Bioimage informatics

**ImageJ-MATLAB: a bidirectional framework for scientific image analysis interoperability**

Mark C. Hiner¹, Curtis T. Rueden¹ and Kevin W. Eliceiri¹,2,*

¹Laboratory for Optical and Computational Instrumentation, Madison, WI 53706, USA and ²Morgridge Institute for Research, Madison, WI 53715, USA

*To whom correspondence should be addressed.

Associate Editor: Robert Murphy

Received on June 17, 2016; revised on September 30, 2016; editorial decision on October 21 2016; accepted on October 25, 2016

**Abstract**

Summary: ImageJ-MATLAB is a lightweight Java library facilitating bi-directional interoperability between MATLAB and ImageJ. By defining a standard for translation between matrix and image data structures, researchers are empowered to select the best tool for their image-analysis tasks.


**Contact:** eliceiri@wisc.edu

**Supplementary information:** Supplementary data are available at Bioinformatics online.

1 Introduction

A proliferation in scientific data analysis tools (The Popularity of Data Analysis Software, 2012) leads to fragmentation: no one option can offer a complete solution. In this article we present ImageJ-MATLAB: providing new options for interoperability between ImageJ (Schneider et al., 2012), a widely used image analysis application, and MATLAB, a ubiquitous tool particularly popular with the signal processing community. The most successful prior attempt in this vein is MIJ/Miji (Sage et al., 2012), a utility collection for launching ImageJ 1.x from within MATLAB and importing images to the workspace. Empowered by recent ImageJ2 efforts (Schindelin et al., 2015), including the availability of the SciJava plugin framework, ImageJ-MATLAB provides an extensible and bidirectional bridge for mutual exchange of data between these two environments. In providing researchers the ability to call MATLAB scripts from ImageJ, and adding a new level of extensibility to the functionality established in MIJ, we hope to strengthen the bond between the ImageJ and MATLAB communities, enabling broader impact for scientific achievement while avoiding unnecessary duplications of effort.

2 Implementation

The fundamental data models of ImageJ and MATLAB are, respectively, Datasets—built on ImgLib2 (Pietzsch et al., 2012)—and matrices. To unify these models, ImageJ-MATLAB is implemented as a SciJava plugin suite—on which ImageJ itself is built. Practically, ImageJ-MATLAB consists of the imagej-matlab and scripting-matlab libraries.

The foundation of ImageJ-MATLAB consists of two Converter plugins: enabling ImageJ Datasets and MATLAB matrices to be freely exchanged within the SciJava framework. Both directions of conversion use the third-party MATLABControl library (https://code.google.com/archive/p/matlabcontrol/) to manage interaction with a MATLAB instance.

The next layers focus on how to expose the aforementioned Converters to the user. On the ImageJ side, a ScriptLanguage plugin allows execution of ‘.m’ scripts directly in the ImageJ script editor with a convenient ‘@matrix’ shorthand to convert active images to matrices. For the MATLAB perspective, we created a MATLABCommand plugin type and paired ‘ImageJ.m’ startup script to provide ImageJ utility functions from the MATLAB command prompt.

3 Results

ImageJ-MATLAB is the next logical evolution of the original MIJ/Miji compatibility layer. ImageJ users can now leverage the power of MATLAB without opening a new application. From within MATLAB, the fixed, built-in commands of MIJ are now defined...
through extensible MATLAB Commands, allowing developers to easily contribute new functionality. In both use cases, a foundation in the SciJava Converter framework will allow ImageJ-MATLAB to grow over time and impact applications beyond ImageJ. For example, KNIME Image Processing (KNIP)—the flagship image analysis toolbox built on the KNIME Analytics platform (Berthod et al., 2008)—integrates with the ImageJ Common data model, and will soon automatically expose parameterized MATLAB scripts as KNIME nodes.

The value of ImageJ-MATLAB is in the expanded toolkit that comes with unifying two powerful applications. Figure 1 illustrates one way to take advantage of these new options: starting from a MATLAB script developed to register a low-resolution H&E image with a high-resolution second-harmonic generation (SHG) image, we convert script inputs and outputs to @Parameters. This effectively turned our MATLAB script into a SciJava module, executable in any SciJava application. When executed in ImageJ the @Parameters are specified by a graphical input panel, and the @OUTPUT image(s) are automatically displayed after the script completes. This example could be further expanded by adding ImageJ pre- and post-processing steps—thresholding, Trainable Weka Segmentation, etc.—and distributing the resulting scripts on an ImageJ update site. This convenience for users to download a script with the click of a button and execute the same combined workflow on their own data is simply not possible in MATLAB alone.

4 Discussion and conclusions

ImageJ-MATLAB is an open-source library facilitating interoperability between the ImageJ and MATLAB applications. Through its use we hope to mitigate wasted effort in both development communities, allowing implementations in either application to become mutually beneficial—ultimately providing users with seamless, interoperable solutions for their analysis needs. This effort establishes MATLAB in the greater SciJava ecosystem, providing extensible plugin mechanisms, and the ImageJ update site infrastructure to encourage future contributions and simplified, standardized distribution. Finally, on a broader level, the techniques applied in building ImageJ-MATLAB can serve as a general template for bringing other scientific applications, such as ITK and CellProfiler, into this growing network of interoperable toolkits.

Acknowledgements

This work would not have been possible without development of the robust SciJava plugin framework, driven by its application in ImageJ. In particular, the authors gratefully thank and acknowledge the efforts of (in alphabetical order): Ellen Arena, Anne Carpenter, Christian Dietz, Wayne Rasband, Jason Swedlow and Pavel Tomancak. Special thanks to Cole Drifka and Adib Keikhosravi for assistance with figures.

Funding

National Institutes of Health RC2-GM092519-01, ‘Grand Opportunities Grant’, ‘Hardening of Biomedical Informatics/Computing Software for

Fig. 1. Registering pancreatic tissue images with differing modalities (H&E, SHG). (A) parameterized script header, (B) original H&E image naively scaled with SHG image overlaid and (C) registered H&E image with same SHG overlay. (D) and (E) are high-contrast renderings of (B) and (C), respectively, to emphasize changes in H&E and SHG alignment.

Robustness and Dissemination’, and additional internal funding from the Laboratory for Optical and Computational Instrumentation (LOCI) and Morgridge Institute for Research.

Conflict of Interest: none declared.

References