AutoMicromanager: A microscopy scripting toolkit for LABVIEW and other programming environments

Ashcroft, B.A.; Oosterkamp, T.H.

AutoMicromanager: A microscopy scripting toolkit for LABVIEW and other programming environments

Brian Alan Ashcroft, and Tjerk Oosterkamp

Citation: Review of Scientific Instruments **81**, 113708 (2010); doi: 10.1063/1.3499370
View online: http://dx.doi.org/10.1063/1.3499370
View Table of Contents: http://aip.scitation.org/toc/rsi/81/11
Published by the American Institute of Physics

AutoMicromanager: A microscopy scripting toolkit for LABVIEW and other programming environments

Brian Alan Ashcroft and Tjerk Oosterkamp
Department of Physics, Leiden University, 2 Niels Bohrweg, 2333 CS Leiden, The Netherlands

(Received 11 June 2010; accepted 20 September 2010; published online 23 November 2010)

We present a scripting toolkit for the acquisition and analysis of a wide variety of imaging data by integrating the ease of use of various programming environments such as LABVIEW, IGOR PRO, MATLAB, SCILAB, and others. This toolkit is designed to allow the user to quickly program a variety of standard microscopy components for custom microscopy applications allowing much more flexibility than other packages. Included are both programming tools as well as graphical user interface classes allowing a standard, consistent, and easy to maintain scripting environment. This programming toolkit allows easy access to most commonly used cameras, stages, and shutters through the Micromanager project so the scripter can focus on their custom application instead of boilerplate code generation. © *2010 American Institute of Physics.* [doi:10.1063/1.3499370]

I. INTRODUCTION

Setting up a custom microscopy solution often requires a period in which LABVIEW or some other programming environment is used to program the required behavior for the components making up the experiment. This process often requires assembling a lot of codes to set up the microscope, camera, stage, shutters, and other equipment on the table. This can require a lot of troubleshooting or result in a code that can only be used for that one application. There are already many commercial microscopy software solutions in the market, allowing acquisition and analysis of images as well as control over a commercial microscope. These packages are limited in two important ways. First, they are often proprietary, making them both expensive and difficult to change. Second, they are often single purposed, mostly either for a certain brand of cameras, or only as a control for cameras in general. Examples of software packages that are designed to run scientific cameras within a LABVIEW environment include SCIENTIFIC IMAGING TOOLKIT,[1] HCIMAGE,[2] SIDX,[3] or DIGITAL OPTICS.[4] Some other projects are trying to provide generalized microscopy support. These include CAMACQJ,[5] MOTMOT,[6] and SCANIMAGE,[7] but these are still specialized for their particular programming language or purpose. AutoMicromanager attempts to replace these packages with a universal programming interface to all scientific microscopy devices by providing a consistent scripting interface to the Micromanager[8] project.

There are a number of software packages on the market that provide a complete microscopy control setup. These include METAMORPH and METAVUE (Ref. 9) (Molecular Devices, CA, USA), NIS-ELEMENTS (Ref. 10) (Nikon, NY, USA), MICROSUITE (Ref. 11) (Olympus, Tokyo, Japan), LAS (Ref. 12) (Leica, Germany), IMAGE-PRO (Ref. 13) (Media Cybernetics, MD, USA), IMAGING WORKBENCH (Ref. 14) (Indec Imaging, CA, USA), and IP LABORATORY (Scanalytics, MD, USA). All of these packages are powerful suites designed to make microscopy possible. They provide a number of drawbacks for the microscopy user who wishes to develop new solutions, and may not allow the ability to swap various microscopy components seamlessly. In addition, most of these solutions will not work directly within the LABVIEW environment which is popular among researchers for its ease of use and instrument control power.

In order to solve this problem, the Micromanager project has been attempting to build a constant application programming interface (API) that can take the complexities of hardware programming and provides a constant and easy to use interface for any application. The Micromanager project uses JAVA to program its graphical user interface (GUI). JAVA is a powerful programming language, but lacks the ability to interface with many other programming languages, notably LABVIEW and IGOR PRO. In addition, the Java Native Interface can make importing native libraries difficult. We present a solution that leverages the power of the Micromanager platform to most of the common scripting and image analysis scripting languages. AutoMicromanager has been tested with LABVIEW (National Instruments, TX, USA), SCILAB,[15] MATLAB (The Mathworks, MA, USA), MATHEMATICA (Wolfram Research, IL, USA), IGORPRO,[16] (D)COM AUTOMATION, and PYTHON (Ref. 17) (using SciPy[18]).

AutoMicromanager was created with C#, a .Net language from Microsoft. The .Net framework allows a great number of programming languages and interfaces to work together including C#, Visual Basic, C++, Iron Python, and Ruby, as well as many more. C#, a language that relies on the .Net framework, was chosen for its power (the unsafe context), flexibility, and for its similarity with JAVA.

II. OVERVIEW OF SUITE

The software consists of five layers as shown in Fig. 1. For the general user, however, only the device objects and the screen GUI layers will matter. These layers provide control over the devices in a microscopy solution. For a full explanation, the bottom layer consists of a number of hardware adapters, which have been created using C++ either by the Micromanager project itself, hardware vendors, or independent laboratories such as ours. These hardware adapters take care of the difficult task of meshing high perfor-

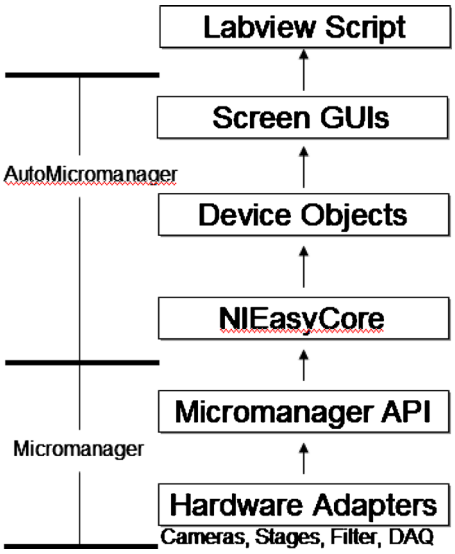

FIG. 1. (Color online) Schematic that shows the modular design of the AutoMicromanager microscopy toolkit. Micromanager interacts with low level C++ hardware adapters and AutoMicromanager allows all the functions to be easily accessed from most available scripting languages at the top level or through the internal GUI controls.

mance C++ routines with the rest of the control software. The hardware adapters are capably maintained by the Micromanager project. The next layer is the Micromanager API forming the middle layer. It controls all the adapters, environment, and so forth. The following layer is the AutoMicromanager layer. This program controls all of the interactions with scripts, the opening and saving of microscopy setups, and a host of other operations. It encapsulates all the devices into easy to use chunks that can be controlled by LABVIEW and other programs. The last layer is the user interface layer. This can either be a set of buttons provided by your custom Labview FrontPage, a custom script for application specific control, or one of the many generic GUIs provided within the AutoMicromanager system which have been designed to run within LABVIEW, (D)COM, or ACTIVEX contexts.

The arrangement of these distinct and independent layers of control of the microscope provides the user with the most control and ease of use. If the user only wishes to provide custom LABVIEW control of one component of the system, then all the other components can be controlled using the default interfaces provided.

III. APPLICATION WITHIN LABVIEW

LABVIEW is the primary target of AutoMicromanager at this time. LABVIEW has excellent facilities to communicate with the .Net framework and this LABVIEW feature has been exploited to provide a powerful set of tools directly into the LABVIEW environment. The first step is to import the .Net controls into the LABVIEW environment. The installation program installs the controls in the Controls Menu accessible from the front page by right clicking on the front page. The NIEasyCore (the interface between AutoMicromanager and LABVIEW) should be placed on the FrontPage to start programming.

AutoMicromanager has many modes of use as has been detailed in the extensive tutorial file available from http://labviewmicroman.sourceforge.net/. The first mode allows all devices to be manually specified in the LABVIEW program. The second and easier method is to load a hardware configuration file into AutoMicromanager as shown in brief here. The hardware configuration file can be created from the standalone program or from the hardware configuration wizard included in the installation package. The scripter should place the NIEasyCore, PictureBoard, and AllDeviceHolder controls on the FrontPage of a blank VI. Now the VI is ready to be wired.

A. Basic setup

The routine "StartEcore" is used to start the Micromanager and AutoMicromanager programs. It takes the location of a configuration file as input, as shown in Fig. 2(a). This file describes all the microscopy hardware, the default settings of the hardware, channel settings, and other microscopy environment settings. This file is not the same as the Micromanager configuration files and can be generated by a VI that is included in the installation package, or by the standalone application. Once all this information is loaded, the information is sent to the AllDeviceHolder [Fig. 2(b)] which will arrange all the device GUI controls. The user can now interact with the microscopy environment.

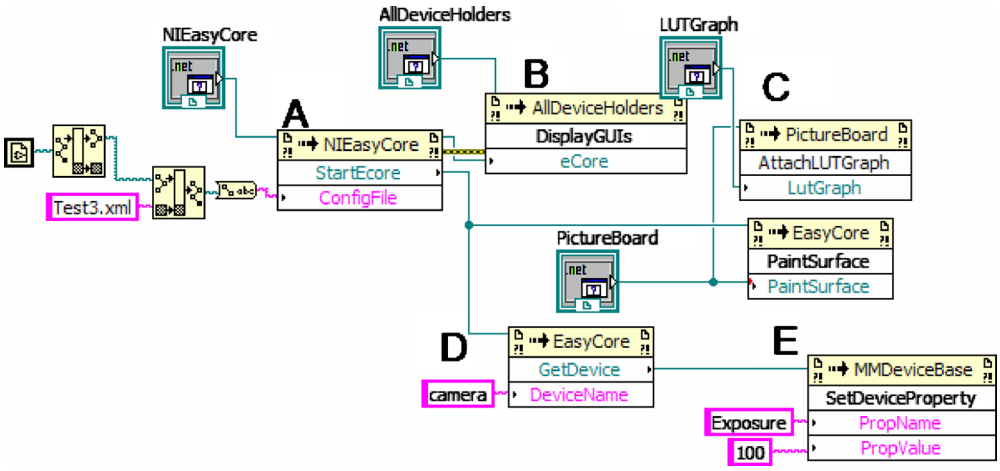

FIG. 2. (Color online) (a) LABVIEW code to start AutoMicromanager, load a hardware configuration file, and start the microscope. (b) Code to load all the device GUI controls for user input. (c) Code to attach a LUT table to the image and then specify the desired image output surface (PaintSurface). (d) Code to request a script interface to the camera device (named in the hardware configuration file). (e) Code to set the camera exposure to 100 ms.

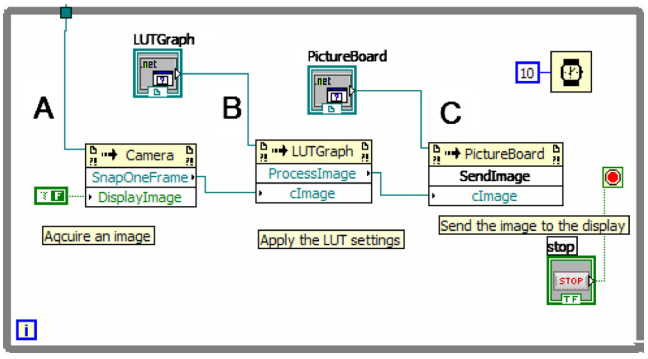

FIG. 3. (Color online) Code to manually acquire image from camera for manipulation or display. (a) Ask camera to acquire one image, (b) attach a LUT graph to the image, and (c) display the image on a PaintSurface.

Finally, the PaintSurface and the look up table (LUTGraph) are registered with NIEasyCore so the Microscopy images can be easily displayed, as shown in Fig. 2(c). This tells the program to send all the images to this surface without the need for LABVIEW to intervene. This setup will allow the LABVIEW VI to control a number of microscopy devices, acquire images, and script custom LABVIEW microscopy solutions.

B. Device control

It may be desirable to control one or more of the microscopy devices from within the LABVIEW program. This is accomplished by asking the NIEasyCore for the desired device that has been registered with the hardware configuration file. As shown in Fig. 2(d), GetDevice can be called after the configuration file is loaded with the name of the desired device. The name of the device is specified in the hardware configuration file. It is then possible to change the microscopy device properties directly from LABVIEW. In Fig. 2(e), it is shown how to change the exposure on a camera. All properties are changed using strings. Any devices that are defined within the AutoMicromanager toolkit can be changed by this method.

C. Image manipulation

Some scripting solutions may wish to examine or manipulate the images that are being acquired. This eventuality can be handled by notifying NIEasyCore to send images to the LABVIEW program. This process is slightly more complicated than the control of the microscopy devices, as was shown in Figs. 3 and 4. In the first case of Fig. 3, the camera device is requested from NIEasyCore and then the camera is requested to acquire an image [Fig. 3(b)] and present the image on the screen [Fig. 3(c)]. The image's pixels can be converted into a LABVIEW array for easy manipulation as described in the tutorials.

In Fig. 4, the LABVIEW program may wish to allow the AutoMicromanager program to perform all the user interactions. The scripter must start AutoMicromanager [Fig. 4(a)] and then notify AutoMicromanager that it wishes to receive notifications (4B). When a notification occurs (4C), the script can get the image (4D), manipulate the image (4D), and then send the image back to autoMicromanager for display (4E).

First, there is a helper control called the NIImageProcessor that handles all the communication of images. This is registered with NIEasyCore so it can begin to get images. Then LABVIEW works in a loop checking whether there is an image ready. Once an image appears, LABVIEW collects the image with GetImage, manipulates the image (in this case saving it to disk), and then returns the image with PutImage and then notifies AutoMicromanager to continue with ImagesAreProcessed. Further image processing options are available, and the CoreImage class contains a number of functions to do basic image processing, background subtraction, and so forth.

IV. OTHER APPLICATIONS

A. PYTHON

The standard Micromanager distribution can already be used with PYTHON, but the ease of integrating .Net projects with PYTHON allows AutoMicromanager to also be used in this programming environment. The Python.Net project allows a user to run .Net code in a native PYTHON environment. This allows access to the Python(x,y) and the other scientific libraries available in the PYTHON world. An example of script is included that shows how to perform a fast Fourier transform on the images acquired from the camera using SciPy in the tutorial section. The conversion from a .Net array to a SciPy array is still time consuming, and a new solution for this performance bottleneck is being completed for future versions.

B. Interior scripting (IGOR PRO/SCILAB examples)

Both IGOR PRO and SCILAB provide interfaces that can be accessed by a client program to perform actions on data sets and to run simple programs. The standalone program provides a method to access these programs from within the AutoMicromanager environment. In this manner, the user can access data and then the image processing and data analysis can be performed from these powerful programs. This method can be used with MATLAB as well as any other programming language that provides automation.

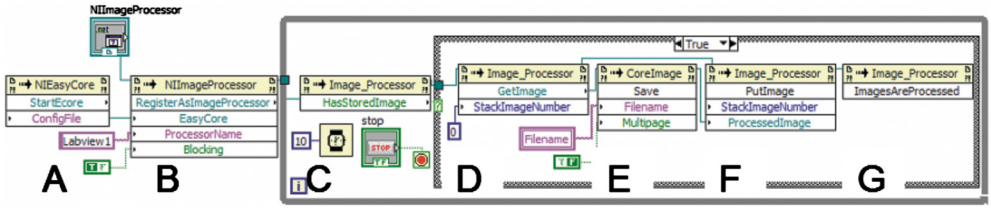

FIG. 4. (Color online) Code to automatically acquire images for manipulation or display. (a) shows loading a hardware config file, (b) requesting notifications for images, (c) checking for available images, (d) getting the image, (e) manipulating image, and (f) returning the image to the AutoMicromanager program. Finally, AutoMicromanager is notified that it can proceed.

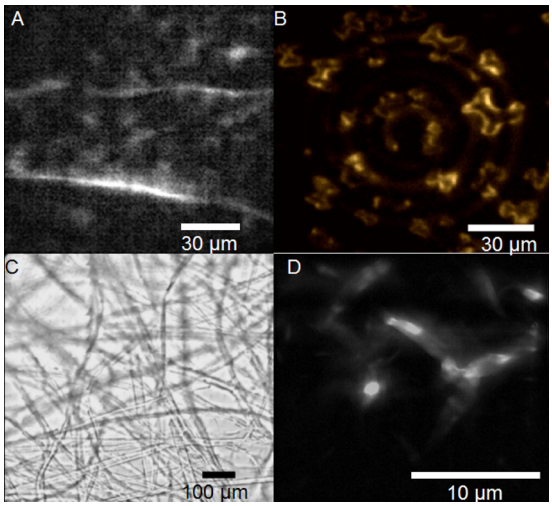

FIG. 5. (Color online) (a) An image of collagen fibers acquired using multifocal, multiphoton microscopy under stochastic scanning. The image was produced by scanning 100 laser foci with galvos driven by a National Instruments DAQ controlled with the internal AutoMicromanager function generator, and images were acquired with a Cascade (Photometrics) camera. (b) Quantum dot crystals illuminated by a two photon microscope, with the laser moving in a circular motion. Image was integrated on the Cascade camera, with the spinning laser beam motivated by a custom device created in the laboratory. The quantum dots were extremely bright and the toolkit acquired this series at 100 frames/s. (c) A brightfield montage of cotton fibers. Image was captured with Cascade camera and an automated PI stage. The image is a composite image from 800 separate images, stitched together. The images were generated by the stack and montage module in the Record and Save tools GUI control. (d) Ridges in quantum dot crystals. Optical slices were acquired by a line scanner of a two photon generating laser spot combined with automated stage motion. The line scanner was created using a galvo mirror, and the stage was stepped perpendicularly with a PI stage. Galvo control of the laser spot is generated by a LABVIEW script, while all the microscopy tasks were handled by AutoMicromanager.

This feat is performed by starting the standalone application with a scripting form added with the hardware setup tool or from the View menu item. A PYTHON scripting engine is included to handle the images and devices. This engine is already set with both a NIEasyCore object as has been defined in the other examples. The sample program is shown in Fig. 5. The standalone program has already loaded a hardware config file and the script shows an example of acquiring an image or device from AutoMicromanager and then sending the information to SCILAB for analysis.

C. ACTIVEX control based solutions

Finally, care has been taken to include a facility to allow any scripting language that can be hosted as an ACTIVEX or (D)COM control to be used within the framework of the standalone application. Many popular and scientifically relevant languages such as MATLAB and IGOR PRO can be used by this method. Sample programs are available from the website that show how to perform a linkage with MATLAB and MATHEMATICA. Once again, MATLAB can already link to the existing Micromanager distribution using JAVA making choosing AutoMicromanager over the Micromanager GUI a matter of taste.

V. CONCLUSION

In the future, AutoMicromanager should be expanded to provide relief from a number of deficiencies. First, the image display is greatly lacking in processing power. Micromanager is linked with ImageJ,[19] providing a great deal of image processing power. SciImage is being developed in parallel with AutoMicromanager in order to address this deficiency, but is still greatly lacking in power. Writing GUI plugins for various components is still a labor intensive project and should be made easier for nonexperts to complete. Last, AutoMicromanager still does not have effective methods to deal with color images coming from the camera. These defects should be improved with the next version of the software package.

We have presented a toolkit that should greatly reduce the time that researchers need to develop a new microscopy solution by automatically handling most of the common boilerplate tasks needed to set up a microscope, while is still giving the user the control over the instruments to perform any advanced microscopy technique. In order to show how efficient the program is at a number of common microscopy tasks, we have shown the results of image acquisition with differing modularity in Fig. 5 with the corresponding figure captions showing the equipment profiles. These figures show that the program can be used for bright field, stage scanning, beam scanning, and many other forms of microscopy with high performance. The program is open source and can be freely downloaded from http://labviewmicroman.sourceforge.net/.

ACKNOWLEDGMENTS

This project has been funded by a starting investigator grant from the European Research Council.

[1] RCubed Software Consulting at ⟨http://www.rcubedsw.com/⟩.
[2] HCIMAGE, Hamamatsu Software at ⟨http://sales.hamamatsu.com/index.php?id=13224248⟩.
[3] Bruxton-SIDX at ⟨http://www.bruxton.com/sidx/⟩.
[4] DIGITAL OPTICS at ⟨http://www.digitaloptics.net/⟩.
[5] CAMACQJ at ⟨http://www.mbl.edu/research/labs/adlc/camacqj/⟩.
[6] A. Straw and M. Dickinson, Source Code Biol. Med. **4**, 5 (2009).
[7] T. Pologruto, B. Sabatini, and K. Svoboda, Biomed. Eng. Online **2**, 13 (2003).
[8] N. Stuurman, N. Amodaj, and R. Vale, Microscopy Today **15**, 42 (2007).
[9] METAMORPH, Life Science Research Imaging Systems, Bioanalytical Software at ⟨http://www.moleculardevices.com/pages/software/metamorph.html⟩.
[10] NIS-ELEMENTS at ⟨http://www.nis-elements.com/⟩.
[11] Olympus MICROSUITE at http://www.leedsmicro.com/oly_microsuite_basic.asp.
[12] MICROSCOPE software, Leica Microsystems at ⟨http://www.leica-microsystems.com/products/microscope-software/⟩.
[13] Media Cybernetics-Image Processing Software | Image Deconvolution | Image Analysis at ⟨http://www.mediacy.com/⟩.
[14] Imaging Workbench Home at ⟨http://www.imagingworkbench.com/⟩.
[15] Home-Scilab WebSite at ⟨http://www.scilab.org/⟩.
[16] WaveMetrics-scientific graphing, data analysis, curve fitting, and image processing software at ⟨http://www.wavemetrics.com/⟩.
[17] PYTHON Programming Language—Official Website at ⟨http://www.python.org/⟩.
[18] E. Jones, T. Oliphant, P. Peterson *et al.*, SciPy: Open Source Scientific Tools for Python, 2001, ⟨http://www.scipy.org⟩.
[19] W. S. Rasband, ImageJ, U.S. National Institutes of Health, Bethesda, Maryland, USA, http://rsb.info.nih.gov/ij/, 1997–2009.