The advances of the medical and biological sciences over recent years, and the growing importance of determining the relationships between structure and function, have made imaging an increasingly important discipline. The ubiquitousness of digital technology — from banal digital cameras to highly specific micro-CT scanners — has made images an essential part of a number of research areas, from nanotechnology to astronomy.

It is common practice for manufacturers of image acquisition devices to include dedicated image processing software, but these programs are usually not very flexible and/or do not allow more complex image manipulations. Image processing programs also are available by themselves. ImageJ holds a unique position because it not only is in the public domain (meaning that its source code is openly available and its use is license-free), but also runs on any operating system. It is attractive because it is easy to use, can perform a full set of imaging manipulations and has a huge and knowledgeable user community.

The program is the brainchild of Wayne Rasband of the Research Services Branch, National Institute of Mental Health, in Bethesda, Md. It is called ImageJ because it is written in the Java language. Its first release, version 0.50, was Sept. 23, 1997, and it is now in version 1.31, released in February.

According to the institute, it has been downloaded from its Web site tens of thousands of times. To the biomedical research community, its availability is particularly important because it is not only is in the public domain (meaning that its source code is openly available and its use is license-free), but also runs on any operating system. It is attractive because it is easy to use, can perform a full set of imaging manipulations and has a huge and knowledgeable user community.
thousands of times, with the current rate being about 24,000 downloads per month. Rasband said that, after working for 10 years on NIH Image, ImageJ’s precursor, he didn’t see a bright future for it, even though Scion Corp. of Frederick, Md., had ported it from a Macintosh-only version to the PC/Windows platform. However, the Scion version wasn’t open source, and he is a strong proponent of open source and platform independence. So he began writing ImageJ in early 1997 after he had become intrigued by the new Java programming language.

**Imaging capabilities**

ImageJ can read most of the common and important formats used in the field of biomedical imaging (see table). If a file format is not currently supported, someone from the international user/developer community usually develops support within days. In addition, ImageJ can be used to acquire images directly from scanners, cameras and other video sources, including cameras that are compatible with TWAIN and FireWire, and frame grabber boards from Cooke, National Instruments and PixelSmart.

The program supports all common image manipulations, including reading and writing of image files, and operations on individual pixels, image regions, whole images and volumes. Volumes, called stacks in ImageJ, are ordered sequences of images that can be operated upon as a whole. It can perform basic operations such as convolution, edge detection, Fourier transform, histogram and particle analyses (including sophisticated statistical processing of groups of particles), editing and color manipulation; and more sophisticated operations such as dilation, erosion and closing of structures, and mathematical operations on sets of images, such as multiplication, exclusive or, and division. In addition, visualization operations, including color space conversions — for example, converting from RGB to Hue Saturation Intensity color space, two- and three-dimensional plotting — and surface and volume rendering, are supported. It also offers core support for analyzing electrophoretic gels.

**Cross-platform**

One of the strong points of ImageJ is its ability to run on different platforms. Statistics covering the last three months, with more than 80,000 downloads, indicate that it is being used mostly with Microsoft operating systems (80 percent), followed by Macintosh platforms (16 percent) and Linux (4 percent). Although these numbers are estimates and can be misleading because someone could download it to one platform and use it on another, it is interesting to note the relatively high number of Macintosh users (the worldwide market share was less than 2 percent in 2003, according to market research company IDC), supporting the long-held view that this type of computer attracts large sections of academia.

Obviously, with a free program, there is no telephone hot line for support. However, a large user base communicates through a mailing list. This way, any user is free to ask questions, put forward suggestions or ideas for new imaging functions and publish solutions. At the time of this writing, this community consisted of more than 1000 users/scientists (the majority) and users/developers (in smaller numbers, but of fundamental importance).

Indeed, the know-how of the community is remarkably high because many members possess a detailed knowledge of the software and the imaging problems that the program can address efficiently. A request on the mailing list is usually all it takes for someone, somewhere in the world, to provide the required help.

For example, Dr. Rex Couture of the department of radiology at Washington University School of Medicine in St. Louis ran into a problem with reading large micro-CT images. These files were about 5 GB, but ImageJ couldn’t read beyond the 4-GB limit at that time. “I posted a query about this,” Couture said. “Within an hour or two, someone working through the night in Europe had found the problem. By 9:30 the next morning, I had a solution to try from Wayne. Since he didn’t have an image file that large, he couldn’t test it, and it didn’t quite work. Twenty-four hours later he had created a test image, and I had a fully debugged new version that solved the problem.” Shortly thereafter, someone in the discussion group also posted a way to open large tiff files.

There has always been a somewhat terse usage instruction on the ImageJ Web site, but getting advice on a specific function required taking the quite intimidating step of asking the mailing list. Now one has the option of using a manual recently produced by Tony Collins at the Wright...
Cell Imaging Facility at Toronto Western Research Institute. This free, very thorough manual has many examples and illustrations. It emphasizes microscopy and accompanies a collection of microscopy-related plug-ins. Although many open-source programs lack an extensive user manual, even novices can find most of the information for their imaging needs in this manual.

Extensions: Macros and plug-ins

The program is virtually limitless because of the availability of user-written macros and plug-ins.

---

**TABLE**

<table>
<thead>
<tr>
<th>Format</th>
<th>Read and write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze (Mayo Clinic’s format)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>AVI uncompressed movies</td>
<td>✓</td>
</tr>
<tr>
<td>Bio-Rad Z-series</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>BMP</td>
<td>✓</td>
</tr>
<tr>
<td>DICOM (uncompressed) read</td>
<td>✓</td>
</tr>
<tr>
<td>DICOM (uncompressed) write</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>DM3 (Gatan Corp. transmission electron microscopy format)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>FDF, VFF (Varian MRI imaging system/EVS 900 micro-CT system formats)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>FITS (NASA format)</td>
<td>read</td>
</tr>
<tr>
<td>GIF (including animated)</td>
<td>✓</td>
</tr>
<tr>
<td>ICO, CUR (MS Windows)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>IPLab (Scanalytics Corp. microscopy format)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>Jpeg</td>
<td>✓</td>
</tr>
<tr>
<td>Jpeg EXIF digital camera header</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>Leica SP multichannel</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>LSM (Zeiss LSM 510 confocal microscopes)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>PCX (Zsoft Corp.)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>PDS (Planetary Data System)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>PGM (P2 and P5)</td>
<td>read</td>
</tr>
<tr>
<td>PICT (Apple Mac)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>PNG</td>
<td>✓</td>
</tr>
<tr>
<td>PSD (Adobe Photoshop)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>QuickTime movies</td>
<td>✓</td>
</tr>
<tr>
<td>Raw (any uncompressed file format not directly supported, 8-, 16-, 24-bit RGB, 32-bit, small- and big-endian)</td>
<td>✓</td>
</tr>
<tr>
<td>SIF (Andor Technology spectroscopy format)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>SPE format (Roper Scientific cameras)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>Sunraster</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>Targa</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>Text images (tab or comma separated)</td>
<td>✓</td>
</tr>
<tr>
<td>Tiff uncompressed and zip compressed, both as 1-, 8-, 12-, 16-bit unsigned, 32-bit (real) or RGB color, single and multiple images (stacks)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>XBM, XPM (X-Windows formats)</td>
<td>(plug-in)</td>
</tr>
<tr>
<td>XLS (Microsoft Excel)</td>
<td>(plug-in)</td>
</tr>
</tbody>
</table>

---

Macros are meant to make it easier to automate oft-repeated tasks, which would be tedious to implement manually. ImageJ has an easy-to-use macro-language that means that knowledge of Java isn’t required for writing simple scripts. For example, a macro can be written that acquires an image every 10 seconds and stores it in a sequence (Figure 3).

Plug-ins are external programs, mostly written in the Java language, that offer image processing capabilities that do not exist in core capabilities of ImageJ. Once implemented, they cannot be distinguished from the program itself. A small cottage industry has sprouted from users/developers who are designing plug-ins for their own use and sharing them with all users. Plug-ins have brought ImageJ from an image processing program to a framework that scientists can use to develop their own imaging solutions.

Plug-ins range from very small and straightforward, such as the Grid plug-in — which simply draws a grid on an image — to complex, practically stand-alone image programs, such as the dendrite-tracing tool NeuronJ, or the surface and volume-rendering plug-in VolumeJ. In fact, the standard imaging capabilities that ImageJ comes with are implemented as plug-ins as well. Developing new plug-ins requires knowing Java language and is thus not for everyone.

However, after using ImageJ for a while, users often need a specific solution for their problem that goes beyond its core capabilities. The Medientechnik und design group in Austria has written a tutorial specifically as an introduction to writing plug-ins for ImageJ.

Most plug-ins that are judged by their developers to be of general use can be published on the ImageJ Web site, which now holds more than 150 plug-ins, written by 98 developers. Some developers may instead make their plug-ins and macros available through personal Web sites; some of these are listed on the ImageJ site.

---

**Imaging library**

A third way of extending ImageJ, which is used by only a small number of more technically advanced developers, is to use its imaging capabilities and plug-ins from their own programs, so that, in technical terms, they are using it as a library of imaging methods. This is called ImageJ’s application programmer’s interface (API), which has been extensively documented so that it is clear how to use and carry out these methods. This way, several online image database servers have been

---

**Figure 3. This simple ImageJ macro acquires an image every 10 seconds and stores it in sequence.**

```java
for (i = 0; i < 5; i++) //5 frames
{
next = getTime () + 10 * 1000; //10s between frames
//Run plug-in that grabs from a Quicktime framegrabber
run ("QT Capture", "grab");
while (getTime () < next) wait (1); //until 10s are over.
```
developed, including one for ophthalmologic telediagnosis.

The applications to which ImageJ has been applied are astounding. Space limitations dictate that only a few salient examples can be given here: Paulo Magalhães and co-workers in the department of biomedical sciences at CNR Institute of Neuroscience, University of Padua, Italy, are using it to study the dynamics of intracellular calcium, while researchers at the Laboratory for Cellular Neurobiology of the Swiss Federal Institute of Technology in Lausanne and the Biomedical Imaging Group at Erasmus MC-University Medical Center Rotterdam in the Netherlands, are using the NeuronJ plug-in, for the automated quantitative analysis of neurons and dendrites (Figure 4). The NeuronJ plug-in is based on recently developed and validated algorithms for detecting and linking elongated image structures specifically for this purpose.

Andrea Mothe and co-workers at the department of zoology at the University of Toronto at Scarborough, in collaboration with Collins from Toronto Western Research Institute, use ImageJ and the VolumeJ plug-in for the 3-D reconstruction of the differential localization of nerve cell gene expression (Figure 5).

Procter & Gamble in Cincinnati is using ImageJ as a framework for Visia, a system that carries out complexion analysis by comparing skin texture analysis of a subject’s facial skin against the texture analyses of other subjects stored in a database. The program uses the ImageJ application programmers interface (API) as a library (Figure 6).

The Biomedical Imaging Group of the Swiss Federal Institute of Technology is
using the program to teach image processing to engineering students,5,6 and the Center for Image Processing in Education of Tucson, Ariz., is using it for teaching basic imaging to high school students.7

Dr. Michael D. Abràmoff and co-workers at the departments of ophthalmology and radiology at University Hospital Utrecht in the Netherlands, have used ImageJ and the FlowJ and VolumeJ plugins for the differentiation of orbital tumors and for measuring the motion of soft tissues in patients.8

Abràmoff also worked with other clinicians to develop the EyeCheck Web site, an online diabetic retinopathy screening project in the Netherlands. It uses ImageJ’s capabilities for storing and displaying retinal images in a telediagnosis environment. The program can accept and store a great variety of image formats, which can then be used for manual grading by ophthalmologists or for semiautomated or automated detection of diabetic retinopathy (Figure 7).

Finally, Dr. Peter Hurd at the department of psychology, University of Alberta in Edmonton, Canada, is using ImageJ to determine the second to fourth digit ratio (index to ring finger) in Anolis carolinensis, or anole lizard, which is thought to reflect the relative concentration of (or sensitivity to) androgens during development (Figure 8).

The program illustrates that imaging is on the boundary between being a field of science and a field of engineering. The solutions being proposed by and implemented in conjunction with users and developers are sometimes engineering solutions (in that they are derived straight

---

**Figure 6.** The Visia complexion analysis system uses ImageJ to analyze how the client compares with a database of women in her age and skin-type groups for the given categories—in this example, spots. It includes scores that indicate the percentage of women who scored below the client’s evaluation for the category. Provided by Procter & Gamble.

**Figure 7.** The EyeCheck Web site provides online diabetic retinopathy screening from retinal color photographs. The left area of the image shows the Web page with patient information. When this page is accessed, an ImageJ applet automatically displays the retinal photographs (in this case, of a normal subject), at right. This allows for user interaction so that, for example, the intensity image can be displayed, as in the bottom part, or specific abnormalities can be marked. Courtesy of EyeCheck.

**Figure 8.** ImageJ is being used to examine the second to fourth digit ratio (index to ring finger) in the anole lizard, which is thought to reflect the relative concentration of or sensitivity to androgens during development. Provided by Peter Hurd, University of Alberta.
from textbook material or a publication) and sometimes scientific solutions (when necessity dictates a scientifically new approach to solve a real-world problem).

A critical and pragmatic reader may expect some drawbacks to ImageJ, and there are some issues. The program requires minimal computer knowledge for installation and first steps, while commercial vendors may offer on-site installation and training. Also, because of the continuous state of development, bugs and “undocumented features” can creep into the distributed version. This can be a problem for the unaware researcher who compares data acquired with the old and new versions. Users usually spot these problems, and corrected versions of the program are made available immediately.

There is a misconception that an imaging program written in Java cannot be fast, and this may divert some potential users from ImageJ. Abràmoff used to share this worry and tested its validity by rewriting some of his convolution routines (which he had originally implemented in Java) in C++ and calling these routines from a plug-in using the ImageJ Java Native Interface API. Although there was a decrease in processing time of about 30 percent, this did not weigh up to the increased development time for the routines.

Furthermore, there is no inherent reason why an algorithm coded in C++ will run significantly faster than the same algorithm coded in Java. The execution speed depends on how good a compiler is at optimizing the generated code. Many people will even argue that the compilers that translate Java byte code into machine code can do a better job of optimization because more information is available to them about the program being compiled and about the machine that it is running on. So the reason why some commercial image processing programs are appreciably faster than ImageJ at convolutions or similar processor-intense operations is related to increased sophistication of their algorithms. The open nature of ImageJ, however, may enable end-users with inspiring insights to develop better and more sophisticated algorithms, without the constraints of proprietary code.

In summary, ImageJ has attracted a varied and dedicated group of users because it is free and expandable, and can operate on any platform. It is especially remarkable how robustly the framework, designed six years ago, has withstood the test of time. Though it is difficult to predict where the program will be five years from now, the evolution will probably be a very interesting and rewarding experience for both users and developers.
Function and dynamics of cells and organelles

Imaging calcium and signal transduction

Dental imaging

Tumor differentiation and soft-tissue motion measurement

Retinal image analysis

Brain and fat tissue imaging

Neuroscience

Craniofacial surgery simulation

Simulation of cell growth patterns

Telediagnosis and image servers