ImageJ and Fiji

J. Goldstein et al., *Scanning Electron Microscopy and X-Ray Microanalysis*,
https://doi.org/10.1007/978-1-4939-6676-9_13

Software is an essential tool for the scanning electron microscopist and X-ray microanalyst (SEMXM). In the past, software was an important optional means of augmenting the electron microscope and X-ray spectrometer, permitting powerful additional analysis and enabling new characterization methods that were not possible with bare instrumentation. Today, however, it is simply not possible to function as an SEMXM practitioner without using at least a minimal amount of software. A graphical user interface (GUI) is an integral part of how the operator controls the hardware on most modern microscopes, and in some cases it is the only interface. Even many seemingly analog controls such as focus knobs, magnification knobs, or stigmators are actually digital interfaces mounted on hand-panel controllers that connect to the microscope control computer via a USB interface.

In addition to its role in data acquisition, software is now indispensable in the processing, exploration, and visualization of SEMXM data and analysis results. Fortunately, most manufacturers provide high-quality commercial software packages to support the hardware they sell and to aid the analyst in the most common materials characterization tasks. Usually this software has been carefully engineered, often at great cost, and smart analysts will take advantage of this software whenever it meets their needs. However, closed-source commercial software suffers from several limitations. Because the source code is not available for inspection, the procedures and algorithms used by the software cannot be checked for accuracy or completeness, and must be accepted as a "black box." Further, it is often very difficult to modify closed source software, either to add missing features needed by the analyst or to customize the workflow to meet specific job requirements. In this regard, open source software is more flexible and more extensible. The cost of commercial software packages can also be a downside, especially in an academic or teaching environment or in any situation where many duplicate copies of the software are required. Clearly a no-cost, open source solution is preferable to a high-cost commercial application if you need to install 50 copies for instructional purposes.

One of the most popular free and open source software packages for SEM image analysis is ImageJ, a Java program that has grown over the decades from a small application started at the National Institutes of Health (NIH) into a large international collaboration with hundreds of contributors and many, many thousands of users (▶ http://imagej.net).

13.1 The ImageJ Universe

ImageJ has grown into a large and multifaceted suite of related tools, and how all these parts fit together (and which are useful for SEM and X-ray microanalysis) may not be immediately obvious. The project began in the late 1970s when Wayne Rasband, working at NIH, authored a simple image processing program in the Pascal programming language that he called *Image*. This original application ran only on the PDP-11, but in 1987 when the Apple Macintosh II was becoming popular, Rasband undertook the development of a Mac version of the tool called *NIH Image*. Largely to enable cross-platform compatibility and to allow non-Macintosh users to run the program, it was again rewritten, this time using the Java programming language. The result was the first version of *ImageJ* in 1997 (Schneider et al. 2012, 2015).

The availability of ImageJ on the Microsoft PC and Unix platforms as well as Macintosh undoubtedly added to its popularity, but just as important was the decision to create an open software architecture that encouraged contributions from a large community of interested software developers. As a result, ImageJ benefitted from a prodigious number of code submissions in the form of macros and plugins as well as edits to the core application itself. Partly to manage this organic growth of the package, partly to reorganize the code base, and in part to introduce improvements that could not be added incrementally, NIH funded the *ImageJ2* project in 2009 to overhaul this widely useful and very popular program, and to create a more robust and more capable foundation for future enhancements (▶ http://imagej.net/ImageJ2).

Both ImageJ and ImageJ2 have benefitted from independent software development projects that interoperate with these programs. The *Bio-Formats* file I/O library as well as other related projects led by the Laboratory for Optical and Computational Instrumentation (LOCI) at the University of Wisconsin (▶ https://loci.wisc.edu) are important resources in the ImageJ universe and have added valuable functionality. The Bio-Formats project responded to the community's need for software that would read and write the large number of vendor-supplied image file formats, mostly for light microscopy (LM). Today the Bio-Formats library goes well beyond LM vendor formats and encompasses 140 different file types, including many useful for SEMXM, such FEI and JEOL images, multi-image TIFFs (useful for EDS multi-element maps), movie formats like AVI for SEM time-lapse imaging, etc. A follow-on LOCI project called SCIFIO aims to extend the I/O library's scope to include N-dimensional files (Hiner et al. 2016). Both projects are closely associated with the Open Microscopy Environment (OME) project and the OME consortium (▶ http://www.openmicroscopy.org). Similarly, the *ImgLib2* project aims to provide a neutral, Java-based computational library for processing N-dimensional scientific datasets of the kind targeted by SCIFIO (Pietzsch et al. 2012).

Given the complexity of this rapidly evolving ecosystem of interrelated and interoperable tools that support ImageJ, it is not surprising that some users find it difficult to understand how all the pieces fit together and how to exploit all the power available in this software suite. Fortunately, there is a simple way to access much of this power: by installing *Fiji*.

13.2 Fiji

Fiji, which is a recursive acronym that stands for "Fiji Is Just ImageJ," is a coherent distribution of ImageJ2 that is easy to install and comes pre-bundled with a large collection of useful

plugins and enhancements to the bare ImageJ2 application (Schindelin et al. 2012). It is often thought of as "ImageJ with Batteries Included." The Fiji website provides several convenient installation packages for both the 32-bit and 64-bit versions of Fiji for common operating systems such as Microsoft Windows (currently Windows XP, Vista, 7, 8, and 10) and Linux (on amd64 and x86 architectures). Pre-built and tested versions for Mac OS X 10.8 (Mountain Lion) and later are also available. By default, these bundles include a version of the Java Runtime Environment (JRE) configured for Fiji's use that can coexist with other instances of Java on the host computer, but "bare" distributions of Fiji are available that will attempt to utilize your computer's existing JRE if that is preferred. Of course, as an Open Source software project, all of the source code for Fiji can be downloaded.

Installation of Fiji is straightforward because it has been configured as a *portable application*, meaning it is designed to run from its own directory as a standalone application. Installation is as simple as downloading the distribution and unpacking it; Fiji does not use an installer, does not copy shared libraries into destination directories scattered around the file system, and it does not store configuration information in system databases (e.g., the Windows registry). Because of this design, once installed it can be moved or copied simply by moving the directory tree. This portability also means it runs quite well from a USB flash drive or removable hard drive.

After launching Fiji you will be presented with the Fiji main window (Fig. 13.1a), which contains the Menu bar, the Tools bar, and a Status bar for messages and other application feedback to the user. Selecting "Update…" or "Update Fiji" on the Help menu will trigger the updater, one of the most useful features of Fiji. Because Fiji is configured by default to start the updater immediately after program launch, for many new users this is the first piece of Fiji

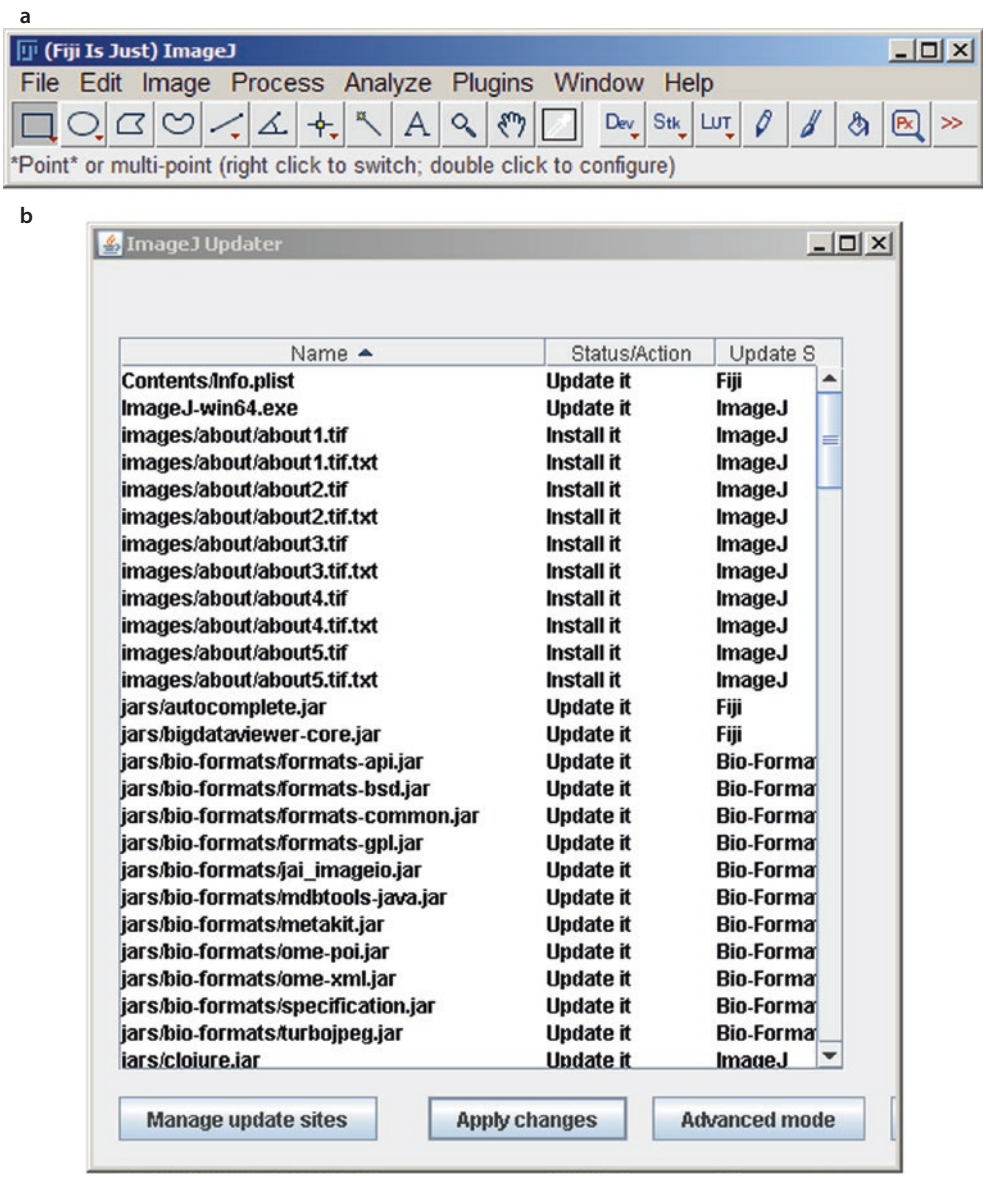

Fig. 13.1 **a** Fiji main window. **b** Fiji updater

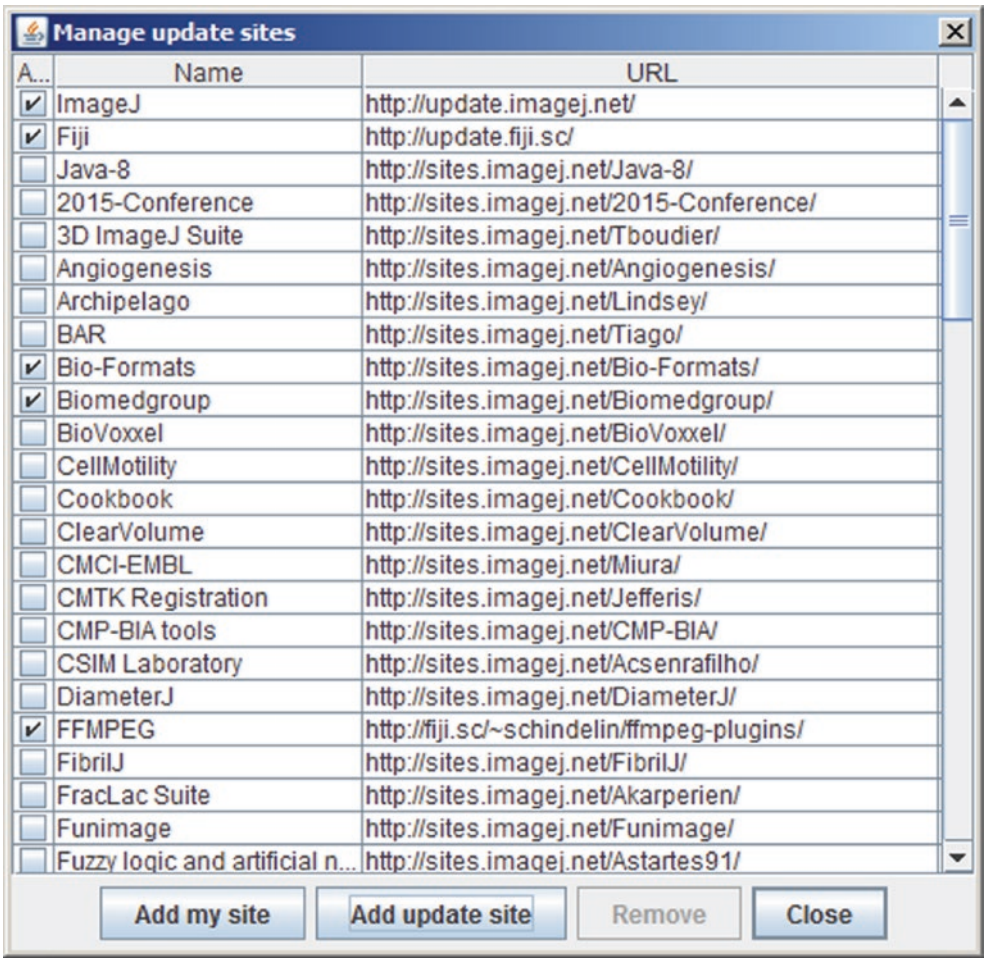

Fig. 13.2 Fiji's Manage Update Sites window

functionality they encounter. Upon activation the updater will scan your local Fiji installation and calculate checksums for everything to see if any components are out-of-date, or if new features have been added since it was last run. It will then confer with the global Fiji code repositories to look for updated Java Archive files (.jar files) and offer to download and install them for the user. Figure 13.1b shows an example of this, where the updater has located numerous changes in the ImageJ, Fiji, and Bio-Formats repositories. By selecting the "Apply changes" button the software will fetch the latest code and apply all the patches to the user's local Fiji installation. Figure 13.2 shows a window listing a selection of available Fiji update sites illustrating the rich community resources.

13.3 Plugins

One of the most powerful features of Fiji is the enormous collection of plugins, macros, and other extensions that have been developed by third-party contributors in the scientific community. Fiji comes with some of the most useful plugins pre-installed, and these are accessible from the Plugins menu item. Hundreds of powerful features are accessible this way, exposed to the user in a series of cascading menus and submenus. Such a large set of choices can be overwhelming at first, but many of the plugins are meant for light microscopy, so the SEM analyst may find it simpler to ignore some of them. However, the Non-local means denoising plugin, the Optic flow plugin, and the myriad of morphological operations under the Plugins|Process menu are all useful for SEM microscopists, as are the dozens of features in the Registration, Segmentation, Stacks, Stitching, Transform, and Utilities submenus.

Sometimes the appearance of a plugin as a single entry in the Fiji menu structure belies the full power of that plugin. Indeed, some of the most impressive plugins available for Fiji might be considered entire image processing packages in their own right. An example of this is the Trainable WEKA Classifier plugin that appears as a single entry on the Segmentation submenu of the Plugins menu. WEKA is an acronym that stands for "Waikato Environment for Knowledge Analysis," a tool

developed by the Machine Learning Group at the University of Waikato in New Zealand (Hall et al. 2009). WEKA is a full-featured and very popular open source software suite written in Java for machine learning (ML) researchers. It provides an open, cross-platform workbench for common ML tasks such as data mining, feature selection, clustering, classification, and regression, going well beyond just image analysis. The Fiji plugin is a gateway into this large array of tools and provides a convenient interface for processing SEM images using a modern machine learning framework (► http://imagej.net/Trainable_Weka_Segmentation).

Some of the most widely used and powerful plugins in Fiji have been back-ported into ImageJ itself, and are available directly from the main application's menu structure. An example of this is the Process menu option known as "Contrast Limited Adaptive Histogram Equalization," or CLAHE (Zuiderveld 1994). First developed in 1994, this algorithm has been implemented in a wide variety of image processing tools. It is designed to amplify local contrast by performing histogram equalization on small subsets (tiles) within the source image, but to limit the allowed amplification to reduce the tendency to magnify the noise in relatively homogeneous patches. Figure 13.3a shows a scanning electron micrograph of microfabricated features on silicon, acquired at 20 keV using an Everhart–Thornley detector. Because of slight misalignment of the raster with the linear features, Moiré contrast is evident in the image as bright edges on some features, and there are pure white and pure black horizontal lines that have been added to simulate contrast artifacts. These extreme limits of intensity preclude the usual brightness/contrast adjustments, but the CLAHE algorithm recovers invisible details without loss of information, Fig. 13.3b.

While it is possible that the ideal software tool for your project is available in Fiji itself (e.g., CLAHE) or in one of the many plugins loaded into Fiji by default (e.g., Trainable Weka Segmentation), it is much more likely that the tool you are looking for is not in the distribution you downloaded from the Fiji website. Only a small fraction of the plugins available to the user have been installed in the menu tree. A much larger collection awaits the user who is willing to explore the many optional Fiji update sites. The Updater window shown in Fig. 13.1b has a "Manage update sites" button at the lower left. If you press this button you are presented with a list of optional plugin repositories, as shown in Fig. 13.4a. When checked, these additional update sites will be accessed and used by the Updater to find new functionality to add into the base distribution. Some of the sites shown in Fig. 13.4a only add one or two items to the Plugins menu, while others import a much larger amount of supplemental code and capability. For example, the "Cookbook" site listed in Fig. 13.4a adds a new top-level menu item to the Fiji main window, as shown in Fig. 13.4b. This new menu contains example code to help new users follow along with a community-written tutorial introduction to ImageJ, available on the ImageJ website (► http://imagej.net/Cookbook).

Occasionally a set of useful plugins will be written by a researcher or contributor who is unable or unwilling to make them available as an update site. The ImageJ website offers free hosting of update sites for any author of plugins, and organizations can run their own Fiji update sites if they wish. If these are not already a selectable option on the Manage update sites list (Fig. 13.4a), the "Add update site" button allows the user to manually follow a third-party update site. As a last resort, plugins may also be manually installed into the Fiji plugins directory, but they will not be automatically updated so this is discouraged.

Thus, there are really four tiers of plugins across the ImageJ universe: (1) core ImageJ plugins that are bundled into the base ImageJ package (more than 1000 plugins in 2016); (2) core Fiji plugins, included by default in the "Batteries Included" Fiji distributions (more than 1000 additional plugins in 2016); (3) plugins available from additional update sites; and (4) plugins that must be located, downloaded, and installed manually. While this last category of plugins is the most likely to be buggy and poorly supported, any plugin written by a co-worker or officemate will often fall into this category, so the code may be highly specific to your task or your organization—don't overlook these!

13.4 Where to Learn More

Learning ImageJ or Fiji can be a daunting task for the beginner, and no attempt was made here to provide even a basic introduction to opening, exploring, manipulating, and saving SEM micrographs or X-ray data. However, there are many excellent resources for learning Fiji on the web, and the community offers several support channels for those who need additional help. Fiji itself has a built-in Help menu with links to the ImageJ and Fiji websites, newsgroups, online documentation, example code, developer tools, guidance documents, etc. The ImageJ Help page maintains links to the ImageJ Forum, Chat Room, and IRC channel as well as pointers to the ImageJ tag on Stack Overflow and Reddit, popular online locations for ImageJ and Fiji questions and answers. Finally, there is a synoptic search engine for many of the above resources at ► http://search.imagej.net.

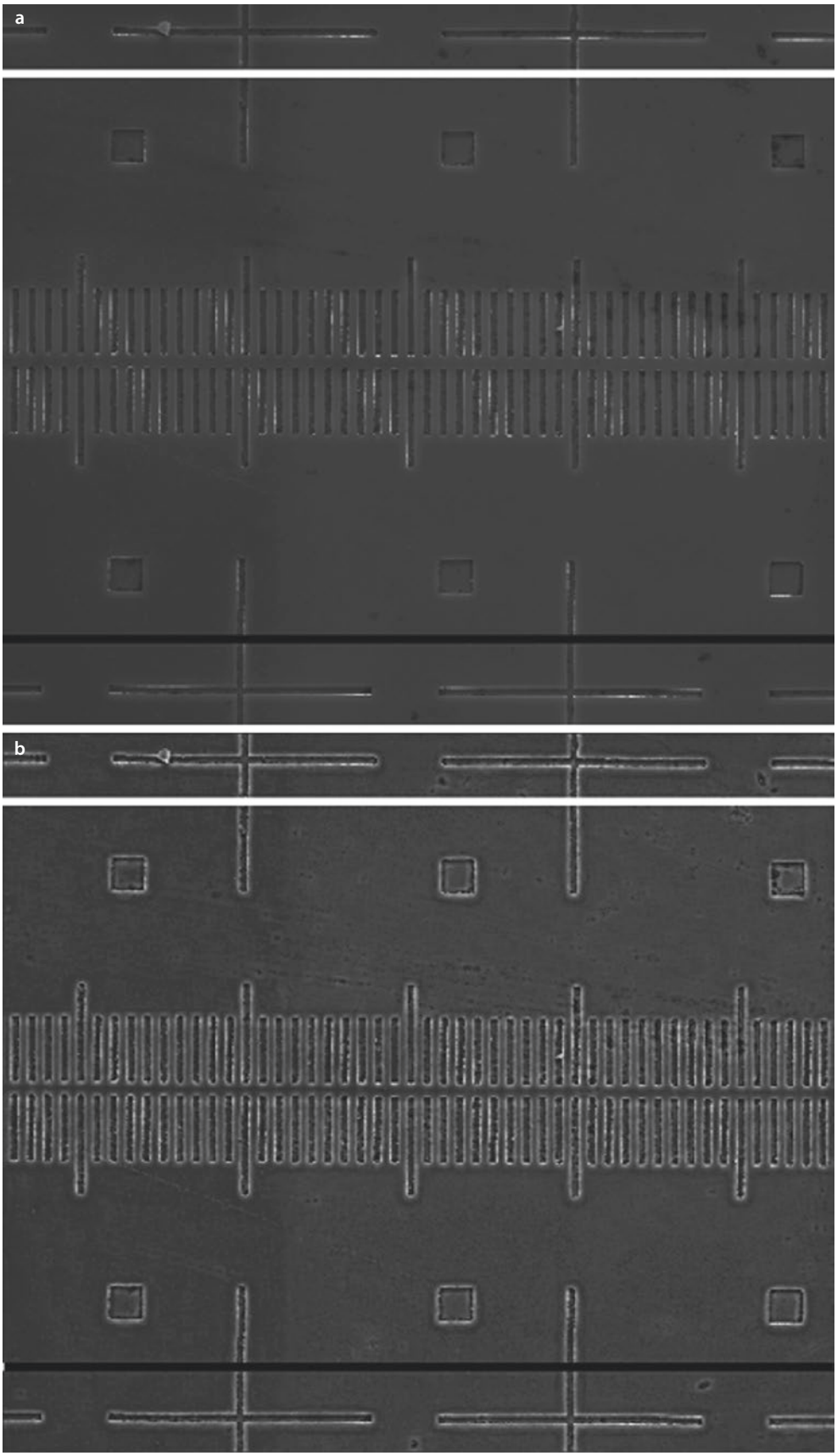

Fig. 13.3 Application of Fiji's CLAHE processing to a low-contrast SEM images **a** and the resulting enhanced image **b**. The images are 256 µm wide

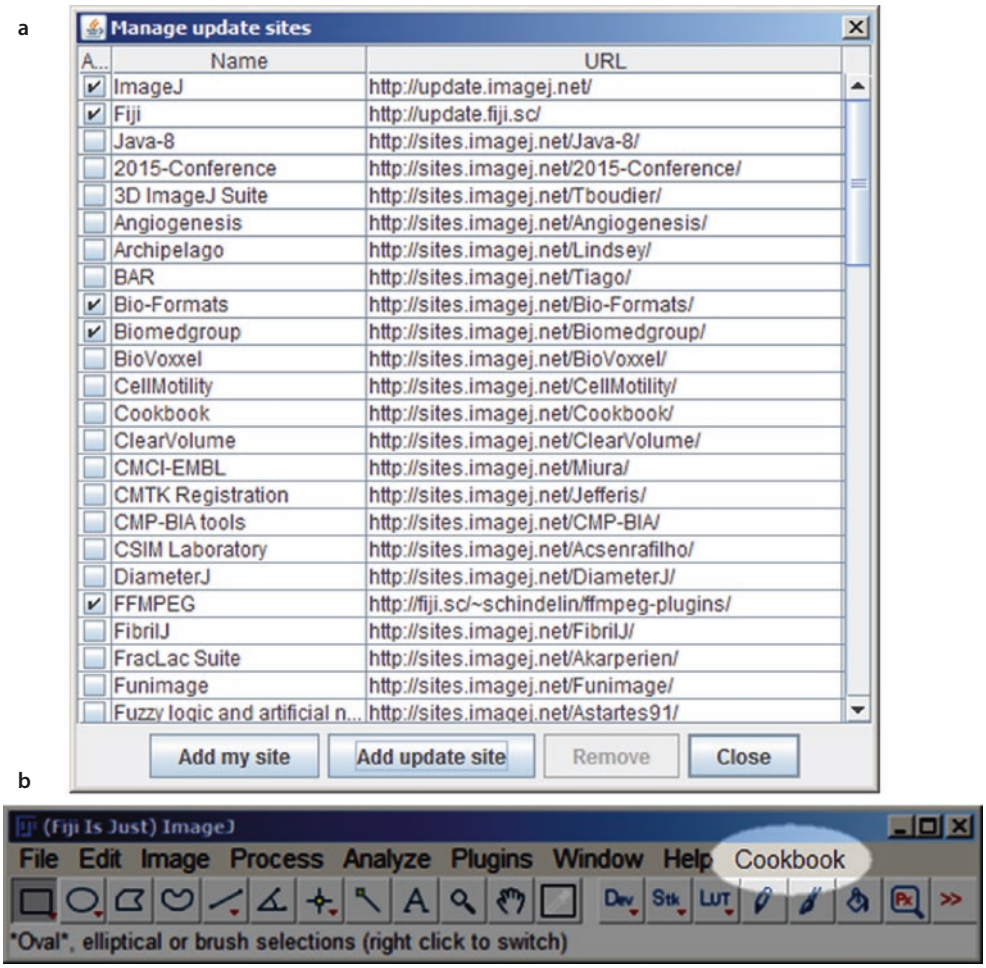

Fig. 13.4 **a** Fiji's Manage Update Sites window, showing some of the many optional plugin repositories available for use. **b** A new top-level menu item called "Cookbook" imported from the Cookbook update site

References

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. ACM SIGKDD Explor Newsl 11(1):10–18

Hiner M, Rueden C, Eliceiri K (2016) SCIFIO [Software]. ► http://scif.io

Pietzsch T, Preibisch S, Tomancak P et al (2012) ImgLib2—generic image processing in Java. Bioinformatics 28(22):3009–3011, ► http://imagej.net/ImgLib2

Schindelin J, Arganda-Carreras I, Frise E et al (2012) "Fiji: an open-source platform for biological-image analysis". Nat Methods 9(7):676–682, ► https://fiji.sc (Note that Fiji is not spelled FIJI)

Schindelin J, Rueden CT, Hiner MC et al (2015) The imageJ ecosystem: an open platform for biomedical image analysis. Mol Reprod Dev 82(7–8):518–529

Schneider CA, Rasband WS, Eliceiri KW (2012) NIH Image to ImageJ: 25 years of image analysis. Nat Methods 9(7):671–675

Zuiderveld K (1994) "Contrast limited adaptive histograph equalization." graphic gems IV. Academic Press Professional, San Diego, pp 474–485