NIH Image to ImageJ: 25 years of Image Analysis

Caroline A. Schneider[1], **Wayne S. Rasband**[2], and **Kevin W. Eliceiri**[1,#]

[1]Laboratory for Optical and Computational Instrumentation, University of Wisconsin at Madison, Madison, WI [2]Section on Instrumentation, National Institutes of Health, Bethesda, Maryland

Abstract

For the past twenty five years the NIH family of imaging software, NIH Image and ImageJ have been pioneers as open tools for scientific image analysis. 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 fifty years have seen tremendous technological advances, few greater than in the area of scientific computing. One of the fields where scientific computing has made particular inroads has been in the area of biological imaging. The modern computer coupled to advances in microscopy technology is enabling new frontiers in biology to be visualized. While the role of the 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 sub fields and specialized tools, and the constant creation and evolution of new tools. Yet 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], is an early pioneer in image analysis. Yet 25 years later the program not only persists but continues to push and drive the field. Interestingly, the program has done so not by continuously reinventing itself but instead 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.

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 National Institutes of Health (NIH) for over 30 years in some form, ImageJ is a product of need and 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 best 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.

NIH Image, the predecessor to ImageJ, was created by Wayne Rasband at NIH in 1987, but the early foundation for this program started even earlier at the beginning of Rasband's career. Rasband received his bachelor's in math from the University of New Mexico in 1965. He was involved early on with the IBM computer punch card systems while still in school and leveraged this expertise to get a job with the State of New Mexico's Department of Automated Processing. While there Rasband was involved in COBOL programming and general systems programming. Shortly after, Rasband was drafted by the 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 Masters in computer science and thus leave the service early. One day in 1970 in the commons at the University of Maryland he saw a notice for a part time programming position at NIH in Bethesda working on the LINC (Laboratory INstrument Computer) created at MIT. Rasband applied for this position, was hired and worked at the NIH until he retired in 2010.

NIH Image: Image analysis on the Mac

At the time 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 Mac IIs. 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 the 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 the PDP-11 minicomputer. The program ran an imaging system that used a rotating drum film scanner to digitize images and a 512×512 frame buffer to display the digitized images and supported a custom-built joystick that could be used to outline objects. The PDP-11 systems were used to analyze gels, autoradiographs, and CT, MRI and PET 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 Apple Macintosh (Mac) II (Apple Inc.) The Mac II with its relative low cost of adoption, widespread use, easy graphic interface and good developer support

provided the ideal program for a new "Image" program. The Mac II had several key additions over the earlier Mac that made his vision of NIH Image possible 1) expansion slots- the ability to add third party acquisition boards 2) Advanced Graphics- the ability to handle not only color but most importantly 8 bit 256 grays, the mainstay format of light microscopy 3) support for the Pascal programming language to allow third party developers to easily develop their own applications.

In spring of 1987, just a few months after Rasband had gotten the Mac II computer, copies of the NIH Image program were handed out on floppy disks to anyone who asked. NIH Image was also promoted on the Mac forum on the CompuServe (CompuServe Information Service) electronic bulletin boards and was 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 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, plugins 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 as autoradiographs, CAT or PET 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 found itself in a diverse range of use cases including spinoffs and related programs like Scion Image (Scion Corporation) for the 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, Microsoft Corporation) community. In the early 90s the PC had caught up to the Mac and had the graphics functionality and extensibility needed to run a program like NIH image but the NIH Image program was Macintosh only. Scion Corporation's products were very popular with NIH Image users as 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 did a full port of 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 since the program was closed source there was no way for Rasband and the community to fix these problems. Scion Image never did achieve 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 any operating system, not just the Mac. He saw increasing interest in the Scion Image program due to it running on Microsoft Windows and yet frustration that it didn't work as well as NIH Image. He also saw the danger in having a separate Windows program both in terms of support and in diluting the user base and plugins. Yet the climate

and timing was 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 it too was being eroded, both due to technology only 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 the PC. Rasband did not want to port NIH Image to the PC and didn't 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"—free 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 and 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 the many years developing NIH Image and ImageJ at NIH occasionally a concerned lawyer or administrator would come see Rasband 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 criteria 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 basically the same for 15 years similar to how NIH Image largely remained the same (Figure 1). Rasband wanted a stable program interface that wouldn't change on users, but he also needed a way to add new functionality based on the needs of his user community. This philosophy of limiting complexity also drove how he decided what functionality to integrate into the program directly or distribute as plugins.

Community contribution of plugins and macros

To facilitate community input into NIH Image and ImageJ, Rasband established a community driven development model with several key elements: 1) user driven need and request on list for Rasband 2) user driven need and another member of the community fixes and contributes back 3) user developer creates their own solution to their own need but then shares with community 4) user feedback on existing feature to either improve functionality or add new functionality. A single developer driven model where all code is developed by one person would have resulted in a monolithic program. While 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 plugins.

Macros are simple custom programming scripts that automate tasks inside a large piece of software. Due to its 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 Rasband 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. The macro language of ImageJ is based on that in NIH Image. Similar to how the Pascal based macro language stayed very constant in NIH Image, ImageJ's macro language has stayed very stable over the last 15 years. Many new commands have been added, but the early commands all still work. While macros are used by programmers, they are especially useful to the bench biologist with approximately 325 macros currently available on the ImageJ website. The macros require little or no programming experience and novel features like the macro recorder directly facilitate this, allowing the user 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 as necessary and share with others. ImageJ has since evolved in its scripting capabilities and now allows other scripting environments to be harnessed such as JavaScript or calling other languages 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 plugins being used by Adobe Photoshop (Adobe Systems Incorporated) 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 imaging processing tools to have plugins and the first with such a large user base. Example plugins included facilities for 3D rendering of images and particle analysis. ImageJ has had plugin support from its inception and plugin numbers have increased rapidly, with over 500 plugins covering a wide range of functions available on the ImageJ website (Figure 2). Many of these plugins are now distributed with the core ImageJ. Rasband's philosophy of limiting complexity drove how he decided what functionality to integrate into the program directly or distribute as plugins. Many of the plugins built into ImageJ are from outside contributors and the decision to

include a plugin in the base distribution was based on whether Rasband thought it would have widespread use.

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 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 couldn't afford an expensive seat license for specialized commercial image analysis packages. But many users of ImageJ also use commercial software, so clearly that is not the only draw and 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. Due to code contributions and add-ons from various sources through its community development model, NIH Image was able to read multiple image types, a rare capability among the early image analysis programs. The majority of the formats were added by users needing support for proprietary formats from microscopes and other imaging equipment. As one of the first programs to widely support proprietary formats, it had the best supported and functional readers, 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 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 (www.openmicroscopy.org) for reading proprietary image formats. While Bio-formats is a general library used by many programs, ImageJ is its biggest user with the Bio-formats ImageJ Plugin used by 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 don't directly use ImageJ but that take advantage of its framework and plugins 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. Due 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 for exporting and opening files and early connections to MATLAB (MathWorks Inc.) are a prime example. ImageJ's third party tool connections have allowed it to be used in image workflows and take advantage of algorithmic strengths provided by MATLAB. ImageJ connectivity with other software programs, such as Imaris, Cell Profiler[5] and Knime[6–8], have also been established. While these collaborations were not specifically envisioned by Rasband when designing the program, they enable a variety of new functionality ranging from automated screening and segmentation-based measurements to sophisticated signal processing analysis, thus further 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 (see Fiji paper[9] for a thorough review) was to design a complete installation 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 were all developed out of targeting a specific community need that NIH Image or ImageJ didn't have, organizing or adding additional tools for convenience in one bundle, or making a custom version that is very use case specific. This is more than just tolerated by Rasband, 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 Inc.) operating system. There was a population of electron microscopists that did not want to change their workflow and ported NIH Image to ImageSXM that runs on OSX with a focus on electron microscopy analysis. Other variants gave rise because of the desire to improve access to new users and provide documentation. MBF_ImageJ was developed by Collins and colleagues to provide a comprehensive user manual with an organized preloaded plugin and macro structure for ImageJ so that users could follow the instructions to do certain steps such as thresholding and 3D 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 plugins; these include MicroManager[10], Icy[11], Cell Profiler[5, 12, 13] and Bio7.

As the ImageJ family of programs moves forward, Rasband continues to play a large part in maintenance and support of ImageJ. While he retired in 2010 after 40 years as a programmer at the Research Services Branch, he now volunteers with the Section of Instrumentation at

NIH and works closely with the Center for Information Technology at the NIH, which hosts the ImageJ website and mailing list. Rasband works to fix bugs, add features requested by users, and manage the website and mailing list. The continued popularity and growth of ImageJ throughout the scientific community has surprised Rasband. The ImageJ website gets about 7,000 visitors a day, and there are about 1,900 subscribers to the ImageJ mailing list. A recent PubMed Central search of "ImageJ" returned over 20,000 papers over its life span. Furthermore, ImageJ has been used in teaching such as with the creation of 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.

In 10 years, Rasband expects to still be working on ImageJ. While 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

References

1. Abramoff, M., Magalhaes, P., Ram, S. Biophotonics International. LAURIN Publishing; 2004. Image processing with ImageJ.

2. Collins TJ. ImageJ for microscopy. Biotechniques. 2007; 43:25–30.

3. Girish V, Vijayalakshmi A. Affordable image analysis using NIH Image/ImageJ. Indian J Cancer. 2004; 41:47. [PubMed: 15105580]

4. Linkert M, et al. Metadata matters: access to image data in the real world. J Cell Biol. 189:777–782.

5. Kamentsky L, et al. Improved structure, function and compatibility for CellProfiler: modular high-throughput image analysis software. Bioinformatics. 27:1179–1180.

6. Lindenbaum P, Le Scouarnec S, Portero V, Redon R. Knime4Bio: a set of custom nodes for the interpretation of next-generation sequencing data with KNIME dagger. Bioinformatics. 2011; 27:3200–3201. [PubMed: 21984761]

7. Jagla B, Wiswedel B, Coppee JY. Extending KNIME for next-generation sequencing data analysis. Bioinformatics. 2011; 27:2907–2909. [PubMed: 21873641]

8. Saubern S, Guha R, Baell JB. KNIME Workflow to Assess PAINS Filters in SMARTS Format. Comparison of RDKit and Indigo Cheminformatics Libraries. Mol Inform. 2011; 30:847–850. [PubMed: 27468104]

9. Schindelin J, et al. Fiji - an Open Source platform for biological image analysis. Nature Methods. 2012 In Press.

10. Edelstein A, Amodaj N, Hoover K, Vale R, Stuurman N. Computer control of microscopes using microManager. Curr Protoc Mol Biol. Chapter 14(Unit14):20.

11. de Chaumont F, et al. Icy: an open bioimage informatics platform for extended reproducible research. Nature Methods. 2012 In Press.

12. Carpenter AE, et al. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. Genome Biol. 2006; 7:R100. [PubMed: 17076895]
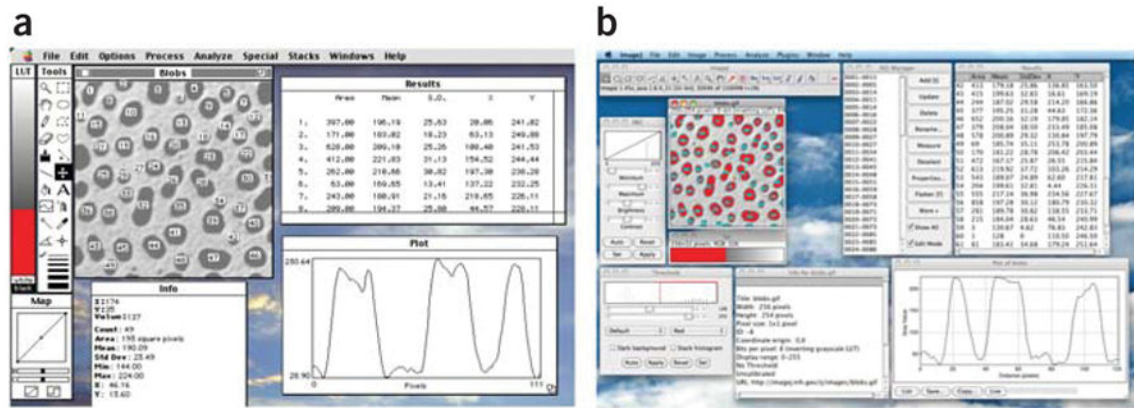
13. Lamprecht MR, Sabatini DM, Carpenter AE. CellProfiler: free, versatile software for automated biological image analysis. Biotechniques. 2007; 42:71–75. [PubMed: 17269487]

14. Burger, W., Burge, MJ. Digital image processing : an algorithmic introduction using Java. Springer; New York, NY: 2010.

Figure 1.
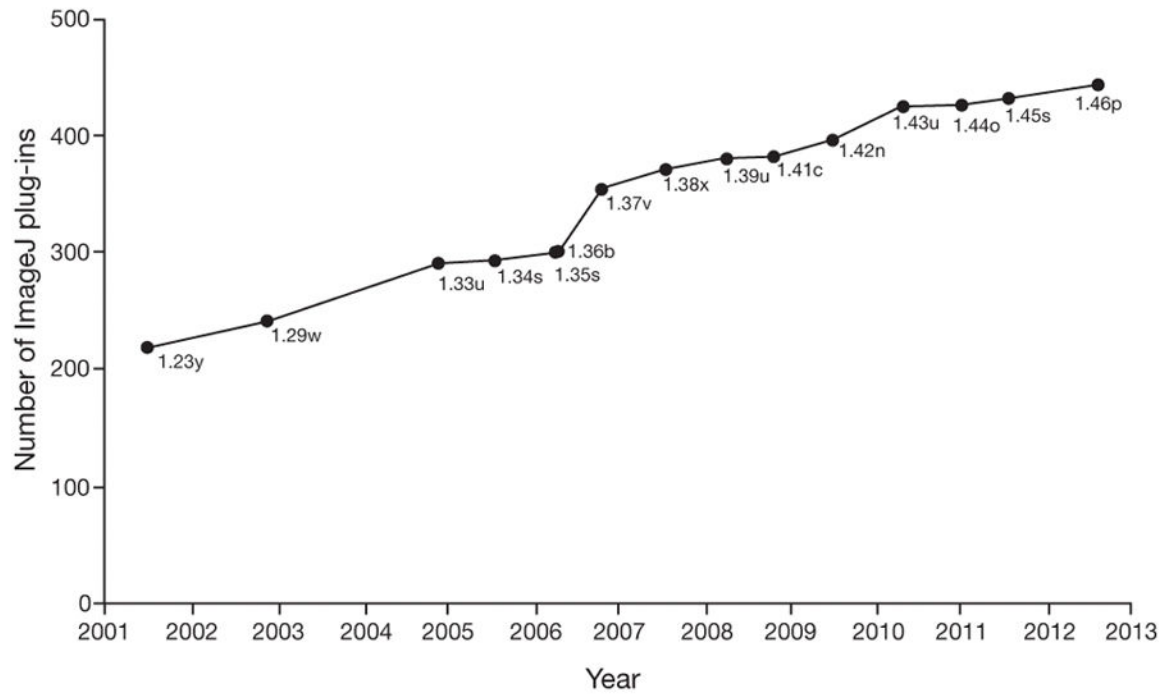Appearance of NIH Image and ImageJ. Screenshots of NIH Image in 199X (a) and ImageJ 1.45s in 2012 (b). Note that while the look and feel is slightly different, the overall feature layout and menu structure is basically the same.

Figure 2.
ImageJ plugins bundled with each ImageJ release over time.

Table 1

List of NIH Image and ImageJ Variants

Program Name	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 OS X 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
ImageJ2×	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	Fiji Is Just ImageJ; 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 which launched ImageJDev
ImageJ2 (ImageJDev)	December 2009	Under development by the ImageJDev project; a complete rewrite of ImageJ; includes ImageJ1 to allow for old-style plugins and macros
MBF_ImageJ	2005	Bundle developed by Tony Collins for light microscopists; plugins from MBF_ImageJ can be installed on Fiji combining the programs
SalsaJ	Unknown	An offshoot of ImageJ intended for astronomy; designed for use in classrooms; available in over 30 languages
CellProfiler		Free, open-source software started by Anne Carpenter and Thouis Jones; aids biologists without computer vision training to quantitatively measure cell images automatically
ICY		Created by the Quantitative Image Analysis Unit at Institute Pasteur, ICY provides integrated software to bridge the gap between users and developers through open-source software and a central website
Bio7		Application used for ecological modeling; integrated development environment; focuses on individual-based modeling and spatially explicit models
Micro-Manager		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