MIJ: Making Interoperability Between ImageJ and Matlab Possible

Daniel Sage[1], Dimiter Prodanov[2], Jean-Yves Tinevez[3], Johannes Schindelin[4]

[1]Biomedical Imaging Group, EPFL, Lausanne, Switzerland; [2]Bio-Nano Electronics Department, Imec, Leuven, Belgium;
[3]Institut Pasteur, Imagopole / PFID, Paris, France; [4]LOCI, University of Wisconsin-Madison, USA.

Abstract

We present a software module MIJ[1] that allows to combine the powerful numerical computation of Matlab[2] and the image-analysis capabilities of ImageJ[3]. Since both ImageJ and Matlab run on multiple operating systems, MIJ is platform-independent. MIJ uses the Java Virtual Machine (JVM) of Matlab to call ImageJ methods from the Matlab console or scripts. In such way, we could bring the richness of ImageJ and of their plugins to the Matlab world.

MIJ contains a collection of static methods to exchange data bidirectionally between Matlab and ImageJ. MIJ has methods to convert and transpose images of ImageJ to 2D matrices and stacks of images to 3D arrays into Matlab. MIJ tries to avoid loss of accuracy by converting the image data to the closest numerical type of Matlab. The only drawback of this interoperability model is the current limitation of the heap memory in Matlab. Depending of the architecture and of the version of Matlab, the heap space can be increased, but it cannot be changed by MIJ.

Since 2011, MIJ is included in the Fiji[4] distribution of ImageJ together with a friendly startup Matlab script called Miji[5]. After calling a single function from Matlab, the full functionality of all the plugins and libraries included in Fiji can be accessed by Matlab scripts, via MIJ. Most notably, Fiji's 3D Viewer can be used to display three- or four-dimensional data in volume-rendering, iso-surface and orthoslice mode[6].

Users have found MIJ very useful even for simple daily tasks, e.g. opening proprietary image formats using the Bio-Formats[7] library or using the image viewer of ImageJ to display a 2D or 3D Matlab's matrices. Imaging professionals prefer usually ImageJ to show, control the rendering and interact with images. On the other hand, displaying plots and graphs is more convenient and flexible in Matlab. Exchanging data between the two platforms can be very valuable for iterative processes which imply image analysis, done in ImageJ, and statistical analysis done in Matlab. MIJ allows also to launch ImageJ macros from Matlab scripts and to run powerful plugins, e.g. TrackMate[8].

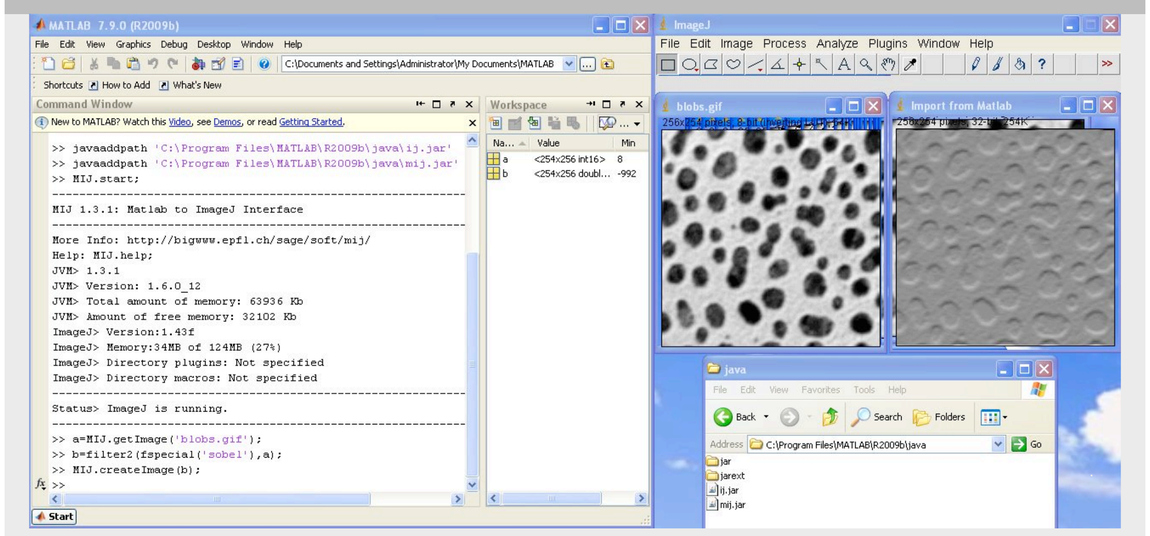
Features

MIJ is an Open-source Java class facilitating interoperability between Matlab and ImageJ/Fiji.

- Run ImageJ in the JVM of Matlab
- Export/Import a 2D Matlab variable as ImageJ's image: the image can be displayed or not
- Export/Import a 3D Matlab variable as ImageJ's stack: the image stack can be displayed or not
- ImagePlus references can be converted to Matlab matrices for further processings using the Java interpreter of Matlab
- Enables to run plugins (setup path)
- Enables to run macros (setup path)
- Additional information from ImageJ can be transferred to Matlab:
  - histogram,
  - table of results,
  - column of results
  - region-of-interests

Application Cases

Using fast Matlab routines and display results on ImageJ



Reading image files using Bio-Formats on Matlab

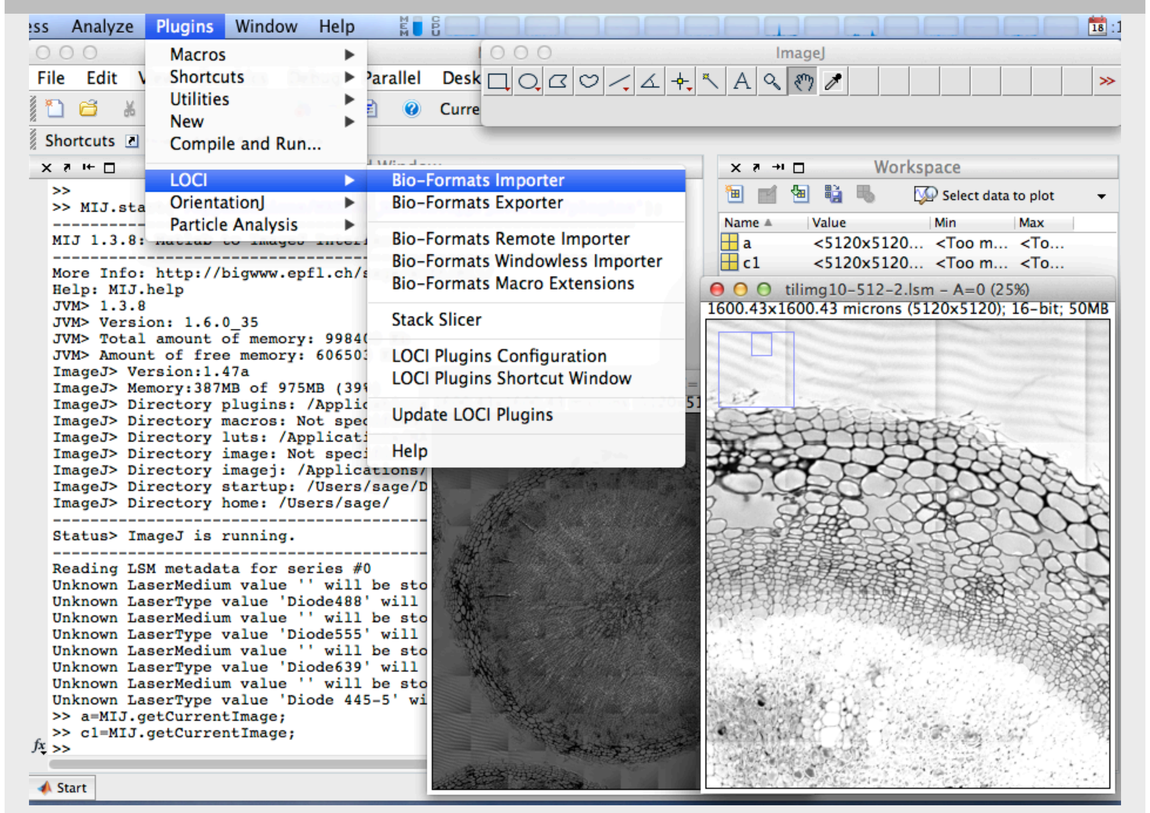
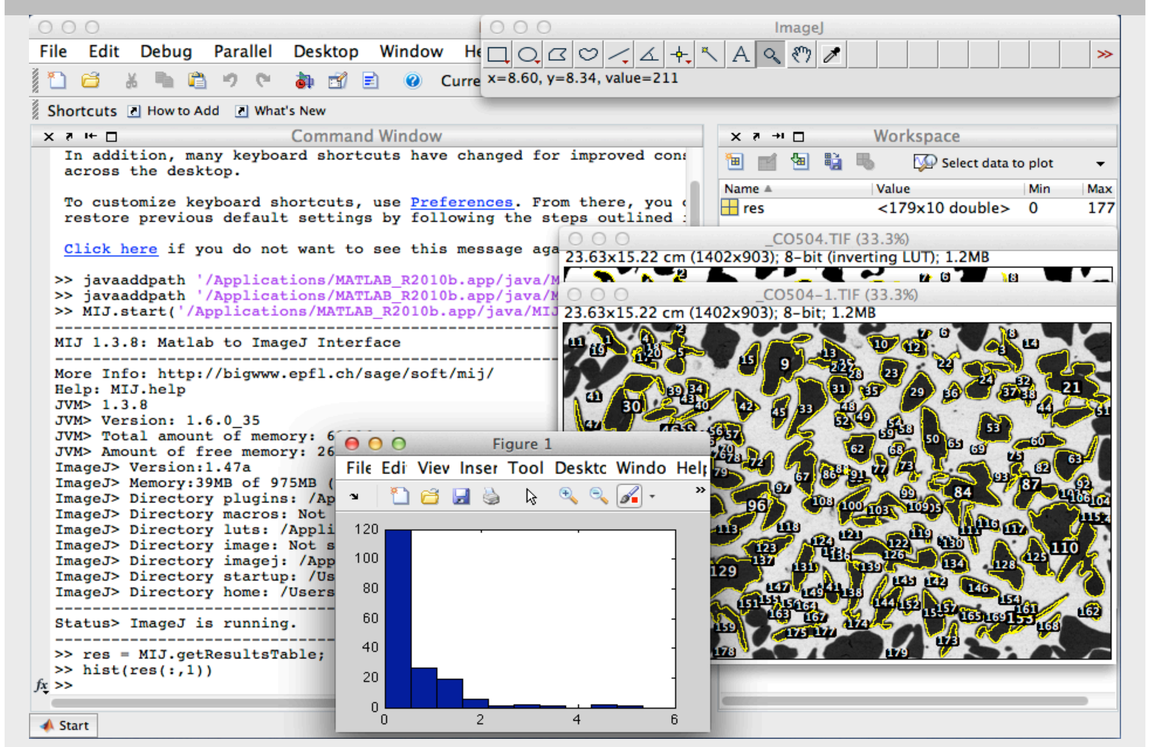


Image-analysis on ImageJ, statistics on Matlab



Mapping

Data structure

Matlab	ImageJ	Conversion
2D array	image	XY matrix transposition
3D array	stack of images	XY matrix transposition, keep Z axis
2D array	table of results	Exact (n columns)
1D array	column of results	Exact
1D array	histogram of image	Exact
2D array	polygon of the ROI	Exact (2 lines, x and y)

Type

Matlab	ImageJ	Conversion
uint8	Grayscale 8-bit	Exact
uint16	Grayscale 16-bit	Exact
uint32	Grayscale 16-bit	Loses precision
uint64	N/A	Not applicable
int8	Grayscale 8-bit	Negative value: lost Positive value: exact
int16	Grayscale 16-bit	Negative value: lost Positive value: exact
int32	Grayscale 16-bit	Negative value: lost Positive value: loses precision
int64	N/A	Not applicable
single	Grayscale 32-bit	Exact
double	Grayscale 32-bit	Loses precision

Technical Issues

Time performance

The data has to be converted, copied and often transposed which requires a overhead time of computation. On a middle-range machine it takes around 100 ms to transfer 10 Mb.

Memory limitation

Matlab provides a Java Virtual Machine (JVM) which interprets the command of the console. In 2012, the version of the JVM is 1.6. The main drawback is that Matlab has limited the Java heap memory (e.g. 256 Mb on Mac OSX, Matlab R2010_b). This limitation restricts the usage of Java for huge datasets like multidimensional imaging purposes. For some operating systems, Matlab proposes to change the limitation by editing a "hidden" file java.opts:
-Xmx1000m

How to use

	ImageJ	Fiji
Download	mij.jar	included
Installation	Copy files into Matlab (Java dir)	automatic
Setup	Add mij.jar and ij.ar in the Java class path	Set the path
Run	`Mij.start('...')`	`Miji;`

Interoperability

Matlab
Statistics analysis
Optimization method
Fast matrix (image) operation
Large base of educated engineers
Broad scientific community
Interpreted language, prototype

ImageJ
Image analysis
User interface. Image interaction
Advanded research plugins
Large base of user (biologist, ...)
Biologist and developper community
Open-source, reproducible

http://bigwww.epfl.ch/sage/soft/mij/

Reference

[1] MIJ: http://bigwww.epfl.ch/sage/soft/mij/
[2] Matlab: http://www.mathworks.ch/
[3] ImageJ: http://rsbweb.nih.gov/ij/
[4] Fiji: http://fiji.sc/
[5] Miji: http://fiji.sc/Miji/
[6] 3D viewer for Matlab: http://www.mathworks.com/matlabcentral/fileexchange/32344-hardware-accelerated-3dviewer-for-matlab
[7] Bio-Formats: http://loci.wisc.edu/software/bio-formats.
[8] TrackMate: http://fiji.sc/TrackMate