Fiji: an open-source platform for biological-image analysis

Johannes Schindelin[1,10], Ignacio Arganda-Carreras[2], Erwin Frise[3], Verena Kaynig[4], Mark Longair[5], Tobias Pietzsch[1], Stephan Preibisch[1,10], Curtis Rueden[6], Stephan Saalfeld[1], Benjamin Schmid[7,10], Jean-Yves Tinevez[8], Daniel James White[1], Volker Hartenstein[9], Kevin Eliceiri[6], Pavel Tomancak[1] & Albert Cardona[5,10]

Fiji is a distribution of the popular open-source software ImageJ focused on biological-image analysis. Fiji uses modern software engineering practices to combine powerful software libraries with a broad range of scripting languages to enable rapid prototyping of image-processing algorithms. Fiji facilitates the transformation of new algorithms into ImageJ plugins that can be shared with end users through an integrated update system. We propose Fiji as a platform for productive collaboration between computer science and biology research communities.

Much primary biological data are acquired as images. In recent years, with the adoption of automated microscopy technologies, the volume and complexity of image data has increased to the point that it is no longer feasible to extract information without using computers. Thus biologists increasingly rely on computer scientists to come up with new solutions and on software to apply those solutions. Starting with Alan Turing's seminal paper on morphogenesis[1], through the development of efficient algorithms for sequence searching[2], to computational whole-genome assembly[3], computer scientists have had a profound impact on biological research. In its simplest form, computerized analysis of images overcomes the limitations and bias of a human observer. More importantly, computers have been essential for processing the images generated during high-throughput microscopy of large numbers and varieties of biological samples under a variety of conditions[4–6]. Additionally, imaging of single intact small organisms is now feasible with high resolution in two dimensions, three dimensions and across time[7], resulting in massive image data sets that are well beyond the scale accessible by subjective observation[8,9]. The analysis of both types of image data requires computer-vision techniques to reconstruct image volumes from many overlapping parts (registration)[10,11], to search for biologically relevant features (segmentation)[12], to follow relevant objects across space and time (tracking)[13,14] and to compare specimens through mapping to anatomical or cellular digital atlases[15,16].

Algorithms to achieve these tasks have been developed for natural images such as an urban street view but must be adapted for biological-image data and transferred into a software platform that is accessible to biologists. Such software should provide intuitive and comprehensive mechanisms to apply an algorithm to biological data and visualize the results in a user-friendly fashion. At the same time, the sheer size of many image data sets demands that algorithms run fast. These needs are tackled by the emerging interdisciplinary field of bioimage informatics that applies computer-vision and other image-processing approaches to biological research questions[17,18].

There are several commercial (for example, Imaris, Volocity and Amira) and open-source (for instance,

[1]Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany. [2]Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. [3]Department of Genome Dynamics, Berkeley Drosophila Genome Project, Lawrence Berkeley National Laboratory, Berkeley, California, USA. [4]Department of Computer Science of the Swiss Federal Institute of Technology Zurich, Zurich, Switzerland. [5]Institute of Neuroinformatics of the University of Zurich and Swiss Federal Institute of Technology Zurich, Zurich, Switzerland. [6]Laboratory for Optical and Computational Instrumentation, University of Wisconsin at Madison, Madison, Wisconsin, USA. [7]Department of Neurobiology and Genetics, University of Wurzburg, Wurzburg, Germany. [8]Institut Pasteur, Imagopole, La plate-forme d'imagerie dynamique, Paris, France. [9]Department of Molecular, Cell and Developmental Biology, University of California, Los Angeles, California, USA. [10]Present addresses: Laboratory for Optical and Computational Instrumentation, University of Wisconsin at Madison, Madison, Wisconsin, USA (J.S.), Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany (B.S.) and Janelia Farm Research Campus, Howard Hughes Medical Institute, Ashburn, Virginia, USA (S.P. and A.C.). Correspondence should be addressed to P.T. (tomancak@mpi-cbg.de) or A.C. (acardona@ini.phys.ethz.ch).

PUBLISHED ONLINE 28 JUNE 2012; DOI:10.1038/NMETH.2019

ImageJ[19], CellProfiler[20], Vaa3D[21], BioImageXD, Icy[22], Konstanz Information Miner (KNIME)[23] and others) software platforms for the analysis of biological images. Commercial platforms often focus on ease of use and broad coverage of image-processing tasks, targeting relatively inexperienced users. Almost invariably the principal details of the image-processing algorithms are hidden, which is undesirable for use in biological research. Conversely, these details are transparent in open-source platforms such as ImageJ, whose long existence, wide adoption and extensible plugin architecture has made it a tool of choice for scientists from a broad range of disciplines. But ImageJ was developed primarily by biologists for biologists, and its architecture does not follow modern software-engineering principles. This makes the platform less attractive for computer scientists to use for delivering new solutions to biologists.

To address this deficiency in ImageJ, we started a new open-source software project Fiji (Fiji is just ImageJ) that updates the underlying architecture of ImageJ and allows researchers to focus on the process of developing innovative, cutting-edge solutions for biological-image analysis. Fiji introduces powerful software libraries for rapid transfer from new algorithms to practical image-analysis tools. Core algorithms available in Fiji can be exploited through a broad range of scripting languages that are familiar to bioinformaticians and simplify the prototyping of new bioimage solutions. Finally, Fiji provides a robust distribution system, which ensures that new algorithms reach its broad user base as soon as possible, initiating an iterative refinement based on communication between developers and users. In summary, Fiji is designed to serve as a software-engineering ecosystem in which computer science and biology research communities can collaborate to turn algorithms into usable programs for solving biological research questions.

How Fiji enhances ImageJ

ImageJ, created by Wayne Rasband at the US National Institutes of Health[19], provides easy installation on arbitrary platforms and a simple user interface. ImageJ is primarily targeted at researchers with minimal computer skills, but because ImageJ's functionality can be easily extended with plugins (software components that can be separately installed to add functionality), it has also been attractive to researchers with training in software development. Over the years an impressive assortment of ImageJ plugins has been developed, touching on most areas of biological-image analysis (http://imagej.nih.gov/ij/).

Fiji maintains compatibility with ImageJ and supplements it with additional core functionality. The Fiji project was created to support the installation and maintenance of one of the more complex ImageJ plugins, TrakEM2, which provides comprehensive solutions for management, registration, segmentation and annotation of large electron microscopy data sets[24]. TrakEM2 outgrew, in its complexity and software-infrastructure demands, the facilities offered by classical ImageJ. Subsequently, several other advanced plugins (4D Viewer[25], selective plane illumination microscopy (SPIM) registration[26], Trainable Segmentation and many more; **Supplementary Table 1**) came to rely on Fiji's modern software-engineering practices such as a software versioning system, inclusion of third-party libraries, automatic updates and straightforward compilation.

The combination of advanced image-analysis solutions and the simplicity and familiarity of ImageJ's user interface has attracted many users to the platform. Fiji is effectively an open-source distribution of ImageJ that includes a great variety of organized libraries, plugins relevant for biological research (**Supplementary Table 1**), scripting languages, extensive tutorials and documentation. The overproliferation and redundancy of plugins in ImageJ can make it difficult to identify solutions suitable for a particular biological problem. Fiji addresses this issue by offering a curated selection of plugins that are organized into categories in the plugin menu (**Supplementary Fig. 1**).

The Fiji project aims to provide useful functionality for a broad range of researchers, from programming-agnostic biologists to bioinformaticians and software engineers to professional computer science researchers (**Fig. 1a**). Fiji enhances ImageJ's ease of installation by bundling all required components into a self-contained package that will run on any computer platform. In addition, biologists skilled in programming can use the many familiar scripting languages available in Fiji to build custom image-processing pipelines. For professional software engineers, Fiji offers the ability to manage source-code, high-performance implementation of algorithms and simple worldwide deployment. Finally, Fiji could become useful to computer scientists as it bundles standard libraries, builds bridges to other platforms (for instance, Matlab through the Miji plugin) and provides facilities for rapid prototyping of data type–agnostic, generic algorithms.

Fiji's source code is hosted in a version controlled source code repository (Git) for people to download and tinker with, and compiles in one step (http://fiji.sc/cgi-bin/gitweb.cgi/). A special 'contrib' branch is accessible for anybody to publish their plugin,

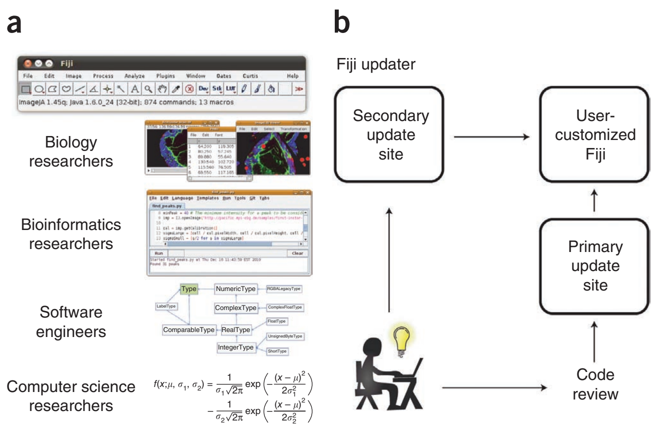

a
b
Fiji updater
Biology researchers
Bioinformatics researchers
Software engineers
Computer science researchers
Type
NumericType
ComplexType
RealType
IntegerType
ComparableType
$f(x;\mu,\sigma_1,\sigma_2)=\frac{1}{\sigma_1\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma_1^2}\right)-\frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma_2^2}\right)$
Secondary update site
User-customized Fiji
Primary update site
Code review

Figure 1 | Fiji as a high-powered distribution of ImageJ. (**a**) Screenshot of Fiji's main window, which is identical to that of ImageJ (top). Biology researchers can interact with multidimensional image data in Fiji's point-and-click interface, which is identical to that of ImageJ. Bioinformaticians can construct image-processing pipelines with scripting languages using the Script Editor plugin (screenshot is shown). Software engineers can use powerful software libraries such as ImgLib (schematic diagram of ImgLib design is shown) to transform mathematical formulations of computer science algorithms to functional programs (mathematical formulation of difference of Gaussian blob detector). (**b**) The Fiji updater workflow. Researchers who created a new plugin can contribute their code to the primary update server in Dresden, Germany, where their contribution will be commented on, enhanced and maintained for the long term by a group of core Fiji developers. Alternatively, anyone can offer their plugins directly through their own secondary update site. Users can freely customize their Fiji installation by choosing what to download.

library or script. Upon successful review, the code is included in the Fiji package system and immediately released to thousands of Fiji users worldwide via an integrated Fiji Updater (**Fig. 1b**). This mechanism provides a formalized way to release new image-analysis solutions to biological researchers while ensuring quality control of the contributed code and its long-term maintenance. Every time Fiji is launched it offers to download available plugin updates from the main server in Dresden, Germany or from alternate update sites. Research groups can also set up their own update sites and offer new plugins to the community by following simple instructions (http://fiji.sc/Adding_Update_Sites). Consequently, users can select which set of plugins they obtain from the various update sites, forming their own customized Fiji installation that suits their image-analysis needs (**Fig. 1b**).

The Fiji project emphasizes communication among users and developers. Fiji plugins are extensively documented via an active wiki, which is the definitive guide to all aspects of Fiji's installation, maintenance, programming and usage (http://fiji.sc/). Feedback and feature requests are posted to a dedicated mailing list. Fiji developers engage in active discussions online, and the core Fiji developers meet regularly for coding sprees called ′hackathons′ at which they work collaboratively on the Fiji source code. These events typically result in dramatic improvement and extension of the Fiji code base, the ′hackathon effect′ (**Supplementary Video 1**).

In short, Fiji focuses on the process of making new solutions practical and bringing them immediately to the users. Next we discuss how scripting languages in Fiji empower biologists to do complex image analysis by themselves and how the Fiji library ImgLib simplifies transfer of abstract algorithms into usable programs.

Scripting and ImgLib for implementing algorithms

Analysis of biological image data typically requires applying multiple algorithms in sequence to many images. Scripting uses a series of simple programming commands (the 'script') to define a sequence of algorithmic operations and then apply them to an image collection.

The simple macro language that allows recording of commands and construction of basic programs made scripting popular among ImageJ users. Fiji builds on that functionality by supporting a broad range of scripting languages (Jython, Clojure, Javascript, JRuby and Beanshell), providing the most comprehensive selection of programming tools on both open-source and commercial bioimaging platforms. These languages can be used without knowledge of Java, and compared to the macro language offer more advanced programming syntax while retaining relative simplicity for the casual programmer.

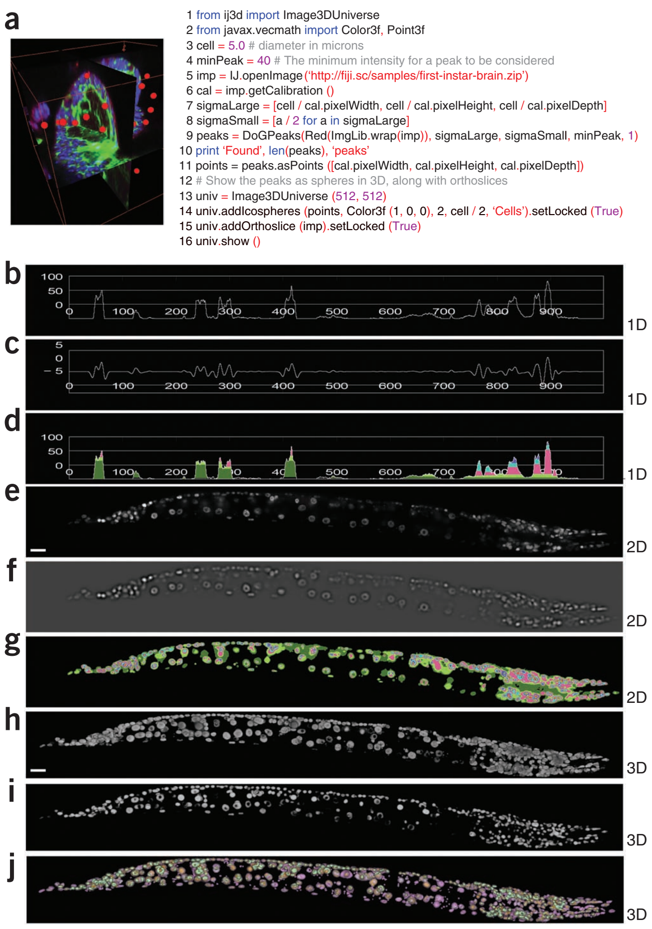
In Fiji, individual scripting commands can be quickly tested by applying them interactively to opened images using the appropriate interpreter plugin (for example, Jython Interpreter). Beyond interactive execution of commands on current images, Fiji also provides a script editor that enables writing, debugging, testing and running of arbitrarily complex scripts in all the above languages and additionally Java itself. With the script editor, users can perform complex tasks with a few lines of code, requiring minimal programming knowledge. It becomes relatively simple (16 lines of code) to load a multichannel three-dimensional (3D) image stack, identify all cells in one channel and visualize the results in the 4D Viewer plugin (**Fig. 2a**). Similarly users can implement a simple task such as swapping fluorescence channels in multidimensional images and apply it to a large collection of images in a directory using scripting commands for file manipulation (**Supplementary Fig. 2**). The details of the code are unimportant (extensive tutorials are available at http://fiji.sc/wiki/index.php/Category:Scripting), but it is crucial to note that all the scripting languages enable users to access Fiji's extensive algorithm libraries that implement advanced image analysis techniques in Java. Thus researchers do not need to become fluent in Java programming and can use their existing scripting skills to apply complex algorithms to their data. Scripts are seamlessly included in Fiji's menu structure and can be publicly distributed through the Fiji updater. These publishing mechanisms ensure validity and long-term viability, and are more convenient for biologists compared to closed scripting environments such as Matlab.

A key feature of the Fiji project is ImgLib (**Fig. 2b–j**)[27], a library for type-, dimension and storage-independent representation of image data that enables writing generic algorithms in a high-performance manner. In ImgLib, image-processing algorithms are implemented just once and can be applied without change to most kinds of images. In other words, ImgLib enables users to apply a complex image-processing algorithm to any image regardless of how many dimensions the image has (one dimension, two dimensions, three dimensions, two dimensions plus time, three dimensions plus time and so on), what the underlying data type is (8-bit, 16-bit and so on) or how the image is stored (in memory, paged to disc or distributed over the net).

We demonstrate the dimensionality independence of ImgLib when applying segmentation algorithms to confocal scans of *Caenorhabditis elegans* larvae expressing fluorescent markers of cell nuclei (**Fig. 2b–j**). Difference of Gaussian (DoG) and maximally stable extremal regions (MSER)[28] are two examples of blob-detection algorithms inspired by computer-vision literature that are suitable for detecting cells in biological images. The DoG is computed by subtracting two consecutive Gaussian convolutions of the image. Intensity maxima and minima in the DoG map represent bright and dark blob detections, respectively. The MSER tree provides a nested hierarchy of blob-segmentation hypotheses, which is particularly relevant in a densely packed cellular field. Both algorithms are applicable for processing of 1D curves, 2D image slices and 3D image volumes (**Fig. 2b–j**). With ImgLib a single, simple piece of computer code (**Supplementary Fig. 3**) can run the algorithms on images of arbitrary dimensions without any modification (**Fig. 2b–j**).

ImgLib is and should be invisible to casual users. It is an 'under the hood' advanced software engineering concept that makes programming easier. ImgLib empowers users and computer science researchers with the abstraction necessary to seamlessly translate the mathematical description of an advanced idea or algorithm into concise code that will run efficiently on large bioimages. Similar libraries that separate the algorithm essence from the actual implementation are available on other platforms such as Insight Segmentation and Registration Toolkit (ITK)[29] and Vision with Generic Algorithms (VIGRA)[30].

Both examples of scripts mentioned above use ImgLib to do the processing steered by simple scripting commands. Equivalent programs in ImageJ's macro language or Java would be much more complex or not possible. This is due to the combination of scripting-language simplicity and the dimension-, type- and storage strategy–independence of ImgLib. Next we showcase how



```
1 from ij3d import Image3DUniverse
2 from javax.vecmath import Color3f, Point3f
3 cell = 5.0 # diameter in microns
4 minPeak = 40 # The minimum intensity for a peak to be considered
5 imp = IJ.openImage('http://fiji.sc/samples/first-instar-brain.zip')
6 cal = imp.getCalibration ()
7 sigmaLarge = [cell / cal.pixelWidth, cell / cal.pixelHeight, cell / cal.pixelDepth]
8 sigmaSmall = [a / 2 for a in sigmaLarge]
9 peaks = DoGPeaks(Red(ImgLib.wrap(imp)), sigmaLarge, sigmaSmall, minPeak, 1)
10 print 'Found', len(peaks), 'peaks'
11 points = peaks.asPoints ([cal.pixelWidth, cal.pixelHeight, cal.pixelDepth])
12 # Show the peaks as spheres in 3D, along with orthoslices
13 univ = Image3DUniverse (512, 512)
14 univ.addIcospheres (points, Color3f (1, 0, 0), 2, cell / 2, 'Cells').setLocked (True)
15 univ.addOrthoslice (imp).setLocked (True)
16 univ.show ()
```

Figure 2 | Scripting and ImgLib. (**a**) Screenshot of 4D Viewer plugin with orthogonal view of the 3D image of *Drosophila melanogaster* first instar brain (left), in which cortex and neuropile glia are green, labeled by Nirvana-Gal4 and UASmcd8GFP, surface glial cells are red, labeled with anti-repo antibody and all nuclei are blue, labeled with Sytox. Red spheres mark the surface glial cells detected using a simple Jython script shown on the right. The script opens a 3D RGB image (line 5) and automatically counts red surface glial cells using the DoG detector (line 9), applying constraints for cell size and labeling intensity (lines 3 and 4). These constraints are expressed as DoG sigma parameters (lines 7 and 8) by extracting image dimensions from metadata (line 6). Cell count is printed in the dialog box (line 10), and cells are subsequently displayed in the 4D Viewer as red spheres of fixed diameter (lines 13–16). (**b**–**j**) Output of ImgLib algorithms MSER (**d**,**g**,**j**) and DoG (**c**,**f**,**i**) for one (**c**,**d**), two (**f**,**g**) and three (**i**,**j**) dimensions. The 3D input image was a confocal stack of *C. elegans* expressing a nuclear marker (**h**); a slice from the stack was used as the 2D input image (**e**) (scale bar, 10 μm), and a line segment from the slice was used as the 1D input image (**b**). MSER and DoG were run on all input images without changing the code. Nested MSER regions representing competing segmentation hypotheses for the nuclei are colored green, red, blue and magenta.

the synergy of Fiji, ImgLib and other libraries results in transformation of abstract algorithms into usable applications for the analysis of biological images.

Advanced image-processing pipelines in Fiji

The growing collection of modern image analysis algorithms in Fiji is a product of interactions between Fiji projects that use computer-vision algorithms to support ongoing biological research. We briefly outline three advanced image-processing pipelines for stitching multidimensional images from tiles (**Fig. 3a**–**e**), reconstruction of massive serial section transmission electron microscopy (ssTEM) mosaics (**Fig. 3f**–**j**) and reconstruction of long-term, time-lapse, multiview recordings of development with selective-plane illumination microscopy (SPIM) (**Fig. 3k**–**o**). Many more examples of projects and synergies can be found on the Fiji wiki (http://fiji.sc/), and we invite biologists with programming skills, professional developers with interest in biology and computer-vision researchers with a new algorithm looking for applications to join the Fiji movement and contribute new approaches, ideas and plugins.

Stitching of confocal stacks. Large biological samples are frequently imaged with high-resolution microscopy techniques as sets of overlapping images or image stacks (**Fig. 3a**). The reconstruction of the entire specimen involves pairwise registration of the overlapping images, with a strategy to minimize the displacement of corresponding image content. The Stitching plugin uses phase correlation to compute the optimal translation among pairs of overlapping multidimensional image tiles (**Fig. 3b**). Subsequently, the plugin uses a global optimization procedure that evenly spreads the error of the registration steps, avoiding the introduction of artifactual deformations and delivering an undistorted representation of the specimen[31]. The reconstructed volume is visualized in the 4D Viewer[25] (**Fig. 3c**), structures are segmented using manual segmentation tools such as Segmentation Editor (**Fig. 3d**), allowing measurements of lengths, areas, volumes of segmented elements (**Fig. 3e**) and quantification of their overlap by colocalization analysis.

ssTEM reconstruction. ssTEM is experiencing a renaissance as a tool of choice for mapping brain micro-circuitry[32,33]. Large sections are typically imaged as sets of overlapping image tiles (**Fig. 3f**) and data sets consisting of hundreds or even thousands of such sections are becoming available. Fiji users can first apply a lens deformation model to pre-process the ssTEM images before registration using the Distortion Correction plugin[34]. Relatively simple ssTEM data sets can be reconstructed using standalone plugins such as Register Virtual Stack Slices, Elastic Alignment and Montage[35] or bUnwarpJ[36]. Arbitrarily large ssTEM data sets consisting of tens of thousands images are best assembled using the integrative TrakEM2 plugin[24] for registration, segmentation and analysis of electron-microscopy data. One of the many avenues for reconstructing ssTEM data in TrakEM2 relies on the scale-invariant feature transform (SIFT)[37] to detect corresponding image content and then solves the globally optimal configuration of massive tile systems[11] (**Fig. 3g**,**h** and **Supplementary Video 2**).

Reconstructed volumes can be segmented into a series of profiles following individual neurons either using the powerful manual segmentation tools (**Fig. 3i**) or automatically with Active Contours[38] and Level Sets[39] algorithms integrated into TrakEM2. Alternatively, to extract the neuronal tracks from electron microscopy data, users can employ the Trainable Segmentation plugin, which leverages the Waikato Environment for Knowledge Analysis (WEKA) library for machine learning[40] to extract statistical properties from user-provided examples and apply them to classify the rest of the image or other similar images[41]. The analysis of the reconstructed ssTEM scans of parts of the nervous system

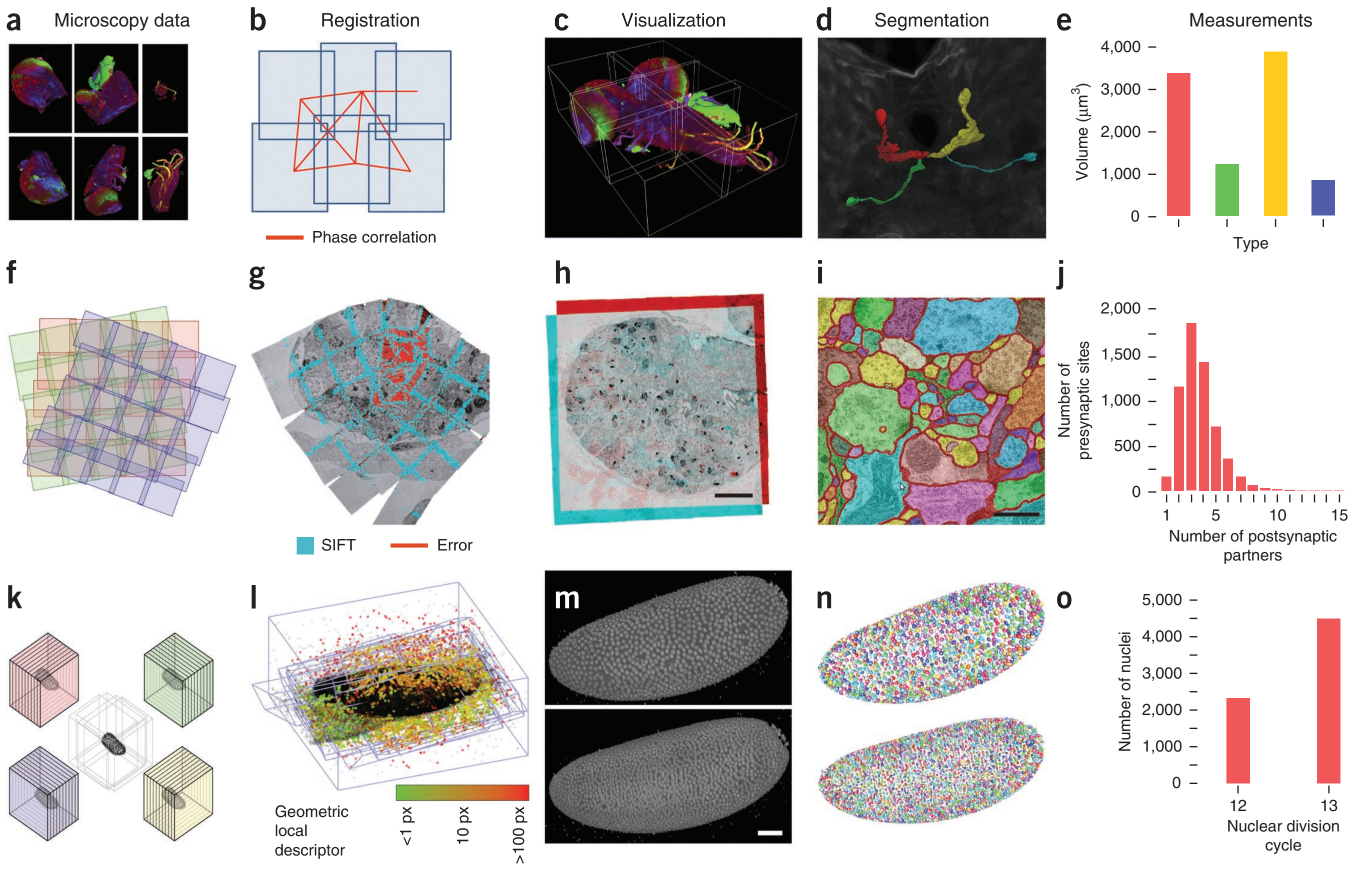
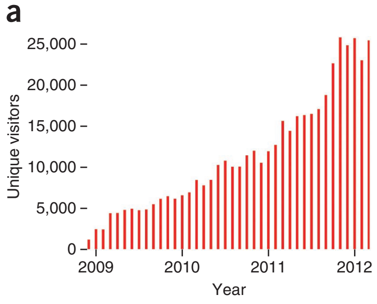


Figure 3 | Fiji projects. (**a**–**e**) Stitching plugin for globally optimal registration of tiled 3D confocal images. The 3D confocal stacks of *D. melanogaster* first instar larval nervous system (**a**) were registered using phase correlation with global optimization (**b**) and visualized in the Fiji 4D Viewer (**c**). Four labeled neurons (color coded in 4D Viewer (**d**)) were segmented using a manual segmentation plugin (segmentation editor) and their volumes were measured (**e**). (**f**–**j**) Globally optimal reconstruction of large ssTEM mosaics using TrakEM2 plugin. Schematic of the ssTEM mosaic; each square is an individual image tile, and independent sections are color-coded (**f**). Screenshot of a video visualizing the progress of global optimization for a single section. SIFT features (SIFT), and residual error signifying displacement of corresponding SIFT features at the current iteration (error) are shown (**g**). Dual-color overlay of two registered consecutive sections showing the entire hemisphere of the larval brain (**h**). Scale bar, 10 µm. Axonal profiles in a small part of a single section in the ssTEM data set were manually segmented using TrakEM2. Each profile is labeled with a different color (**i**). Scale bar, 0.5 µm. Relationship between numbers of presynaptic partners and postsynaptic sites extracted manually with TrakEM2 from a micro-cube of the registered data (**j**). (**k**–**o**) Plugin suite for processing of multiview SPIM data. Schematic representation of multiview (4D) SPIM imaging showing 3D stacks of the same specimen acquired from different angles (**k**). Progress of the global optimization of multiview SPIM acquisition of a *D. melanogaster* embryo (**l**). Corresponding geometric bead descriptors are colored according to their residual displacement at the current iteration of the optimizer. The resulting reconstructed embryo at the 12th (top) and 13th (bottom) nuclear division cycle shown as a 3D rendering in Fiji's 3D viewer (**m**). Scale bar, 50 µm. Results of the DoG segmentation of the nuclei marked with His-YFP (**n**); same stages as in **m**. Each nucleus is marked with a different color. Quantification of the nuclear counts at the 12th and 13th nuclear division in the embryo shown in **m** and **n** (**o**).

requires quantitative comparisons across different samples[42] such as counting individual synaptic connections between segmented neurons (**Fig. 3j**). Ultimately high-resolution TEM scans need to be registered to low-resolution light microscopy data for correlative analysis and tools such as the Simple Neurite Tracer plugin[43], the Virtual Insect Brain (VIB) Protocol and Interactive Image Transformations using various landmark correpondences are ready to be adapted for this purpose. The flexibility and integrative nature of TrakEM2 with its unique ability to reconstruct very large electron microscopy data sets on commodity hardware cannot be found on any other bioimaging platform to our knowledge.

Processing of multiview SPIM data. Modern developmental biologists want to image developing embryos *in toto* with resolution sufficient to distinguish and track individual labeled cells throughout development. SPIM is an emerging technique to achieve these goals, and Fiji provides a comprehensive pipeline for acquisition, reconstruction, segmentation, tracking and analysis of terabyte-size long-term, time-lapse, multiview SPIM data sets.

For image acquisition with SPIM, Fiji synergizes with another large ImageJ offshoot project, Micro-Manager[44], to provide open-source steering software for custom OpenSPIM microscopes (P.T.; unpublished results) to deliver 3D stacks of the same specimen imaged from different angles directly into the Fiji environment (**Fig. 3k**). After acquisition, the multiview data can be reconstructed into a continuous 3D volume using fluorescent beads in the rigid agarose medium as fiduciary markers (bead-based SPIM Registration plugin[26]). The constellations of fluorescent beads form local geometric descriptors that are matched across views and used to create a 3D image by globally optimizing



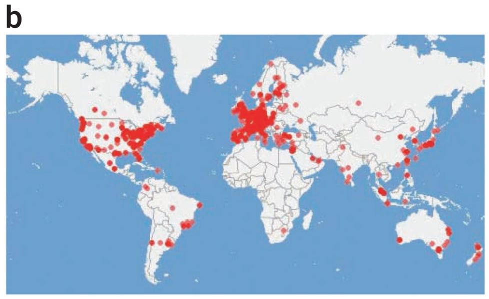


Figure 4 | Fiji usage. (**a**) A chart showing the number of unique visitors to the Fiji wiki per month over the last three years. (**b**) World map with locations of computers that updated Fiji between 20 March 2012 and 27 March 2012.

local affine transformation models (**Fig. 3l** and **Supplementary Video 3**). The reconstructed images can be optionally fused into a single output volume (**Fig. 3m**) using the 'multiview fusion' plugin. Interactive DoG segmentation that is part of the bead-based SPIM registration plugin can be used to segment and count the nuclei in the reconstructed specimen (**Fig. 3n**). The registration process, the segmentation results (**Fig. 3o**) and the tracking of nuclei across time (**Supplementary Video 4**) can be visualized using the Fiji 4D Viewer[25]. All this is happening with truly massive long-term, time-lapse, multiview SPIM data sets in the terabyte range and to our knowledge makes Fiji currently the only widely available solution for dealing with this type of data.

Fiji-plugin integration. The various Fiji plugins interact on two principal levels. Trivially, the output of one plugin can serve as input for another because they all run in a common platform. More elaborate integration is possible because most Fiji methods are increasingly implemented as software libraries using ImgLib for data representation. For instance, the confocal stack, ssTEM mosaic and multiview SPIM registration plugins are using a common global optimizer capable of accepting different representations of image content (phase correlation, SIFT and local geometric descriptors) and minimizing the displacement of correspondences using the transformation model appropriate for the given imaging modality (translation only, rigid and affine). The integration of standalone plugins into tightly cooperating suites such as TrakEM2 is an ongoing trend in all Fiji projects. Casual users benefit from such integration by being able to seamlessly combine solutions from diverse areas of bioimage analysis and create efficient analysis pipelines to interpret their images.

Conclusion

ImageJ has for a long time been the tool of choice for biologists who need basic as well as advanced image analysis. Over the lifetime of ImageJ, a revolution in microscopy brought about an order of magnitude increase in typical image sizes and two or more orders of magnitude increase in throughput. The Fiji project provides biologists with powerful tools for generation of advanced image-processing pipelines, via scripting languages and feature-rich libraries for processing large quantities of large images, while building on the simplicity of ImageJ.

Over the past two years the Fiji project has achieved remarkable international recognition (**Fig. 4a**). Analysis of Fiji updater records reveals that Fiji is used in every major academic research center throughout the world (**Fig. 4b**). The number of Fiji downloads and the access to the Fiji wiki (**Fig. 4a**) has grown steadily over the past years currently with 20,000 estimated users. Several prominent scientific institutions use Fiji as an open-source solution for image-processing needs of their researchers. As the community grows, new projects and new talented researchers join, expanding the ability of biologists to cope with image data.

Many image-analysis solutions available for Fiji have been developed uniquely under Fiji and do not have a comparable alternative in other platforms. Some of the standard image-processing or visualization tasks implemented in Fiji plugins are addressed, sometimes more comprehensively, by other platforms (for instance, visualization in three dimensions (Vaa3D[21]) or trainable segmentation (CellProfiler[20])). However, in Fiji, these tools are embedded in a large ecosystem of plugins that can interact through a common infrastructure, for example, by assembling diverse image-processing steps into complex pipelines with the use of scripting languages.

The reliance on Java invites the question of whether Fiji can deal efficiently with computing-intensive tasks on top of large biological-image data. It is true that careful algorithm implementations in the C/C++ programming language will sometimes execute faster than the equivalent Java program. In other cases, Java outperforms C/C++ (for example, http://fiji.sc/Why_Java). It is equally true that specialized implementations will typically outperform generic ones. Many of the flagship Fiji projects described above deal with massive terabyte-sized data sets and yet deliver interactive performance on standard hardware. Important advantages of Java are the ease of installation on a variety of platforms, portability, stability and backwards compatibility that are harder to achieve with packages built with C. Thus, Fiji strikes a balance between usability, performance and flexibility, and as such it occupies a unique position in the landscape of biological-image analysis where it synergistically complements other tools as well as provides unique and new capabilities.

Fiji is open not only with respect to its source code but also in its relationship with other platforms. It aims to communicate and integrate with other bioimage analysis software that outperform Fiji on a particular task. An example of such integration is the interface to Matlab and ITK, the availability of ImgLib for adoption in any Java-based platform (for instance, KNIME and ImageJ2) and the participation of developers from diverse platforms in Fiji-centered hackathons (representatives of ImageJ2, CellProfiler, Micro-Manager, ITK, KNIME, Icy, BioImageXD and OMERO attended the last two hackathons). Most importantly, in the future, Fiji will reconnect to its roots by becoming the outer, application-oriented layer of the ImageJ2 (http://developer.imagej.net/) project, the aim of which is to redesign the very core of ImageJ to meet the demands of next-generation biology research. Fiji will focus on developing state-of-the-art plugins inspired by the needs of biological research projects, and ImageJ2 will put an emphasis on creating the infrastructure necessary to prototype, realize, distribute and deploy the new algorithm solutions in a sustainable manner. The two projects have a common starting point, the classical ImageJ environment, and are committed to work together, exchanging code and coordinating design decisions. The process of creating a powerful synergy between Fiji and ImageJ2 has already begun by incorporating the flagship Fiji plugins and libraries (such as the Fiji Updater and ImgLib) into ImageJ2, forming a basis of a long-term partnership that will combine professional software engineering of the ImageJ2 core and biological application–driven plugin development in Fiji.

Note: Supplementary information is available in the online version of the paper.

ACKNOWLEDGMENTS
We thank W. Rasband for developing ImageJ and helping thousands of scientists, those who contributed to the Fiji movement by financing and organizing the hackathons, namely G.M. Rubin for hackathons at Janelia Farm, I. Baines for hackathons at Max Planck Institute of Molecular Cell Biology and Genetics in Dresden, R. Douglas for a hackathon at Institute of Neuroinformatics in Zurich, F. Peri and K. Miura for the hackathon at European Molecular Biology Laboratory, and International Neuroinformatics Coordinating Facility for Fiji image-processing school, W. Pereanu for the confocal image of the larval fly brain, M. Sarov for the confocal scan of *C. elegans* larva, the scientists who released their code under open-source licenses and made the Fiji project possible. We want to thank Carl Zeiss Microimaging for access to the SPIM demonstrator. K.E. and C.R. were supported by US National Institutes of Health grant RC2GM092519. J.S. and P.T. were funded by Human Frontier Science Program Young Investigator grant RGY0083. P.T. was supported by The European Research Council Community's Seventh Framework Programme (FP7/2007-2013) grant agreement 260746.

COMPETING FINANCIAL INTERESTS
The authors declare no competing financial interests.

Published online at http://www.nature.com/doifinder/10.1038/nmeth.2019. Reprints and permissions information is available online at http://www.nature.com/reprints/index.html.

1. Turing, A.M. The chemical basis of morphogenesis. 1953. *Bull. Math. Biol.* **52**, 153–197, discussion 119–152 (1990).
2. Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990).
3. Myers, E.W. *et al.* A whole-genome assembly of *Drosophila*. *Science* **287**, 2196–2204 (2000).
4. Neumann, B. *et al.* Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature* **464**, 721–727 (2010).
5. Collinet, C. *et al.* Systems survey of endocytosis by multiparametric image analysis. *Nature* **464**, 243–249 (2010).
6. Shariff, A., Kangas, J., Coelho, L.P., Quinn, S. & Murphy, R.F. Automated image analysis for high-content screening and analysis. *J. Biomol. Screen.* **15**, 726–734 (2010).
7. Megason, S.G. & Fraser, S.E. Imaging in systems biology. *Cell* **130**, 784–795 (2007).
8. Keller, P.J., Schmidt, A.D., Wittbrodt, J. & Stelzer, E.H. Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy. *Science* **322**, 1065–1069 (2008).
9. Fowlkes, C.C. *et al.* A quantitative spatiotemporal atlas of gene expression in the *Drosophila* blastoderm. *Cell* **133**, 364–374 (2008).
10. Anderson, J.R. *et al.* A computational framework for ultrastructural mapping of neural circuitry. *PLoS Biol.* **7**, e1000074 (2009).
11. Saalfeld, S., Cardona, A., Hartenstein, V. & Tomancak, P. As-rigid-as-possible mosaicking and serial section registration of large ssTEM datasets. *Bioinformatics* **26**, i57–i63 (2010).
12. Murray, J.I. *et al.* Automated analysis of embryonic gene expression with cellular resolution in *C. elegans*. *Nat. Methods* **5**, 703–709 (2008).
13. Fernandez, R. *et al.* Imaging plant growth in 4D: robust tissue reconstruction and lineaging at cell resolution. *Nat. Methods* **7**, 547–553 (2010).
14. Bao, Z. *et al.* Automated cell lineage tracing in *Caenorhabditis elegans*. *Proc. Natl. Acad. Sci. USA* **103**, 2707–2712 (2006).
15. Long, F., Peng, H., Liu, X., Kim, S.K. & Myers, E. A 3D digital atlas of *C. elegans* and its application to single-cell analyses. *Nat. Methods* **6**, 667–672 (2009).
16. Peng, H. *et al.* BrainAligner: 3D registration atlases of *Drosophila* brains. *Nat. Methods* **8**, 493–500 (2011).
17. Swedlow, J.R. & Eliceiri, K.W. Open source bioimage informatics for cell biology. *Trends Cell Biol.* **19**, 656–660 (2009).
18. Peng, H. Bioimage informatics: a new area of engineering biology. *Bioinformatics* **24**, 1827–1836 (2008).
19. Abramoff, M.D., Magalhaes, P.J. & Ram, S.J. Image processing with ImageJ. *Biophotonics International* **11**, 36–42 (2004).
20. Carpenter, A.E. *et al.* CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.* **7**, R100 (2006).
21. Peng, H., Ruan, Z., Long, F., Simpson, J.H. & Myers, E.W. V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nat. Biotechnol.* **28**, 348–353 (2010).
22. de Chaumont, F., Dallongeville, S. & Olivo-Marin, J.-C. ICY: a new open-source community image processing software in. *IEEE Int. Symp. on Biomedical Imaging* 234–237 (2011).
23. Berthold, M.R. *et al.* KNIME: the Konstanz Information Miner. in *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)* 319–326 (Springer, 2007).
24. Cardona, A. *et al.* TrakEM2 software for neural circuit reconstruction. *PLoS One* (in the press).
25. Schmid, B., Schindelin, J., Cardona, A., Longair, M. & Heisenberg, M. A high-level 3D visualization API for Java and ImageJ. *BMC Bioinformatics* **11**, 274 (2010).
26. Preibisch, S., Saalfeld, S., Schindelin, J. & Tomancak, P. Software for bead-based registration of selective plane illumination microscopy data. *Nat. Methods* **7**, 418–419 (2010).
27. Preibisch, S., Tomancak, P. & Saalfeld, S. in *Proc. ImageJ User and Developer Conf.* **1**, 72–76 (2010).
28. Matas, J., Chum, O., Urban, M. & Pajdla, T. Robust wide baseline stereo from maximally stable extremal regions. *Image Vis. Comput.* **22**, 761–767 (2004).
29. Ibanez, L., Schroeder, W., Ng, L. & Cates, J. *The ITK Software Guide* (Kitware Inc., 2005).
30. Köthe, U. Reusable software in computer vision. in *Handbook of Computer Vision and Applications* Vol. 3 (eds. Jähne, B., Haussecker, H. & Geissler P.) 103–132 (San Diego: Academic Press, 1999).
31. Preibisch, S., Saalfeld, S. & Tomancak, P. Globally optimal stitching of tiled 3D microscopic image acquisitions. *Bioinformatics* **25**, 1463–1465 (2009).
32. Cardona, A. *et al.* An integrated micro- and macroarchitectural analysis of the *Drosophila* brain by computer-assisted serial section electron microscopy. *PLoS Biol.* **8**, e1000502 (2010).
33. Bock, D.D. *et al.* Network anatomy and *in vivo* physiology of visual cortical neurons. *Nature* **471**, 177–182 (2011).
34. Kaynig, V., Fischer, B., Müller, E. & Buhmann, J.M. Fully automatic stitching and distortion correction of transmission electron microscope images. *J. Struct. Biol.* **171**, 163–173 (2010).
35. Saalfeld, S., Fetter, R., Cardona, A. & Tomancak, P. Elastic volume reconstruction from series of ultrathin microscopy sections. *Nature Methods* advance online publication, doi:10.1038/nmeth.2072 (10 June 2012).
36. Arganda-Carreras, I. *et al.* Consistent and elastic registration of histological sections using vector-spline regularization. in *Lecture Notes in Computer Science* **4241**, 85–95 (Springer, 2006).
37. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**, 91–110 (2004).
38. Sethian, J.A. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science* (Cambridge University Press, 1999).
39. Kass, M., Witkin, A. & Terzopoulos, D. Snakes: active contour models. *Int. J. Comput. Vis.* **1**, 321–331 (1988).
40. Hall, M. *et al.* The WEKA data mining software: an update. *SIGKDD Explor.* **11**, 10–18 (2009).
41. Kaynig, V., Fuchs, T.J. & Buhmann, J.M. Geometrical consistent 3D tracing of neuronal processes in ssTEM data. *Med. Image. Comput. Comput. Assist. Interv.* **13**, 209–216 (2010).
42. Cardona, A. *et al.* Identifying neuronal lineages of *Drosophila* by sequence analysis of axon tracts. *J. Neurosci.* **30**, 7538–7553 (2010).
43. Longair, M.H., Baker, D.A. & Armstrong, J.D. Simple Neurite Tracer: open source software for reconstruction, visualization and analysis of neuronal processes. *Bioinformatics* **27**, 2453–2454 (2011).
44. Edelstein, A., Amodaj, N., Hoover, K., Vale, R. & Stuurman, N. Computer control of microscopes using μManager. in *Current Protocols in Molecular Biology* (John Wiley & Sons, Inc., 2010).