µManager: Open Source Software for Light Microscope Imaging

Nico Stuurman, Nenad Amdodaj and Ron Vale
University of California, San Francisco, CA
nico@cmpmail.ucsf.edu

Introduction. No longer are biologists content to peer through the oculars of a microscope. Eyes have been replaced by digital cameras. Instead of manual control of microscope stages, filters and light sources, modern microscopes have become fully robotic. Image acquisition and robotic movements require computer control. With such equipment it is possible to automatically record images at multiple wavelengths, carry out time-lapse recordings of living cells using very short exposure times and low light doses, and even to record images at multiple positions (*e.g.* from a multi-well plate) without operator intervention. Clearly, computer control of light microscope image acquisition is making many new experimental imaging strategies possible in the biological sciences.

A very important part of this technological advancement is the software that controls the equipment. There are currently about 15 commercial software packages available for light microscope image acquisition. However, each package usually controls a limited number of hardware components. Thus, in a lab with equipment from different vendors, it is often necessary to buy and master the use of multiple software packages for controlling multiple microscopes. Many of the software packages available cram a large number of features into the package, making it hard to figure out how to operate the software. Moreover, the software is expensive, perhaps not a problem for well-funded laboratories, but an impediment for smaller labs and teaching environments. Lastly and most importantly, it is not possible for end-users to modify or adapt the software (other than through nonstandard and often ineffective 'macro' solutions). Cutting edge research will always demand novel imaging strategies — which cannot possibly have been forseen during the creation of a software package — thus full adaptability of the software is extremely important for many current and future experiments.

To remedy this situation, our lab has started a project to develop an Open Source microscope image acquisition software package that we have named 'µManager'. µManager is available, free of charge, along with its source code at http://micro-manager.org. The software was designed in a highly modular way, making it easy to add support for new hardware or to otherwise adapt the software (discussed below). µManager runs as a plugin to ImageJ (http://rsb.info.nih.gov/ij/), which is a widely used, freely available, image analysis package written in Java by Wayne Rasband at the NIH. Like ImageJ, µManager works on Windows, Mac OS X, and Linux (however, not all hardware is supported on each platform due to a scarcity of device drivers for platforms other than Windows). µManager already works with many hardware components (microscopes from Zeiss and Nikon, cameras from Hamamatsu, Andor, Roper/Photometrics, QImaging and DVC, and peripherals from Ludl, Prior, ASI and Sutter; for a current list see: http://micro-manager.org/support.php). Through a simple to use interface, microscopists can take individual snap-shots, perform time-lapse imaging, take *z*-series and multi-channel images as well as use any combination of these acquisition strategies.

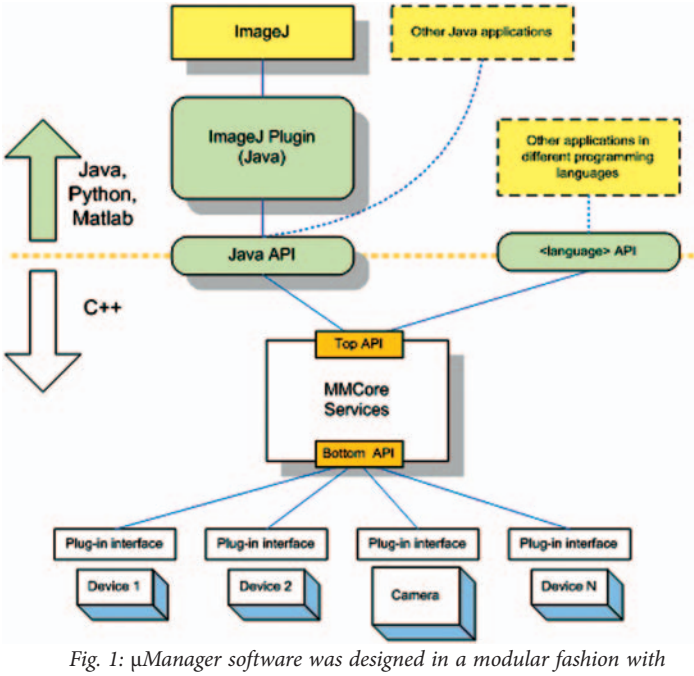

Fig. 1: µManager software was designed in a modular fashion with three distinct layers.

An Open Standard for microscope device control through modular design. The µManager software was designed in a modular fashion with three distinct layers (Fig. 1). At the bottom there is a layer of 'Device adapters' that interfaces to the actual device drivers or talks to the hardware through serial, USB, or other ports. Next, there is the central 'Core' that integrates all basic functionality. The Core exports its capabilities to higher levels through an interface that is computer language independent. Therefore, it is possible to use the µManager functionality from many different environments. For instance, one can interface with µManager from Matlab, a popular development environment (for a project aiming to develop µManager in a complete Matlab toolbox, see http://www.mcb.ucdavis.edu/faculty-labs/scholey/Roy/Roboscope.html). Both the Core and the device adapters were written in C++, and can be compiled on many different platforms. We developed the user interface in Java, which also runs on multiple platforms. Because of the modular structure of the µManager software, it is straight-forward to add support for new hardware devices. Once a new device adapter is written, it can be directly used by the existing µManager software (*i.e.*, no changes to the rest of the software are required). To assist users and instrumentation companies in writing these device adapters, we have written a device developer's kit (http://www.micro-manager.org/downloads.php?object=devkit). A programmer with some experience writing C++ code can use this kit to develop device adapters, independent from the µManager development team, most likely in about a week's time. Several companies have already contributed (DVC, Qimaging) or are working (Scion) on µManager adapters for their hardware. We also have received device adapters from µManager users and hope that more will be contributed in the near future. Through these community efforts, the list of hardware supported by µManager will continue to grow, and the modular design of µManager guarantees that the user can add support for hardware (which is not the case with closed source, commercial software).

Another interesting consequence of the modular architecture of the µManager software is that third party software can also interface with the µManager Core and therewith automatically gains control of all hardware that works with µManager. Due to the liberal licensing terms, even commercial software packages can use this approach. Thus, the µManager device interface can grow into an open-access industry standard, enabling commercial companies and researchers to extend its functionality without restrictions. Writing a µManager device adapter is all that is required to make a device work with multiple software packages under multiple computer Operating Systems. Clearly, adaptation of such a standard (as opposed to the current industry-wide trend of providing Windows-only software development kits for microscope hardware) will reduce development cost, guarantee inter-operability of all hardware and software adapting to the standard, and therefore be beneficial to microscopists and instrumentation companies alike.

Using the µManager software. Upon startup, the user is first asked to specify a configuration file. This configuration file tells the software what hardware is connected to the computer and also lists a number of configuration presets (shortcuts to set multiple parts of the microscope system to a specific state). The configuration file is a human readable text file and can be edited directly, although the built-in configuration wizard

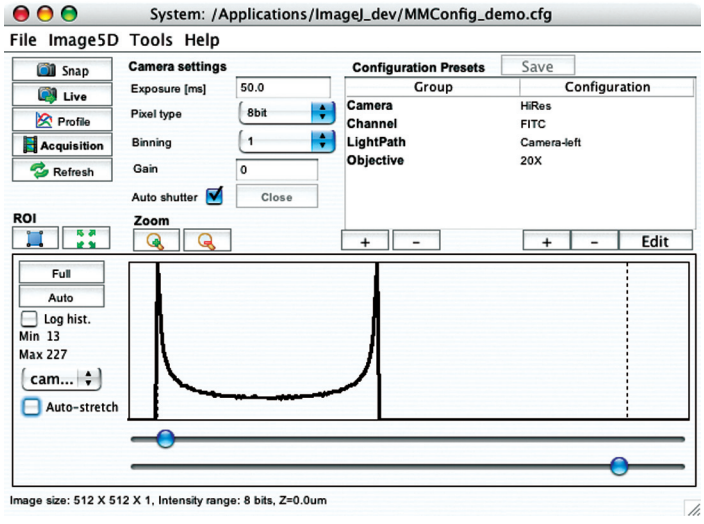

Fig. 2: main µManager window will become active

and preset editors make it easier to alter this file (see below). A 'demo' configuration file, which defines 'virtual' (emulated) cameras and other equipment is included, making it possible to test the software without using real hardware. To instruct µManager about the hardware attached to the computer, the user can run the hardware configuration wizard (found under the 'Tools' menu). This wizard goes through a number of steps to figure out the hardware setup and will eventually save the information in a configuration file.

Once the configuration file is loaded, the main µManager window will become active (Fig. 2). This window lets the user take images ('Snap'), get continuous data from the camera ('Live'), set the exposure time, change binning and gain of the camera, and set (and delete) regions of interest. A histogram of the last acquired image is shown in the bottom part of the application window. The images acquired with the 'Snap' and 'Live' buttons are displayed in ImageJ windows and ImageJ can operate on these images as well. Setting a region of interest (ROI) is as straight forward as drawing a box in an acquired image and pressing the ROI button (the subsequent acquisitions will only use this ROI). µManager is not continuously querying all attached hardware to find out whether changes have taken place, but the 'Refresh' button will instruct µManager to query the attached hardware about its current state.

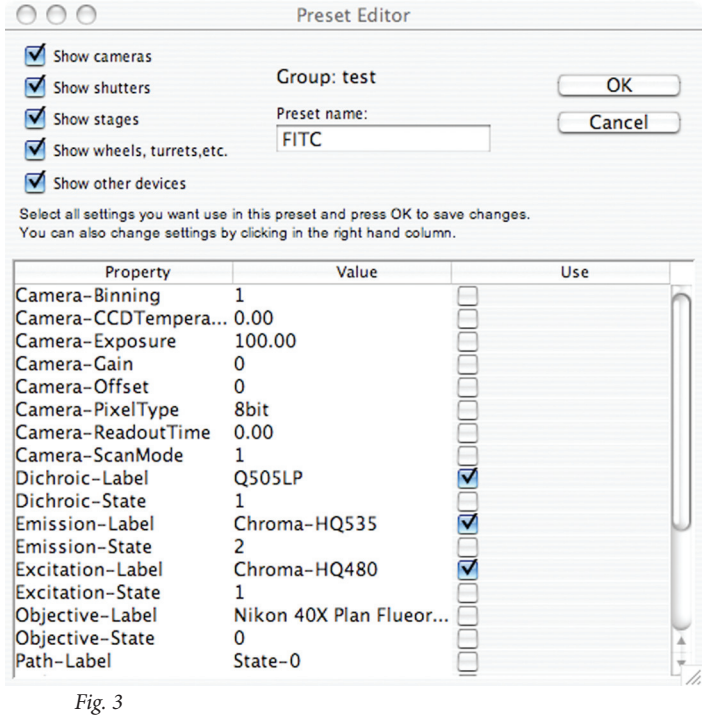

Fig. 3

All components of the system can be controlled individually. To do so, select Tools -> Device/property browser. A window will open showing all components, which can be set to a different state simply by entering a new value and pressing the 'Enter' button. In most systems, there are a large number of properties that can be set. Checkboxes in the top of the window allow the user to restrict the number of properties to only those of use to the investigator. Nevertheless, changing components through the device/property browser is cumbersome; therefore µManager uses configuration presets to make it faster to operate and automate the microscope.

The right top part of the window harbors configuration presets. These are shortcuts allowing one to set multiple components of the system to a desired state. For instance, one could define a configuration preset that moves the excitation filter wheel, the dichroic mirror and the emission filter wheel all simultaneously into pre-defined positions. An obviously useful configuration preset would be called 'FITC' (in the group 'Channel') that moves filters needed for FITC imaging in place. Testing out the demo-configurations (see Fig. 2) will provide an idea of what can be accomplished; however, it is most insightful to create a new configuration. To do so, Press the '+' button towards the center of the window. After naming the new configuration group, the user is presented with a window listing all the properties in the system (Fig. 3). Check the boxes in the 'Use' column for those components that should be set in this preset. Name the preset in top of the window. Set the selected properties to the desired state and press OK to create the preset. Multiple configuration presets can be made in each group. By placing the mouse on top of the name of a configuration preset, a 'tooltip' will appear showing which equipment will be affected (and how) by the preset. Pressing the 'Save' button next to the configuration presets will save the presets to the configuration file.

The Acquisition button in the main windows opens up the 'Multi-dimensional Acquisition' window, which lets one setup time-lapses, *z*-series and multi-channel imaging. Channels are actually the configuration presets defined above, so that creative configuration presets enable interesting acquisition strategies. Acquired images open up in a multi-dimensional image window (Fig. 4, based on the ImageJ Image5D plugin by Joachim Walter).

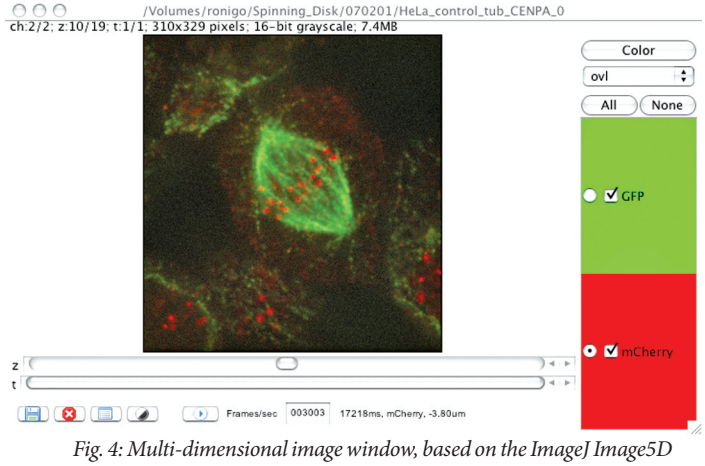

Fig. 4: Multi-dimensional image window, based on the ImageJ Image5D plugin by Joachim Walter.

The future. Our aim in starting the µManager project was to develop a useful tool for microscopists. We hope that µManager will gather even more support as a growing community of users, developers, and instrumentation companies all contribute to this increasingly potent tool for microscopy. Such a community effort appears to be the best way to create an affordable, yet powerful and adaptable software for microscope control. Be a part of this project by downloading the software and giving it a try (it can be installed side by side with existing software). Your feedback (preferably to the µManager mailing list: https://lists.sourceforge.net/lists/listinfo/micro-manager-general) is important for the future of µManager.

µManager development was supported by a grant from the Sandler Foundation. ■