Received: 19 October 2020 | Revised: 4 November 2020 | Accepted: 6 November 2020
DOI: 10.1002/pro.3993

TOOLS FOR PROTEIN SCIENCE

The ImageJ ecosystem: Open-source software for image visualization, processing, and analysis

Alexandra B. Schroeder[1,2,3] | Ellen T. A. Dobson[1] | Curtis T. Rueden[1] | Pavel Tomancak[4,5] | Florian Jug[4,6,7] | Kevin W. Eliceiri[1,2,3,8]

[1]Laboratory for Optical and Computational Instrumentation, Center for Quantitative Cell Imaging, University of Wisconsin at Madison, Madison, Wisconsin

[2]Morgridge Institute for Research, Madison, Wisconsin

[3]Department of Medical Physics, University of Wisconsin at Madison, Madison, Wisconsin

[4]Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany

[5]IT4Innovations, VŠB – Technical University of Ostrava, Ostrava, Czech Republic

[6]Center for Systems Biology Dresden, Dresden, Germany

[7]Fondazione Human Technopole, Milan, Italy

[8]Department of Biomedical Engineering, University of Wisconsin at Madison, Madison, Wisconsin

Correspondence
Kevin W. Eliceiri, Laboratory for Optical and Computational Instrumentation, Center for Quantitative Cell Imaging, University of Wisconsin, 1,675 Observatory Drive, Madison,WI 53719.
Email: eliceiri@wisc.edu

Funding information
Chan Zuckerberg Initiative; Deutsche Forschungsgemeinschaft, Grant/Award Numbers: JU 3110/1-1, TO563/8-1; European Regional Development Fund, Grant/Award Number: CZ.02.1.01/0.0/0.0/16_013/0001791; German Federal Ministry of Research and Education, Grant/Award Numbers: 01IS18026C, 031L0102; Morgridge Institute for Research; National Institute of General Medical Sciences, Grant/Award Number: P41-GM135019; Retina Research Foundation Walter H. Helmerich Professorship

Abstract
For decades, biologists have relied on software to visualize and interpret imaging data. As techniques for acquiring images increase in complexity, resulting in larger multidimensional datasets, imaging software must adapt. ImageJ is an open-source image analysis software platform that has aided researchers with a variety of image analysis applications, driven mainly by engaged and collaborative user and developer communities. The close collaboration between programmers and users has resulted in adaptations to accommodate new challenges in image analysis that address the needs of ImageJ's diverse user base. ImageJ consists of many components, some relevant primarily for developers and a vast collection of user-centric plugins. It is available in many forms, including the widely used Fiji distribution. We refer to this entire ImageJ codebase and community as the ImageJ ecosystem. Here we review the core features of this ecosystem and highlight how ImageJ has responded to imaging technology advancements with new plugins and tools in recent years. These plugins and tools have been developed to address user needs in several areas such as visualization, segmentation, and tracking of biological entities in large, complex datasets. Moreover, new capabilities for deep learning are being added to ImageJ, reflecting a shift in the bioimage analysis community towards exploiting artificial intelligence. These new tools have been facilitated by profound architectural changes to the ImageJ core brought about by the ImageJ2 project. Therefore, we also discuss the contributions of ImageJ2 to enhancing multidimensional image processing and interoperability in the ImageJ ecosystem.

KEYWORDS

Fiji, image analysis, ImageJ, imaging, microscopy, open source software

1 | INTRODUCTION

ImageJ is a widely-used open-source software that allows users to visualize, inspect, quantify, and validate scientific image data.[1] Imaging-based methods serve a crucial role in the life sciences and have undergone tremendous growth in the past decades.[2] As novel imaging modalities emerge and datasets become more complex, having reproducible and reliable methods to interpret biological images is the core of image analysis, where "seeing is a matter of learning".[2] Image analysis allows users to extract information from images in a reproducible manner. For this to occur, the algorithms and parameters used must remain open and consistent. If any component of the algorithm is proprietary, researchers may be unable to reproduce their work or track changes with full transparency, which is imperative to the scientific process.[2] To address this need, open source software is advantageous, and this is one of the main reasons for ImageJ's widespread use.

Novel imaging modalities offer enhanced resolution, specificity, and coverage and have contributed to many of the tremendous biological advancements over the past several decades.[3–7] Modern research therefore requires methods for the efficient and robust manipulation, interpretation, and visualization of such advanced, multidimensional imaging data.[8] This is important for a wide array of biological studies that include quantifying the proximity of fluorescent-labeled proteins,[9] tracking cell fates over time,[10] automating cell counting,[11] tracking invading cancer cells,[12] collecting whole-slide information,[13] quantifying and characterizing cells, such as microglia, within the brain,[14] or registering multiview light sheet fluorescence microscopy datasets to study development.[15,16] Image analysis also serves a crucial biomedical role for diagnostic interpretation.[17–20] As the prevalence of large multidimensional datasets continues to grow, the ability to manually take measurements not only becomes impractically time-consuming, but the sensitivity, accuracy, objectivity, and reproducibility of doing so can become greatly inhibited.[21,22] Because of these challenges, we will highlight in this review current developments within the ImageJ ecosystem that address the handling of large, multidimensional datasets, including annotating and performing advanced image analysis techniques.

Historically, the predecessor to ImageJ, NIH Image, was developed by Wayne Rasband in 1987 at the National Institutes of Health. Rasband was enthusiastic about the concept of open code sharing on the electronic bulletin boards of the time and the possibilities offered by the release of the Apple Macintosh II with advanced graphics and support for the Pascal programming language. His goal was to allow third party developers to customize and create their own domain-specific analysis routines[23] and let users drive the applications of NIH Image. In 1995, when Sun Microsystems released the Java programming language that could run on any operating system, the transition of NIH Image to Java was initiated, and the first release of the succeeding platform, known as ImageJ, occurred.[23]

Figure 1 shows the evolution of the ImageJ ecosystem. ImageJ has a simple and single toolbar that opens when the program is running, and this toolbar has remained much the same since its early years. Next to this remarkable stability, ImageJ is also uniquely flexible. Its core feature from the very beginning has been, and still is, that it can be extended by users to fulfill the specific needs of their analyses. Thus, ImageJ—as designed in its first incarnation as NIH Image, as well as the first generation of the cross-platform ImageJ,[23] henceforth referred to as ImageJ1 to disambiguate it from ImageJ2—can adapt to changing times and technologies, and it continues to do so to this day.

Fiji (Fiji is Just ImageJ), a "batteries-included" distribution of ImageJ (https://imagej.net/Welcome) bundling many plugins which facilitate scientific image analysis, was introduced in 2007 to ease installation and development of more complex plugins but also to be ready-to-use as-is by biologists.[24] Fiji developers introduced several key components such as an automatic update mechanism[24] (https://imagej.net/Updater), a script editor (https://imagej.net/Script_Editor), and a more powerful image data model[26] (https://imagej.net/ImgLib2). Concurrently, the ImageJ2 project sought to improve ImageJ's core architecture, introducing an improved plugin mechanism (https://imagej.net/SciJava_Common) enabling greater extensibility.[27] The two projects joined forces, and now Fiji builds upon the ImageJ2 framework to which it contributed several important libraries. The marriage of Fiji and ImageJ2 did not mean divorce from ImageJ1; on the contrary, ImageJ2 provides backwards compatibility with ImageJ1 via the ImageJ Legacy bridge (https://imagej.net/ImageJ_Legacy), allowing existing ImageJ1 plugins to continue working as-is in the new system. ImageJ is compatible with macOS, Linux, and

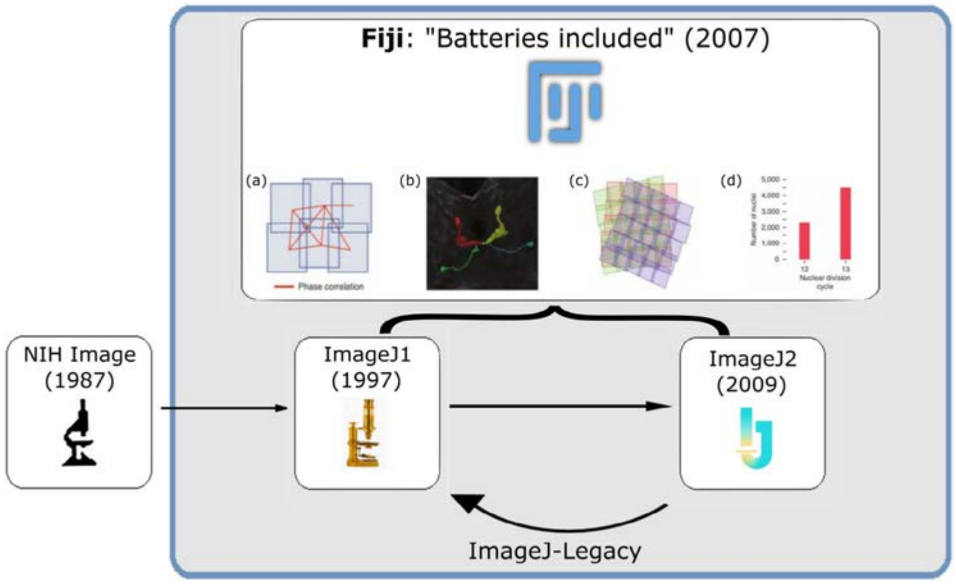


FIGURE 1 A diagram of the evolution of NIH Image to the current ImageJ ecosystem including release dates from 1987 until present. Fiji[24] leverages the updated ImageJ2 framework, which contains elements from ImageJ1 via the optional ImageJ Legacy bridge that facilitates full backwards compatibility. Fiji comes prepackaged with many plugins and tools including (a,c) registration, (b) segmentation, and (d) measurement tools. Thumbnail images are courtesy of the Fiji community[24,25]

Windows and comes in several "flavors"—or packages. The current ImageJ ecosystem includes the original ImageJ1, the updated ImageJ2, the Fiji distribution of ImageJ, and community plugins, shipped individually via Update Sites,[28] which extend the functionality of ImageJ. More than 200 listed Update Sites (https://imagej.net/Update_Sites) in ImageJ's central registry provide a convenient vehicle for users to obtain the latest releases of novel ImageJ tools and specialized plugins. The Fiji distribution includes a broad collection of preloaded plugins available for scientific image analysis and is recommended for new users.[29] Many advanced plugins (for a small selection, see Table I) rely on Fiji's modern distribution that includes accessible compilations, automatic updates, and third-party libraries.

The ImageJ ecosystem lends utility to a wide range of users with varying programming expertise: from the bench biologist with little-to-no programming experience taking advantage of the prepackaged plugins, to power users who customize their image analysis pipelines with scripts that automate entire workflows within ImageJ. One of the most unique and attractive elements of ImageJ is how easily customizable and adaptable it is even for novice users. Users with no formal programming experience can record actions using the Macro Recorder[23] by performing them manually in ImageJ. From there, with the Script Editor, users can expand their code snippets into scripts that can be easily shared with others. On the software engineering side, the modular structure of ImageJ2 enables programmers to extend ImageJ in a wide variety of ways, including the ability to use components of ImageJ as building blocks for their own software tools and to build bridges with other platforms.[23,24]

While ImageJ has been utilized predominantly by biologists working in a research setting,[47–51] it has also served users from diverse nonbiological research fields, including geologists[52] and astronomers.[53] As of July 2020, a simple PubMed search of the term "ImageJ" yields 2,626 results that represent publications within biology,[11] but also art,[54,55] dentistry,[20,56] medicine,[18,21] and geology.[52] Even historians have used ImageJ, for example, to reconstruct inner woodworm holes to study the interior state of precious medieval sculptures.[55] Thus, the ImageJ ecosystem offers highly valuable, inclusive, and versatile image analysis capabilities across many disciplines.

Rather than being controlled by a single group of developers, development is community-driven; users and developers interact to best optimize their code for a particular problem. The ImageJ wiki (https://imagej.net) serves as the main information repository for both users and developers. Additionally, the recently introduced Scientific Community Image Forum (https://forum.image.sc) serves as a discussion forum and a source of helpful advice for many open-source toolkits, including ImageJ and Fiji.[57] Virtually everything in the ImageJ ecosystem can be found in freely-accessible GitHub repositories, facilitating contributions and collaborative development. Individuals may modify any particular component of the ImageJ ecosystem, be it core library or particular plugin, and publish these changes to GitHub where other users and developers review the new code, provide feedback, and incorporate useful additions into the ecosystem.[58] The ImageJ wiki, the forum, and GitHub (https://github.com/imagej and https://github.com/fiji) are the three pillars on which the ImageJ ecosystem and its diverse, vibrant user and developer community rests. Together they allow for maintenance, growth, and

TABLE I Representative specialized plugins and libraries in ImageJ ecosystem

Plugin or library	Description	Representative contributors (limited to three)
BigDataViewer[30] (https://imagej.net/BigDataViewer)	Inspects large image files from local and remote data sources while maintaining the image metadata	Tobias Pietzsch
BigStitcher[31] (https://imagej.net/BigStitcher)	Software package that allows simple and efficient alignment of multitile and multiangle image datasets, for example acquired by light sheet, widefield or confocal microscopes	Stephan Preibisch, David Hörl
BigWarp[32] (https://imagej.net/BigWarp)	Tool for manual, interactive, landmark-based deformable image alignment, using Java implemented thin plate splines to build a dense deformation model	John Bogovic, Stephan Saalfeld, Tobias Pietzsch
Bio-Formats[33] (https://imagej.net/Bio-Formats)	Standalone Java library for reading and writing life sciences image file formats capable of parsing both pixels and metadata for a large number of formats	Melissa Linkert, Curtis Rueden, Sébastien Besson
ClearVolume[34] (https://imagej.net/ClearVolume)	Real-time, accelerated 3D-volume rendering for multichannel image processing, inspection during acquisition	Florian Jug, Loic Royer, Martin Weigert,
CSBDeep[35] (https://imagej.net/CSBDeep)	Open-source machine learning plugin compatible with Fiji, python, KNIME that can restore and segment fluorescence images	Uwe Schmidt, Deborah Schmidt, Martin Weigert
DeepImageJ[36] (https://deepimagej.github.io/deepimagej/)	User-friendly plugin that enables the use of a variety of pretrained deep learning models in ImageJ and Fiji	Carlos García-López-de-Haro, Estibaliz Gómez-de-Mariscal, Daniel Sage
Fijiyama[37] (https://imagej.net/Fijiyama)	3D registration tool for multimodal time-lapse imaging	Romain Fernandez,Cédric Moisy
FLIMJ[38,39] (https://imagej.net/FLIMJ)	Fluorescence lifetime imaging microscopy analysis and image processing	Dasong Gao, Paul Barber, Curtis Rueden
ImageJ Ops[17] (https://imagej.net/Ops)	Framework for reusable image processing operations, ops extends Java's mantra of "write once, run anywhere" to image processing algorithms	Curtis Rueden, Christian Dietz, Alison Walter
ImageJ-OpenCV (https://github.com/imagej/imagej-opencv)	Library that enables OpenCV to be used from ImageJ commands and scripts and provides converters between ImgLib2 and OpenCV data structures	Gabriella Turek, Curtis Rueden
ImageJ-TensorFlow (https://github.com/imagej/imagej-tensorflow)	This project enables TensorFlow to be used from ImageJ commands and scripts	Curtis Rueden, Deborah Schmidt, Benjamin Wilhelm
ImageJ updater (https://imagej.net/Updater)	Keeps users up-to-date with all components of ImageJ, including both plugins and the core components (libraries) needed by those plugins	Johannes Schindelin, Curtis Rueden, Deborah Schmidt

(Continues)

TABLE I (Continued)

Plugin or library	Description	Representative contributors (limited to three)
ImgLib2[26] (https://imagej.net/ImgLib2)	General-purpose, multidimensional image processing library and core to ImageJ2 and Fiji	Tobias Pietzsch, Curtis Rueden, Stephan Saalfeld,
Labkit (https://imagej.net/Labkit)	Advanced ImageJ2 plugin for image labeling and segmentation, featuring a user-friendly interface to work on large datasets	Matthias Arzt
MaMuT[40] (https://imagej.net/MaMuT)	Annotation and multiviewing of big data, combining the power of BigDataViewer and TrackMate; requires manual or semi-automated annotation	Jean-Yves Tinevez, Tobias Pietzsch
Mastodon[41] (https://github.com/mastodon-sc/mastodon)	Facilitates computational analysis within cellular, developmental, or stem cell biology to provide interactive inspection that is fast and responsive; semi-automatic and fully automatic tracking data; extract numerical features to gather statistics	Tobias Pietzsch, Jean-Yves Tinevez
SCIFIO[42] (https://imagej.net/SCIFIO)	Flexible framework for SCientific image format input and output, that is, a library for reading and writing N-dimensional image data	Curtis Rueden, Mark Hiner,
SciJava (https://imagej.net/SciJava)	Collaboration of projects providing software for scientific computing—An effort to cooperate and reuse code when feasible; includes ImgLib2, SCIFIO, bio-formats, ImageJ2, or Fiji	Curtis Rueden, Johannes Schindelin, Mark Hiner
SciView[43,44] (https://imagej.net/SciView)	Visualize large 3D datasets, integrates ImageJ ops and ImageJ mesh to mesh data	Kyle Harrington, Ulrik Günther, Tobias Pietzsch
Script editor (https://imagej.net/Script_Editor)	Script editor and interpreter for SciJava scripting languages within ImageJ and Fiji	Johannes Schindelin, Curtis Rueden, Albert Cardona
Trainable Weka Segmentation[45,46] (https://imagej.net/Trainable_Weka_Segmentation)	Plugin that combines a collection of machine learning algorithms with a set of selected image features to produce pixel-classifications	Ignacio Arganda-Carreras, Johannes Schindelin, Matthias Arzt
TrakEM2[47] (https://imagej.net/TrakEM2)	Plugin for morphological data mining, three-dimensional modeling and image stitching, registration, editing, and annotation	Albert Cardona, Stephan Saalfeld

adaptation of the ImageJ ecosystem to the changing image analysis needs of the scientific community.

This review emphasizes new and ongoing developments within the ImageJ ecosystem, and particularly within Fiji, that are useful for biologists working with advanced imaging techniques; they include machine learning with TensorFlow,[59] the hosting of deep learning models with DeepImageJ,[36] and even the training of TensorFlow-based artificial neural networks from within Fiji with CSBDeep.[60,61] We will also cover tasks such as tracking millions of cells in light sheet microscopy recordings of animal development with Mastodon[41] and MaMuT,[40] and 3-dimensional (3D) image data visualization with ClearVolume[34] and SciView.[42,43] These plugins were selected to highlight the diversity of the ImageJ ecosystem. In this review, we also aim to explain some of the latest advancements in the ImageJ ecosystem, revealing how this set of open source tools is adapting to the ever-changing requirements in the fast-paced field of bioimage analysis.

2 | DISCUSSION

ImageJ's capabilities range from common tasks such as opening images of various formats, annotating and processing images, and executing simple workflows on images, to advanced projects involving visualizing and analyzing large image data and implementing machine learning algorithms. The following sections address common tools used in ImageJ and review frequently-used image analysis techniques, as well as ways in which users can extend ImageJ and customize it through macros and scripting. We then highlight some of the current efforts in Fiji to handle large and complex datasets and the ongoing efforts to implement machine and deep learning approaches. Finally, we introduce the newer, more advanced features of ImageJ2, including the ImageJ Ops algorithm framework.

2.1 | Opening and annotating images

While ImageJ is a useful image analysis tool for research in the life sciences, it also serves a broader audience as a basic image viewer and annotation tool. Bio-Formats, included with the Fiji distribution of ImageJ, allows users to open images saved in a broad variety of file formats.[33] This ability to open almost any image generated by a microscope often makes ImageJ the tool of choice for the “first-look” at acquired image data. Brightness and contrast adjustment, image resizing, and application of lookup tables (LUTs) are common tasks that users frequently apply to freshly opened images. Next, the users may want to apply simple transformations, such as rotating the images or cropping, as well as adding various annotations to highlight a specific region of interest (ROI) or structures present in the images. Users may directly inspect pixel intensities, as well as the signal intensity within ROIs, by generating histograms. Most bioimage analysis tools offer a similar basic functionality, yet users often turn to ImageJ as it can open a particular format or because they are familiar with ImageJ's graphical user interface.

The ability to work with image stacks is one of the core ImageJ functionalities, deeply ingrained into the architecture of ImageJ1 and one of the factors in its long-time use. Stacks of images obtained at various depths for a specific field-of-view can be collapsed into a single image by a variety of approaches, most commonly by selecting the highest pixel value along the z direction of the stack, the so-called Maximum Intensity Projection. This is just one example of the functionality of ImageJ in manipulating image stacks. It is particularly useful for fluorescence imaging of 3-dimensional (3D) specimens to capture the fluorescence signal at various depths and focal planes.

For scientific publication and presentations, it is imperative to include information about the real size of the structures that are being displayed. The users of ImageJ can either import the size of pixels in microns from the image metadata or set the scale manually if this information is not available. This facilitates direct measurement of the size of structures present in the image. Additionally, users can use the pixel size data to place scale bars in the images for presentations or publications.

2.2 | Common image analysis techniques: Segmentation, tracking, and registration

ImageJ provides users the ability to segment,[62] track particles,[63] and register[64] their datasets. Segmentation, that is, object detection/delineation, allows biologists to computationally separate certain regions in their images, such as the total area occupied by cells in brightfield microscopy images.[62] Particle tracking algorithms, that is, identifying and linking objects across multiple images over time, can, for example, assess mitochondrial transport movements in cortical and hippocampal neurons.[63] Image registration, that is, transforming different sets of image data into one coordinate system, is crucial for generating high-resolution stacks of thick 3D fluorescence microscopy data where the sample may be moved during acquisition to accommodate sectioning; one such example is serial imaging of the *Drosophila* brain where accurate stitching and registration is needed.[15,64,65]

More advanced techniques and tools are available for segmentation, including Trainable WEKA Segmentation (https://imagej.net/Trainable_Weka_Segmentation), which is an ImageJ plugin that leverages machine learning provided by the open WEKA library to segment previously unseen image data.[44] The Trainable WEKA Segmentation plugin allows users to train classifiers simply by “tracing” labels on selected pixels in the training images.[46] The Fiji plugin Labkit (https://imagej.net/Labkit), also implementing pixel classification functionality via the WEKA library, additionally offers a user-friendly and efficient way to label large 2D and 3D image datasets thanks to improvements in architecture and scalability in the BigDataViewer[30] and the SciJava[66] component collection.

Segmenting objects is the first step in tracking analysis, which allows researchers to follow the segmented objects' movements over time.[29,67,68] Objects or structures of interest are separated from the background signal and then can be tracked frame-to-frame. Researchers are typically interested in tracking particular labelled biological entities, such as cells or sub-cellular structures. TrackMate (https://imagej.net/TrackMate) is a Fiji plugin that facilitates manual, semi-automated, or fully-automated single particle tracking.[69] This plugin can be utilized to track particles in 2D and 3D datasets and accommodates multichannel images. The user interface allows for separate execution of the image segmentation and particle-linking steps.[29,70] Users are provided with algorithmic results for tracking, as well as quantitative results and plots.[29,70] TrackMate includes several tracking algorithms, such as LAP tracking,[69] and experienced users can implement their own custom algorithms. The tracking data can be exported and analyzed in third party packages outside of ImageJ. The development of the Mastodon and MaMuT plugins, which allow for cell tracking and segmentation within large datasets, will be highlighted later in Section 2.4.

There are many advanced tools available in ImageJ for registration. TrakEM2 (https://imagej.net/TrakEM2) allows users to data mine their images based on morphological features and perform 3D modeling, stitching, registration, and annotation.[71] TrakEM2 was developed to segment neurons and reconstruct neuronal circuits by skeletonizing neuronal arbors based on nodes and directional edges that incorporate the degree of certainty in the continuity of edges.[71] Fijiyama (https://imagej.net/Fijiyama) is a 3D registration tool useful for multimodal time-lapse imaging.[37,72] When monitoring data changes over time, particularly if acquired on two different imaging systems, changes in position and orientation and registering different highlighted views of structures can be challenging if not impossible, yielding a compromised result. Fijiyama combines image stacks in a coherent manner that allows the user to explore their data using the ImageJ 3D Viewer.[37,73] Recent updates include support for registering 4D or 5D multichannel or multitime images, and future efforts will include new transformation models, as well as extended support for images where intensity may be modulated upon adjusting size.[72] The BigWarp tool (https://imagej.net/BigWarp) allows for interactive, manual landmark-based image alignment.[32] The BigDataViewer[30] (https://imagej.net/BigDataViewer) allows users to visualize and navigate their datasets, using landmark pair placement to display the effects of the warp.[32]

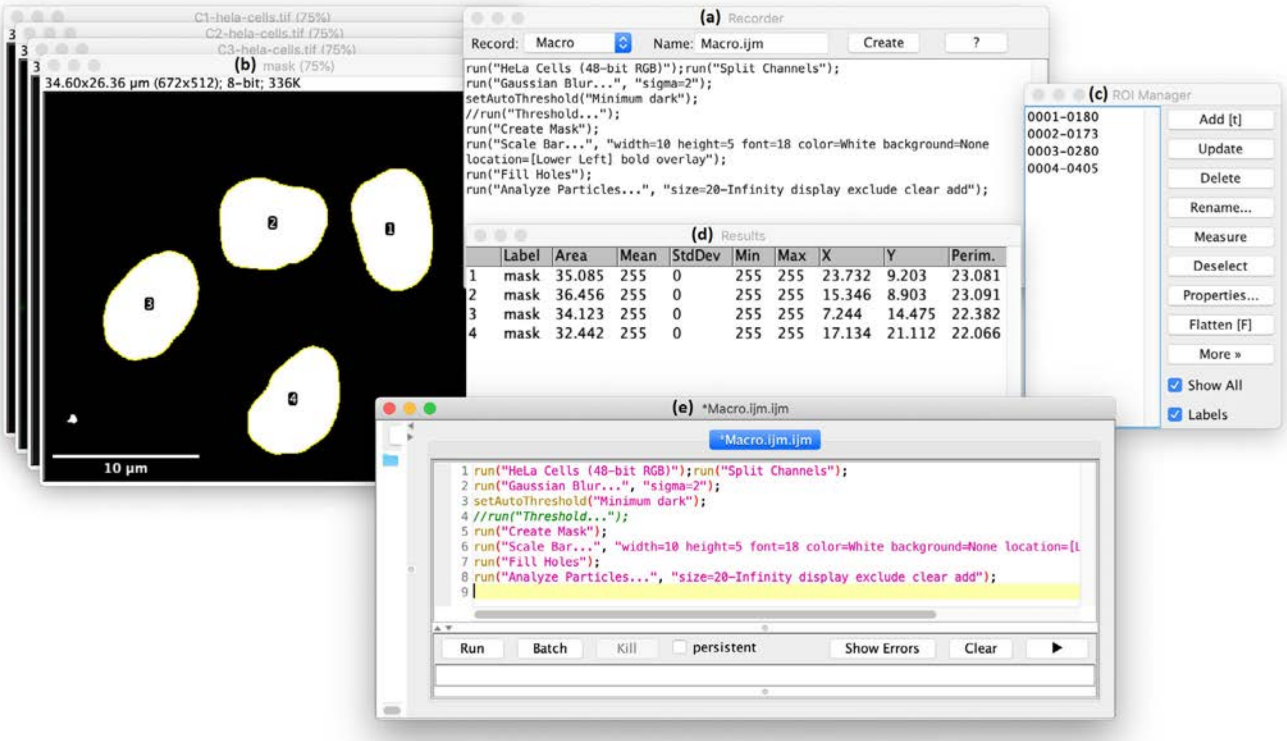
2.3 | Scripting and plugins

Scripts allow users, even those with only basic programming knowledge, to automate repetitive image analysis tasks and package simple solutions into short, named programs that can be linked as shortcuts to the ImageJ toolbar or menu for quick and uncomplicated access.[74] The most frequently used types of scripts in ImageJ are "macros", a specific type of a script written in the ImageJ1 macro language. The ImageJ1 macro language was designed to be easy for new programmers to learn and debug,[75,76] but even users with no programming experience can use the Macro Recorder, which allows users to record a series of manually executed ImageJ commands directly into code[76] (Figure 2). Such recorded sequences of events executed using the ImageJ GUI can then be adapted in the Script Editor into fully functional scripts[75,77] that can be saved and added to the ImageJ menu if desired. Detailed tutorials on how to record, implement, and publish ImageJ macros (https://imagej.net/Tutorials) are freely available on the ImageJ wiki. A testament to the popularity of this approach amongst ImageJ users are the thousands of available scripts listed on ImageJ1 website.[76]

In addition to the ImageJ1 macro language, Fiji introduced facilities to write scripts in the Script Editor in several popular scripting languages, including JavaScript, Clojure, BeanShell, Groovy, Python/Jython, Ruby/JRuby, R, and Scala.[77] Not all available languages can be directly recorded via the Macro Recorder and therefore require intermediate programming skills. However, the programming capabilities in these established scripting languages are vastly more advanced compared to the ImageJ1 macro language. Since there are excellent online tutorials and templates (small script examples including all the syntactic requirements to start writing a specific solution) and many biologists have significant experience with scripting languages, this mechanism of generating workflows in the ImageJ ecosystem is increasingly prevalent.

2.4 | Current efforts within ImageJ: Handling big data and specialized plugins for analysis

Challenges arise when opening large, multidimensional datasets using ImageJ1 (e.g., whole-slide digital pathology images or stitched mosaics). Image datasets have gotten larger and larger, usually being multiple gigabytes, often reaching terabytes. Such data is difficult to open, cumbersome to view, annotate, or quantify using ImageJ's standard image window. Fiji's BigDataViewer (BDV) plugin



```
run("HeLa Cells (48-bit RGB)");run("Split Channels");
run("Gaussian Blur...", "sigma=2");
setAutoThreshold("Minimum dark");
//run("Threshold...");
run("Create Mask");
run("Scale Bar...", "width=10 height=5 font=18 color=White background=None
location=[Lower Left] bold overlay");
run("Fill Holes");
run("Analyze Particles...", "size=20-Infinity display exclude clear add");
```

```
0001-0180
0002-0173
0003-0280
0004-0405
```

	Label	Area	Mean	StdDev	Min	Max	X	Y	Perim.
1	mask	35.085	255	0	255	255	23.732	9.203	23.081
2	mask	36.456	255	0	255	255	15.346	8.903	23.091
3	mask	34.123	255	0	255	255	7.244	14.475	22.382
4	mask	32.442	255	0	255	255	17.134	21.112	22.066

```
run("HeLa Cells (48-bit RGB)");run("Split Channels");
run("Gaussian Blur...", "sigma=2");
setAutoThreshold("Minimum dark");
//run("Threshold...");
run("Create Mask");
run("Scale Bar...", "width=10 height=5 font=18 color=White background=None location=[L
run("Fill Holes");
run("Analyze Particles...", "size=20-Infinity display exclude clear add");

```

FIGURE 2 Example of how to generate simple code in ImageJ/Fiji. After opening (a) the Macro Recorder (Plugins › Macros › Record...), all subsequent manual commands are saved as code, in this case, the ImageJ1 macro language. In this example, (b) the "HeLa Cells" sample image available in Fiji was opened and channels split. Using the nuclei channel (C3), the image was filtered with a Gaussian Blur, and a mask was generated. A scale bar (10 µm) was also added. By calling the command Analyze Particles (Analyze › Analyze Particles...), nuclei were identified as regions of interest (ROIs) and were saved to (c) the ROI Manager and (d) a list of measurements appeared in the Results window. Measurements can be adjusted/selected via Set Measurements (Analyze › Set Measurements...). When a user clicks "Create" in the Macro Recorder, the generated code is added to the (e) Script Editor and, once there, can easily be edited, saved, and reused

has solved these problems for the ImageJ ecosystem by allowing users to navigate arbitrarily large image files from local and remote data sources.[30] The key to BDV's performance on big image data is the use of hierarchical data structures (accompanied by suitable data formats such as HDF5 or N5) that store image data as non-overlapping chunks (referred to as "blocks") in a multi-resolution pyramid. BDV requests and loads only the blocks that the user is currently interacting with and only at the appropriate resolution. Additionally, using ImgLib2's sophisticated caching infrastructure, BDV keeps track of recently used blocks to avoid loading the same data over and over again. BigDataViewer considers each imaging channel or acquisition angle from a multi-parametric image as a separate source that users can manipulate independently. All this functionality relies on the powerful ImgLib2 library.[26] After the original work on BDV, many additional functions have been added, with some still targeting visualization of large data and others concerning large data handling. Today, many research software developers in the ImageJ ecosystem use these functions, which were ultimately incorporated as core ImageJ components. The BigStitcher (https://imagej.net/BigStitcher) software package facilitates efficient alignment for multitile and multiangle imaging datasets acquired on light sheet, widefield, or confocal microscopes.[31] BigStitcher supports file sizes that can be as large as many terabytes of data, such as those often produced during light sheet microscopy acquisition. BigStitcher is integrated within BDV, and users can interactively display and process input images regardless of file size.[31,78]

The power of BDV is best demonstrated via tools such as Mastodon[41] and MaMuT[40] that offer interactive segmentation and tracking of cells within large developmental bioimage datasets. In the context of studies of development, researchers are often interested in following many cells over time in 3D, quantifying their behaviors, and ultimately extracting developmental lineages through the division of cells as they build a tissue or entire organism over time. These tasks are often performed on multiterabyte 4D image acquisitions, and the challenges to display the data and run analysis on it in such contexts are formidable.

MaMuT (https://imagej.net/MaMuT) is an ImageJ plugin that allows efficient manual annotation of cellular lineages in the context of such big image datasets.[40] It combines BDV's large data management capabilities with TrackMate's infrastructure for managing tracking results, described in Section 2.2. It also offers supervised semi-automatic tracking of individual cells through the imaged volumes. In this way, it has been used to extract a complete developmental lineage of an appendage of the crustacean *Parhyale hawaiensis*, imaged with light sheet microscopy, answering long-standing biological questions about appendage morphogenesis.[40] In order to complete this work, it was necessary to isolate one appendage from the rest of the large, light sheet acquisition. MaMuT was unable to handle the many millions of cell detections in this complete dataset. To address such limitations, the original MaMuT authors are currently developing Mastodon (https://github.com/mastodon-sc/mastodon), a tool capable of dealing with developmental lineages that consist of tens of millions of cells.[41] Once finished, Mastodon will enable the analysis of automated tracking solutions generated by arbitrary tracking algorithms on large image data. As any such automated method will result in extremely large lineage trees that are full of errors, biologists will also be able to use Mastodon's highly optimized, ergonomic, and collaborative user interface to interactively correct such imperfect solutions. Figure 3 shows a glimpse of what Mastodon will be able to do by visualizing results of the automated tracking of nuclei in a 3D cartographic projection of a developing *Tribolium castaneum*, recorded by multiview light sheet microscopy.

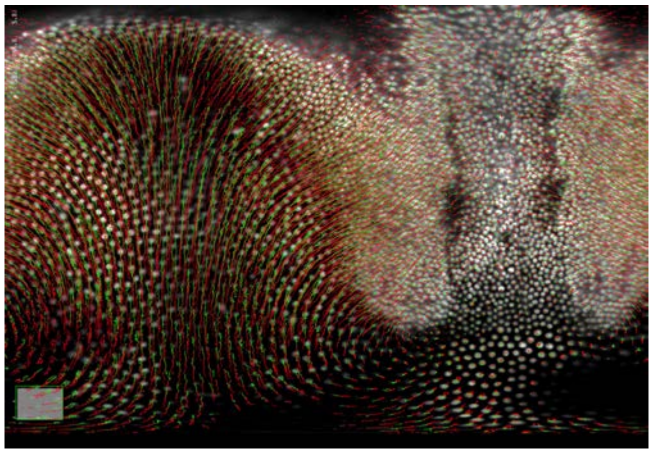


FIGURE 3 Using Mastodon to perform nuclei tracking in a 3D cartographic projection of a developing *Tribolium castaneum* using an image of the blastoderm shortly after its formation (left side: dorsal side; right side: ventral side). Each time-point includes more than 500 tracked cells, visualized here in white fluorescence, amounting to more than a quarter of a million detections. The trajectory of each cell is represented by a line that fades from green to red backwards in time. The cartographic projection does not preserve lengths, and thus there is no uniform scale on the image. The cells on this image have a diameter that ranges from 6 μm (ventral) to 22 μm (dorsal). Images provided with the coordination of Drs. Jean-Yves Tinevez (Institut Pasteur, Paris), Tobias Pietzsch and Vladimir Ulman, and courtesy of Dr. Akanksha Jain and Dr. Mette Handberg-Thorsager, (Tomancak lab, MPI-CBG, Dresden)

The challenging, large, and complex developmental lineaging data highlights the need for better, more performant image visualization tools in the ImageJ ecosystem. To that end, several recent updates within the ImageJ framework have been made or are currently under development to handle different analysis aspects of increasingly large datasets. To better visualize 3D datasets, the SciView plugin (https://imagej.net/SciView) is currently under development, with preview releases available through the ImageJ Updater.[42,43] The SciView plugin integrates a combination of features from the more recent ImageJ2 libraries, including ImageJ Ops and ImageJ Mesh, allowing the user to interact with the image and mesh data. Figure 4 highlights an example of the SciView plugin for volume rendering and image segmentation.

The ultimate frontier is to make these types of tools, the advanced data visualization offered by SciView and the lineage tracing of MaMuT and Mastodon, work in real time during data acquisition. The ClearVolume plugin (https://image.net/ClearVolume) offers real-time, GPU-accelerated volume rendering for multichannel image processing, which is particularly useful for light sheet microscopy datasets.[34] The Fiji integration of ClearVolume enables users to conveniently and very efficiently render and navigate through moderately-sized, multichannel volumes and movies.

Once user data is compiled for visualization and navigation, signal quantification and measurement tools may be utilized for subsequent analyses, such as lifetime measurements. The fluorescence lifetime of a molecule is sensitive to the fluorophore's immediate environment, including pH, oxygen concentration, and proximity to other molecules exhibiting Förster resonance energy transfer.[79] Fluorescence lifetime imaging microscopy (FLIM) and spectral lifetime imaging microscopy (SLIM) are two techniques that offer the ability to assess the state of a fluorescent molecule to gain insight into physiological changes such as metabolism.[80–82]

The recently-developed FLIMJ plugin (https://imagej.net/FLIMJ) reflects updated efforts to allow users easily

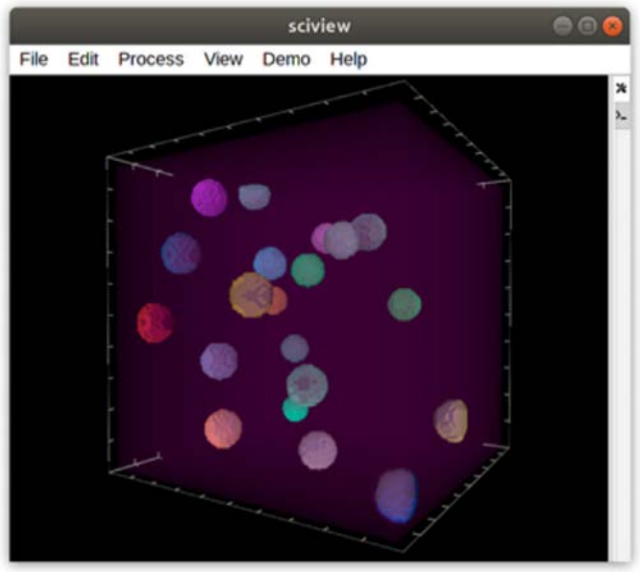


FIGURE 4 A SciView example visualization with synthetic data of a 3D image containing 20 hypothetical spherical regions of interest, representing artificial cells. The synthetic cell data is shown as a volume rendering. Regions of interest are shown as mesh-based geometries generated by ImageJ Ops' implementation of the marching cubes algorithm. Color identifies nonoverlapping cells within the represented 100 x 100 x 100 pixels bounding box. Image courtesy of Dr. Kyle Harrington, Helmholtz Imaging Platform, Max Delbrueck Center for Molecular Medicine in the Helmholtz Association, Berlin, Germany and Dr. Ulrik Guenther, Center for Advanced Systems Understanding Görlitz/Center for Systems Biology, Dresden

to analyze FLIM data (Figure 5).[38] FLIMJ accommodates single, double, or triple exponential fits with integrated noise models; it can fit individual pixels, entire images-per-pixel, or perform global analysis across the entire image.[38] Fit methods include Levenberg–Marquardt Algorithm (LMA), Rapid Lifetime Determination (RLD), Bayes and others, and statistical models including Poisson, Gaussian, Maximum likelihood estimation (MLE), and others. Users are also able to select the image-spatial binning size, thresholds, and the number of components. Batch processing can be performed to analyze datasets that include many FLIM images.[38,39] FLIMJ-Ops plugs the FLIMlib library into ImageJ Ops (which will be covered in detail below), allowing users, via Jupyter notebooks, to invoke the Ops of FLIMJ-Ops directly for processing.

2.5 | Machine learning in ImageJ

Machine learning applies artificial intelligence to train computer systems to automate processes and learn from experience to improve models without explicit programming.[83] Deep learning, the sub-field of machine learning that trains artificial neural networks (ANNs) on a body of labeled training data to solve various data analysis tasks, is currently one of the most potent techniques in the arena of artificial intelligence.[83] Computer vision seeks to equip computers with high-levels of understanding to automate tasks that human vision can perform.[84] These techniques are often applied to very large, multivariable datasets that emphasize improving predictive accuracy, and now machine learning libraries are available in ImageJ via new plugins and libraries such as ImageJ-OpenCV (https://github.com/imagej/imagej-opencv) and ImageJ-TensorFlow[85] (https://github.com/imagej/imagej-tensorflow).

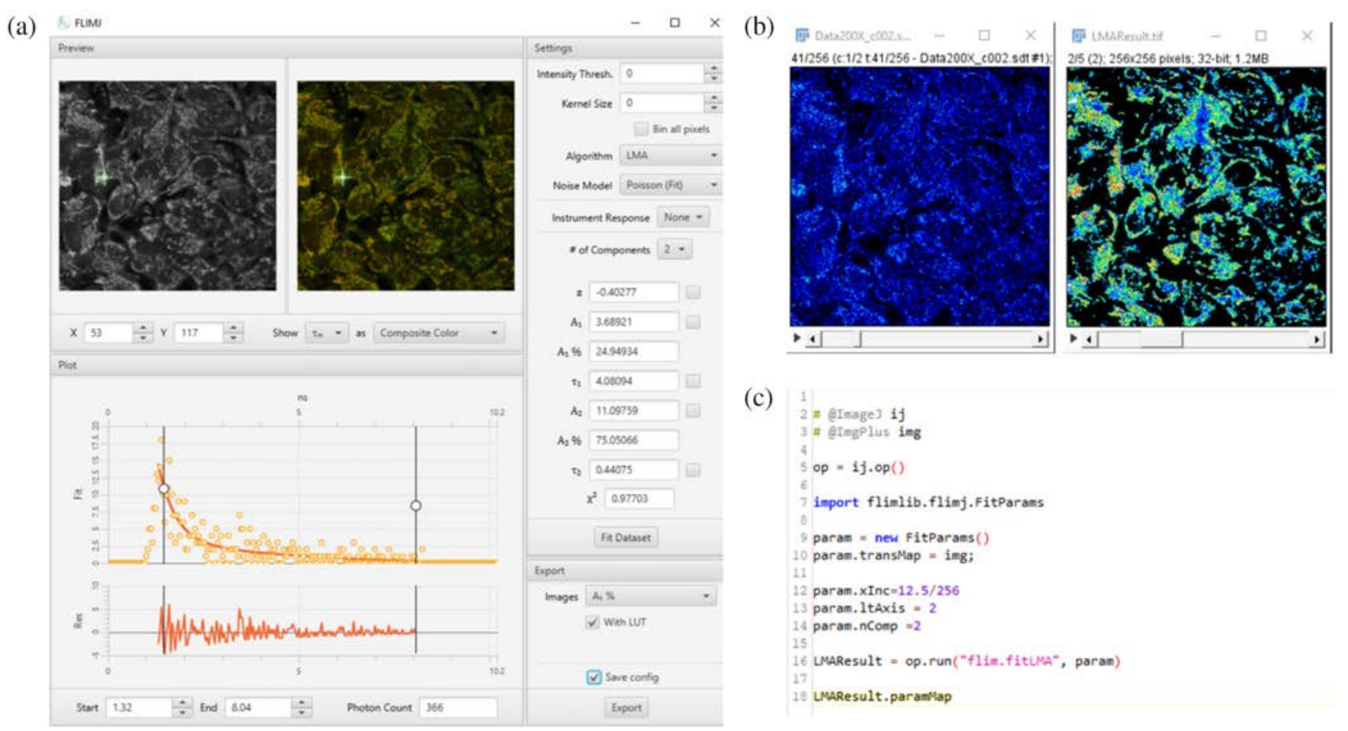
2.5.1 | OpenCV

OpenCV (https://imagej.net/OpenCV) is a library comprised of thousands of computer vision and machine learning algorithms.[86] The OpenCV library supports image processing, feature detection, object detection, machine learning, and video analysis. Dominguez et al. created a package that combines ImageJ with OpenCV to leverage the robust computer vision algorithms of OpenCV with a simple ImageJ GUI for defining regions of interest.[21] The ImageJ-OpenCV library helps to convert between ImgLib2 and OpenCV data using the SciJava framework.[87]

2.5.2 | TensorFlow and ImageJ-TensorFlow

Deep learning is used in various ways in the ImageJ ecosystem, enabled by developments such as the open-source ImageJ-TensorFlow library[85] (https://github.com/imagej/imagej-tensorflow). ImageJ-TensorFlow translates between ImageJ images (backed by ImgLib2) and TensorFlow tensors,[59] thereby enabling ImageJ commands and scripts to utilize TensorFlow's data-flow model that maps computations onto a broad scale of hardware platforms. This might include inference measurements made on all kinds of devices, including mobile devices, single machines, or large systems with thousands of graphics processing units (GPUs).[59] A more in-depth description of the computer workflow and models is discussed by Abadi et al.[59]

TensorFlow has been used in medicine to interface with DICOM images to classify normal versus Parkinson's disease groups,[88] to detect early and advanced glaucoma on fundus photographs,[89] and for



Intensity Thresh. 0
Kernel Size 0
Bin all pixels
Algorithm LMA
Noise Model Poisson (Fit)
Instrument Response None
# of Components 2
z -0.40277
A₁ 3.68921
A₁ % 24.94934
τ₁ 4.08094
A₂ 11.09759
A₂ % 75.05066
τ₂ 0.44075
χ² 0.97703
Images A₁ %
[✓] With LUT
[✓] Save config
X 53 Y 117 Show τₘ as Composite Color
Start 1.32 End 8.04 Photon Count 366

```

# @ImageJ ij
# @ImgPlus img

op = ij.op()

import flimlib.flimj.FitParams

param = new FitParams()
param.transMap = img;

param.xInc=12.5/256
param.ltAxis = 2
param.nComp =2

LMAResult = op.run("flim.fitLMA", param)

LMAResult.paramMap
```

FIGURE 5 An example of the FLIMJ plugin. (a) The FLIMJ-UI consists of six panels: two image panels, one fitting, one residual, one parameter, and one export panel. The fitting panel shows one pixel's intensity decay curve (yellow bubbles) and the fit-curve (red). Fluorescence lifetime imaging analysis is then performed to analyze the (b) autofluorescence lifetime signal from mammalian HeLa cells as the input dataset. (c) The fitting of the same dataset can also be done from direct FLIMJ-Ops functions. A minimalistic script can run any available fit-function of choice. Image courtesy of Dr. Jenu Chacko, Laboratory for Optical and Computational Instrumentation, Center for Quantitative Cell Imaging, University of Wisconsin-Madison

segmenting boundaries on multiphoton images of skin cells.[90] Most interestingly, the integration of TensorFlow via ImageJ-TensorFlow not only enables the execution of pretrained deep learning models, but it can also enable the training of ANNs from within the ImageJ ecosystem, for example, within CSBDeep (see below).

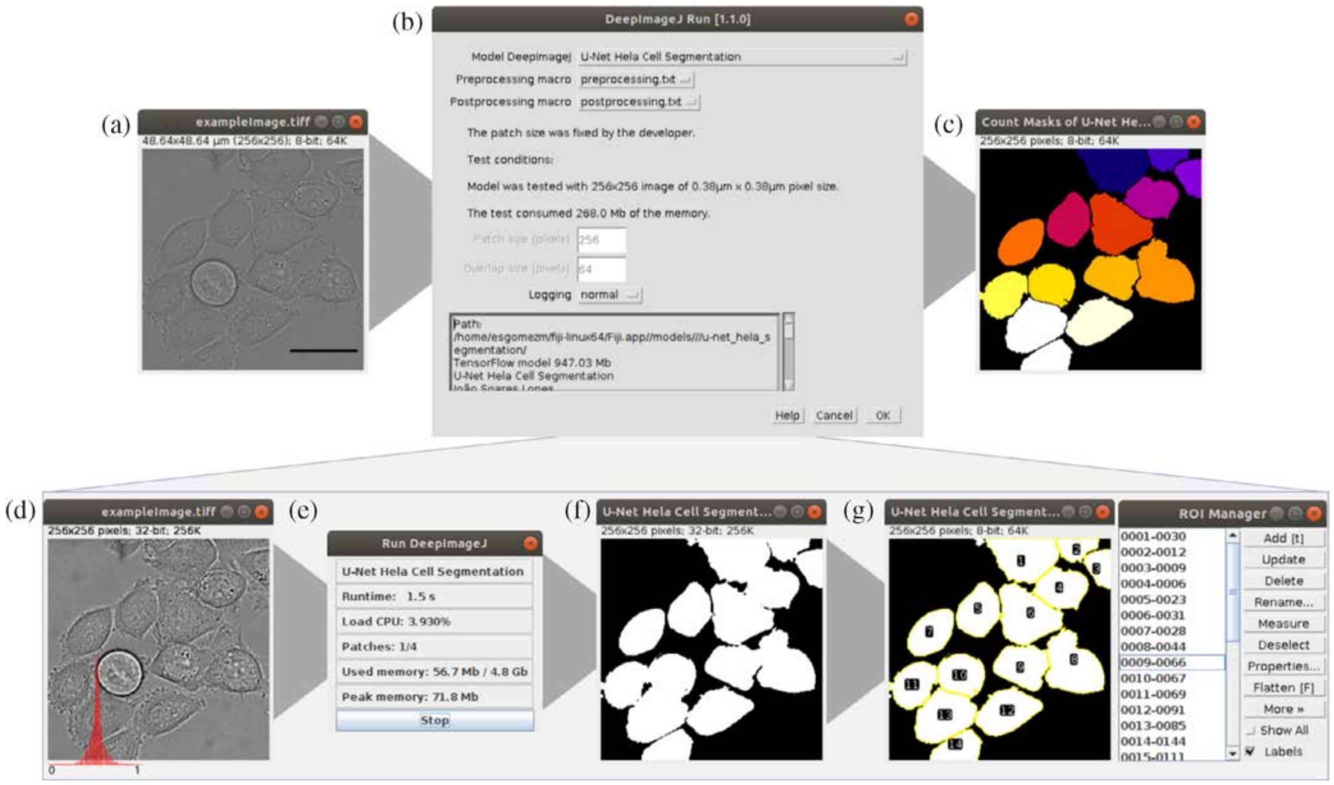
2.5.3 | DeepImageJ

In recent years, deep learning has proven its superiority to other approaches in tasks such as object detection and classification, image segmentation, face recognition, and many more.[91] Deep learning solutions in the microscopy image domain include image restoration with Content-Aware Image Restoration (CARE)[35] (https://imagej.net/CARE), denoising datasets with Noise2Void[92] (https://imagej.net/N2V), object and pattern recognition, and segmentation.[93]

DeepImageJ (https://deepimagej.github.io/deepimagej/) is a plugin that enables access to deep learning models from within Fiji or ImageJ (Figure 6).[36] Users need no experience with deep learning but can simply download available deep learning models from a central repository and embed them directly into ImageJ or Fiji.[36] The execution of the pretrained deep learning model makes use of the Java bindings of TensorFlow.[36] Since many models require certain pre- and post-processing of the image data, DeepImageJ also hosts and serves those in the form of ImageJ macros or scripts. Gómez-de-Mariscal and García-López-de-Haro et al. show several examples of data processing using DeepImageJ, including recovering fluorescence isotropy from cells,[94] virtually staining unlabeled cells with a hematoxylin and eosin filter with VirtualStain,[95] and segmenting HeLa cells with the U-Net feature.[96]

2.5.4 | CSBDeep

CSBDeep (https://imagej.net/CSBDeep and https://csbdeep.bioimagecomputing.com/) is an open-source library with extensive deep learning functionality in Python and Java (Fiji). Originally developed for CARE purposes, it is now also used in a plethora of other research and software projects that incorporates denoising tools like Noise2Void,[92] DenoiSeg[97] (https://imagej.net/DenoiSeg) and others.[61] Initially, Weigert


(a)
exampleImage.tiff
48.64x48.64 μm (256x256); 8-bit; 64K
(b)
(c)
Count Masks of U-Net He...
256x256 pixels; 8-bit; 64K
(d)
exampleImage.tiff
256x256 pixels; 32-bit; 256K
0
1
(e)
(f)
U-Net Hela Cell Segment...
256x256 pixels; 32-bit; 256K
(g)
U-Net Hela Cell Segment...
256x256 pixels; 8-bit; 64K
1
2
3
4
5
6
7
8
9
10
11
12
13
14

DeepImageJ Run [1.1.0]

Model DeepImageJ: U-Net Hela Cell Segmentation
Preprocessing macro: preprocessing.txt
Postprocessing macro: postprocessing.txt

The patch size was fixed by the developer.

Test conditions:

Model was tested with 256x256 image of 0.38μm x 0.38μm pixel size.

The test consumed 268.0 Mb of the memory.

Patch size (pixels): 256
Overlap size (pixels): 64
Logging: normal

```
Path:
/home/esgomezm/fiji-linux64/Fiji.app//models///u-net_hela_s
egmentation/
TensorFlow model 947.03 Mb
U-Net Hela Cell Segmentation
João Soares Lopes
```

Help | Cancel | OK

Run DeepImageJ

U-Net Hela Cell Segmentation
Runtime: 1.5 s
Load CPU: 3.930%
Patches: 1/4
Used memory: 56.7 Mb / 4.8 Gb
Peak memory: 71.8 Mb
Stop

ROI Manager

0001-0030
0002-0012
0003-0009
0004-0006
0005-0023
0006-0031
0007-0028
0008-0044
0009-0066
0010-0067
0011-0069
0012-0091
0013-0085
0014-0144
0015-0111

Add [t] | Update | Delete | Rename... | Measure | Deselect | Properties... | Flatten [F] | More » | Show All | [✓ Labels]

FIGURE 6 An example of using DeepImageJ to segment HeLa cells in a differential interface contrast (DIC) microscopy image using a trained U-Net. Panels (a–c) depict the ImageJ steps visualized by the user and (d–g) show the automatically executed steps defined by the model developer and executed inside of DeepImageJ. (a) The image in this figure is a DIC microscopy image of HeLa cells (scale bar =15 μm). (b) The DeepImageJ Run GUI allows the selection of a suitable model from the user's local machine and of any preprocessing and postprocessing ImageJ macros given within the selected trained model (in this case, "U-Net HeLa Cell Segmentation"). DeepImageJ displays some important information about the model's configuration such as the patch size (input size admitted by the model's architecture) and the overlap size (size of the image borders that need to be discarded in the output image due to the size of the convolutions in the network). (c) The final result of the segmentation using a trained U-Net is shown. (d) The DIC image intensity values were normalized (preprocessing step). (e) A dialog box displays the time and memory information about the U-Net inference process. (f) The output of the U-Net is shown, and (g) Watershed segmentation was used to obtain uniquely labeled cells (postprocessing). The raw DIC input image source is from the Cell Tracking Challenge (http://celltrackingchallenge.net/2d-datasets/), captured by Dr. G. van Cappellen, Erasmus Medical Center, Rotterdam, The Netherlands. Figure courtesy of Dr. Daniel Sage, School of Engineering (STI), Institut de Microtechnique (IMT), and Laboratoire d'Imagerie Biomédicale (BIG), Ecole Polytechnique Fédérale de Lausanne

et al. showed how fluorescence microscopy images could be restored from images with 60 times fewer photons acquired, showing how resolution could be enhanced even after under-sampling the images.[35] Their work also demonstrated how to resolve granular structures smaller than the diffraction limit at 20-times-higher frame rates.[35]

Today, CSBDeep has become the basis for many other useful applications.[61,92,97,98] Additionally, the Fiji-integrated Java parts of CSBDeep are now capable (via the use of ImageJ-TensorFlow) to not only run saved deep learning models, but even train such models as Noise2Void and DenoiSeg.[97] It is important to note here that all existing code is open and fully available to others. The intention of the developers of CSBDeep is to enable the wider developer community to deploy their deep learning solutions to the ImageJ ecosystem, which would be of great utility for all users.

2.5.5 | Summary of plugin and library contributions in ImageJ ecosystem

From navigating large datasets to handling multiview and multidimensional imaging data to implementing machine learning and deep learning algorithms, ImageJ has responded to the feedback of researchers and risen to meet their data handling needs. Table I includes some key plugins and tools that we have highlighted in this review. All items in the table are available in Fiji and most are currently ImageJ-compatible or plan to be. The following section introduces how changes to the ImageJ software framework via ImageJ2 enhance the handling of more complex datasets and facilitate interoperability (Table I).

2.6 | ImageJ2 and ImageJ ops

Developments in modern microscopy and imaging are a driving force for change to the ImageJ framework. As multimodal and multidimensional microscopy have evolved, so too have the needs of ImageJ users. ImageJ2 (https://imagej.net/ImageJ2) is a major effort to redesign ImageJ's foundation to accommodate new imaging methods, including polarized light microscopy,[99] light sheet microscopy,[100] large stitched datasets, and fluorescence lifetime imaging microscopy.[38] Because it was designed to reflect the needs of the user community and allow for collaborations to foster open development, ImageJ2 improves upon ImageJ's: extensibility, that is, its potential to be extended and enhanced with new features; modularity, that is, the extent to which its functionality is accessible as reusable building blocks rather than only a single monolithic application; interoperability, that is, its ability to be used in combination with other programs and workflows; and generality and scalability, that is, its capability to manage more complex data types and larger image file sizes.[23,27]

ImageJ2 provides mechanisms for accessing ImageJ functionality from other software tools, including from Python (https://github.com/imagej/pyimagej) and Jupyter Notebook (https://github.com/imagej/imagej-notebook), from Node.js (https://github.com/imagej/imagej-node), from MATLAB (https://doi.org/10.1093/bioinformatics/btw681), from the command line (https://imagej.net/Scripting_Headless), and via web services (https://imagej.net/Server). Such interoperability allows for collaboration between bioimage analysis tools to facilitate improved automation, where the respective strengths of each software package may be effectively leveraged.

ImageJ2 is more than just a desktop application; it is a collection of reusable libraries and plugins built upon the foundational layer of SciJava (https://imagej.net/SciJava), an umbrella for software components which are broader than only image processing.[66] The SciJava layer houses ImageJ's new plugin framework and application container (https://imagej.net/SciJava_Common) as well as the SciJava Script Editor (https://imagej.net/Script_Editor) and script language plugins. SciJava components also provide random-access data input/output interfaces and plugins, expression parsing, common user interface elements, and native library utilities, to name a few examples.

ImageJ2 comes bundled with the ImageJ Updater to ensure that users have seamless access to the latest updates, as well as to new plugins when additional Update Sites are enabled.[27] ImageJ2 has extensible support for scientific image file types thanks to the SCIFIO (Scientific Image Format Input and Output) library[42] and is able to handle N-dimensional data through the integration of the ImgLib2 multidimensional image processing library that supports numeric and nonnumeric data types.[26] The Bio-Formats library,[33] which is included with Fiji, plugs into SCIFIO, integrating its additional file formats more seamlessly into ImageJ.

The main library of ImageJ2 in terms of image processing algorithms is ImageJ Ops (https://imagej.net/Ops), which serves as an extensible framework for reusable operations.[27] A primary goal of Ops is to make it easier and simpler for programmers to implement algorithms, such that they can be used as is from any SciJava-compatible software project.[101] Like a function call in most programming languages, an operation is invoked by its name and arguments, but unlike a standard function call, Ops performs a “matching” process to determine the best function implementation to use based on the request. For example, for smaller kernel sizes, naïve convolution is faster than convolving via a Fast Fourier Transform (FFT), but FFT-based convolution is a better choice as the kernels become larger. The Ops framework will pick the best approach based on the arguments given, in a manner enabling developers to extend ImageJ with additional implementations handling new cases or improving performance. For users, all routines available in ImageJ Ops are accessible in a consistent way from scripts, categorized by functionality into namespaces, deterministic in behavior, that is, always giving the same result for the same inputs, and usable without a graphical display for example, from the command line or on nodes of a compute cluster.

3 | CONCLUSIONS

ImageJ has grown from a simple tool to analyze two dimensional images, into a widely utilized platform for modern biological image analysis. This growth is due to the tireless work of the creator of ImageJ1, Wayne Rasband, the contributions of the emergent and synergistic Fiji and ImageJ2 communities, and the vast contingent of users and developers adopting the ImageJ ecosystem for their work. It was developed with the intent to be adaptable by the users, for the users, and that tradition of help and accessibility continues through the ImageJ wiki, Scientific Community Image Forum, and GitHub repositories. Open-source access to code repositories allows developers to adapt existing solutions to increasingly advanced image datasets, drawing from advances in the computer vision and machine learning fields. Cutting edge solutions are pushed through the ImageJ Updater to tens of thousands of users, who offer feedback directly to the developers at scale. As new ecosystems based on different programming paradigms and technologies emerge, unconstrained by decades of legacy code, the challenge for the ImageJ ecosystem is to remain relevant. This review highlights ongoing efforts within the ImageJ ecosystem to adapt to the emergent needs and challenges in modern biological image analysis. While the features and adaptability of the ImageJ infrastructure and the large numbers of users and developers invested in this software make it a powerful tool, the way forward will clearly be to continue building bridges to emerging platforms for the benefit of the larger scientific community.

ACKNOWLEDGMENTS

We thank Dr. Daniel Sage, Dr. Jean-Yves Tinevez, Dr. Jenu Chacko, Dr. Kyle Harrington and Dr. Ulrik Guenther for assistance with figures. We also thank all members of the ImageJ ecosystem community for their contributions, especially the main developers of the many functionalities and plugins we have listed in Table I.

AUTHOR CONTRIBUTIONS

Alexandra Schroeder: Writing-original draft; writing-review and editing. **Ellen Dobson:** Writing-original draft; writing-review and editing. **Curtis Rueden:** Writing-original draft; writing-review and editing. **Pavel Tomancak:** Writing-original draft; writing-review and editing. **Florian Jug:** Writing-original draft; writing-review and editing. **Kevin Eliceiri:** Conceptualization; resources; supervision; writing-original draft; writing-review and editing.

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

ORCID

Kevin W. Eliceiri https://orcid.org/0000-0001-8678-670X

REFERENCES

1. Cardona A, Tomancak P. Current challenges in open-source bioimage informatics. Nat Methods. 2012;9:661–665.
2. Guiet R, Burri O, Seitz A. Open source tools for biological image analysis. In: Rebollo E, Bosch M, editors. Computer Optimized Microscopy: Methods and Protocols. Methods in Molecular Biology. New York, NY: Springer, 2019; p. 23–37.
3. Sahl SJ, Hell SW, Jakobs S. Fluorescence nanoscopy in cell biology. Nat Rev Mol Cell Biol. 2017;18:685–701.
4. Planchon TA, Gao L, Milkie DE, et al. Rapid three-dimensional isotropic imaging of living cells using Bessel beam plane illumination. Nat Methods. 2011;8:417–423.
5. Weber M, Mickoleit M, Huisken J. Light sheet microscopy. Methods Cell Biol. 2014;123:193–215.
6. Giepmans BNG, Adams SR, Ellisman MH, Tsien RY. The fluorescent toolbox for assessing protein location and function. Science. 2006;312:217–224.
7. Zipfel WR, Williams RM, Webb WW. Nonlinear magic: Multiphoton microscopy in the biosciences. Nat Biotechnol. 2003;21:1369–1377.
8. Eliceiri KW, Berthold MR, Goldberg IG, et al. Biological imaging software tools. Nat Methods. 2012;9:697–710.
9. López-Cano M, Fernández-Dueñas V, Ciruela F. Proximity ligation assay image analysis protocol: Addressing receptor-receptor interactions. Methods Mol Biol. 2019;2040:41–50.
10. Caldon CE, Burgess A. Label free, quantitative single-cell fate tracking of time-lapse movies. MethodsX. 2019;6:2468–2475.
11. Grishagin IV. Automatic cell counting with ImageJ. Anal Biochem. 2015;473:63–65.
12. Pijuan J, Barceló C, Moreno DF, et al. In vitro cell migration, invasion, and adhesion assays: From cell imaging to data analysis. Front Cell Dev Biol. 2019;7:107.
13. Della Mea V, Baroni GL, Pilutti D, Di Loreto C. SlideJ: An ImageJ plugin for automated processing of whole slide images. PLoS One. 2017;12:e0180540.
14. Young K, Morrison H. Quantifying microglia morphology from photomicrographs of immunohistochemistry prepared tissue using ImageJ. J. Vis. Exp. 2018;136:57648.
15. Preibisch S, Saalfeld S, Schindelin J, Tomancak P. Software for bead-based registration of selective plane illumination microscopy data. Nat Methods. 2010;7:418–419.
16. Preibisch S, Amat F, Stamataki E, et al. Efficient Bayesian-based multiview deconvolution. Nat Methods. 2014;11:645–648.
17. Madabhushi A, Lee G. Image analysis and machine learning in digital pathology: Challenges and opportunities. Med Image Anal. 2016;33:170–175.
18. Teigen LM, Kuchnia AJ, Nagel E, et al. Impact of software selection and ImageJ tutorial corrigendum on skeletal muscle measures at the third lumbar vertebra on computed Tomography scans in clinical populations. JPEN J Parenter Enteral Nutr. 2018;42:933–941.
19. Selvan AN, Cole LM, Spackman L, Naylor S, Wright C. Hierarchical cluster analysis to aid diagnostic image data visualization of MS and other medical imaging modalities. Methods Mol. Biol. 2017;1618:95–123.
20. Brookes SJ. Using ImageJ (Fiji) to analyze and present X-ray CT images of enamel. Methods Mol. Biol. 2019;1922:267–291.
21. Domínguez C, Heras J, Pascual V. IJ-OpenCV: Combining ImageJ and OpenCV for processing images in biomedicine. Comput Biol Med. 2017;84:189–194.
22. Meijering E, Cappellen G. van quantitative biological image analysis. In: Shorte SL, Frischknecht F, editors. Imaging cellular and molecular biological functions. Principles and practice. Berlin, Heidelberg: Springer, 2007; p. 45–70.
23. Schneider CA, Rasband WS, Eliceiri KW. NIH image to ImageJ: 25 years of image analysis. Nat Methods. 2012;9:671–675.

24. Schindelin J, Arganda-Carreras I, Frise E, et al. Fiji: An open-source platform for biological-image analysis. Nat Methods. 2012;9:676–682.
25. Anon Introduction to Fiji. Available from http://imagej.github.io/presentations/fiji-introduction/#/2
26. Pietzsch T, Preibisch S, Tomančák P, Saalfeld S. ImgLib2—Generic image processing in Java. Bioinformatics. 2012;28:3009–3011.
27. Rueden CT, Schindelin J, Hiner MC, et al. ImageJ2: ImageJ for the next generation of scientific image data. BMC Bioinf. 2017;18:529.
28. Schindelin J. The ImageJ ecosystem: An open platform for biomedical image analysis; 2015. Available from http://onlinelibrary.wiley.com/doi/10.1002/mrd.22489/full
29. Arena ET, Rueden CT, Hiner MC, Wang S, Yuan M, Eliceiri KW. Quantitating the cell: Turning images into numbers with ImageJ. WIREs Dev Biol. 2017;6:e260.
30. Pietzsch T, Saalfeld S, Preibisch S, Tomancak P. BigDataViewer: Visualization and processing for large image data sets. Nat Methods. 2015;12:481–483.
31. Hörl D, Rojas Rusak F, Preusser F, et al. BigStitcher: Reconstructing high-resolution image datasets of cleared and expanded samples. Nat Methods. 2019;16:870–874.
32. Bogovic JA Robust registration of calcium images by learned contrast synthesis. 2016. Available from https://ieeexplore.ieee.org/document/7493463
33. Linkert M, Rueden CT, Allan C, et al. Metadata matters: Access to image data in the real world. J Cell Biol. 2010;189:777–782.
34. Royer LA, Weigert M, Günther U, et al. ClearVolume: Open-source live 3D visualization for light-sheet microscopy. Nat Methods. 2015;12:480–481.
35. Weigert M, Schmidt U, Boothe T, et al. Content-aware image restoration: Pushing the limits of fluorescence microscopy. Nat Methods. 2018;15:1090–1097.
36. Gómez-de-Mariscal E, García-López-de-Haro C, Donati L, Unser M, Muñoz-Barrutia A, Sage D. DeepImageJ: A user-friendly plugin to run deep learning models in ImageJ. bioRxiv:799270; 2019.
37. Fernandez R, Moisy C. Fijiyama: A registration tool for 3D multimodal time-lapse imaging. Bioinformatics. 2020;btaa846. https://doi.org/10.1093/bioinformatics/btaa846
38. Gao D, Barber PR, Chacko JV, Sagar MAK, Rueden CT, Grislis AR, Hiner MC, Eliceiri KW (2020) FLIMJ: An open-source ImageJ toolkit for fluorescence lifetime image data analysis. bioRxiv:2020.08.17.253625.
39. Anon FLIMJ. ImageJ. Available from https://imagej.net/FLIMJ
40. Wolff C Multi-view light-sheet imaging and tracking with the MaMuT software reveals the cell lineage of a direct developing arthropod limb; 2018. Available from https://elifesciences.org/articles/34410
41. Anon mastodon-sc/mastodon. Mastodon Science; 2020. Available from https://github.com/mastodon-sc/mastodon
42. Hiner MC, Rueden CT, Eliceiri KW. SCIFIO: An extensible framework to support scientific image formats. BMC Bioinf. 2016;17:521.
43. Anon SciView. ImageJ. Available from https://imagej.net/SciView
44. Günther U, Harrington KIS (2020) Tales from the Trenches: Developing SciView, a new 3D viewer for the ImageJ community. ArXiv200411897 Cs. Available from http://arxiv.org/abs/2004.11897
45. Arganda-Carreras I, Kaynig V, Rueden C, et al. Trainable Weka segmentation: A machine learning tool for microscopy pixel classification. Bioinf Oxf Engl. 2017;33:2424–2426.
46. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: An update. ACM SIGKDD Explor Newslett. 2009;11:10–18.
47. Cardona A, Saalfeld S, Schindelin J, et al. TrakEM2 software for neural circuit reconstruction. PLoS One. 2012;7:e38011.
48. Martiel J-L, Leal A, Kurzawa L, et al. Measurement of cell traction forces with ImageJ. Methods Cell Biol. 2015;125:269–287.
49. Brazill JM, Zhu Y, Li C, Zhai RG. Quantitative cell biology of neurodegeneration in drosophila through unbiased analysis of fluorescently tagged proteins using ImageJ. J. Vis. Exp. 2018;138:58041.
50. Patel A, Li Z, Canete P, et al. AxonTracer: A novel ImageJ plugin for automated quantification of axon regeneration in spinal cord tissue. BMC Neurosci. 2018;19:8.
51. Hayes JA, Papagiakoumou E, Ruffault P-L, Emiliani V, Fortin G. Computer-aided neurophysiology and imaging with open-source PhysImage. J Neurophysiol. 2018;120:23–36.
52. Lormand C, Zellmer GF, Németh K, et al. Weka trainable segmentation plugin in ImageJ: A semi-automatic tool applied to crystal size distributions of Microlites in volcanic rocks. Microsc Microanal Off J Microsc Soc Am Microbeam Anal Soc Microsc Soc Can. 2018;24:667–675.
53. Collins KA, Kielkopf JF, Stassun KG, Hessman FV. ASTROIMAGEJ: Image processing and photometric extraction for ultra-precise astronomical light curves. Astron J. 2017;153:77.
54. Voras ZE (2017) Binding Medium Alteration and its Effect on Fine Art Paintings as Observed by Surface Analysis. Available from https://search.proquest.com/docview/1972679309/abstract/5F652F1A30D44387PQ/1
55. Kapitany K, Somogyi A, Barsi A. Inspection of a medieval WOOD sculpture using computer TOMOGRAPHY. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 2016;XLI-B5:287–291.
56. Williams C, Wu Y, Bowers DF. ImageJ analysis of dentin tubule distribution in human teeth. Tissue Cell. 2015;47:343–348.
57. Rueden CT, Ackerman J, Arena ET, et al. Scientific community image forum: A discussion forum for scientific image software. PLoS Biol. 2019;17:e3000340.
58. Anon How to contribute to an existing plugin or library. ImageJ. Available from https://imagej.net/How_to_contribute_to_an_existing_plugin_or_library
59. Abadi M. TensorFlow: Large-scale machine learning on heterogeneous distributed systems; 2016. Available from https://arxiv.org/abs/1603.04467
60. Anon CSBDeep. ImageJ. Available from https://imagej.net/CSBDeep
61. Anon CSBDeep. CSBDeep. Available from https://csbdeep.bioimagecomputing.com/tools
62. Čepa M. Segmentation of total cell area in brightfield microscopy images. Methods Protoc. 2018;1:43.

63. Miller KE, Liu X-A, Puthanveettil SV. Automated measurement of fast mitochondrial transport in neurons. Front Cell Neurosci. 2015;9:435.
64. Murtin C, Frindel C, Rousseau D, Ito K. Image processing for precise three-dimensional registration and stitching of thick high-resolution laser-scanning microscopy image stacks. Comput Biol Med. 2018;92:22–41.
65. Saalfeld S. Computational methods for stitching, alignment, and artifact correction of serial section data. Methods Cell Biol. 2019;152:261–276.
66. Anon SciJava. ImageJ. Available from https://imagej.net/SciJava
67. Chenouard N, Smal I, de Chaumont F, et al. Objective comparison of particle tracking methods. Nat Methods. 2014;11:281–289.
68. Meijering E. Cell segmentation: 50 years down the road [life sciences]. IEEE Signal Process Mag. 2012;29:140–145.
69. Jaqaman K, Loerke D, Mettlen M, et al. Robust single particle tracking in live cell time-lapse sequences. Nat Methods. 2008; 5:695–702.
70. Tinevez J-Y, Perry N, Schindelin J, et al. TrackMate: An open and extensible platform for single-particle tracking. Methods. 2017;115:80–90.
71. Cardona A TrakEM2 software for neural circuit reconstruction; 2012. Available from https://doi.org/10.1371/journal.pone.0038011
72. Anon Fijiyama. ImageJ. Available from https://imagej.net/Fijiyama
73. Schmid B A high-level 3D visualization API for Java and ImageJ. 2010. Available from http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-274
74. Anon Introduction into Macro Programming. ImageJ. Available from https://imagej.net/Introduction_into_Macro_Programming
75. Mutterer J, Rasband W. ImageJ Macro Language Programmer's Reference Guide v1.46d.:45.
76. Anon ImageJ. Available from https://imagej.nih.gov/ij/macros/
77. Anon Scripting. ImageJ. Available from https://imagej.net/Scripting
78. Anon BigStitcher. ImageJ. Available from https://imagej.net/BigStitcher
79. Anon The bh TCSPC Handbook 8th ed. Becker Hickl GmbH. Available from https://www.becker-hickl.com/literature/handbooks/the-bh-tcspc-handbook/
80. Berezin MY, Achilefu S. Fluorescence lifetime measurements and biological imaging. Chem Rev. 2010;110:2641–2684.
81. Lakowicz JR, Szmacinski H, Nowaczyk K, Johnson ML. Fluorescence lifetime imaging of free and protein-bound NADH. Proc Natl Acad Sci U S A. 1992;89:1271–1275.
82. Bird DK, Eliceiri KW, Fan C-H, White JG. Simultaneous two-photon spectral and lifetime fluorescence microscopy. Appl Optics. 2004;43:5173–5182.
83. Bi Q, Goodman KE, Kaminsky J, Lessler J. What is machine learning? A primer for the epidemiologist. Am J Epidemiol. 2019;188:2222–2239.
84. Ballard DH, Brown CM. Computer Vision. Englewood Cliffs, NJ: Prenice-Hall, 1982.
85. Yang SJ Assessing microscope image focus quality with deep learning; 2018. Available from https://doi.org/10.1186/s12859-018-2087-4
86. Anon OpenCV. ImageJ. Available from https://imagej.net/OpenCV
87. Anon imagej/imagej-opencv. ImageJ; 2020. Available from https://github.com/imagej/imagej-opencv
88. Zhang YC, Kagen AC. Machine learning Interface for medical image analysis. J Digit Imaging. 2017;30:615–621.
89. Ahn JM, Kim S, Ahn K-S, Cho S-H, Lee KB, Kim US. A deep learning model for the detection of both advanced and early glaucoma using fundus photography. PLoS One. 2018;13: e0207982.
90. Cai S, Tian Y, Lui H, Zeng H, Wu Y, Chen G. Dense-UNet: A novel multiphoton in vivo cellular image segmentation model based on a convolutional neural network. Quant Imaging Med Surg. 2020;10:1275–1285.
91. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015; 521:436–444.
92. Krull A, Buchholz T-O, Jug F. Noise2Void - learning denoising from single noisy images. Paper presented at: Conference on Computer Vision and Pattern Recognition (CVPR); 2019; pp. 2129–2137.
93. Maier A, Syben C, Lasser T, Riess C. A gentle introduction to deep learning in medical image processing. Z Für Med Phys. 2019;29:86–101.
94. Wang H, Rivenson Y, Jin Y, et al. Deep learning enables cross-modality super-resolution in fluorescence microscopy. Nat Methods. 2019;16:103–110.
95. Rivenson Y, Wang H, Wei Z, et al. Virtual histological staining of unlabelled tissue-autofluorescence images via deep learning. Nat Biomed Eng. 2019;3:466–477.
96. Falk T, Mai D, Bensch R, et al. Author correction: U-net: Deep learning for cell counting, detection, and morphometry. Nat Methods. 2019;16:351.
97. Buchholz T-O, Prakash M, Krull A, Jug F (2020) DenoiSeg: Joint denoising and segmentation. ArXiv200502987 Cs . Available from http://arxiv.org/abs/2005.02987
98. Schmidt U, Weigert M, Broaddus C, Myers G (2018) Cell detection with star-convex polygons. ArXiv180603535 Cs 11071:265–273.
99. McQuilken M, Mehta SB, Verma A, Harris G, Oldenbourg R, Gladfelter AS. Polarized fluorescence microscopy to study cytoskeleton assembly and organization in live cells. Curr Protoc Cell Biol. 2015;67:4.29.1–4.29.13.
100. Pitrone PG, Schindelin J, Stuyvenberg L, et al. OpenSPIM: An open-access light-sheet microscopy platform. Nat Methods. 2013;10:598–599.
101. Anon ImageJ Ops. ImageJ. Available from https://imagej.net/ImageJ_Ops

How to cite this article: Schroeder AB, Dobson ETA, Rueden CT, Tomancak P, Jug F, Eliceiri KW. The ImageJ ecosystem: Open-source software for image visualization, processing, and analysis. *Protein Science*. 2021;30:234–249. https://doi.org/10.1002/pro.3993