doi:10.1017/S1431927619001442

ImageJ for the Next Generation of Scientific Image Data

Curtis T. Rueden[1] and Kevin W. Eliceiri[1,2*]

[1.] Laboratory for Optical and Computational Instrumentation, University of Wisconsin at Madison, Madison, Wisconsin, USA.
[2.] Morgridge Institute for Research, Madison, Wisconsin, USA.
* Corresponding author: eliceiri@wisc.edu

Quantifying images is now a critical, widespread need in biological research as light microscopy experiments continue to grow in scale, size, dimensionality, scope, modality, and complexity.
It is becoming clear that given the complexity, and heterogeneity of the modern image dataset, there cannot be a single software solution. Different imaging processing and visualization approaches need access not only to the data but also to each other. There needs to be compatibility not only in file import and export but interoperability in preserving and communicating what was done to the image. There is a great opportunity in achieving this interoperability, tools that can talk to each other not only enable new biological discovery but also efficiencies in sharing code and in many cases more precise workflows. We present our efforts towards interoperability and extensibility in the ImageJ consortium. We are actively developing key software libraries like ImgLib and ImageJ Ops that are utilized to analyze and visualize biological image data, to the developmental benefit of not only of the applications but the libraries themselves. We also overview the two major development efforts of the ImageJ [1,2] family of image analysis, FIJI [3] and ImageJ2 [4].

The ImageJ distribution Fiji ("Fiji Is Just ImageJ") was conceived to address the need of an easy-to-install image-processing package for biologists, based on ImageJ. Always inspired by researchers' needs, Fiji bundles many ImageJ plugins and offers tutorials and documentation in the ever-growing Fiji Wiki. To facilitate the interaction between biologists and programming experts, Fiji has an update function that allows for a rapid turn-around time between development and usage of Fiji components.
Fiji not only caters for regular users but also for advanced ones, offering powerful scripting languages in addition to the ImageJ macro language. Expert programmers benefit from Fiji's Script Editor which allows to edit and run not only scripts but also Java plugins without the need to leave ImageJ. To avoid duplication of efforts and to provide better separation of concerns, Fiji collaborates closely with other projects such as Bio-Formats, ImageJ2 and ImgLib.

Any successful software project, after a period of sustained growth and the addition of functionality outside the scope of the program's original intent, will benefit from a subsequent period of scrutiny and refactoring, and ImageJ is no exception. ImageJ2 is a new version of ImageJ seeking to strengthen both the software and its community. Internally, it is a total redesign of ImageJ, but it is backwards compatible with ImageJ 1.x via a "legacy layer" and features a user interface closely modeled after the original. Under the hood, ImageJ2 completely isolates the image processing logic from the graphical user interface (UI), allowing ImageJ2 plugins to be used in many contexts, including headless in the cloud or on a server such as OMERO, or from within another application such as KNIME, ICY or CellProfiler (a Python application). ImageJ2 has an N-dimensional data model driven by the powerful ImgLib2 library, which supports image data expressed in an extensible set of numeric and non-numeric types, and accessed from an extensible set of data sources. ImageJ2 is driven by a state-of-the-art, collaborative development process, including version control, unit testing, automated builds via a continuous integration system, a

https://doi.org/10.1017/S1431927619001442 Published online by Cambridge University Press

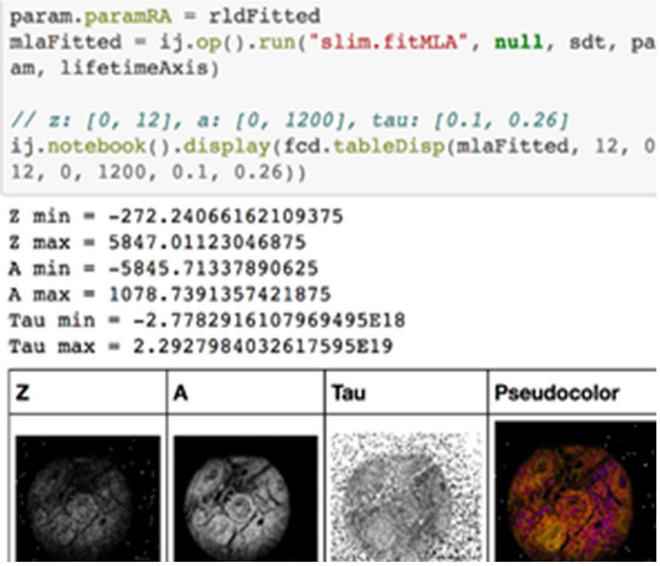
bug tracker and more. We are collaborating closely with related projects including Fiji, CellProfiler, KNIME, Bio-Formats and OMERO, and are striving to deliver a coherent software stack reusable throughout the life sciences community and beyond. The result is well-designed, community-driven software accessible to users yet powerful enough for programmers. ImageJ2 is available separately but the preferred distribution for biologists is via FIJI where it is pre-bundled.

Meeting current and future needs of scientific image analysis requires a flexible and extensible data model, supporting arbitrary dimensions, data types and image sizes. To this end, we chose to use the ImgLib2 library, which provides an extensible, interface-driven design that supports numeric and non-numeric data types. It also provides great flexibility regarding the source and structure of data. Out of the box, ImgLib2 provides several data sources and organizations, including use of N-dimensional array "cells." However, the library remains general enough that alternative structures are also feasible.

Another very recent ImageJ2/FIJI development is on the image processing side with the development of the Ops framework (Figure 1). ImageJ Ops is a framework for reusable image processing operations that extends Java's mantra of "write once, run anywhere" to image processing algorithms. We describe how Ops helps with minimizing complexity, how it helps maintain good performance by allowing for extensible optimization, and allows for effective sharing and reuse of imaging analysis functions [5].

References:

[1] CA Schneider, WS Rasband and KW Eliceiri. Nature Methods **9** (2012), p. 671.
[2] ET Arena et al., Wiley Interdiscip Rev Dev Biol. **6** (2017), p 260.
[3] J Schindelin et al., Nature Methods **9** (2012), p 676.
[4] J Schindelin et al., BMC Bioinformatics. **18** (2017), p 529.
[5] The authors acknowledge funding from NIH grants R03 EB008516 and RC2 GM092519 and from the Morgridge Institute for Research.



```
param.paramRA = rldFitted
mlaFitted = ij.op().run("slim.fitMLA", null, sdt, pa
am, lifetimeAxis)

// z: [0, 12], a: [0, 1200], tau: [0.1, 0.26]
ij.notebook().display(fcd.tableDisp(mlaFitted, 12, 0
12, 0, 1200, 0.1, 0.26))
```

```
Z min = -272.24066162109375
Z max = 5847.01123046875
A min = -5845.71337890625
A max = 1078.7391357421875
Tau min = -2.7782916107969495E18
Tau max = 2.2927984032617595E19
```

Figure 1. ImageJ based Ops workflow showing input parameters and resulting output of a Fluorescence Lifetime Imaging (FLIM) fit.

https://doi.org/10.1017/S1431927619001442 Published online by Cambridge University Press