



SOFTWARE TOOL ARTICLE

REVISÉD Microscope-Cockpit: Python-based bespoke microscopy for bio-medical science [version 2; peer review: 3 approved, 3 approved with reservations]

Mick A. Phillips ¹⁻³, David Miguel Susano Pinto ¹, Nicholas Hall ¹, Julio Mateos-Langerak ⁴, Richard M. Parton ¹, Josh Titlow¹, Danail V. Stoychev ¹, Thomas Parks², Tiago Susano Pinto¹, John W. Sedat⁵, Martin J. Booth ⁶, Ilan Davis¹, Ian M. Dobbie ^{1,7}

¹Micron Advanced Bioimaging Unit, Department of Biochemistry, University of Oxford, South Parks Road, Oxford, OX1 3QU, UK

²Diamond Light Source, Harwell Science and Innovation Campus, Didcot, OX11 0DE, UK

³Division of Structural Biology, The Henry Wellcome Building for Genomic Medicine, Roosevelt Drive, Oxford, OX3 7BN, UK

⁴IGH, Univ Montpellier, Montpellier, 34396, France

⁵Department of Biochemistry & Biophysics, UCSF, San Francisco, CA, 94158, USA

⁶Department of Engineering Science, University of Oxford, Oxford, OX1 3PJ, UK

⁷Integrated Imaging Center, Department of Biology, Johns Hopkins University, 3400 N. Charles St., Baltimore, Maryland, 21218, USA

v2 First published: 08 Apr 2021, 6:76
<https://doi.org/10.12688/wellcomeopenres.16610.1>
 Latest published: 17 Jan 2022, 6:76
<https://doi.org/10.12688/wellcomeopenres.16610.2>

Abstract

We have developed “Microscope-Cockpit” (Cockpit), a highly adaptable open source user-friendly Python-based Graphical User Interface (GUI) environment for precision control of both simple and elaborate bespoke microscope systems. The user environment allows next-generation near instantaneous navigation of the entire slide landscape for efficient selection of specimens of interest and automated acquisition without the use of eyepieces. Cockpit uses “Python-Microscope” (Microscope) for high-performance coordinated control of a wide range of hardware devices using open source software. Microscope also controls complex hardware devices such as deformable mirrors for aberration correction and spatial light modulators for structured illumination via abstracted device models. We demonstrate the advantages of the Cockpit platform using several bespoke microscopes, including a simple widefield system and a complex system with adaptive optics and structured illumination. A key strength of Cockpit is its use of Python, which means that any microscope built with Cockpit is ready for future customisation by simply adding new libraries, for example machine learning algorithms to enable automated microscopy decision making while imaging.

Open Peer Review

Approval Status

	1	2	3	4	5	6
version 2						
(revision)						
17 Jan 2022			view	view	view	view
version 1						
08 Apr 2021	view	view				


1. **Kyle Harrington** , Max Delbrueck Center for Molecular Medicine, Berlin, Germany
2. **Jason R Swedlow** , University of Dundee, Dundee, UK
- Emil Rozbicki**, Glencoe Software, Inc, Seattle, USA
- William Moore**, University of Dundee, Dundee, UK
- Sébastien Besson** , University of Dundee,


Keywords


Microscope-Python, Bespoke microscope, Microscope hardware device control, Free and open source software, Imaging, Machine Learning (ML), Artificial Intelligence (AI), Adaptive optics (AO), Super resolution microscopy

Dundee, UK

3. **Joe Knapper** , University of Glasgow, Glasgow, UK

4. **Marijonas Tutkus** , Vilnius University Life Sciences Centre, Vilnius, Lithuania
Center for Physical Sciences and Technology, Vilnius, Lithuania

5. **Adam Wollman** , Newcastle University, Newcastle upon Tyne, UK

6. **Xavier Casas Moreno** , KTH Royal Institute of Technology, Stockholm, Sweden

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding authors: Ilan Davis (ilan.davis@bioch.ox.ac.uk), Ian M. Dobbie (ian.dobbie@jhu.edu)

Author roles: **Phillips MA:** Conceptualization, Software; **Susano Pinto DM:** Conceptualization, Software, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing; **Hall N:** Software; **Mateos-Langerak J:** Software; **Parton RM:** Conceptualization, Resources; **Titlow J:** Resources; **Stoychev DV:** Software; **Parks T:** Software; **Susano Pinto T:** Software; **Sedat JW:** Conceptualization; **Booth MJ:** Conceptualization, Supervision; **Davis I:** Conceptualization, Funding Acquisition, Project Administration, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing; **Dobbie IM:** Conceptualization, Funding Acquisition, Project Administration, Software, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: Martin Booth declares a significant interest in Aurox Ltd., whose microscopes were used in this work.

Grant information: This research was funded by Wellcome strategic awards, a senior fellowship and an investigator award to I.D. [091911/Z/10/Z], [091911/Z/10/A], [107457/Z/15/Z], [096144/Z/17/Z] and [209412/Z/17/Z]; Wellcome funding to the Wellcome Trust Centre for Human Genetics [105605/Z/14/Z] and [203141/Z/16/Z]; an MRC/EP SRC/BBSRC Next-generation Optical Microscopy [MR/K01577X/1] to I.D.; GIS IBISA [#2015-28]; and by the CNRS/MITI [Défi Imag'In 2015]. D.S. is funded by a BBSRC iCASE grant with Aurox as the industrial partner [BB/M011224/1]. N.H. was supported by funding from the Engineering and Physical Sciences Research Council (EPSRC) and Medical Research Council (MRC)[EP/L016052/1].

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2022 Phillips MA *et al.* This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Phillips MA, Susano Pinto DM, Hall N *et al.* **Microscope-Cockpit: Python-based bespoke microscopy for biomedical science [version 2; peer review: 3 approved, 3 approved with reservations]** Wellcome Open Research 2022, 6:76 <https://doi.org/10.12688/wellcomeopenres.16610.2>

First published: 08 Apr 2021, 6:76 <https://doi.org/10.12688/wellcomeopenres.16610.1>

REVISED Amendments from Version 1

After reviewers comments we added an additional section on the saved file image format, comments on the newly published Pycro-Manager extension to uManager and an additional table comparing the features of our software with other available options for computer control of microscope hardware.

Any further responses from the reviewers can be found at the end of the article

Introduction

Why we need Cockpit

Biomedical research has benefited from significant engineering advances that have increased the speed and sensitivity of many types of scientific instrumentation. Microscope technology, in particular, is evolving rapidly through advances in optical techniques and image processing, enabling major scientific discovery. However, the rapid adoption and application of advanced microscopies by biomedical scientists is constrained, especially when it relies on commercialisation to make the necessary technology accessible to the end-users.

It can often take several years to develop innovations from research instruments into user-friendly, off-the-shelf systems¹⁻⁴. We report a collaboration between the University of Oxford and the University of California San Francisco (UCSF) that grew out of the OMX microscope project, dating from the early 1990s, which is extensively documented in the supplement of 1.

To accelerate technology adoption, many labs are building custom microscopes based on newly developed imaging methods, such as light sheet⁵, lattice light sheet³, 3DSIM^{1,6}, STED⁷, and MINFLUX⁸, or to exploit new analysis techniques, such as STORM analysis^{9,10}, 3B¹¹, and SIMFLUX¹². These instruments require flexible, computerised control of a wide range of individual components (lasers, mirror actuators, filters, objectives, detectors, etc.) allowing more rapid deployment of novel techniques, and often providing faster and more sensitive operation than commercial systems^{3,13,14}. However, integration of these components into a single user-friendly platform that is focused on users' scientific application, rather than dictated by the control infrastructure, can be a major challenge. There have been three general solutions adopted by the community for software control of bespoke microscope hardware.

1. Using individual manufacturer-provided packages for each piece of hardware. This can provide a straightforward solution for simpler systems, but fundamental incompatibilities between each vendor's control models can prevent integration of multiple components.
2. Using [LabVIEW](#), a commercial visual programming tool from a leading hardware manufacturer, National Instruments (Austin, Texas, United States).
3. Use of a dedicated microscope control package, of which several are available, with the most widely used being [Micro-manager](#) (µManager): an open source Java and C++

based software platform for controlling a range of microscope hardware such as stands, cameras, and stages. A less widely-adopted solution is a bespoke platform written in Python to control the OMX microscope^{1,15,16}, and this forms the starting point of the Microscope-Cockpit project described in this manuscript.

An important feature for custom microscopes is the freedom to design the system around a specific set of experiments, as defined by the user. With a specific application in mind, an instrument can be optimised to make the best possible compromises for that application. For example, imaging deep into biological tissue can involve significant optical aberrations leading to degraded image quality. This can be compensated for by the application of adaptive optics techniques, at a cost of increased complexity and marginally decreased light efficiency. However, it is desirable that this capability is easily enabled and exploited by microscope users, typically biologists with limited experience of the underlying engineering, and be reproducible over months or years of use. Many such custom microscope designs have been published¹⁷⁻²². In all of these cases, control of the many hardware components in a complex and timed manner is challenging, particularly for time sensitive applications involving living specimens. Correction settings must be measured and applied in coordination while simultaneously minimising additional exposure of the sample.

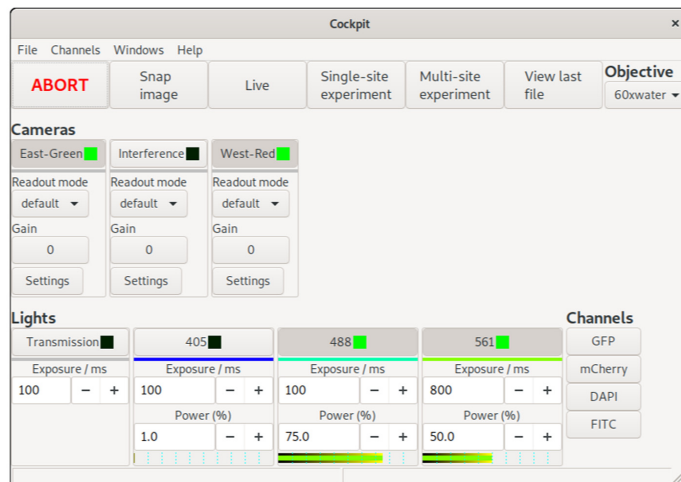
Finally, the democratisation of artificial intelligence (AI)-based image analysis approaches means that a wealth of tools are potentially available to users. However, current systems, with some specific exceptions, are not well suited for the integration of bespoke novel AI-approaches with the imaging process. This limitation can be significantly reduced by an appropriately flexible package for control of bespoke systems.

Philosophy and implementation of Cockpit

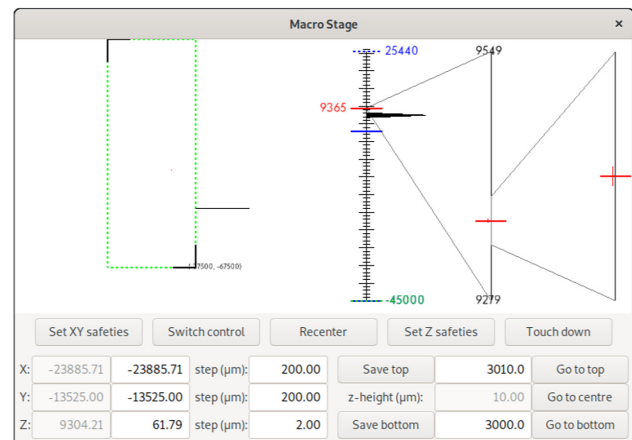
We set out to create a package that is suitable for biologists to carry out a range of experiments in a time efficient manner, at scale. At the same time, the software has to control a wide range of electronic and optical hardware devices and microscope types. Many bespoke microscope systems that lack eyepieces, such as systems built by physical scientists in collaboration with biologists, make sample navigation particularly challenging. Improving navigation on this type of system is a major feature of Cockpit.

There are several existing software control approaches including manufacturer-supplied packages, custom control software, in LabVIEW or a more traditional programming language, or the open source program µManager, as discussed earlier. We feel that Cockpit has several distinct advantages over these other approaches. We compare the relative merits of Cockpit and a range of alternatives in the discussion.

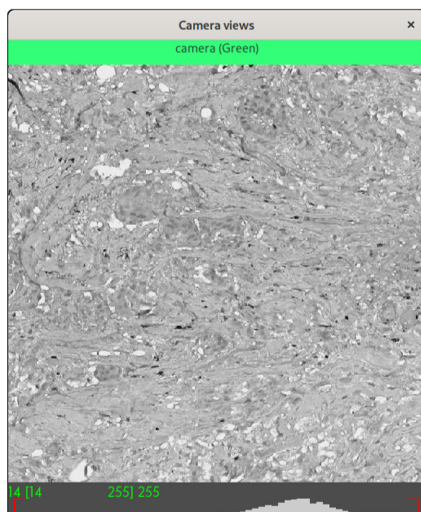
Cockpit is an open-source software package that makes it easy to control multiple devices through a single [Python](#)-based user interface ([Figure 1](#)). Cockpit supports multiple platforms, and the similarity of its interface across operating systems and widget toolkits is shown in [Figure 2](#). It provides an innovative



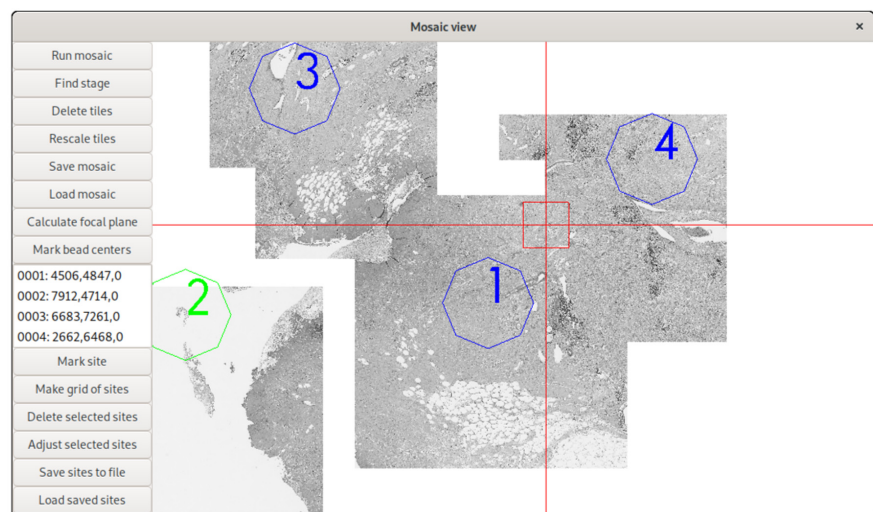
(a) Main Window



(b) Macro Stage



(c) Camera View



(d) Mosaic View

Figure 1. The main Cockpit Graphical User Interface (GUI) components: **(a)** The main window provides quick access to a number of functions and shows the status of all the devices, as well as the channels buttons to load predefined configurations. **(b)** The macro stage window provides an overview of the position of all stages, including nested stages and Z position. **(c)** The camera view shows the last image taken from each active camera and their histograms. **(d)** The mosaic view displays all mosaic images taken, saved locations, and dramatically eases navigation.

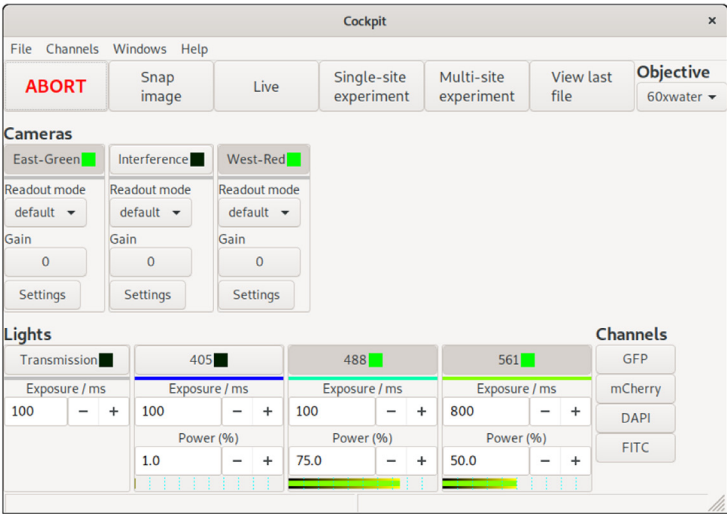
mosaic tool, allowing large areas of a microscope specimen to be imaged and then navigated in real time. This is similar to [Micro-Magellan](#)²³, but with the additional feature of keeping even thousands of images in Graphic Processing Unit (GPU) memory for instant access at various levels of detail. It utilises [Python-Microscope](#)²⁴ to communicate with a wide range of microscope hardware and is able to exploit a master timing control device to provide hardware triggering for digital devices and synchronised analogue voltages. This enables highly reproducible timing over a range of time scales even in extremely complex experiments with multiple devices.

The simple interface design ([Figure 1](#)) enables users to focus on collecting the data they require without having to devise

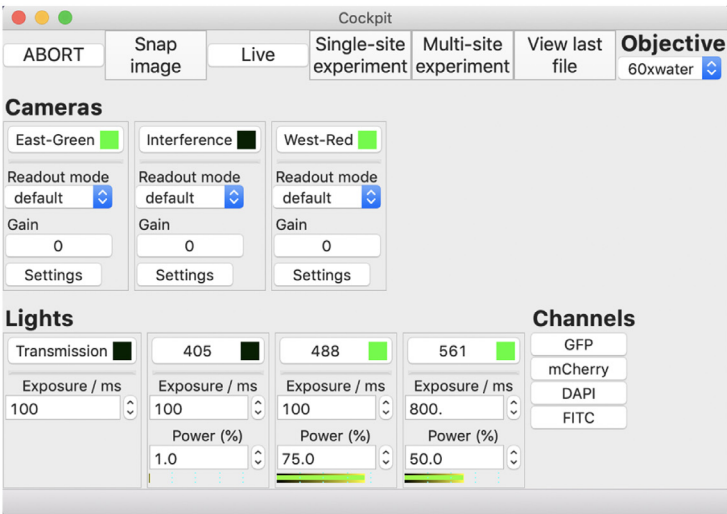
complex control infrastructure for systems of any level of complexity. It also enables easy navigation of even large sample regions. This can be especially important on bespoke microscopes which often lack eyepieces. For these types of system, Cockpit includes a touchscreen-optimised display window, providing large buttons and simple control over most functions ([Figure 3](#)).

The principal design goals for the Cockpit interface were to:

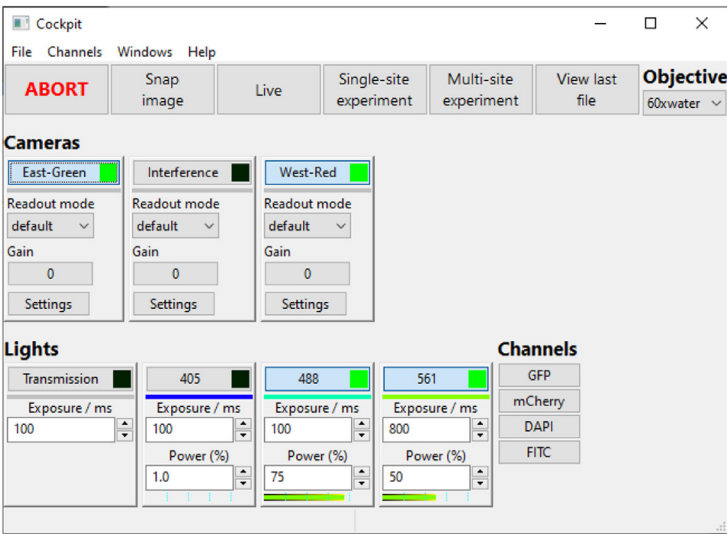
- Provide an easy-to-use, simple “Cockpit” control program for a wide range of microscopes.
- Hide from the user the complexity required to control an advanced microscope.



(a) GNU/Linux (Gnome)



(b) macOS



(c) Windows 10

Figure 2. The Cockpit main window under different operating systems.

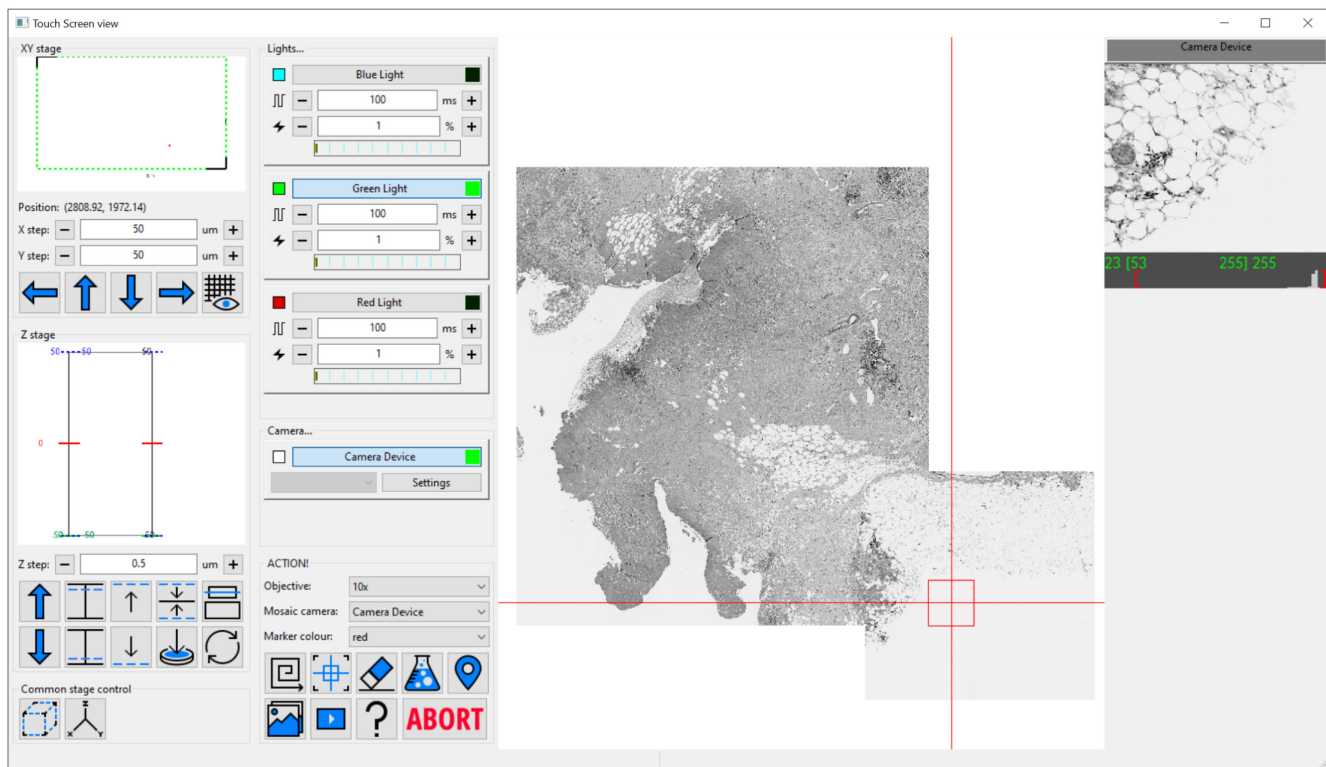


Figure 3. The Cockpit touchscreen window with the main GUI components in one window and large touchscreen friendly buttons. The left panel houses stage position indicators and controls, next panel over has devices like cameras and lights along with a set of large buttons at the bottom to run mosaics, centre the field of view on the current stage position, erase mosaic images, open the experiment dialog, drop a marker at the current position, display live video, run an experiment, a help button, and finally abort which stops an experiment or mosaic. The majority of the window is the mosaic display panel which can zoom from an overview up to the level of single pixels. Finally on the right is the last snapped image from each enabled camera, only one in this case.

- Minimise the effort required to change hardware or integrate new hardware.
- Enable high-precision hardware timing signals to optimise experimental control, facilitate complex systems and maximise speed.

Cockpit provides an integrated microscope control package that enables all these features while being user friendly and portable across many different microscopes and common operating systems.

The first version of the Cockpit software was developed in UCSF to drive the OMX microscope^{1,15,16}, which was later commercialised. This code was re-written at UCSF and later released as open source under a BSD license. From this concept we have extensively developed the implementation of Cockpit to realise its full potential.

Implementation

Cockpit is implemented in the Python programming language²⁵. The code consists of four main components: devices, handlers, interfaces, and the Graphical User Interface (GUI). Devices

represent the individual physical devices, handlers represent control components of the devices, handlers from single or multiple devices are aggregated into interfaces, and the GUI component allows user interaction with the different interfaces.

Devices and handlers

The hardware control side of Cockpit has two parts: devices and handlers. A Cockpit device maps to a single physical device while handlers abstract the control components of the device to be used. For example, an XY stage device provides two handlers, one per axis. A laser device also provides two handlers, a light source handler and a light power handler: the light source handler controls the light source state — on or off — while the light power handler controls its intensity.

Most of the Cockpit code does not interact with Cockpit devices directly. Rather, when the Cockpit program starts, it constructs a “depot” object whose role is to obtain and keep track of the different device handlers. This architecture enables rapid changes in the choices of hardware devices, so they can be very easily added or removed from a complex system as it is being developed. This approach prevents potential

conflicts or confusions such as addressing devices that do not exist or adding devices that lack drivers.

Within this architecture, any Python class can be used as a device; it is only required to provide one or more Cockpit handlers. Built-in Cockpit devices include adaptors for the Python-Microscope package, thus supporting all of its available devices. Cockpit also supports the Aerotech PRO115 linear stage from the [Soloist CP Controller](#), the PI M-687 XY stage from the PI C-867.262 controller, the [Picomotor controller 8743-CL](#), the [small 512x512 Spatial Light Modulator \(SLM\)](#) and [liquid crystal polarization rotator](#) from Meadowlark Optics, and the [SR470 laser shutter system](#) from Stanford Research Systems.

Cockpit itself does not require a direct connection to the hardware. Indeed, most Cockpit devices are implemented as remote objects with the help of [Pyro](#), a Python package for remote procedure calls. The use of Pyro enables the distribution of devices over the network to different computers, even running different operating systems.

Interfaces and GUI

Cockpit defines interfaces that are high-level abstractions for the system, with each controlling one or more handlers. For example, the stageMover interface is a unified translation manager that coordinates all translation stages, hiding the complexity of handling different stages for different axes and nested stages on the same axis. For instance, a system may include an XY stage, a coarse Z stage, and a high precision fast piezo Z stage. The GUI component provides different views onto the multiple interfaces and handlers, each exposing different levels of complexity.

Cockpit has a multi-window GUI, with each window serving a different purpose ([Figure 1](#)). This provides the flexibility to distribute the windows over multiple monitors in whatever form best fits the system being used. The main window provides an at-a-glance view of the state of the various devices; the mosaic window enables navigation of the sample relative to an overview formed of tiled images; the macro stage window displays the position of the different stages; and the camera view the current images from all active cameras. In addition, Cockpit also has a simpler single-window touchscreen interface ([Figure 3](#)) which duplicates a subset of the functionality provided on the other windows. This window is optimised for touchscreen interaction with large buttons for the most useful subset of functionality.

Cockpit uses [wxPython](#), a Python wrapper to [wxWidgets](#), to provide a user interface that has a native look and feel on the different platforms ([Figure 2](#)). Within wxPython, we make extensive use of [OpenGL](#) to enable high performance display and interaction with large images or image mosaics up to gigapixel sizes.

Data acquisition experiments

In Cockpit, the acquisition of an image series, such as a Z-stack or a time lapse sequence, is defined by an experiment. Cockpit experiments are based around the concept of an

action table which must be run with hardware triggers. The experiment settings are converted into a list of device actions, which is parsed into a list of pre-computed analogue values and digital levels for each time point on a timing device. The action table is uploaded to the timing device and then started, running the experiment utilising hardware triggers and analogue voltages, where appropriate, to control all active devices. This approach, adapted from Carlton *et al.*¹, is able to produce both digital triggers and analogue signals with precise timing.

Effective execution of the experiment requires accurate timing. In order to ensure that experiment timing is independent of the host computer's performance or the Python implementation, the timing functionality is performed using dedicated hardware. This enables time-critical operation of the microscope imaging tasks. The timing system has been implemented on several different hardware platforms, a digital signal processing board (Innovative Integration M67-160 with an A4D4 daughter board), an NI FPGA board (National Instruments cRIO-9068), and a Red Pitaya single board computer (STEMlab 125-14). In this way, we have created a universal and adaptable microscope platform with outstanding timing precision and accuracy.

Output image files

Experimental images are saved into files utilising the '.dv' file format, typically multiple wavelengths, Z-slices, and time points into a single file. Images from the mosaic window and single snaps can also be saved into this format, with the mosaic saved files having an associated text file defining XYZ positions of each collected image. The '.dv' format is an extension of the mrc file format, defined in detail in the MRC/CCP4 2014 (https://www.ccpem.ac.uk/mrc_format/mrc2014.php) file format specification²⁶. The CCP4 consortium of the EM community continue to support and extend this file format. This support includes file validators and a detailed specification, which is compatible with the files used here but not identical.

The optical microscopy specific metadata are covered in the documentation. Although relatively uncommon, the file format is supported by the Bio-Formats (<https://www.openmicroscopy.org/bio-formats/>) project allowing import of '.dv' files, along with the associated metadata into software using this library including ImageJ, OMERO, and Matlab. Additionally, the Chromaggon image alignment tool will read and write '.dv' files and it is the native format for DeltaVision microscopes utilising the commercial package SoftWoRx.

System requirements

Cockpit requires Python 3.5 or later, and the Python packages [PyOpenGL](#), [Pyro4](#), [Freetype Python](#), [Matplotlib](#) (RRID:SCR_008624)²⁷, [Python-Microscope](#)²⁴, [NumPy](#) (RRID:SCR_008633)²⁸, [pySerial](#), [SciPy](#) (RRID:SCR_008058)²⁹, and [wxPython](#). These are all free and open-source software that are available for all widely-used operating systems. Hence, Cockpit can be used across GNU/Linux, macOS, and Windows.

The Cockpit interface incorporates some assumptions as to what is required for a minimal microscope. It requires at least one camera, a computer controlled X, Y and Z translation

establishes the conditions required for the desired images, such as choice of laser lines and their intensities, exposures, number of channels and cameras, Z sectioning range, and points to visit on the guide map. The use of a touchscreen enables rapid setting up of the conditions for the experiment, focusing on the most important features of the microscope hardware without providing full functionality.

In a second phase of acquisition, a more detailed software control window is used with a conventional keyboard to define any more advanced experimental settings, such as numeric values for time lapse interval which are difficult to set from a touch interface. Finally the experiment is run.

In this way, we have created a unique, near-instantaneous control environment that allows the user to start an imaging session by exploring the entire slide quickly and marking regions to visit and image in order to achieve their scientific goals. Our software provides a novel way for the user to plan their entire work flow rapidly and with efficient specimen usage.

Previously published systems and microscopes in development using Cockpit

To exemplify the flexibility of Cockpit, we present a number of systems that currently run using the software as the user interface. Along with previously published systems, we have a number of microscopes at earlier stages of development where Cockpit provides the user interface.

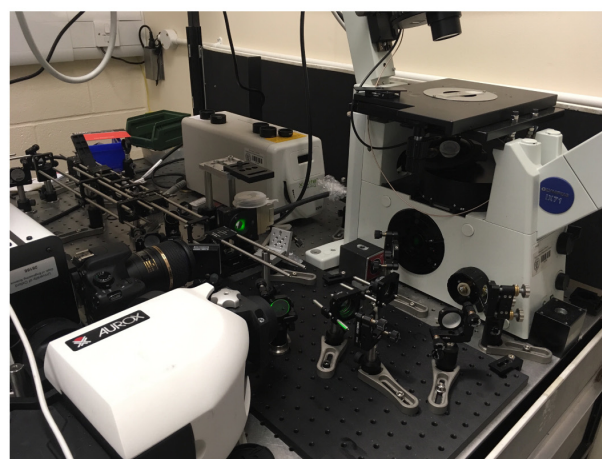
Cockpit has already been deployed in three systems reported in previous publications. The first is a cryo-super resolution microscope for correlative imaging, CryoSIM^{30,31} (Figure 5(a)). The second demonstration was a laser-free spinning disk confocal microscope using adaptive optics (AO) for aberration correction³² (Figure 5(b)). Additionally, Cockpit was the basis of the control software in the implementation of a novel technique, IsoSense, for improved aberration correction in widefield microscopes and structured illumination³³.

A simple widefield microscope

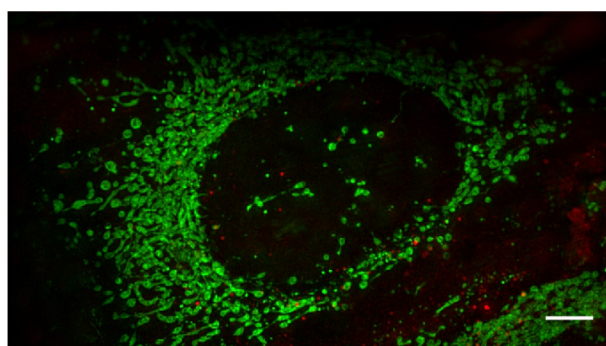
We present a simple, compact, fully automated, portable inverted microscope based around the Zaber MVR system. This



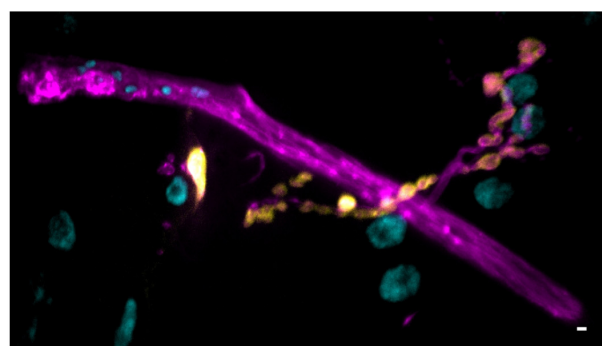
(a) CryoSIM



(b) Aurox AO



(c) Image from CryoSIM



(d) Image from Aurox AO

Figure 5. Previously published systems that use Cockpit: **(a)** the CryoSIM system; **(b)** the Aurox Clarity AO laser free spinning disk confocal system; **(c)** image acquired on CryoSIM of cryo preserved HeLa cell labelled with MitoTracker green and LysoTracker red; **(d)** image acquired on the Aurox Clarity system with AO correction at high optical sectioning. Image is a maximum intensity projection of a 20 μm z-stack. Sample was a *Drosophila* neuromuscular junction with DLG in yellow, HRP in magenta, and DAPI in cyan. Scale bars 5 μm .

provides a bare-bones inverted stand with motorised X, Y, and Z axes, a multi-position motorised filter turret, and LED-based fluorescence illumination. The system has a Ximea camera with a small physical footprint (approximately 30 mm cube) and a Red Pitaya single board computer running as a hardware timing device (Figure 6).

This system is able to take fluorescence images in three colours, using the three illumination LEDs and a quad-bandpass dichroic filter set. The system has 4× and 63× air objectives. Although changing the objective involves physically removing the objective and its mount, this is easily achieved due to the spring-loaded kinematic mounting system. The Cockpit mosaic function allows large areas of a sample to be mapped quickly (Figure 4(a)), and then regions of interest can be selected and marked for return later or the mosaic stopped and z-stacks in one or more channels collected. If a live sample is used the system can be used for time lapse imaging.

This system provides a demonstration of the power and portability of Cockpit and it functions as a test bed for new software developments on practical hardware. The system's small size and portability enable easy transport of the system for demonstrations and collaborative projects in other laboratories.

A complex structured illumination microscope with adaptive optics

In a further system, we are using Cockpit to maximise the performance of complex upright widefield structured illumination microscope, referred to as DeepSIM (Figure 7(a)). This incorporates not only translation stages, light sources, and cameras, but also a spatial light modulator (SLM) and a deformable

mirror (DM) for AO. This set-up is required to perform super-resolution live imaging experiments deep in tissue specimens, such as on the *Drosophila* larval neuro-muscular junction preparation, a powerful model system for understanding synapse biology on a molecular level. In order to achieve this imaging the system has to synchronise the illumination lasers, the SLM, a polarisation rotator, the DM, the Z stage position, and the cameras. The lasers, SLM, DM, and cameras receive digital triggers from the Red Pitaya, while the Z-position and polarisation rotator state are controlled via analogue voltages, all synchronised at the μ s level. Cockpit provides a simple user interface allowing selection of different experimental parameters, such as exposure time, laser power, Z step, and stack size (Figure 7(b)). The experiment module creates the relevant signals and timing information, transfers this to the Red Pitaya which then controls the hardware during the experiment.

Cockpit also provides a user-friendly interface to Microscope-AOTools³⁴, facilitating use of a wavefront sensor for calibration of the DM then adaptively correcting sample induced aberrations via image-based metrics, using so-called sensorless AO. This interface also implements IsoSense³³ to improve aberration detection over a wide range of complex samples.

Although this system is extremely complex with a multitude of devices the user interface is clean and easy to use. The system has sensible defaults, thus minimising the expertise and interaction required to collect experimental data.

Example simple extension in Python

We include an example script which demonstrates how the functionality of Cockpit can easily be extended in Python. The script finds DAPI stained cell nuclei in images, utilising the

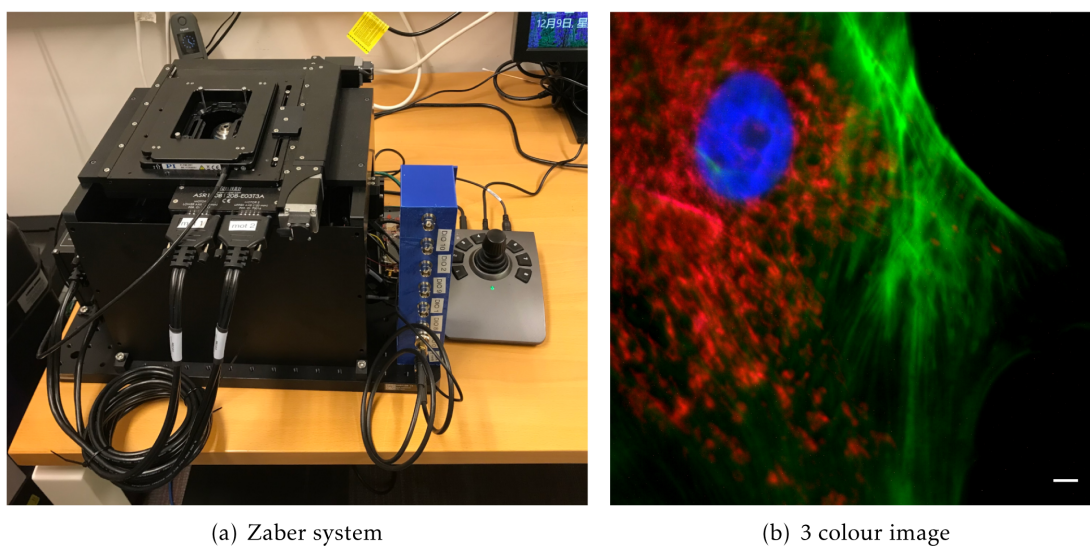


Figure 6. The Zaber microscope. (a) An image of the system, showing its compact size with a 300×450 mm footprint, (b) a three colour image showing the nucleus stained with DAPI in blue, the actin in green and mitochondria stained with MitoTracker Red CMXRos in Red. Scale bar 5 μ m.

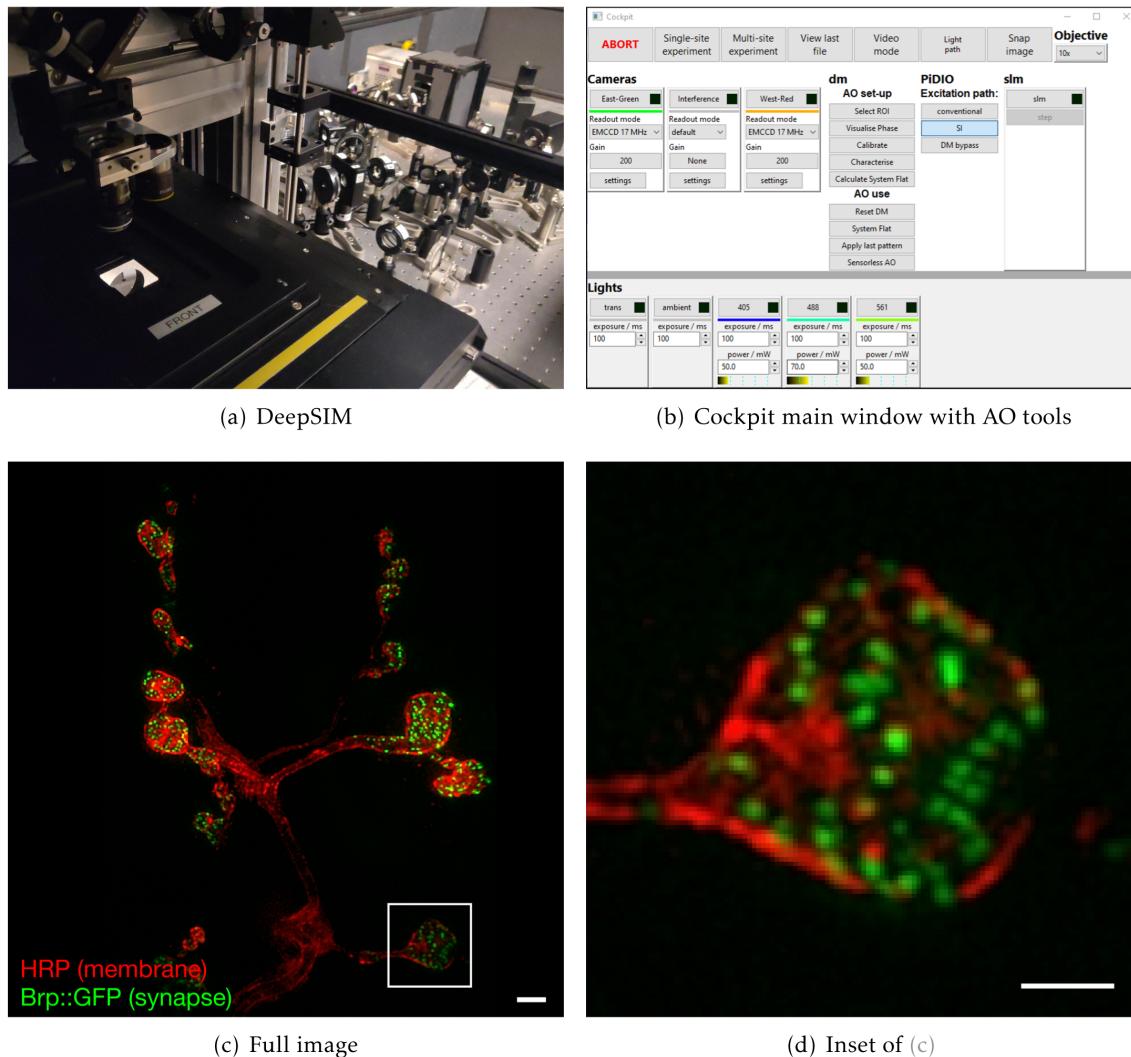


Figure 7. Example data from the DeepSIM system using Cockpit to image deep ($< 20 \mu\text{m}$) in *Drosophila* neuro-muscular junctions. The system uses sensorless AO to correct for sample induced aberrations. (a) system view from the stage; (b) main Cockpit window with the additional controls for the SLM and AO devices; (c) image of individual synapses at the *Drosophila* neuro-muscular junction using AO-SIM imaging, with Cy3 labelled HRP antibody labelling the neuronal membrane in red and Brp::GFP in the synapse in green; (d) zoomed in region of (c). Scale bars are $2 \mu\text{m}$.

mosaic functionality to scan large areas and the point marking features to record the centroids of the detected nuclei. The script utilises the OpenCV³⁵ framework to detect large roughly circular objects in images. Images taken for the mosaic are also trapped by the code which Gaussian blurs to reduce noise, binarise, and finally applies a Hough transform to find circular objects. The locations of these objects in the image are then transformed into stage coordinates and added to the marked point list. The Python code, a test image, and detailed instructions on how to setup a simulated microscope to run this without any microscope hardware is detailed at <https://doi.org/10.5281/zenodo.5745648>.

For this simple example the parameters are fixed and tuned to the data set used, and select a subset of nuclei within a

size range and with defined circularity. The current code also doesn't reliably detect nuclei at the edges between images in the mosaic scan. However, it can easily detect a large number of cell positions to create a point list which could then be run through a multi-site experiment to collect data such as a 3D, multi-channel stack, or time lapse on a large number of cells semi-automatically.

Discussion

We have developed a Python-based GUI for controlling bespoke microscopes which can map the entire slide or dish, allowing a real time exploration of the entire specimen landscape. It also connects to separate hardware timing devices to enable high precision timing of even complex devices and experiments. A key property of our software is that it is focused

around the needs of the user and their experimental design and workflow. Finally, using Python means that the microscope control can easily be modified to make use of the extensive machine learning algorithms available in easy to integrate libraries. This means that a microscope system that is controlled by Cockpit can readily be adapted to make decisions using machine learning algorithms during the acquisition process in order to modify the imaging conditions.

Existing control options

Cockpit has been introduced as an alternative to a number of existing microscope control options that are already available. Most published systems take one of three approaches: 1) utilise individual software packages provided by the device manufactures, 2) use LabVIEW to integrate LabVIEW-based drivers (VIs) provided by the manufacturers into a single custom GUI, or 3) use the open source microscope control package μ Manager³⁶. We will discuss the relative merits of each of these options in turn.

Using individual software packages to control each device is simple and direct. However this approach has several severe drawbacks. Each hardware device needs its own software, which takes up screen space and computer resources. This also removes the ability of different hardware parts to automatically interact with each other, for instance changing a stage's position once a camera has finished collecting images. Producing Z-stacks, or multi-channel images on such a setup is awkward, slow and open to human error. These factors can lead to corruption of the results.

LabVIEW is a visual programming tool that allows the construction of "programs" by connecting modules with wires and building graphical interfaces. Many manufacturers provide software to allow their hardware to be used in LabVIEW, through so-called virtual instruments (VIs). Building clean and simple visual framework to control highly complex advanced optical systems in LabVIEW requires advanced expertise in the program's use. Because such skills are usually beyond the ability of most scientists in an academic setting, many one-off complex bespoke systems, built to demonstrate a new principle, are very hard to use routinely by experimental scientists. Despite their obvious importance, such systems can rarely be reproduced and adopted by others for routine use.

μ Manager is an open source generalised microscope control interface which works on top of [ImageJ](#), a popular image analysis program. ImageJ is written in [Java](#), and μ Manager is written in a combination of [Java](#) and [C++](#). μ Manager's approach is most directly comparable to Cockpit. Many hardware device manufacturers provide μ Manager compatible libraries and instructions on how to connect and control their hardware. The package comes with mechanisms to run basic experiments, and the ability sequence commands. It should be noted that Pycro-Manager³⁷ has recently been developed and released to allow much closer integration between Python and μ Manager. Pycro-Manager seems likely to fill a similar position to Microscope-Cockpit, allowing direct Python based control of

imaging experiments, integration with online analysis, and a range of other functionality. As this is a recent development, we have not explored this package in any detail.

Being based on ImageJ, μ Manager, by design, includes a large application with multiple windows. This also means that the majority of the interface must be written in Java; however the system also needs a C/C++ layer to interface the Java code to C/C++ based system libraries. Cockpit is written in pure Python, relying on the Python-Microscope package for hardware interfacing. Producing a strict separation of the user interface from the hardware control components. This has the additional benefit that connected hardware may be physically located on another computer, increasing scalability and allowing devices requiring incompatible hardware or software to still be seamlessly integrated.

The mosaic window in Cockpit is similar in concept to Micro-Magellan²³, a μ Manager plugin. Being able to record a large area view, possibly from multiple areas, and have that available for instant navigation. The mosaic functionality dramatically speeds up experiment setup and finding the correct regions of interest, especially on bespoke systems with no eyepieces. Our mosaic interface utilises modern fast GPUs to enable instant access to even very large mosaic maps. Additionally, the touch screen interface allows easy and intuitive access to most of the functionality using a visual grammar that is very familiar to everyone.

The dedicated timing device interface allows fast and repeatable timing for both simple and complex experiments. It is useful in general but absolutely indispensable for experiments like the live cell adaptive optics SIM experiment ([Figure 7\(c\)](#)).

The standardised device interfaces from Python-Microscope mean that replacing one device with another of the same type is simply a matter of changing the address of the device in a configuration file.

[Table 1](#) shows a comparison between various features of the previously described microscope control software options. We have ranked features of the control software options with a traffic light colour scheme with green being best and red least good. The table includes the approaches mentioned above along with Matlab, a common alternative to LabVIEW, but a more conventional programming package with a range of extensions and with a large support base, and MetaMorph (<https://www.moleculardevices.com/products/cellular-imaging-systems/acquisition-and-analysis-software/metamorph-microscopy>) / SlideBook (<https://www.intelligent-imaging.com/slidebook>) included as representatives of commercial generalised microscope control packages.

Cockpit is still under active development, both within our labs and at several other laboratories across multiple countries. We are working on adding online image analysis to further enhance the mosaic functionality with machine learning, enabling the system to capture a large area of the sample quickly

Table 1. A symbolic representation of the relative strengths of different microscope control approaches in a number of areas with a traffic light colour scheme, with red worst, and green best. Separate control programs involve using a separate control program for each piece of hardware. In general, this software is free, however some components such as cameras might require a separate software purchase from the manufacturer. MetaMorph and SlideBook are examples of generalised commercial software designed for microscope control and provided by third parties, these are only two examples of a range of such packages.

Program	Usability	Ease of setup	Flexibility	Scalability	Complex functionality	Cost
Cockpit	Green	Orange	Green	Green	Green	Green
uManager	Green	Orange	Green	Orange	Orange	Green
Separate Control programs	Red	Green	Red	Red	Red	Orange
Matlab	Red	Red	Orange	Red	Red	Red
Labview	Red	Red	Orange	Red	Orange	Red
Metamorph/Slidebook	Green	Orange	Orange	Orange	Orange	Red

at low-resolution and then automatically identify features of interest for 3D-multi-channel acquisition.

Conclusions

We have developed Cockpit, a new paradigm for user-based software control of complex bespoke microscopes. The software is highly adaptable by a user or engineer with experience in Python. The key advantage of our approach is that a user is presented with a clean and simple interface that hides the complexity of the hardware, so they can focus on their experimental design and obtain data from the instrument with a high precision and reproducible workflow. We hope that the community will adopt this package and help us to continue to develop it.

Methods

CryoSIM imaging

The CryoSIM system has previously been published in detail^{30,31}. It was used to image HeLa cells grown on carbon coated gold EM grids under standard tissue culture conditions. Cells were labelled with MitoTracker Green (Thermo Fisher) at 100 nM and LysoTracker Red DND-99 (Thermo Fisher) at 50 nM after 30 minutes of incubation, to give green mitochondria and red lysosomes. The grids with live cells on them were then blotted and plunge frozen in liquid nitrogen cooled liquid ethane (Leica EM GP2) before being transferred to liquid nitrogen storage. Once frozen, grids were preserved in liquid nitrogen at cryogenic temperatures on the imaging systems to prevent thawing and detrimental ice crystal formation. Structured Illumination Microscopy (SIM) images were collected with a 100x 0.9NA objective, 488 nm and 561 nm laser excitation, and emission collected at 525/50 and 605/70 on two Andor iXon EMCCD cameras. Images were reconstructed using SoftWoRx (GE Healthcare) and image quality was assessed with SIMcheck³⁸.

Aurox AO imaging

The Aurox Clarity AO system is as previously published, the system utilised an Olympus IX70 microscope with a

60x 1.42NA objective. Illumination was provided by a CoolLED p-300³². This system was used to image *Drosophila melanogaster* neuro-muscular junctions (NMJ). The samples were prepared by following the protocol form Brent *et al.* 2009³⁹. 3rd instar *Drosophila melanogaster* larvae (Oregon-R strain) were dissected in HL3 buffer with 0.3mM Ca²⁺ to prepare a larval fillet. After this, larvae were fixed with 4% paraformaldehyde and blocked using 1% BSA³⁹. Larvae were stained overnight with 1:100 Horseradish Peroxidase (HRP) conjugated to Alexa568 fluorophore to visualise the neurons, and primary mouse antibody against Discs large (DLG) to visualise the postsynaptic density. The next day, the larvae were counter stained with secondary antibody to detect the DLG (1:200 donkey anti-mouse conjugated to Alexa488 fluorophore was used), as well as 1:1000 from 1 mg mL⁻¹ stock DAPI to visualise the nuclei. The larvae were then washed and mounted in 65% vectashield.

Zaber imaging

This system is based around a Zaber MVR motorised inverted microscope with Zeiss optics, and 4x 0.13NA and 63x 0.75NA Air objectives. Fluorescence illumination comes via 3 LEDs (385 nm, 473 nm and 568 nm) and a quad bandpass dichroic filter set (Chroma #89402). The system has a small format (Approx 30 mm cube) Ximea (model MQ042MGCM) camera and a Red Pitaya single board computer running as a hardware timing device. Multi channel images were taken by sequentially illuminating with the three different LEDs without changing the quad dichroic filter cube. This system was used to collect the example data for the simulated microscope and test Python script available at <https://doi.org/10.5281/zenodo.5745648>.

DeepSIM imaging

DeepSIM is a custom made upright structured illumination microscope with AO specifically designed for SIM super resolution imaging deep into live tissue (manuscript in preparation). On the DeepSIM system images were taken using 488 nm and 561 nm laser illumination. The system includes a Structure Light Modulator (SLM, Meadowlark) to provide structured

illumination for SIM imaging. The system also includes a deformable mirror (Alpao DM-69) for aberration correction. The system has a 60x 1.1NA water dipping objective and samples were imaged in aqueous buffer. The AO components were calibrated and controlled using Microscope-AOTools³⁴ to reduce aberrations and produce good SIM imaging at depth in biological samples.

The samples were prepared by following the protocol³⁹. 3rd instar *Drosophila melanogaster* larvae (Bruchpilot (Brp)-GFP strain) were dissected in HL3 buffer with 0.3mM Ca²⁺ to prepare a so-called larval fillet, and the larval brains were removed. After this, larvae were stained for 15 minutes with 1:50 Horseradish Peroxidase (HRP) conjugated to Cy3 fluorophore to visualise the neurons, washed with HL3 buffer with 0.3mM Ca²⁺ and imaged in HL3 buffer with 0mM Ca²⁺ to prevent the larvae from moving.

Data availability

No data are associated with this article.

Software availability

Software available from: <https://pypi.org/project/microscope-cockpit/>

Source code available from: <https://github.com/MicronOxford/cockpit>

Archived source code at time of publication (version 2.9.1): <https://zenodo.org/record/4542863>²⁵

Example Python script and test data set along with instructions on how to set it up are available at <https://doi.org/10.5281/zenodo.5745648>

Cockpit is distributed under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Acknowledgements

We are grateful to the original UCSF software team led by John Sedat, including Melvin Jones, Chris Weisiger and Sebastian Haase for the initial writing of the Python code used to control the OMX and OMX-T microscopes, which was subsequently released as open source software and formed the starting point for this project. We wish to thank many colleagues who have provided helpful suggestions, discussions and testing during development of Cockpit, the Micron Advanced Bioimaging Unit particularly Mantas Žurauskas and Andrew Jefferson, Maria Harkiolaki from the Diamond Light Source beamline B24. Oscar Davis for help on the interface look and feel. An earlier version of this article can be found on bioRxiv (<https://doi.org/10.1101/2021.01.18.427171>).

References

- Carlton PM, Boulanger J, Charles K, et al.: **Fast live simultaneous multiwavelength four-dimensional optical microscopy.** *Proc Natl Acad Sci U S A.* 2010; **107**(37): 16016–16022.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Stelzer EHK: **Light-sheet fluorescence microscopy for quantitative biology.** *Nat Methods.* 2015; **12**(1): 23–26.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Chen BC, Legant WR, Wang K, et al.: **Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution.** *Science.* 2014; **346**(6208): 1257998.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Gustafsson MG: **Surpassing the lateral resolution limit by a factor of two using structured illumination microscopy.** *J Microsc.* 2000; **198**(Pt 2): 82–87.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Pitrone PG, Schindelin J, Stuyvenberg L, et al.: **OpenSPIM: an open-access light-sheet microscopy platform.** *Nat Methods.* 2013; **10**(7): 598–599.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Markwirth A, Lachetta M, Mönkemöller V, et al.: **Video-rate multi-color structured illumination microscopy with simultaneous real-time reconstruction.** *Nat Commun.* 2019; **10**(1): 4315.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Zdankowski P, McGloin D, Swedlow JR: **Full volume super-resolution imaging of thick mitotic spindle using 3D AO STED microscope.** *Biomed Opt Express.* 2019; **10**(4): 1999–2009.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Eilers Y, Ta H, Gwosch KC, et al.: **MINFLUX monitors rapid molecular jumps with superior spatiotemporal resolution.** *Proc Natl Acad Sci U S A.* 2018; **115**(24): 6117–6122.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Holden SJ, Uphoff S, Kapanidis AN: **DAOSTORM: an algorithm for high-density super-resolution microscopy.** *Nat Methods.* 2011; **8**(4): 279–280.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Marsh RJ, Pfisterer K, Bennett P, et al.: **Artifact-free high-density localization microscopy analysis.** *Nat Methods.* 2018; **15**(9): 689–692.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Rosten E, Jones GE, Cox S: **ImageJ plug-in for bayesian analysis of blinking and bleaching.** *Nat Methods.* 2013; **10**(2): 97–98.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Crossen J, Hinsdale T, Thorsen RØ, et al.: **Localization microscopy at doubled precision with patterned illumination.** *Nat Methods.* 2020; **17**(1): 59–63.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Nadella KMNS, Roš H, Baragli C, et al.: **Random-access scanning microscopy for 3D imaging in awake behaving animals.** *Nat Methods.* 2016; **13**(12): 1001–1004.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- York AG, Chandris P, Nogare DD, et al.: **Instant super-resolution imaging in live cells and embryos via analog image processing.** *Nat Methods.* 2013; **10**(11): 1122–1126.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Dobbie IM, King E, Parton RM, et al.: **Omx: A new platform for multimodal, multichannel wide-field imaging.** *Cold Spring Harb Protoc.* 2011; **2011**(8): 899–909.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Schermelleh L, Carlton PM, Haase S, et al.: **Subdiffraction multicolor imaging of the nuclear periphery with 3D structured illumination microscopy.** *Science.* 2008; **320**(5881): 1332–1336.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Booth MJ: **Adaptive optical microscopy: the ongoing quest for a perfect image.** *Light Sci Appl.* 2014; **3**(4): e165.
[Publisher Full Text](#)

18. Liu TL, Upadhyayula S, Milkie DE, *et al.*: **Observing the cell in its native state: Imaging subcellular dynamics in multicellular organisms.** *Science*. 2018; **360**(6386): eaaq1392.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
19. Turcotte R, Liang Y, Tanimoto M, *et al.*: **Dynamic super-resolution structured illumination imaging in the living brain.** *Proc Natl Acad Sci U S A*. 2019; **116**(19): 9586–9591.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
20. Gould TJ, Burke D, Bewersdorf J, *et al.*: **Adaptive optics enables 3D STED microscopy in aberrating specimens.** *Opt Express*. 2012; **20**(19): 20998–21009.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
21. Kner P, Winoto L, Agard DA, *et al.*: **Closed loop adaptive optics for microscopy without a wavefront sensor.** In: *Three-Dimensional and Multidimensional Microscopy: Image Acquisition and Processing XVII. Proc SPIE Int Soc Opt Eng*. 2010; **7570**: 757006.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
22. Kner P, Sedat JW, Agard DA, *et al.*: **High-resolution wide-field microscopy with adaptive optics for spherical aberration correction and motionless focusing.** *J Microsc*. 2010; **237**(2): 136–147.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
23. Pinkard H, Stuurman N, Corbin K, *et al.*: **Micro-Magellan: open-source, sample-adaptive, acquisition software for optical microscopy.** *Nat Methods*. 2016; **13**(10): 807–809.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
24. Susano Pinto DM, Phillips MA, Hall N, *et al.*: **Python- Microscope – a new open-source Python library for the control of microscopes.** *J Cell Sci*. 2021; **134**(19): jcs258955.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
25. Mick, Pinto DMS, Dobbie I, *et al.*: **MicronOxford/cockpit release-2.9.1.** (Version release-2.9.1) *Zenodo*. 2021.
<http://www.doi.org/10.5281/zenodo.4542863>
26. Cheng A, Henderson R, Mastrorade D, *et al.*: **Mrc2014: Extensions to the mrc format header for electron cryo-microscopy and tomography.** *J Struct Biol*. 2015; **192**(2): 146–150.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
27. Hunter JD: **Matplotlib: A 2D graphics environment.** *Comput Sci Eng*. 2007; **9**(3): 90–95.
[Publisher Full Text](#)
28. Harris CR, Millman KJ, van der Walt SJ, *et al.*: **Array programming with NumPy.** *Nature*. 2020; **585**(7825): 357–362.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
29. Virtanen P, Gommers R, Oliphant TE, *et al.*: **SciPy 1.0: Fundamental algorithms for scientific computing in python.** *Nat Methods*. 2020; **17**(3): 261–272.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
30. Kounatidis I, Stanifer ML, Phillips MA, *et al.*: **3D correlative cryo-structured illumination fluorescence and soft X-ray microscopy elucidates reovirus intracellular release pathway.** *Cell*. 2020; **182**(2): 515–530.e17.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
31. Phillips MA, Harkiolaki M, Pinto DMS, *et al.*: **CryoSIM: super resolution 3D structured illumination cryogenic fluorescence microscopy for correlated ultra-structural imaging.** *bioRxiv*. 2020.
[Publisher Full Text](#)
32. Hussain SA, Kubo T, Hall N, *et al.*: **Wavefront-sensorless adaptive optics with a laser-free spinning disk confocal microscope.** *J Microsc*. 2020.
[PubMed Abstract](#) | [Publisher Full Text](#)
33. Žurauskas M, Dobbie IM, Parton RM, *et al.*: **IsoSense: frequency enhanced sensorless adaptive optics through structured illumination.** *Optica*. 2019; **6**(3): 370–379.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
34. Hall N, Titlow J, Booth MJ, *et al.*: **Microscope-AOtools: a generalised adaptive optics implementation.** *Opt Express*. 2020; **28**(20): 28987–29003.
[PubMed Abstract](#) | [Publisher Full Text](#)
35. Bradski G: **The OpenCV Library.** *Dr. Dobbs Journal of Software Tools*. 2000.
[Reference Source](#)
36. Edelstein A, Amodaj N, Hoover K, *et al.*: **Computer control of microscopes using µmanager.** *Curr Protoc Mol Biol*. 2010; **92**(1): 14–20.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
37. Pinkard H, Stuurman N, Ivanov IE, *et al.*: **Pycro-manager: open-source software for customized and reproducible microscope control.** *Nat Methods*. 2021; **18**(3): 226–228.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
38. Ball G, Demmerle J, Kaufmann R, *et al.*: **SIMcheck: a toolbox for successful super-resolution structured illumination microscopy.** *Sci Rep*. 2015; **5**: 15915.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
39. Brent JR, Werner KM, McCabe BD: **Drosophila larval NMJ dissection.** *J Vis Exp*. 2009; (24): 1107.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)

Open Peer Review

Current Peer Review Status:      

Version 2

Reviewer Report 21 June 2023

<https://doi.org/10.21956/wellcomeopenres.19384.r58035>

© 2023 Moreno X. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Xavier Casas Moreno 

KTH Royal Institute of Technology, Stockholm, Sweden

The article presents Microscope-Cockpit, a software solution for microscope control that uses a software model for controlling multiple devices and implements a graphical user interface (GUI) which is user friendly for biologists. The authors present several use-cases that are promising and position Cockpit as a state-of-the-art tool. I have some comments that I would like the authors to address, after which the article should be ready for indexing.

The authors compare Microscope-Cockpit with different state-of-the-art alternatives, such as LabView, uManager and custom code. However, in the recent years (2021-2023) there have been different initiatives to provide similar solutions in Python. How does Microscope-Cockpit relate to these efforts, and how do they complement each other? Some examples are ImSwitch, PycroManager, PYME, miEye, etc. In my opinion mentioning these will strengthen the current efforts in Python for microscope control.

The authors use the dataformat ".dv". I miss a comparison between this and the state-of-the-art in microscopy research (tiff, hdf5 and ome-zarr). They are explained and compared here: <https://www.nature.com/articles/s41592-021-01326-w>¹. In case someone wants to provide support for these other formats, how difficult would it be to do so in Microscope-Cockpit?

It is mentioned that Cockpit does not require a direct connection since it uses Pyro. Is Pyro always needed or is it an option when controlling remote objects? How does it work when the devices are in the local machine? And how are both approaches compared in terms of latency?

The authors explain briefly the software model of Microscope-Cockpit, using devices and handlers, and the interfaces. I think it would be helpful to have a figure with a diagram of the software model, to see how the different layers relate to each other. Also, is the software model implementing a software design pattern?

Thank you in advance.

References

1. Moore J, Allan C, Besson S, Burel JM, et al.: OME-NGFF: a next-generation file format for expanding bioimaging data-access strategies. *Nat Methods*. 2021; **18** (12): 1496-1498 [PubMed Abstract](#) | [Publisher Full Text](#)

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Super-resolution microscopy, including software for microscope control.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 02 June 2023

<https://doi.org/10.21956/wellcomeopenres.19384.r58034>

© 2023 Wollman A. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Adam Wollman 

Newcastle University, Newcastle upon Tyne, England, UK

Microscope-Cockpit is an open source python software package for controlling bespoke microscopes. Although there is existing software for this (e.g. micro-manager), more competition in this space can only be a good thing for the microscopy community. Cockpit also has a number of advantages over other open source microscopy software. It is entirely based in python, making

it natively compatible with lots of existing image software in python and easily built upon by the large python community. It has a touch screen interface and it is explicitly designed around mosaic imaging of multiple connected fields of view. This latter feature also leads to its two small possible weak points. The choice of file format is understandable considering mosaic imaging but it would have been nice to see native support for other image formats e.g. ome.tif . The software also appears to require a motorised x,y,z stage which again is understandable for mosaic imaging but not a requirement for lots of other types of microscope. Also, could the authors clarify, does this statement, 'In addition, experiments require a device that can be programmed as source of hardware triggers, supplying digital signals and, optionally, analogue voltages.' mean Cockpit will also only work with a control board for hardware triggers? Or will it work to some degree without?

Otherwise Cockpit will be an extremely useful tool for the microscopy community going forward.

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Microscopy development, image analysis software

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 01 June 2023

<https://doi.org/10.21956/wellcomeopenres.19384.r58030>

© 2023 Tutkus M. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

**Marijonas Tutkus** ¹ Vilnius University Life Sciences Centre, Vilnius, Lithuania² Center for Physical Sciences and Technology, Vilnius, Lithuania

This paper describes “Microscope-Cockpit” (Cockpit), which is adaptable open-source user-friendly Python-based Graphical User Interface (GUI). For hardware control it is using “Python-Microscope” (Microscope), and it includes coordinated control of hardware devices. It was verified using several bespoke microscopes, including a simple widefield system and a complex system with adaptive optics and structured illumination. It is written in the text that this system allows for using several cameras, but I did not see any actual demonstration of simultaneous operation with several cameras.

This Cockpit GUI allows for control of bespoke microscope systems, navigation of the entire slide landscape, and automated acquisition without the use of eyepieces. One of the functions that I miss a lot in this work is any sample auto-focusing pipeline. Sample's Z drift correction using IR laser back reflection or similar approaches became very popular and important for many experiments.

A key strength of Cockpit is its use of Python, which means that any microscope built with Cockpit is ready for future customisation by simply adding new libraries, for example machine learning algorithms to enable automated microscopy decision making while imaging. However, I would like to make a note that making a universal software platform for any bespoke microscopy system is a challenging task, and probably each new case will require readjustments of the Cockpit.

In this paper several software packages are compared to the Cockpit (such as LabVIEW, Matlab, µManager). It is important to mention that there are many more open-source alternatives that µManager that are able to do either hardware control or data analysis or both. Most of them are listed [here](#). Recently newly developed open-source Python-based systems, such as the Sync-Scope,¹ and miEye² appeared. They both allow for similar functionality as the Cockpit.

Other specific comments.

1. It is written on page 3 “Many bespoke microscope systems that lack eyepieces, such as systems built by physical scientists in collaboration with biologists, make sample navigation particularly challenging.”. To be precise, it all depends on what is the sample. If one is looking into diffraction limited spots that are dim, looking through eyepieces would not help to navigate around the sample.
2. On page 4 it is written that “the principal design goals for the Cockpit interface were to” “Hide from the user the complexity required to control an advanced microscope”. I think it should be intuitive, not necessarily hidden.
3. On page 7: “Experimental images are saved into files utilising the ‘.dv’ file format,”. I would like to see explanation why authors decided not to use tiff or zarr formats.
4. On page 7: “the mosaic window enables navigation of the sample relative to an overview formed of tiled images;”. Does this GUI window allows for any data analysis on site? e.g. applying different LUT, adjusting intensity scale, etc.

5. On page 7: "Cockpit requires Python 3.5 or later". Different Python versions often create compatibility problems. Did you check functionality of the Cockpit in different versions of Python?
6. On page 10: "The system has 4× and 63× air objectives. Although changing the objective involves physically removing the objective and its mount, this is easily achieved due to the spring-loaded kinematic mounting system.". Doesn't that mean that one has to remove the sample when changing objectives? In such a case, returning back to the original positions would be challenging.
7. On page 10: "If a live sample is used the system can be used for time lapse imaging.". Probably any dynamic sample can be used for time lapse imaging regardless it is living cells or just molecules in vitro.
8. On page 11: "We have developed a Python-based GUI for controlling bespoke microscopes which can map the entire slide or dish, allowing a real time exploration of the entire specimen landscape.". I suggest using "interactive exploration".
9. On page 12: "This has the additional benefit that connected hardware may be physically located on another computer, increasing scalability and allowing devices requiring incompatible hardware or software to still be seamlessly integrated.". In such case latency might become an issue. Please, provide explanation on this matter.
10. On page 12: "We have ranked features of the control software options with a traffic light colour scheme with green being best and red least good. The table includes the approaches mentioned above along with Matlab, a common alternative to LabVIEW, but a more conventional programming package with a range of extensions and with a large support base, and MetaMorph (<https://www.moleculardevices.com/products/cellular-imaging-systems/acquisition-and-analysis-software/metamorph-microscopy>) / SlideBook (<https://www.intelligent-imaging.com/slidebook>) included as representatives of commercial generalised microscope control packages.". I think, Igor Pro is also used relatively widely in the community to control microscope's hardware and perform data analysis.

References

1. Casas Moreno X, Silva MM, Roos J, Pennacchietti F, et al.: An open-source microscopy framework for simultaneous control of image acquisition, reconstruction, and analysis.*HardwareX*. 2023; **13**: e00400 [PubMed Abstract](#) | [Publisher Full Text](#)
2. Alsamsam MN, Kopūstas A, Jurevičiūtė M, Tutkus M: The miEye: Bench-top super-resolution microscope with cost-effective equipment.*HardwareX*. 2022; **12**: e00368 [PubMed Abstract](#) | [Publisher Full Text](#)

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: single-molecule biophysics, super-resolution microscopy, image analysis, open-source microscopy.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 01 June 2023

<https://doi.org/10.21956/wellcomeopenres.19384.r58037>

© 2023 Knapper J. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Joe Knapper 

Physics and Astronomy, University of Glasgow, Glasgow, Scotland, UK

I'd like to thank the authors for sharing this work, based around making the control of microscopy systems more intuitive for the user based on a GUI for Python-Microscope. I agree with the suggestions of the first round of reviews, and agree with the authors that these changes improved the article. While I recommend the article and tool are ready for publication, I would like to raise the following questions.

The inclusion of Table 1 gives more context for Microscope-Cockpit, but I hope the authors can share how it was assessed? As a subjective ranking, is it based on the feedback of multiple users and developers, public forums, etc?

Can the authors expand on the ease of use of exporting the '.dv' format? I note they list it is supported in ImageJ, OMERO, Matlab and Chromagnon, but as this work is Python-based, can the authors suggest a (preferably native) Python-compatible solution? omero-py, pyimagej, or ideally something only requiring a pypi installation? The first sentence of "Output image files" is also hard

to follow - does a single '.dv' file contain multiple images, wavelengths, z-slices and time points?

The GPU-based mosaic for multiple large area scans appears to be an important feature of this work, and very useful for potential users. Can the authors clarify if this is positioning and blending images together based on areas of overlap, or simply placing images directly based on the stage position?

Finally, a note to the authors that currently it appears that Python ≥ 3.10 is incompatible, based on wxPython depending on attrdict. Whether this is addressed in the paper or the program is up to the authors.

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Python, microscope control, image tiling

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 29 September 2021

<https://doi.org/10.21956/wellcomeopenres.18310.r45668>

© 2021 Swedlow J et al. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

**Jason R Swedlow**

¹ Centre for Gene Regulation and Expression, School of Life Sciences, University of Dundee, Dundee, UK

² Centre for Gene Regulation and Expression, School of Life Sciences, University of Dundee, Dundee, UK

Emil Rozbicki

¹ Glencoe Software, Inc, Seattle, WA, USA

² Glencoe Software, Inc, Seattle, WA, USA

William Moore

¹ OME, University of Dundee, Dundee, UK

² OME, University of Dundee, Dundee, UK

Sébastien Besson

¹ Centre for Gene Regulation and Expression, School of Life Sciences, University of Dundee, Dundee, UK

² Centre for Gene Regulation and Expression, School of Life Sciences, University of Dundee, Dundee, UK

This paper introduces Microscope-Cockpit, a graphical user interface software that has been built on top of the historical software used to control the OMX microscope. It aims to provide a clean and user friendly interface allowing to create complex experimental set-up. Several use cases are included in the article to support the capacity of the software to adapt to different experimental set-ups. Overall the tool described in this article is scientifically valid but we would like to suggest some revisions detailed below.

The Implementation section details the Python requirements of Microscope-Cockpit. Amongst those packages, the Python-microscope library appears as one of the most critical dependencies since it contains all the logic and support for handling and controlling microscopy devices. As part of this review, it came to our attention that this library has been recently published (<https://doi.org/10.1242/jcs.258955>)¹ including a supplementary figure Fig S1 which cross-references Fig 1 of this paper. The article should thus be amended to cross-reference the Python-microscope publication wherever appropriate.

The article makes no mention of the file format written by Microscope-Cockpit. The Python-microscope paper does not include any detail about the choice of data output for end-users either. From our minimal testing of the application, the only option available to the user is to write data as "DV files (.dv)", a proprietary file format. Is this the only choice available to the user? If so, this feels like a limitation of the current implementation which contradicts the claim for the open and universal mission of Cockpit. We suggest a paragraph should be added to the Implementation section that discusses the Data Generation/Output including a description of the file format as well as the tooling available for end-users of the acquisition system to read and analyze imaging data generated by Cockpit.

In the section discussing Cockpit in relationship Micro-Manager, a proposed key strength of Cockpit is its Python implementation allowing the "strict separation of the user interface from the hardware control components" while "the majority of [Micro-Manager] interface must be written in Java; however the system also needs a C/C++ layer to interface the Java code to C/C++ based system libraries". This argument omits Pycro-Manager, a Python bridge for Micro-Manager, published earlier this year (<https://doi.org/10.1038/s41592-021-01087-6>)² and mentioned in the

aforementioned Python-microscope publication. The section mentioned above should be amended to cite this paper and include Pycro-Manager in the comparison with Cockpit. Much is made of the ability to use Python for custom modifications and the integration of third-party libraries. The text mentions that “the microscope control can easily be modified to make use of the extensive machine learning algorithms available in easy to integrate libraries” but this sentence rather suggests the extensibility is a property of the underlying python-microscope library rather than the Cockpit interface. The Cockpit reference documentation (<https://www.micron.ox.ac.uk/software/cockpit/>) primarily describes the installation and the configuration of the software. To support the extensibility claim, we would suggest either to introduce a public example of Python extension to the default Cockpit UI and/or an amendment to the online documentation that would demonstrate this extension which could be linked from the article.

Together with python-microscope paper, this paper introduces a new open-source solution for microscope control and imaging acquisition. Both in the introduction and in the discussion, Cockpit is compared to several equivalent commercial and open-source solutions, including the well-established Micro-Manager. This comparison is important and particularly useful for end users and imaging facility managers who need to make informed decisions e.g. when investing in new equipment or technologies. We suggest to make the outcome of this discussion more explicit and prominent e.g. under a form of a summary table that would include the different available ecosystems (individual hardware solutions, LabView, python-microscope/Microscope-Cockpit and Micro-Manager/Pycro-Manager) and compare their respective advantages and drawbacks.

References

1. Pinto DMS, Phillips MA, Hall N, Mateos-Langerak J, et al.: Python-microscope: A new open source Python library for the control of microscopes. *J Cell Sci.* 2021. [PubMed Abstract](#) | [Publisher Full Text](#)
2. Pinkard H, Stuurman N, Ivanov I, Anthony N, et al.: Pycro-Manager: open-source software for customized and reproducible microscope control. *Nature Methods.* 2021; **18** (3): 226-228 [Publisher Full Text](#)

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

No

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: bioimaging, image informatics, software engineering, data management

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however we have significant reservations, as outlined above.

Author Response 14 Dec 2021

Ian Dobbie

Response to review from authors.

Authors responses are in *italics*

New text in the manuscript are in **bold**

The Implementation section details the Python requirements of Microscope-Cockpit. Amongst those packages, the Python-microscope library appears as one of the most critical dependencies since it contains all the logic and support for handling and controlling microscopy devices. As part of this review, it came to our attention that this library has been recently published (<https://doi.org/10.1242/jcs.258955>)¹ including a supplementary figure Fig S1 which cross-references Fig 1 of this paper. The article should thus be amended to cross-reference the Python-microscope publication wherever appropriate.

We thank the reviewers for pointing this out. The Python-Microscope publication has been accepted at JCS since we submitted the article. We have revised the manuscript to reference the published article rather than the biorXiv preprint.

The article makes no mention of the file format written by Microscope-Cockpit. The Python-microscope paper does not include any detail about the choice of data output for end-users either. From our minimal testing of the application, the only option available to the user is to write data as "DV files (.dv)", a proprietary file format. Is this the only choice available to the user? If so, this feels like a limitation of the current implementation which contradicts the claim for the open and universal mission of Cockpit. We suggest a paragraph should be added to the Implementation section that discusses the Data Generation/Output including a description of the file format as well as the tooling available for end-users of the acquisition system to read and analyze imaging data generated by Cockpit.

We thank the reviewers for raising this point. While it is correct that Microscope-Cockpit only saves images in the '.dv' file format, it is a slight variant of the open MRC file format commonly used in the electron microscopy community. The MRC format is extensively documented at https://www.ccpem.ac.uk/mrc_format/mrc2014.php. The slight variations to the encoded metadata reflect differences between the relevant data in optical and electron microscopy. Crucially, there is considerable support within open source software that supports '.dv' files, including Bio-Formats for importing data into ImageJ, OMERO and Matlab. Therefore, users can easily convert to other formats and use popular programs to analyse the data output by Microscope-Cockpit based microscopes. We have revised the manuscript to address this point

and changed the software package documentation to include an explanation of this as well as information about the software that supports the '.dv' format, and explicit definition of what metadata we save in the files. We have added a section on output image files which explicitly mentions these points:

Output image files

Experimental images are saved into files utilising the '.dv' file format, typically multiple wavelengths, Z-slices and time points into a single file. Images from the mosaic window and single snaps can also be saved into this format, with the mosaic saved files having an associated text file defining XYZ positions of each collected image. The '.dv' format is an extension of the mrc file format, defined in detail in the MRC/CCP4 2014 file format specification. The CCP4 consortium of the EM community continue to support and extend this file format. This support includes file validators and a detailed specification, which is compatible with the files used here but not identical.

The optical microscopy specific metadata are covered in the documentation. Although relatively uncommon, the file format is supported by the Bio-Formats project allowing import of '.dv' files, along with the associated metadata into software using this library including ImageJ, OMERO and Matlab. Additionally, the Chromagnon image alignment tool will read and write '.dv' files and it is the native format for DeltaVision microscopes utilising the commercial package SoftWoRx.

In the section discussing Cockpit in relationship Micro-Manager, a proposed key strength of Cockpit is its Python implementation allowing the "strict separation of the user interface from the hardware control components" while "the majority of [Micro-Manager] interface must be written in Java; however the system also needs a C/C++ layer to interface the Java code to C/C++ based system libraries". This argument omits Pycro-Manager, a Python bridge for Micro-Manager, published earlier this year (<https://doi.org/10.1038/s41592-021-01087-6>)² and mentioned in the aforementioned Python-microscope publication. The section mentioned above should be amended to cite this paper and include Pycro-Manager in the comparison with Cockpit.

We thank the reviewers for pointing out this paper which was again published since our original submission and we have now added discussion about Pycro-Manager to the paper. We have revised the manuscript by adding the following text to the discussion section:

It should be noted that Pycro-Manager [pinkard2021] has recently been developed and released to allow much closer integration between Python and uManager. Pycro-Manager seems likely to fill a similar position to Microscope-Cockpit, allowing direct Python based control of imaging experiments, integration with online analysis and a range of other functionality. As this is a recent development, we have not explored this package in any detail.

Much is made of the ability to use Python for custom modifications and the integration of third-party libraries. The text mentions that “the microscope control can easily be modified to make use of the extensive machine learning algorithms available in easy to integrate libraries” but this sentence rather suggests the extensibility is a property of the underlying python-microscope library rather than the Cockpit interface. The Cockpit reference documentation (<https://www.micron.ox.ac.uk/software/cockpit/>) primarily describes the installation and the configuration of the software. To support the extensibility claim, we would suggest either to introduce a public example of Python extension to the default Cockpit UI and/or an amendment to the online documentation that would demonstrate this extension which could be linked from the article.

The reviewers make a good point that would significantly strengthen the paper. We have written a short script that utilises some of the embedded functionality to find and mark cell nuclei in a typical DAPI stained sample. The code, along with a test image set, config files and detailed instructions on how to set it up are collected into a zenodo repository and available via a doi (10.5281/zenodo.5745648) link. In fact, this simulated microscope approach is a very powerful approach to setting up and testing such scripts. We have also added the following text to the manuscript.

Example simple extension in Python

We include an example script which demonstrates how the functionality of cockpit can easily be extended in python. The script finds DAPI stained cell nuclei in images, utilising the mosaic functionality to scan large areas and the point marking features to record the centroids of the detected nuclei. The script utilises the OpenCV framework to detect large roughly circular objects in images. Images taken for the mosaic are also trapped by the code which Gaussian blurs to reduce noise, binarise and finally applies a Hough transform to find circular objects. The locations of these objects in the image are then transformed into stage coordinates and added to the marked point list. The Python code, a test image and detailed instructions on how to setup a simulated microscope to run this without any microscope hardware is detailed at <https://doi.org/10.5281/zenodo.5745648>.

For this simple example the parameters are fixed and tuned to the data set used, and select a subset of nuclei within a size range and with defined circularity. The current code also doesn't reliably detect nuclei at the edges between images in the mosaic scan. However, it can easily detect a large number of cell positions to create a point list which could then be run through a multi-site experiment to collect data such as a 3D, multi-channel stack, or time lapse on a large number of cells semi-automatically.

Together with python-microscope paper, this paper introduces a new open-source solution for microscope control and imaging acquisition. Both in the introduction and in the discussion, Cockpit is compared to several equivalent commercial and open-source solutions, including the well-established Micro-Manager. This comparison is important and particularly useful for end users and imaging facility managers who need to make informed decisions e.g. when investing in new equipment or technologies. We suggest to make the

outcome of this discussion more explicit and prominent e.g. under a form of a summary table that would include the different available ecosystems (individual hardware solutions, LabView, python-microscope/Microscope-Cockpit and Micro-Manager/Pycro-Manager) and compare their respective advantages and drawbacks.

We thank the reviewers for this suggestion and we have added a table of feature comparison between a range of microscope control options as a traffic light colour coded table and the following text to the manuscript.

Table 1 shows a comparison between various features of the previously described microscope control software options. We have ranked features of the control software options with a traffic light colour scheme with green being best and red least good. The table includes the approaches mentioned above along with Matlab, a common alternative to LabVIEW, a more conventional programming package with a range of extensions and with a large support base, and MetaMorph/ SlideBook included as representatives of commercial generalised microscope control packages.

[New table included in revised version]

Table 1: A symbolic representation of the relative strengths of different microscope control approaches in a number of areas with a traffic light colour scheme, with red worst and green best. Separate control programs involve using a separate control program for each piece of hardware. In general, this software is free, however some components such as cameras might require a separate software purchase from the manufacturer. MetaMorph and SlideBook are examples of generalised commercial software designed for microscope control and provided by third parties, these are only two examples of a range of such packages.

Competing Interests: No competing interests were disclosed.

Reviewer Report 14 September 2021

<https://doi.org/10.21956/wellcomeopenres.18310.r45664>

© 2021 Harrington K. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Kyle Harrington 

¹ Max Delbrueck Center for Molecular Medicine, Berlin, Germany

² Max Delbrueck Center for Molecular Medicine, Berlin, Germany

What about the availability of open source drivers through micromanager? Although it is

somewhat recognized that the micromanager API/architecture is not ideal, it is certainly recognized that the large number of already existing drivers is a key strength. How is the availability of drivers for Cockpit?

How does Cockpit compare to Pycro-manager (Pycro-Manager: open-source software for customized and reproducible microscope control, Pinkard *et al.*, 2021, Nature Methods),¹ which enjoys the benefits of the existing micromanager ecosystem while also providing a Python interface?

Overall, Cockpit seems like a strong contribution. I see a lot of potential, but it is not clear that it is quite competitive with micromanager yet (which is fine, that will come with time). Integration with a tool like napari would be a great choice for broadening the usability of Cockpit. It is nice to see that Cockpit is being tested on multiple systems, although as mentioned it would be good to see a fair comparison between Cockpit's current (and future planned) drivers relative to a mature project like micromanager.

References

1. Pinkard H, Stuurman N, Ivanov I, Anthony N, et al.: Pycro-Manager: open-source software for customized and reproducible microscope control. *Nature Methods*. 2021; **18** (3): 226-228 [Publisher Full Text](#)

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Image analysis, Computer science, Software engineering, Computational biology, Python, Java, C++, Lisp

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 14 Dec 2021

Ian Dobbie

Response to review from Authors.

What about the availability of open source drivers through micromanager? Although it is somewhat recognized that the micromanager API/architecture is not ideal, it is certainly recognized that the large number of already existing drivers is a key strength. How is the availability of drivers for Cockpit?

This is an interesting idea and we have considered it. There are two major arguments against this approach. Firstly this would require importing the whole of the uManager infrastructure, the java JVM etc... Secondly our brief exploration appeared to show that matching devices to the fixed device type specific API in Python-Microscope, the library we use to interface directly to the hardware, would involve substantial effort for each device. We felt it was more productive to apply this effort to implementing the device directly in Python. This probably took roughly the same effort and enabled us to avoid the uManager overhead.

How does Cockpit compare to Pycro-manager (Pycro-Manager: open-source software for customized and reproducible microscope control, Pinkard *et al.*, 2021, Nature Methods), which enjoys the benefits of the existing micromanager ecosystem while also providing a Python interface?

We thank the reviewer for pointing out this paper which was published since our original submission and we have now added discussion about Pycro-Manager to the paper. We have revised the manuscript by adding the following text to the discussion section:

It should be noted that Pycro-Manager [pinkard2021] has recently been developed and released to allow much closer integration between Python and uManager. This seems likely to fill a similar position as Microscope-Cockpit, allowing direct Python based control of imaging experiments, integration with online analysis and a range of other functionality. As this is a recent development we have not explored this package in any detail.

Competing Interests: No competing interests were disclosed.