly Rousseau [d], Niamh Burke [e], Jerome Mutterer [f], David Rousseau [a,*]

[a] Université d'Angers, LARIS, UMR IRHS INRAe, 49000 Angers, France
[b] Aix Marseille Université, CNRS, IMM, Microscopy Core Facility, Marseille, France
[c] IGBMC, CNRS, INSERM, Université de Strasbourg, Illkirch, France
[d] Sorbonne Université, Paris Brain Institute - ICM, INSERM, CNRS, AP-HP, Hôpital de la Pitié Salpêtrière, Paris, France
[e] School of Medicine, University College Dublin, Dublin, Ireland
[f] IBMP, CNRS, Université de Strasbourg, Strasbourg, France

ARTICLE INFO

Keywords:
Computational imaging
Python library
Microscope

ABSTRACT

We describe a new software library written in the Python programming language to perform computational imaging on the so-called EnderScope, an automated imaging system based on the 3-axis motion stage of a 3D printer. This library is designed to be easy to use for beginner users and provides high-level functions to access used devices. We provide practical use cases and example codes using the library as an educational tool for testing, experimenting or developing computational imaging and smart microscopy algorithms.

Code metadata

Current code version	1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00110
Permanent link to Reproducible Capsule	
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Raspberry Pi 4+ with Debian Bookworm OS; Python 3.10+, pyserial, jupyter, ipywidgets, numpy, scipy, matplotlib, python-opencv
If available Link to developer documentation/manual	https://github.com/mutterer/enderscopy/
Support email for questions	jerome.mutterer@cnrs.fr david.rousseau@univ-angers.fr

1. Motivation and significance

The EnderScope is an open hardware, automated microscopy imaging system originally developed for investigating microplastics pollution [1]. The EnderScope is made of a 3D printer on which a camera is fixed to provide a 3D automated imaging system. The original EnderScope version was provided with a ready to use graphical user interface that allows moving the printer's axis, capturing single images thanks to the Raspberry Pi camera with fixed focus, and one function for automating the acquisition over a larger sample area. Fixed illumination using an LED ring was controlled independently of the Raspberry Pi, using an Arduino.

Several performance and reliability indicators like the achievable optical resolution, depth of field, measured LED emission wavelength, diagonal relative illumination, distortion effects or x and y-axis repositioning errors have been characterized for the original Enderscope device and are available in [1] supplementary information.

As any open hardware, the EnderScope can be replicated and modified to adapt to new potential uses. In this article, we investigate the possibility to perform acquisition of the same scene with a range of optical parametric conditions. This allows to envision computational imaging. This field of imaging encompasses a large range of methods [2] including imaging systems where the raw data acquired are

* Corresponding author.
E-mail address: david.rousseau@univ-angers.fr (David Rousseau).

https://doi.org/10.1016/j.softx.2025.102210
Received 18 February 2025; Received in revised form 8 May 2025; Accepted 19 May 2025
Available online 4 June 2025

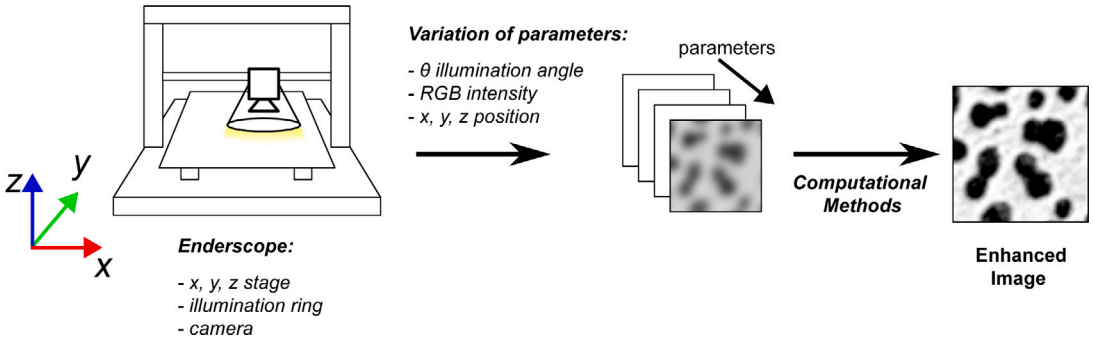


Fig. 1. Computational imaging with the EnderScope: Acquiring sets of images under varying conditions allows applying computational imaging methods to produce images with enhanced information content.

not directly interpretable to human eyes such as magnetic resonance imaging (MRI acquired in the Fourier transform domain) or computed tomography (CT acquired in the Radon transform domain). Here, we consider yet another type of computational imaging coined as epsilon imaging [3]. This type of computational imaging operates with situations where a batch of images acquired with various optical parametric conditions are processed to produce an improved final image as described in Fig. 1. While CT scan and MRI are not easily accessible in educational context, the epsilon imaging is a much more accessible way to computational imaging. In order to promote this usage, we created a Python library to simplify controlling the instrument programmatically. The library allows users to create software representations (objects) of components like stages, cameras, illumination devices, or other serially connected hardware. These objects act as simple building blocks, allowing users to perform tasks like 'moving the stage' or 'taking an image' without needing to deal with the underlying complexity of the code.

While many other microscope control libraries already exist [4–6], with more than 26 listed at http://smartmicroscopy.org, these libraries usually address high performance, complex microscopy setups and are designed to potentially make use of the largest possible variety of devices. This translates into some degree of complexity in the libraries' code and their software dependencies. In contrast, we focused on the minimal useful code to support a specific kind of device, so that even the non expert users feel invited to try their hand at taking programmatic control of an automated imaging device and experiment with image acquisition and computational imaging.

The article is structured as follows. We first describe the developed library and then illustrate its use for various computational imaging setups before final discussion.

2. Software description

2.1. Principles

The EnderScope as described in [1] consists in the 3-axis stage of a 3D printer, a camera, an illumination device driven by an Arduino board, all connected to a Raspberry Pi computational unit (Fig. 2). The 3D printer and illumination devices are connected to the computer using serial over USB, and the camera is connected directly to the Pi's CSI camera connector. The printer main board runs the standard firmware derived from the 'Marlin' version (available at https://marlinfw.org/) and is responsible for receiving G-Code commands and driving the 3 stepper motors accordingly to achieve axis movements. In the original EnderScope version, the LED control is done through a small breadboard with a push button, placed on top of the optics assembly. The need for this extra wiring is removed in this new version. The Arduino microcontroller driving the illumination LEDs runs a dedicated firmware that can listen to commands received over the serial port and does the tuning of each LED colors and intensities. The enderscope.py Python library provides objects for automated microscopy workflows, such as a 3-axis stage, a camera, and an illumination device. Each object includes easy-to-use functions that handle the low-level operations of the devices. Each device can be created and tested alone, and we provide example Jupyter Notebooks to experiment with them.

2.1.1. Serial device helper class

Both the stage and the illumination devices need to be connected to the host Raspberry Pi using a serial over USB connection. A SerialDevice class was written to facilitate serial port discovery and device creation. Sensible defaults are provided for port configuration, but can be overridden if necessary.

2.1.2. Stage class

The prototype 3-axis stage in the EnderScope is an Ender-3 3D printer [7]. The movement of a 3D printer's axis is usually achieved by sending the printer commands in the G-Code machine control language. G-Code needs to be sent to the printer over serial port, and is executed by the printer, translated to axis movements or specific functions like print bed heating or fan speed control. Although modern G-Code has some macro-scripting capability, the numbered commands somewhat lack readability and are widely unknown to most students. Our module provides explicit methods to initialize the stage to a known home position, move it along the 3 axis by given relative or absolute amounts, move it to specified positions, setting the speed of each axis stepper motor, and waiting for the stage to be settled at the last requested position, which permits synchronization with other devices.

- `Stage.home()`: this function uses the G-code G28 command to return one or more axes to their original position, typically used for calibration or resetting.
- `Stage.move_absolute(x, y, z=None)`: in absolute mode, the G-code G90 command interprets all coordinates as positions in logical coordinate space.
- `Stage.move_relative(x, y, z=None)`: in relative mode, the G-code G91 command interprets all coordinates as relative to the last position.
- `Stage.move_position([x, y, [z]])`: this function moves the stage to the specified position using the provided coordinates (x, y, and optionally z).
- `Stage.move_axis(axis, distance)`: this function moves the stage along the specified axis (x, y, or z) over a given distance using a relative command code.
- `Stage.move_towards(direction, distance)`: this function moves the stage in the specified direction (north, south, east, west, up, or down) over a given distance using a relative command code.

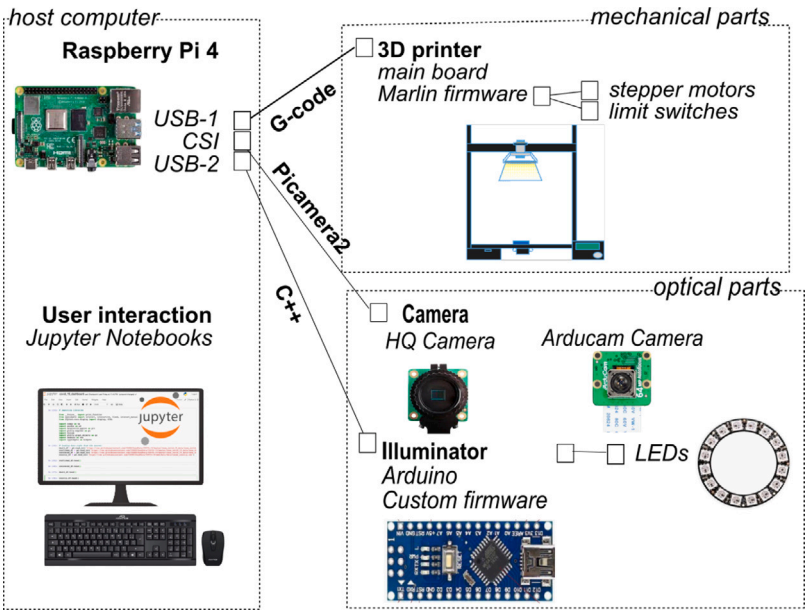


Fig. 2. Description of how to start the EnderScope.

- `Stage.finish_moves()`: this function completes the current moves by ensuring that the stage has finished moving before proceeding.
- `Stage.get_position()`: this functions retrieves the current position of the stage in the form of coordinates (x,y et z). Uses the M114 G-code command.
- `Stage.write_code(gcode)`: this function sends a raw G-code command to the stage.
- `Stage.set_speed(speed, axis)`: this function sets the movement speed of the stage on a specified axis among 'x', 'y', or 'z'. Uses the M203 G-code command.
- `Stage.read_params()`: this function reads the full list of the devices current parameters, such as feed rates, acceleration settings and more. Uses the M502 G-code command.

2.1.3. Enderlights class

This class represents an illumination device. Whereas the original illuminator was manually switched on and off, we propose a new Arduino firmware that accepts serial commands to achieve illumination tasks. The minimal tasks included are a global shutter and setting the illumination colors and intensity. Additional examples include a more advanced illuminator with different modes and parameters to achieve e.g. controlled directional illumination.

- `Enderlights.shutter(boolean)`: This function opens or closes a virtual shutter based on a boolean.
- `Enderlights.red(value)`: this function sets the red level.
- `Enderlights.green(value)`: this function sets the green level.
- `Enderlights.blue(value)`: this function sets the blue level.
- `Enderlights.color(r, g, b)`: the color function sets the RGB levels by adjusting the red, green, and blue components.
- `Enderlights.mode(m)`: this function switches the operating mode. (mode=0: all LEDs are driven identically; mode=1: the ring is divided in 4 segments specified using bits in parameter p; mode=2: uses a single LED specified by parameter p)
- `Enderlights.parameter(p)`: this function is used to define or modify a specific parameter of the current operating mode.

2.1.4. Control panel

While the main goal of this library is to invite users to take programmatic control of the system, we provide a minimal set of graphical user interface elements in a simple control panel, in the form of iPywidgets [8] to move the printer axis, home the system and record and reuse stage positions. This control panel allows one to get a first manual control of the stage and explore the stage's coordinate space. As an additional example application, we provide an extended version of the control panel and device model that could be used to create a fully functional user interface to use the EnderScope without programming (see Fig. 3).

2.1.5. Cameras

We successfully used the Raspberry Pi Camera Module 2, Module 3 and High Quality Camera which can all be controlled from Python using the Picamera2 module (see Table 1). Future development might include a generic Camera class with additional camera modules.

2.1.6. Control synchronization

Proper image acquisition requires reliable knowledge of the system's current state. For example, after the stage axes were moved, we need to make sure the stage is still prior to capturing images to prevent stage motion blur. Stage movements can take up to several seconds to complete, but the Marlin firmware by default acknowledges command receipt not command completion. This allows to smoothly process commands in queue [9]. To address this feature, any movement sensitive task should be preceded by a blocking Stage.finish_moves() call. The camera capture commands used in the example notebooks also display blocking behavior, so that further instructions will not be executed before the image object is returned. Finally, the enderlights.ino Arduino firmware also has this blocking behavior, ensuring correct illumination is set before proceeding with next steps of acquisition. These measures ensure that all tasks are run in the correct sequence regardless of their specific execution times.

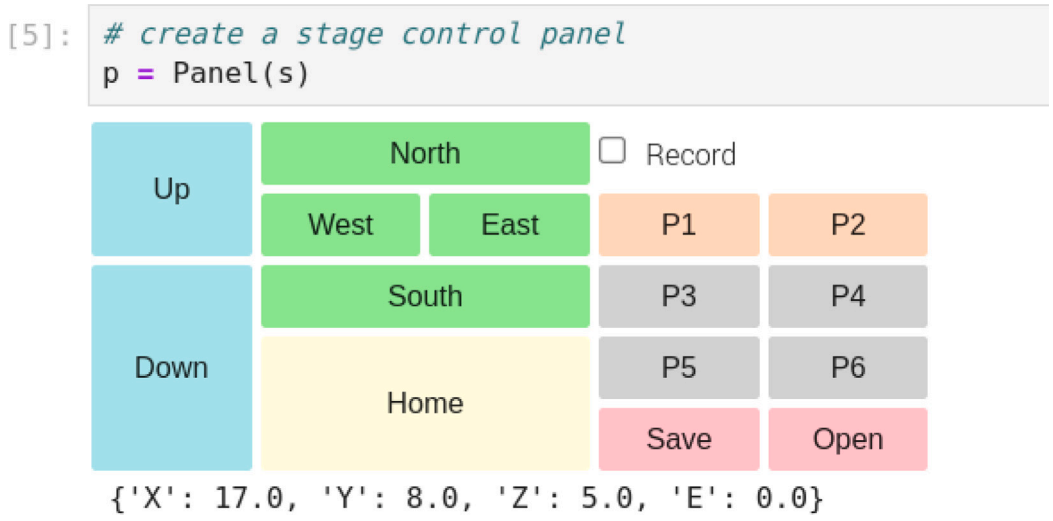


Fig. 3. A basic EnderScope control panel.

Table 1
Comparison of tested Raspberry Pi Camera Modules and ArduCam 64MP Camera. Full list of available options is available at https://www.raspberrypi.com/documentation/accessories/camera.html and https://www.arducam.com/.

Specification	Camera module v2	Camera module 3	HQ camera	64 MP autofocus camera
Sensor resolution	3280 × 2464	4608 × 2592	4056 × 3040	9152 × 6944
Lens Mount	N/A	N/A	C/CS- or M12	CS
Focus	Manual	Motorized	Manual	Manual/Auto
Focal length	3.04 mm	4.74 mm	Depends on lens	5.1 mm
Horizontal Field of View (FoV	62.2°	66°	Depends on lens	68°
Vertical Field of View (voV)	48.8°	41°	Depends on lens	56°

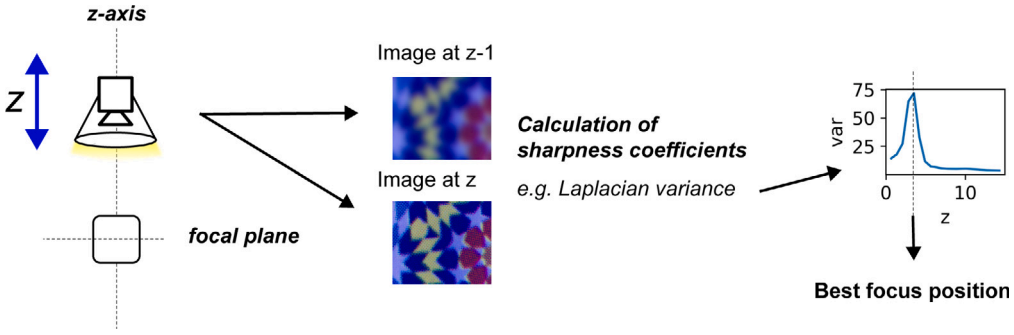


Fig. 4. Simple autofocus.

3. Computational imaging illustrative examples

A full set of examples describing how to use the library is provided in the repository linked in code metadata Table. These include basic examples for creating the relevant objects and more advanced examples combining these objects to perform various computational imaging tasks. We now shortly describe more advanced computational imaging examples also provided with the library.

3.1. Simple autofocus

A range of images is acquired around the current Z position as shown in Fig. 4. Laplacian variance is used as an estimate of image focus quality. The stage is moved to the Z coordinate with the maximum focus score. Any variant for the image focus quality can be tested here [10]. A video demo is accessible at https://www.youtube.com/watch?v=VoJjsEalR7A

3.2. Shadow or reflection removal using multi-angle illumination

Our illumination device is composed of 16 independent RGB LEDs. The pattern and colors of used LEDs can be set, with provided examples modes using a single LED, or adjacent sets of 4 LEDs illuminating the sample from one or several $\pi/2$ ring sector. Series of images can be captured with varying illumination angles. Computing different projections across the image dataset allows e.g. removal of shadows if using a maximum intensity projection, or removal of bright light reflections when using minimum intensity projection (see Fig. 5).

3.3. Smartscan (overview and region-of-interest scan)

In this example, a first overview image is captured with the camera moved at the maximum height along the Z axis. Under these conditions, the camera field of view is 18x18 cm, which covers most of the base plate. Objects of interest can be identified in this image, yielding a set of

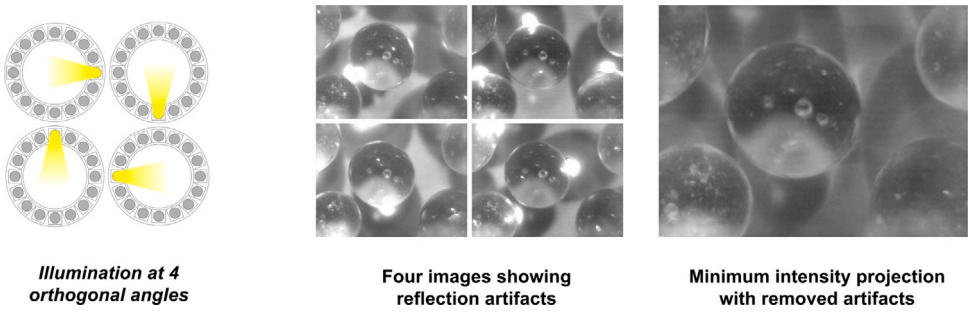


Fig. 5. Multi-angle illumination acquisition using silica gel beads.

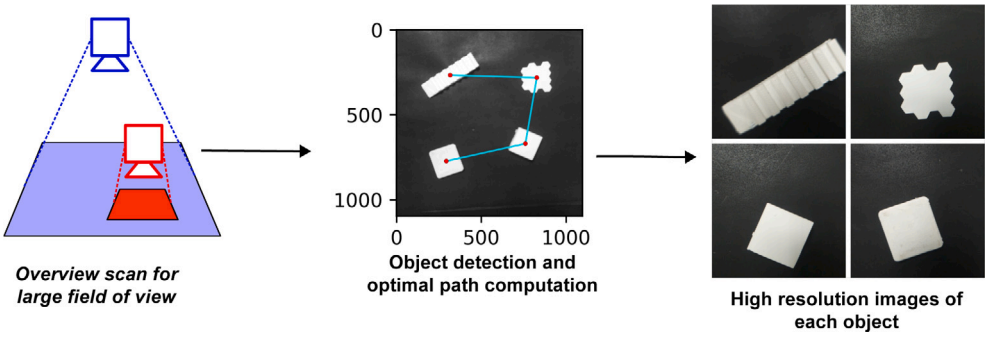


Fig. 6. Smart Scan principle.

objects centroids. The optimal path linking those centroids is computed using a Traveling Salesman Problem (TSP) algorithm [11], allowing higher resolution images of the identified objects to be acquired, at optimized speed, after moving the camera closer to the objects (see Fig. 6). A video demo is accessible at https://www.youtube.com/watch?v=qtj_kktQ8ec.

4. Impact

The EnderScope system has been published recently in [1]. The proposed library constitutes an extension of the original imaging system to computational imaging. This extension provides a very low-cost hardware and software combination that can be used for educational purposes as well as for research matters.

This library is particularly beneficial in an educational setting, making computational imaging more accessible to researchers and students with minimal coding experience. By providing an intuitive interface and reducing the complexity of hardware control, it enables users to engage with the software of the EnderScope without being overwhelmed by technical barriers. However, even though the user experience has been simplified, this does not prevent the user from being able to learn about programming, as the code is not hidden. The jupyter notebooks and EnderScope library are freely available online https://github.com/mutterer/enderscopy/, meaning that researchers can view, modify and experiment with the code. Additionally, this library has the potential to make the EnderScope more suitable for community use. As suggested in [1], the EnderScope is suitable for use both within and beyond formal research environments, including both research laboratories and use by members of the public. By simplifying control and customization, this library makes the EnderScope an even more viable tool for citizen science projects. The combination of both the EnderScope and the library presented in this article can introduce researchers to the concept of designing custom hardware tailored to their specific experimental needs. The EnderScope was initially designed to be easy to build, while this library should make the EnderScope even easier to use. By lowering the barrier to entry for researchers to build, use and potentially even modify open instruments such as the EnderScope, it encourages users to expand their arsenal of available tools—not just relying on off-the-shelf equipment, but creating bespoke solutions for their unique research challenges. For example, modifications can be made to the EnderScope such as replacing non-coherent illumination with laser-based illumination to allow for techniques like digital holography, ptychography, or lensless imaging. Additional customizations could be explored to allow for polarized light microscopy.

While the intentionally all-code and notebooks solution we describe in this article is meant to invite users to experiment with the system and adapt it to their own application, it also requires some degree of code proficiency which could sadly drive away some potential users. Thus, future developments might include integration into established imaging software like μManager, the development of application-specific simplified physical control panels.

The impact of this combined system has already been demonstrated at several science festivals for the general public, as well as to several scientific communities at an international scale. This includes the Bioimaging community during conferences including MIFOBIO 2023 and Trends in Microscopy 2025 (https://imabio-cnrs.fr/mifobio2023/ and https://gerbi-gmb.de/event/tim2025/), the image processing community during the IPRIA 2025 (https://youtu.be/MYbDuIdUElg). The accessibility of the library proposed here should enable to further leverage the capabilities of the EnderScope and to significantly expand its users community.

5. Conclusions

We have provided a library that enables to extend the capabilities of the recently introduced EnderScope (a 3D printer turned into a camera positioned on a 3-axis robotized arm). The proposed library allows to perform computational imaging at low cost with this open hardware. It offers high-level methods for accessing essential stage and illumination functions which allows users to focus on computational imaging methods development. This article provides several examples such as autofocus, artifact of illumination correction or smart scanning, with more being available in the associated online material. We expect that this will facilitate prototyping of smart imaging systems

and computational imaging methods in both educational and research contexts.

CRediT authorship contribution statement

Sirine Gharbi: Writing – original draft, Software. **Emeric Poiraud:** Software. **Hugo Le Guenno:** Data curation. **Erwan Grandgirard:** Data curation. **Charly Rousseau:** Data curation. **Niamh Burke:** Writing – review & editing, Data curation. **Jerome Mutterer:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **David Rousseau:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge France-BioImaging infrastructure supported by the French National Research Agency ANR-24-INBS-0005 FBI BIOGEN and the French National Infrastructure on Plant Phenotyping ANR PHENOME 11-INBS-0012.

References

[1] Burke Niamh, Müller Gesine, Saggiomo Vittorio, Hassett Amy Ruth, Mutterer Jérôme, Ó Súilleabháin Patrick, et al. EnderScope: a low-cost 3D printer-based scanning microscope for microplastic detection. Philos Trans A 2024;382(2274):20230214.

[2] Bouman Charles A. Foundations of computational imaging: a model-based approach. SIAM; 2022.

[3] Raskar Ramesh. Computational photography: Epsilon to coded photography. In: LIX fall colloquium on emerging trends in visual computing. Springer; 2008, p. 238–53.

[4] Susano Pinto David Miguel, Phillips Mick A, Hall Nicholas, Mateos-Langerak Julio, Stoychev Danail, Susano Pinto Tiago, et al. Python-Microscope–a new open-source Python library for the control of microscopes. J Cell Sci 2021;134(19):jcs258955.

[5] Edelstein Arthur D, Tsuchida Mark A, Amodaj Nenad, Pinkard Henry, Vale Ronald D, Stuurman Nico. Advanced methods of microscope control using μManager software. J Biol Methods 2014;1(2).

[6] Casas Moreno Xavier, Al-Kadhimi Staffan, Alvelid Jonatan, Bodén Andreas, Testa Ilaria. ImSwitch: Generalizing microscope control in python. J Open Source Softw 2021;6(64).

[7] Shenzhen Creality 3D Technology Co. Ltd. Ender3 device description. 2025, https://github.com/Creality3DPrinting/Ender-3. [Accessed 27 January 2025].

[8] Jupyter widgets community. ipywidgets, a GitHub repository. 2025, https://github.com/jupyter-widgets/ipywidgets. [Accessed 27 January 2025].

[9] Marlin FirmWare. Marlin's command queue concept. 2025, https://github.com/MarlinFirmware/Marlin/blob/bugfix-2.1.x/docs/Queue.md. [Accessed 6 May 2025].

[10] Pertuz Said, Puig Domenec, Garcia Miguel Angel. Analysis of focus measure operators for shape-from-focus. Pattern Recognit 2013;46(5):1415–32.

[11] Toaza Bladimir, Esztergár-Kiss Domokos. A review of metaheuristic algorithms for solving TSP-based scheduling optimization problems. Appl Soft Comput 2023;110908.