NIH Image to ImageJ: 25 years of image analysis

Caroline A Schneider, Wayne S Rasband & Kevin W Eliceiri

For the past 25 years NIH Image and ImageJ software have been pioneers as open tools for the analysis of scientific images. We discuss the origins, challenges and solutions of these two programs, and how their history can serve to advise and inform other software projects.

The last 50 years have seen tremendous technological advances, few greater than in the area of scientific computing. One of the fields in which scientific computing has made particular inroads has been the area of biological imaging. The modern computer coupled to advances in microscopy technology is enabling previously inaccessible realms in biology to be visualized. Although the roles of optical technologies and methods have been well documented, the role of scientific imaging software and its origins have been seldom discussed in any historical context. This is due in part to the relative youth of the field, the wide variety of imaging software tools available, sheer diversity of subfields and specialized tools, and the constant creation and evolution of new tools.

In this great diversity and change, one software tool has not only survived but thrived. The scientific image-analysis program, ImageJ[1,2], known in previous incarnations as NIH Image[3], was an early pioneer in image analysis. Twenty-five years after its introduction the program not only persists but continues to push and drive the field. It does so not by continuously reinventing itself but by sticking to a core set of design principles that have allowed it to become a modern image-processing platform and yet retain an interface that a user from over 20 years ago would recognize and readily use.

Caroline A. Schneider and Kevin W. Eliceiri are at the Laboratory for Optical and Computational Instrumentation, University of Wisconsin at Madison, Madison, Wisconsin, USA. Wayne S. Rasband is at the Section on Instrumentation, US National Institutes of Health, Bethesda, Maryland, USA.
e-mail: eliceiri@wisc.edu

Given the great success and impact of ImageJ, one would expect that this application was a software initiative with official backing and formal planning by a central funding body. Despite its original name, NIH Image, and its home at the US National Institutes of Health (NIH) for over 30 years in some form, ImageJ is a product of need, user-driven development and collaboration—rather than a specific plan by the NIH to create it at the onset.

ImageJ became what it is through years of collaborative effort, and NIH nurtured it by providing the resources to support the primary programmer, Wayne Rasband, throughout this period. In this current age of careful oversight and scrutiny from administrative bodies, the story of ImageJ and the independent track that Rasband had in its development is both interesting and telling for other projects. To best understand this, one needs to look at how ImageJ started.

Rasband created NIH Image, the predecessor to ImageJ, at the NIH in 1987, but the foundation for this program was laid even earlier at the beginning of Rasband's career. Rasband received his bachelor's degree in math from the University of New Mexico, Albuquerque, in 1965. He was involved early on with the IBM computer punch card systems while still in school. He leveraged this expertise to get a job with the State of New Mexico's Department of Automated Processing, where he performed common business-oriented language (COBOL) programming and general systems programming. Shortly thereafter, Rasband was drafted by the US Army and assigned to the Pentagon. While there, Rasband became aware of a University of Maryland graduate program that would allow him to pursue his master's degree in computer science and thus leave the service early. One day in 1970, in the commons at the University of Maryland, College Park, he saw a notice for a part-time programming position at the NIH in Bethesda, Maryland, USA, to work on the laboratory instrument computer (LINC) created at the Massachusetts Institute of Technology. Rasband applied for this position, was hired and worked at the NIH until he retired in 2010.

NIH Image: image analysis on the Mac

When Rasband began working at the Research Services Branch at the National Institute of Mental Health, part of the intramural campus of the NIH, most scientific data processing was done on mainframe computers, and the personal computer revolution was just beginning. There was no image-analysis program for the Macintosh computer, and Rasband had just obtained one of the first Apple Macintosh (Mac) II computers. Rasband realized that it had the appropriate hardware and low-level software to be an ideal base for a small, low-cost image-analysis system; all it needed was some software for image analysis. Rasband decided to write that software in support of the imaging analysis needs he saw at the time: chiefly, better access in terms both of ease of adoption and cost.

It was his goal to have a low-cost image-analysis system that the average bench scientist could afford and deploy. Rasband wanted to create a system that was smaller and more affordable than his earlier software systems that required the $150,000 PDP-11 minicomputers in use at the time. He had developed an image-analysis program called "Image" for this platform. The program ran an imaging system that used a rotating drum film scanner to digitize images and a 512 × 512 pixel frame buffer to display the digitized images, and it supported a custom-built joystick that could be used to outline objects. The PDP-11 systems were used to analyze gels, autoradiographs, and computed tomography, magnetic resonance and positron emission tomography images.

As a successor to 'Image', Rasband set out to build a program that would provide the same utility but could be used on the desktop computers that were just becoming widely available, chief among them the Mac II. With its relative low cost of adoption, widespread use, easy graphic interface and good developer support, the Mac II was the ideal platform for a new 'Image' program. The Mac II had several key additions over the earlier Mac that made Rasband's vision of NIH Image possible, specifically (i) expansion slots: the ability to add third-party acquisition boards, (ii) advanced graphics: the ability to handle not only color but most importantly 8-bit 256 gray colors, the mainstay format of light microscopy, and (iii) support for the Pascal programming language to allow third-party developers to easily develop their own applications.

In the spring of 1987, just a few months after Rasband had obtained his Mac II, he handed out copies of the NIH Image program on floppy disks to anyone who asked. Rasband also promoted NIH Image on the Mac forum on the CompuServe Information Service electronic bulletin boards and made the program available on several Mac bulletin board systems. Rasband wanted to create a general-purpose extensible image-analysis program that could be used by anyone who wanted to capture, display and enhance images, and he never targeted a specific biological application or type of imaging such as microscopy. His goal was to let the users drive the applications for NIH Image.

Rasband continued to develop the program, but—through innovative concepts such as mailing lists, free reusable code, plug-ins and macros—he also encouraged the users to develop their own code to address their own application needs. Medical researchers were some of the first users of the program because autoradiographs, computed axial tomography or positron emission tomography scans and X-rays could be viewed, analyzed and notated. As NIH Image became increasingly used in many fields—biological microscopy being the largest—the functionality of the program and application base grew.

The move to other operating systems

As the code could be freely used in any form, NIH Image was used in a variety of cases, including spinoffs and related programs such as Scion Image (Scion Corporation) for the personal computer (PC) platform. Scion Image was a notable effort by the Scion Corporation to address an unmet need—providing an NIH Image for the PC (Microsoft Windows operating system) community. In the early 1990s the PC had caught up to the Mac and had the graphics functionality and extensibility needed to run a program such as NIH Image, but the NIH Image program was only available for the Mac. Scion Corporation's products were very popular with NIH Image users because they made a frame-grabber board that was the principal way users collected their images in NIH Image, whether from a gel imager or analog microscopy camera. Scion saw the opportunity to expand its hardware framegrabber market to the PC by making a Windows version of NIH Image. On their own, with no input from Rasband, they fully ported the Pascal-based NIH Image to the C programming language and released the resulting program as Scion Image. Unfortunately, users found it to be 'buggy', and because the program was closed-source, there was no way for Rasband and the community to fix these problems. Scion Image never achieved a large user base, and the need for NIH Image for Windows largely remained unmet.

After NIH Image had been established, Rasband started thinking about expanding its capabilities to other operating systems. He saw increasing interest in the Scion Image program because it ran on Microsoft Windows and also saw the frustration that it did not work as well as NIH Image did. He also saw the danger in having a separate Windows program, both in terms of support and in diluting the user base and plug-ins.

Yet the climate and timing were such that he felt he had to have a solution beyond the Mac platform. The late 1990s was a notable period in Apple history as the Mac was in a period of decline, with the PC rapidly gaining ground. In scientific research, the Mac still had a loyal following, but this following, too, was being eroded both owing to technologies only being available on the PC platform and the lower hardware cost of the PC. Rasband faced a major challenge: how to continue a program for the Mac and yet support one for the PC. Rasband did not want to port NIH Image to the PC and did not want to maintain two programs or trust a third party to maintain one.

In 1995 Sun Microsystems created the Java programming language and runtime environment in a bid to create an operating system–agnostic programming platform that would allow programmers to 'write once, run anywhere', freeing them from having to choose what operating system to support. Rasband found this idea appealing and liked the idea of maintaining a single code base that could run in any operating system with the Java runtime environment installed or on a Web browser as a Java applet, thus allowing a single program to be run not only on the Mac and Windows platforms but also on the Unix operating system that was becoming popular among scientists. Furthermore, after using Pascal for over 20 years, Rasband was ready to try another programming language.

In the transition of NIH Image to Java, Rasband wanted to retain the elements of NIH Image that had made it so successful but felt the software deserved a new name; he chose ImageJ to maintain the connection to NIH Image but with a "J" to indicate its Java foundation.

The transition from NIH Image to ImageJ was not without its problems, however, as the cross-platform compatibility proved difficult at times. The first public implementation of Java had many rough edges. Instead of 'write once, run everywhere', Rasband found himself writing once and debugging everywhere. As one of the first end-user scientific programs to widely use Java, there were many difficulties in getting ImageJ to work properly on different platforms and Java environment distributions. As an early Java adopter, Rasband had to tackle many software-interface issues—from talking to native hardware code for data acquisition to dealing with varying levels of Java support on different operating systems. But over time, as the Java runtime environments improved and coding problems were solved, porting NIH Image to Java set the stage for ImageJ to achieve even greater success.

During Rasband's many years developing NIH Image and ImageJ at NIH, occasionally a concerned lawyer or administrator would come see him with questions or concerns about the open nature of ImageJ and its commercial potential. Nothing came of these infrequent meetings, and Rasband was left unfettered to develop the program as he wanted.

A driving design criterion of both NIH Image and ImageJ was to keep the program simple with no complex user interfaces. Upon opening ImageJ, just a single toolbar appears, and it is from this straightforward interface that all of the capabilities of ImageJ can be found and used. The ImageJ toolbar has stayed essentially the same for 15 years, similar to how NIH Image has remained largely the same (**Fig. 1**). Rasband wanted a stable program interface that would not change, but he also needed a way to add new functionality based on user needs. This philosophy of limiting complexity also drove how he decided what functionality to integrate into the program directly or distribute as plug-ins.

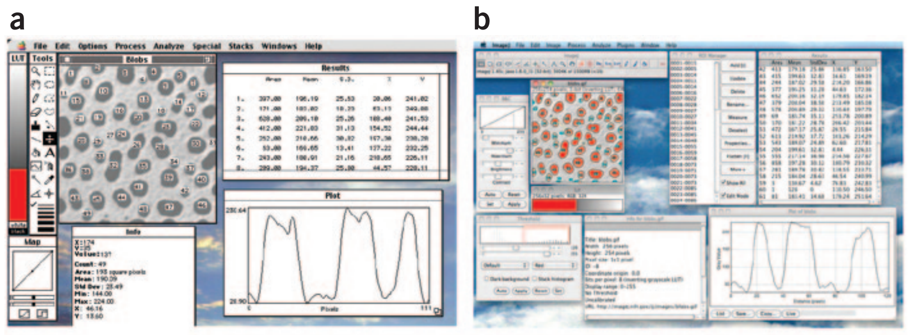
a
File Edit Options Process Analyze Special Stacks Windows Help
Tools
Blobs
Results
Plot
Info
Map
b

Figure 1 | Appearance of NIH Image and ImageJ. (**a**,**b**) Screenshots of NIH Image 1.62, released in 1999 (**a**), and ImageJ 1.45, released in 2011 (**b**). Although the look is slightly different, the overall feature layout and menu structure is basically the same.

Plugins and macros

To facilitate community input into NIH Image and ImageJ, Rasband established a community-driven development model with several key elements: (i) user-driven need and request for Rasband to address; (ii) user-driven need that another member of the community can address; (iii) user developer can create a solution to his or her own need but then share it with the community, and (iv) user feedback can be provided on an existing feature to either improve functionality or add new functionality.

A single developer–driven model such that all code is developed by one person would have resulted in a monolithic program. Although this would provide the simplicity of having only one way of doing things, the breadth and depth of the solutions would be greatly attenuated. Rasband instead chose a more flexible approach that would allow users to add functionality on their own, but in a manner that would allow the functionality to be shared with others. This was accomplished through the use of macros and plug-ins.

Macros are simple, custom programming scripts that automate tasks inside a large piece of software. Because of macros' rather basic programming format, general users can create macros with no formal programming experience. Rasband added a macro language to NIH Image in 1989 after he saw an article titled "Building your own C interpreter." He realized he could use the source code that was included in the article to create a Pascal language interpreter for NIH Image.

When Rasband later developed ImageJ, he based the macro language on the one in NIH Image. Similar to how the Pascal-based macro language remained very constant in NIH Image, ImageJ's macro language has remained very stable over the last 15 years. Many new commands have been added, but the early commands all still work. Although macros are used by programmers, they are especially useful to the bench biologist, with ~325 macros currently available on the ImageJ website.

The use of macros requires little or no programming experience. New features such as the macro recorder directly facilitate this, allowing users to record any actions they manually do. This recording is put into a macro syntax that users can execute for future application of this workflow, modify it as necessary and share it with others. ImageJ has since evolved in its scripting capabilities and now allows other scripting environments to be harnessed, such as JavaScript, or other languages to be called, such as Python, through an ImageJ Jython Bridge.

In many cases, linking together existing functionality using macros is insufficient for a necessary application and users need to add new functionality. In 1993, Rasband saw the great utility of plug-ins being used by Photoshop (Adobe) to add new functionality to that software and decided to add these modular software elements to NIH Image. NIH Image was one of the first scientific image-processing tools to have plug-ins and the first with such a large user base. Example plug-ins included facilities for three-dimensional rendering of images and particle analysis. ImageJ has had plug-in support from its inception, and the number of plug-ins has increased rapidly, with over 500 (May 2012) plug-ins that cover a wide range of functions available on the ImageJ website (**Fig. 2**). Some of these plug-ins are distributed with the core ImageJ and most are available for separate download and install by the user. Rasband's philosophy of limiting complexity drove how he decided what functionality to integrate into the program directly in the menus, distribute as core plug-ins that come prepackaged with ImageJ or make available as downloads from the ImageJ website. Many of the plug-ins built into ImageJ are from outside contributors, and the decision to include a plug-in in the base distribution was based on whether Rasband thought it would have widespread use. Additional ImageJ plug-ins are available at third-party websites with links to these resources from the ImageJ website.

It is important to note that Rasband never sought to replace commercial image-analysis solutions. In part, this is because a good part of the functionality of NIH Image or ImageJ was created as a result of there not being another solution, commercial or open-source, to do it. Of course, out of necessity to be a full-

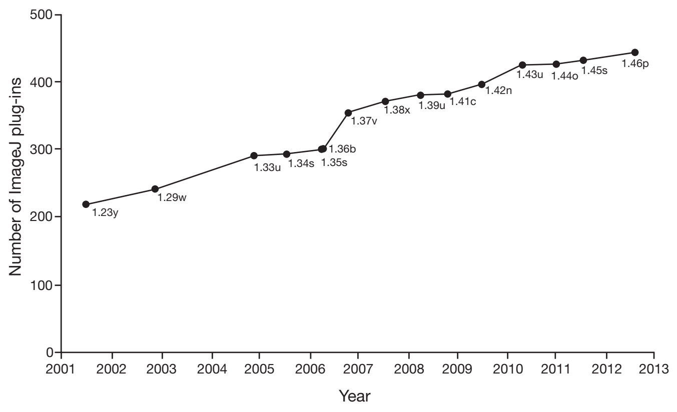

Figure 2 | ImageJ plug-ins bundled with each ImageJ release over time. Each data point is labeled with the version number and letter.

featured program, NIH Image and ImageJ recapitulated many of the features present in a commercial image-processing program such as Adobe Photoshop. Certainly, many of the NIH Image and ImageJ users were first attracted to using the software because they could not afford an expensive seat (per computer) license for specialized commercial image-analysis packages. But many users of ImageJ also use commercial software, so clearly that is not the only draw. In fact, many imaging-software companies also use and recommend ImageJ. As well, many commercial tools have emulated the key concepts of ImageJ, for example, most modern analysis programs now offer some sort of scripting functionality.

File format challenges

One of the main challenges of image-analysis programs is being able to open any of the myriad image file formats that have been developed over the years. Owing to code contributions and add-ons from various sources through its community development model, NIH Image could read multiple image types—a rare capability among the early image-analysis programs. Needing support for proprietary formats from microscopes and other imaging equipment, users added support for the majority of formats.

As one of the first programs to widely support proprietary formats, NIH Image had the best-supported and functional readers, which are modular software code used to read a file format and translate it into the open formats used by the software. These readers led to the development of a reader code used not only in NIH Image and ImageJ but other programs as well.

A major example of this—and a vast improvement to ImageJ's ability to read and parse proprietary image data—was the advent of Bio-Formats[4], a library from the Open Microscopy Environment (http://www.openmicroscopy.org/) for reading proprietary image formats. Whereas Bio-Formats is a general library used by many programs, ImageJ is its biggest user with the Bio-Formats ImageJ plug-in used in over 30,000 laboratories. ImageJ has been far more than just a user of Bio-Formats; without the community-driven model of ImageJ and the resulting vetting and testing process for every new format, Bio-Formats arguably would not have the performance and functionality it currently has. In this way ImageJ continues to benefit other programs that do not directly use ImageJ but that take advantage of its framework and plug-ins and other code such as Bio-Formats.

Integration with other tools

Biologists often need to use a variety of different software to acquire and analyze data, and connectivity between these tools can be crucial. Owing to the Mac-only support of NIH image and its pioneering status, there were few early examples of NIH Image connecting with external programs. There were several prominent examples mediated by export of an open file format, however, such as the export of a .csv file for statistical analysis. From the beginning of ImageJ there was interest in directly connecting to external toolkits without the need to export and open files, and early connections to Matlab (MathWorks) are a prime example. ImageJ's third-party tool connections have allowed it to be used in image workflows and take advantage of algorithm capabilities provided by Matlab.

ImageJ connectivity with other software programs, such as Imaris, Cell Profiler[5] and Knime[6–8], has also been established. Although Rasband did not specifically envision these collaborations when designing the program, they have enabled a variety of new functionality ranging from automated screening and segmentation-based measurements to sophisticated signal processing analysis, thus extending the utility of ImageJ.

A prominent example of how ImageJ has been adopted by the community is Fiji (Fiji Is Just ImageJ) and ImageJ2. The goal of Fiji[9] was to design a complete installation that was identical on any platform and which was easy to download and unpack. ImageJ2 (http://developer.imagej.net/), the next generation of ImageJ, is an NIH-funded collaboration between several institutions, groups and individuals, including Rasband. The ImageJ2 collaboration hopes to create more extensibility, modularity and interoperability as well as extend ImageJ community resources. ImageJ2 retains the interface of ImageJ but adds new architecture to remove some of the current limitations of ImageJ, such as data types, image size and dimensions. In addition to Fiji and ImageJ2, several other variants and programs based on ImageJ are currently available (**Table 1**). These variants all developed from targeting a specific community need that NIH Image or ImageJ did not have, organizing or adding additional tools for convenience in one bundle or making a custom version that is very use case–specific.

Rasband more than just tolerated this; he has encouraged it as another mechanism for addressing the diverse needs of the ImageJ analysis community. For example, when NIH Image core development ceased in favor of focusing on ImageJ, this resulted in NIH Image not being ported to the OSX (Apple) operating system. There was a population of electron microscopists that did not want to change their workflow and ported NIH Image as a new program, ImageSXM that runs on OSX with a focus on electron-microscopy analysis.

Other variants arose because of the desire to improve access to new users and provide documentation. MBF_ImageJ was developed by Tony Collins and colleagues to provide a comprehensive user manual

Table 1 | List of NIH Image and ImageJ variants

	Date initiated	Description
NIH Image	1987	The predecessor of ImageJ, created by Rasband; made for the Macintosh; no longer under active development
ImageSXM	May 1993	A version of NIH Image for OSX extended by Steve Barrett; intended to handle loading, display and analysis of images from the scanning microscope
ImageJ	1997	The current version of ImageJ developed by Rasband; sometimes called ImageJ1
ImageJ2x	Unknown	An offshoot of ImageJ; modified to use Swing interface; no longer under active development
ImageJA	July 2005	An offshoot of ImageJ developed by Johannes Schindelin; used as the core of Fiji
Fiji	December 2007	A 'batteries included' distribution of ImageJ popular in the life sciences
ImageJX	March 2009	Created by Grant Harris to discuss improvements to ImageJ; formed the basis of an application to NIH that launched ImageJDev
ImageJ2 (ImageJDev)	December 2009	Under development by the ImageJDev project; a complete rewrite of ImageJ; includes ImageJ1 to allow for old-style plug-ins and macros
MBF_ImageJ	2005	Bundle developed by Tony Collins for light microscopists; plug-ins from MBF_ImageJ can be installed on Fiji, combining the programs
SalsaJ	Unknown	An offshoot of ImageJ intended for astronomy applications; designed for use in classrooms; available in over 30 languages
CellProfiler	2006	Free, open-source software started by Anne Carpenter and Thouis Jones; aids biologists without computer-vision training to quantitatively measure cell images automatically
Icy	2011	Created by the Quantitative Image Analysis Unit at Institut Pasteur, Icy provides integrated software to bridge the gap between users and developers through open-source software and a central website
Bio7	Unknown	Application used for ecological modeling; integrated development environment; focuses on individual-based modeling and spatially explicit models
μManager	2005	Open-source microscopy software; controls automated microscopes; comprehensive imaging solution when used with ImageJ; developed by Arthur Edelstein, Ziah Dean, Henry Pinkard and Nico Stuurman

with an organized preloaded plug-in and macro structure for ImageJ so that users could follow the instructions to do certain steps such as thresholding and three-dimensional rendering. ImageJA was developed to allow for an applet version of ImageJ that could be run in any web browser, and this is now integrated into Fiji. SalsaJ was a targeted version of ImageJ with an interface and content for astronomy users. There have also been several attempts to extend the functionality and data model of ImageJ, including ImageJX and ImageJ2X. These are no longer active initiatives, but ideas from those projects have been incorporated in current ImageJ efforts, including the ImageJ2 project. Other applications are not variants of ImageJ but use components of ImageJ, such as the plug-ins; these include μManager[10], Icy[11], CellProfiler[5,12,13] and Bio7.

As the ImageJ family of programs moves forward, Rasband continues to play a large part in maintaining and supporting ImageJ. Although he retired in 2010 after 40 years as a programmer at the Research Services Branch, he now volunteers with the Section of Instrumentation at the NIH and works closely with the Center for Information Technology at the NIH, which hosts the ImageJ website and mailing list. Rasband fixes bugs, adds features requested by users, and manages the website and mailing list.

The continued popularity and growth of ImageJ throughout the scientific community has surprised Rasband. The ImageJ website has ~7,000 visitors a day, and there are ~1,900 subscribers to the ImageJ mailing list as of May 2012. A recent PubMed Central search (May 2012) of "ImageJ" returned over 20,000 papers. Furthermore, ImageJ has been used in teaching, such as in an image-processing textbook[14] that illustrates imaging processing examples using ImageJ. Rasband hopes to see the continued use and evolution of ImageJ as a teaching and research tool as more people recognize and understand its capabilities.

Ten years from now, Rasband expects to still be working on ImageJ. Although the program and its variants will continue to develop, and other programs will be developed based on ImageJ, he expects the program and its variants to retain the two fundamental hallmarks of ImageJ: flexibility and extensibility developed over 25 years ago. He also expects ImageJ to continue to be used for diverse applications ranging from materials science and soil science, astronomy and climate science, to medical imaging and crystallography.

ACKNOWLEDGMENTS
We thank members of our research groups and software projects for helpful feedback on the manuscript, in particular C. Rueden, J. Fong and J. Schindelin for input on the manuscript and help with the figures. This work was supported in part by NIH grant RC2 GM092519 (to K.W.E.), but the opinions expressed are solely those of the authors.

COMPETING FINANCIAL INTERESTS
The authors declare no competing financial interests.

1. Abramoff, M., Magalhaes, P. & Ram, S. *Biophotonics Int*. **11**, 36–42 (2004).
2. Collins, T.J. *Biotechniques* **43**, 25–30 (2007).
3. Girish, V. & Vijayalakshmi, A. *Indian J. Cancer* **41**, 47 (2004).
4. Linkert, M. *et al. J. Cell Biol.* **189**, 777–782 (2010).
5. Kamentsky, L. *et al. Bioinformatics* **27**, 1179–1180 (2011).
6. Lindenbaum, P., Le Scouarnec, S., Portero, V. & Redon, R. *Bioinformatics* **27**, 3200–3201 (2011).
7. Jagla, B., Wiswedel, B. & Coppee, J.Y. *Bioinformatics* **27**, 2907–2909 (2011).
8. Saubern, S., Guha, R. & Baell, J.B. *Mol. Inform.* **30**, 847–850 (2011).
9. Schindelin, J. *et al. Nat. Methods* **9**, 676–682 (2012).
10. Edelstein, A., Amodaj, N., Hoover, K., Vale, R. & Stuurman, N. *Curr. Protoc. Mol. Biol.* 14.20.1–14.20.17 (2010).
11. de Chaumont, F. *et al. Nat. Methods* **9**, 690–696 (2012).
12. Carpenter, A.E. *et al. Genome Biol.* **7**, R100 (2006).
13. Lamprecht, M.R., Sabatini, D.M. & Carpenter, A.E. *Biotechniques* **42**, 71–75 (2007).
14. Burger, W. & Burge, M.J. *Digital Image Processing: an Algorithmic Introduction Using Java* (Springer, 2010).