Community Experience Distilled

Image Processing with ImageJ

Second Edition

Extract and analyze data from complex images with ImageJ, the world's leading image processing tool

Jurjen Broeke
José María Mateos Pérez
Javier Pascau

[PACKT] open source*
PUBLISHING
community experience distilled

Image Processing with ImageJ

Second Edition

Extract and analyze data from complex images with ImageJ, the world's leading image processing tool

Jurjen Broeke

José María Mateos Pérez

Javier Pascau

[PACKT] open source
PUBLISHING
community experience distilled

BIRMINGHAM - MUMBAI

Image Processing with ImageJ

Second Edition

First published: September 2013

Second edition: November 2015

Production reference: 1241115

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78588-983-7

`www.packtpub.com`

Credits

Authors
Jurjen Broeke
José María Mateos Pérez
Javier Pascau

Reviewer
Jan Eglinger

Commissioning Editor
Neil Alexander

Acquisition Editor
Manish Nainani

Content Development Editor
Sumeet Sawant

Technical Editor
Parag Topre

Copy Editor
Karuna Narayanan

Project Coordinator
Shweta H Birwatkar

Proofreader
Safis Editing

Indexer
Monica Ajmera Mehta

Graphics
Disha Haria

Production Coordinator
Conidon Miranda

Cover Work
Conidon Miranda

About the Authors

Jurjen Broeke has a PhD in neuroscience from Vrije Universiteit (VU) Amsterdam and uses live-cell imaging techniques to study the fundamental processes of neuronal function. As a neuroscientist, he studies the processes involved in neural communication. Besides acquiring images, Jurjen also develops software to analyze dynamics in ImageJ, MATLAB, and R. When not enjoying the outdoors and taking pictures, he develops technical hardware and software solutions in the Department of Functional Genomics at VU.

José María Mateos Pérez is a Spanish postdoctoral fellow at the Montreal Neurological Institute (`http://www.mcgill.ca/neuro/`), where his main research lines deal with neurodevelopment and machine learning applied to clinical prediction. He has also been an experienced ImageJ user and has developed several macros and plugins. One of them, jClustering, has been published in PLOS ONE, a peer-reviewed journal. When he has enough time to procrastinate, he also likes to develop data analysis tools in Python and R.

Javier Pascau received his PhD from Polytechnic University in Madrid in 2006 and is currently a visiting professor at Carlos III University, Madrid. He has been a part of Biomedical Imaging and Instrumentation Group, a research laboratory with a multidisciplinary team of engineers, physicists, biologists, and physicians located both in the university as well as Hospital General Universitario Gregorio Marañón (biig.uc3m.es). Javier's research and teaching cover areas such as medical image processing, analysis, quantification, and multimodal registration, both in preclinical and clinical environments. He has been involved in the development of small animal PET and CT devices. In the last few years, Javier has led several projects on intraoperative radiation therapy and image-guided surgery. He has authored more than 40 papers published in peer-reviewed journals over the last 15 years.

I want to thank all my colleagues at the university and the hospital, since my knowledge on image processing is the result of multiple interactions in this multidisciplinary environment.

About the Reviewer

Jan Eglinger works as an image processing specialist at Friedrich Miescher Institute for Biomedical Research in Basel, Switzerland. Jan received a master's degree in biotechnology from ESBS in Strasbourg, France, and a PhD in cell biology from MPI-CBG in Dresden, Germany. He has been contributing to Fiji and ImageJ development since 2010.

Table of Contents

Preface

Advances in image processing are vital for the science and technology communities. However, as images become larger and more complex, even more advanced processing techniques are required. Automation becomes necessary too so that you can perform simple tasks easily and focus on more sophisticated issues. ImageJ is here to help—as one of the key powerful tools in the development of image processing, it lets you extract even more useful data from your images.

What this book covers

Chapter 1, Getting Started with ImageJ, takes a look at the origin and use of ImageJ and discusses how to download and install it on different platforms. We will also take a look at the basic folder structure of ImageJ installation and configure it to be used.

Chapter 2, Basic Image Processing with ImageJ, discusses the different image types that are supported by ImageJ. You will also learn how to load images from a disk or URL. We will take a look at the anatomy of an image window in ImageJ and the information that can be viewed. It will also deal with image scaling, calibration, lookup tables, adjusting image size, and adjusting channels.

Chapter 3, Advanced Image Processing with ImageJ, investigates the processing of different types of images. We will take a look at different sources of noise that can corrupt images and degrade their quality. You will also learn how to apply different corrections to images to fix these problems.

Chapter 4, Image Segmentation and Feature Extraction with ImageJ, looks at the ways to separate an image into a foreground and background. We will consider different methods to set the threshold in grayscale and color images.

Chapter 5, Basic Measurements with ImageJ, considers some methods to measure the parameters within images and time series. We will apply some of the techniques discussed in previous chapters to extract data from our images. You will also learn how to visualize dynamic data in a single image (kymographs).

Chapter 6, Developing Macros in ImageJ, discusses how to create a macro using a recorder to discover the commands and functions that we can apply. Next, we will take a look at processing a folder full of images and saving the resulting images to the hard disk. Finally, we will look at the Batch Process mode, which allows ImageJ to process a folder in a similar way.

Chapter 7, Explanation of ImageJ Constructs, looks at the framework of macros and plugins that are available in ImageJ. We will discuss some of the constructs that the ImageJ API exposes for use in scripting and plugins. Finally, we will describe how to set up an IDE to develop ImageJ and plugins using it as a standalone or Maven-based project.

Chapter 8, Anatomy of ImageJ Plugins, takes a look at the anatomy of plugins for ImageJ1.x and ImageJ2. We will also take a look at some of the specific constructs that are used in plugins for both frameworks. This chapter examines how to compile, run, and debug plugins using the IDE or tools provided by ImageJ.

Chapter 9, Creating ImageJ Plugins for Analysis, develops a plugin from scratch using the Maven system and NetBeans IDE. We will discuss how to add a basic user interface to our plugin, allowing the user to change some of the parameters that influence the way the plugin functions. We will also add an external library to provide additional functionality that was not present in ImageJ.

Chapter 10, Where to Go from Here, sums up the topics that are discussed in previous chapters and provides further resources that are available for you to continue developing your own plugins. The chapter also looks at some of the more advanced techniques that are available for developers.

What you need for this book

You'll need the following software for the book:

ImageJ 1.4x or Fiji

- NetBeans 8.0.2+

Who this book is for

This book is created for engineers, scientists, and developers eager to tackle image processing with one of the leading tools in the field for image processing and analysis. No prior knowledge of ImageJ is needed. Familiarity with Java programming will be needed for readers to code their own routines using ImageJ.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "The two most important folders are the `macros` and `plugins` folders."

A block of code is set as follows:

```
varmyTools = newMenu("My awesome tools",
newArray("Macro_1", "Macro_2", "-", "Macro_3"));

macro"My awesome tools - C037T0b11MT7b09aTcb09t" {
  cmd = getArgument();
  if(cmd== "Macro_1")
  runMacro("/PATH/TO/Macro_1_tool");
  else if(cmd == "Macro_2)"
  runMacro("/PATH/TO/some_other_tool");
}
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "We can now perform the particle analysis by selecting **Analyze | Analyze Particles…** from the menu."

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Getting Started with ImageJ

Welcome to the second edition of *Image Processing with ImageJ*. ImageJ is a versatile and open source software package designed for scientific image processing and analysis. It is written in the Java programming language, allowing for a uniform cross-platform experience. It is based on the NIH Image software package on the Macintosh platform, developed in 1987 by Wayne Rasband. Rasband, who is still an active contributor of ImageJ, published the first ImageJ distribution in 1997. It was developed as a project to provide a solution to a problem. In 2012, ImageJ celebrated its twenty-fifth birthday with a publication in the journal Nature Methods.

ImageJ distributions

Currently, there are different distributions that are based on or are extensions of the original ImageJ. The basic ImageJ package is available on the ImageJ website at the National Institute of Health (`http://imagej.nih.gov/ij/download.html`). The current version of the package is **version 1.50b**, and the website is updated monthly. This is the core distribution of ImageJ, which contains the main interface and all the basic tools to load, view, process, and export images and data. Other distributions contain this core package and most of its features, but you need to add additional features and plugins to create an optimized interface for specific fields. Some of these other distributions are still easily recognizable as ImageJ, while others offer a completely different interface.

For different scientific fields, different distributions were developed based on the core of ImageJ. One of the major distributions for the life sciences is called **Fiji** (Fiji Is Just ImageJ), which can be found on the Fiji website (`http://fiji.sc/Fiji`). The basis of Fiji is ImageJ, but it comes with a large complement of preinstalled features (macros and plugins) that are commonly used for image processing in the life sciences. It is focused on fluorescence microscopy, with built-in tools for segmentation, visualization, and co-localization. It also contains plugins for image registration, particle tracking, and super-resolution processing and reconstruction. It also has an extensive library of image formats that can be opened. This library includes proprietary image formats from all the major acquisition software packages via the **Bio-Formats** plugin, as described in the upcoming section. The advantages of this distribution are the large number of supplied plugins that come with it as well as a very user-friendly script editor. It also has an extensive update mechanism for both ImageJ as well as some plugins.

For the field of astronomy, a different distribution of ImageJ was developed, named **AstroImageJ** (`http://www.astro.louisville.edu/software/astroimagej/`). This distribution takes the core implementation of ImageJ and supplements it with specific plugins and macros developed for analysis in the field of astronomy. It is not directly compatible with ImageJ. The core of ImageJ was slightly modified for this distribution.

An example of a distribution derived from ImageJ but with a different user interface is **Icy** (`http://icy.bioimageanalysis.org/`). The Icy distribution has integrated ImageJ, and many plugins are compatible. However, not every plugin developed for ImageJ will work within Icy and vice versa. In the Icy distribution, there is a strong emphasis on cellular and spot detection and tracking. There is also a strong emphasis on plugin development. Plugins that are developed for the Icy platform will have documentation and automated updating implemented by design. There are also possibilities for users to directly provide feedback to the developers from within the interface, which is a feature not present within other distributions based on ImageJ. A disadvantage may be that it requires several external libraries to be installed, most importantly VTK, which can cause issues on Linux systems.

Another distribution that uses ImageJ not only for the processing of data but also aids in the acquisition of data is called **µManager**, which can be found at `https://www.micro-manager.org/`. It is loaded from within ImageJ as a plugin, but provides a unique interface geared towards image acquisition and hardware control. Camera and microscope drivers allow the control of supported hardware used in image acquisition, which can then be fed directly to ImageJ for processing and analysis. An example of the use of µManager is in the Open SPIM project, where it is used to control a DIY light sheet microscope, acquire images, and process them.

The uses of ImageJ

ImageJ is a great tool to process images and perform analysis. It is used in many scientific peer-reviewed publications, with over 1000 articles in diverse fields such as life sciences, astronomy, and physics. In life sciences, it is used to quantify medical images to aid in the detection of pathological markers. It is also used to process and quantify data from single-cell or single-molecule experiments using super-resolution techniques such as **STORM** and **PALM**. In physics and engineering, it is used to quantify and visualize data obtained from atomic force microscopy. For astronomy, ImageJ is used to analyze images obtained from telescopes and satellites and visualize data obtained from observatories. NASA's Jet Propulsion Laboratory hosts a central node with a good collection of data that is available for download at `http://pds.jpl.nasa.gov/`. It contains information on the planetary missions as well as other research fields such as atmospheres or asteroids.

As it supports a large number of different image formats, it is a great image viewer and allows a great number of pixel-based operations. It also supports images with bit depths greater than 8 or 16 bits per channel. However, it is not meant for anything other than pixel-based operations. If you wish to use vector-based operations, then ImageJ is not the tool for you (unless you wish to develop this functionality).

Besides the common tools for image processing, such as cropping, rotating, and scaling, it supports images with multiple dimensions. Images with up to five dimensions can be processed and saved. These dimensions can include channels (multiple colors), frames (time points), and slices (*Z* planes), and any combination of these dimensions. Currently, multipoint acquisitions are not supported (different locations in a larger *XY* space). It is also possible to change the intensity of pixels displayed by adjusting the brightness and contrast, or the color-coding of the pixels (**Lookup Tables**). More advanced techniques to correct image acquisition artifacts, such as background and bleaching, are also available.

The default image format of ImageJ is the **Tagged Image File Format** (**TIFF**). This format allows for the storage of multidimensional data and supports many meta-information fields for calibration, data acquisition information, and descriptions. It can also store information about elements such as overlays. Graphical annotations are placed on the image in a separate layer. Measurements will benefit from the calibration included in the image, allowing for a fast feedback of sizes in the appropriate unit. It is, however, less suited for different kinds of mixed data such as video files. Using the **FFMPEG** plugin allows you to open and save the image data of a video but not the audio track(s). Also, editing is limited to a small set of transitions and layering techniques. For editing videos with image and sound, non-linear editors are available. They allow for greater control.

It can also be used as an image-conversion tool. Many image formats can be read natively by ImageJ, and with the help of a plugin, many proprietary formats can be opened. Once the image is opened, it can be saved to any of the supported export formats supported by ImageJ, including, but not limited to TIF, JPG, and PNG for images and AVI and MOV for **time series** and **Z-stacks**. It can also be used to change the order and/or color of images exported by other software. It is, however, not meant as a general photo editor or nonlinear video editor, as it lacks some of the specialized tools required for these workflows.

The current state of ImageJ

Currently, ImageJ has been cited in more than 200 publications since the beginning of 2015, in fields ranging from physics and engineering to medicine and biology. Many publications are about newly developed plugins that were specifically developed to solve a problem within a certain subfield of science. On the ImageJ website, the page that lists plugins has more than 1000 plugins available. A few research institutes even have collections of multiple plugins available that were developed there as research projects. Most, if not all, are open source plugins with the full source code available. You can adjust and customize the code to suit your needs.

ImageJ2

ImageJ is still under active development, and new features and bug fixes are added to the core distribution on a regular basis. Currently, the development of a revised system for ImageJ is being developed. It is called **ImageJ2**. The goals of ImageJ2 are to better support multidimensional data as well as create a more extensible platform that can be used as a library instead of a standalone application. It will also create a more consistent environment for development and extension. One of the features being developed is the updating mechanism for ImageJ. Currently, it is possible to update ImageJ automatically using a central repository, and one of the goals of ImageJ2 is to expand this option to plugins and other features and allow tracking of bugs and features. However, one of the core requirements in the new ImageJ2 system is backward compatibility. This goal means that plugins developed now will stay functional in future releases of ImageJ. The current status is indicated as **beta**, which means the plugin is functional but may still contain bugs and is not optimized for performance yet.

SciFIO and OME-XML

Other developments related to image are those related to image formats and standards. Currently, all major commercial acquisition platforms store image data in unique proprietary file formats. The SCIFIO project is aimed at creating an extensible and integrated interface to handle images of different formats. It will support more image formats and allow for additional options to be set when importing the data, such as autoscaling, loading metadata, and loading the data in different ImageJ image types. However, it is still under active development, and some of the features do not quite work in a production environment (yet).

The **OME-XML** (**Open Microscopy Environment-XML**) project is aimed at creating a file format that contains all the image and metadata in a standardized format. This would facilitate the exchange of microscope image data, regardless of the equipment used for acquisition. It is mainly focused on the exchange of microscopy data in the field of life sciences. It contains all the experimental and setup data as well as the pixel data in a single file specification.

Bio-formats

Besides the OME-XML format, which is focused on integrating acquisition and processing across multiple acquisition platforms, there is also active development of the plugin used to import many image formats currently in existence. This plugin, called Bio-Formats, is mostly focused on image formats from the life sciences. However, it also supports **FITS** data, which is used in the field of astronomy and space exploration. It currently supports (to different degrees) 140 different image formats and converts them to the OME-XML format for use in ImageJ.

Integrated environment for acquisition and processing

As ImageJ is such an extensible application for acquisition, processing, and analysis, it is impossible to deal with all the options and extensions. In this edition, I will focus on image processing and analysis. I recommend the Fiji distribution for people beginning with ImageJ, as it contains a large number of useful features that allow you to get off to a running start. Another advantage is the presence of the script editor supplied with Fiji, which has many features that some of the larger Java development suites also provide. These features mainly include syntax highlighting and smart indenting. The editor also includes a selection of macro and plugin templates that allow for a basic framework to start with.