An Introduction to Image Analysis in Microscopy

CECAD Imaging Facility Course, v1.1

Nikolay Kladt

March 9, 2015

Imaging Facility
Cluster of Excellence – Cellular Stress Responses in Aging-Associated Diseases
University of Cologne

Preface

Goals

Working with digital images is part of modern microscopy and many imaging techniques rely on digital image processing. Storing, handling, processing and quantifying therefore become part of a scientific procedure and basic knowledge is essential to protect results integrity and prevent scientific misconduct.

> *Scientific image analysis is not photo editing.* (Rossner and Yamada [2004])

In this course, basic knowledge and techniques for digital image processing are covered, focusing on microscopy image analysis and maintaining scientific integrity.

Acknowledgments

This script is heavily based on great existing resources that already cover most of the content of this course. These are cited, where applicable, as most content is very basic and covered in pretty much every textbook or tutorial. However, two resources deserve specific mentioning:

1. The Fiji Website:
   This website contains a lot of information about how to use Fiji, as well as necessary theoretical background. This should be one of the main resources if you want to continue your studies after this course.

2. The textbook 'Basics of Image Processing and Analysis' by Kota Miura, Centre for Molecular & Cellular Imaging, EMBL Heidelberg.
   This textbook has been used in courses for many years and this course content has been heavily influenced by the EMBL textbook. If you are interested, check out the more advanced EMBL course materials that cover Macro programming or Plugin development or the EMBL BIAS courses.

Why a different textbook?

In general, the EMBL textbooks cover all aspects and could have been used for this course. However, we wanted to include information on manuscript figure preparation

and image data integrity. Also, we adjusted the existing course contents to the duration of our introductory course.

ImageJ & Fiji

This textbook was written using the Fiji distribution of ImageJ (IJ version 1.49h, Java 1.6.0_24). Fiji, ImageJ and plugins are the hard work of their respective developers. Whenever you publish work using Fiji, ImageJ, or various plugins, make sure to cite their work (see websites for more information). This not only gives credit to the developers, but also helps them to maintain funding and working on these projects in the future!

Contents

Module 1

Publishing microscopy data

In this module, we will work through some necessary basics of image processing and explore common ways to visualize microscopy data. After this module, you should feel comfortable working with your images in Fiji and be able to use this knowledge to prepare your data for publication.

Digital images are ubiquitous and nearly everybody is used to taking pictures with a digital camera or smart-phone and also storing and processing these images. The same way that digital imaging has replaced analog cameras in our daily lives, light microscopes have transformed to digital imaging systems[1]. Even the most simple light microscopes are usually equipped with a digital camera and a screen to display, take and store images - also allowing quick and simple image processing tasks without the need for additional software (e.g. brightness & contrast adjustments, white balancing). More complex light microscopes, such as confocal laser scanning microscopes (CLSM), already require sophisticated software interfaces to set up imaging parameters and visualize the results. Finally, there are also types of microscopes that depend on digital image processing (e.g. stochastic optical reconstruction microscope, STORM or single plane illumination microscopy, SPIM).

As a result of these advancements, modern digital microscopy offers more than just 'pretty pictures' - there are many imaging techniques that rely on computational analysis as well as methods for image enhancement, restoration and quantitative analysis. Modern microscopy cannot be understood without the basics of digital images and digital image processing. This knowledge is not only essential to make full use of the capabilities, but also to understand limitations – a very important prerequisite to maintain scientific validity.

[1] In this course, we do not discuss the technologies that actually generate digital images in microscopes. If you want to learn about various microscopy techniques, we recommend looking at the online iBiology Microscopy Course (www.ibiology.org/ibioeducation/taking-courses/ibiology-microscopy-course.html, 25-02-2015)

Raster and vector images

Typically, most pictures you work with (scanned documents, photos taken with a digital camera, pictures found on the web) are raster graphics (see Fig. 1.1). This means that these images are made up of a grid of *pixels* (picture elements). Each pixel has a value that represents some property of the image (usually brightness). The range of this value is called *bit-depth* and determines possible values at each pixel. Sometimes, the term *bitmap image* is used for a raster image. For example, the image shown in figure 1.1 illustrates a raster image with 420 x 420 pixels and a bit-depth of 1 byte. Therefore, we have a total of 176400 pixels, multiplied with 1 byte taking up a total of 172 kilobytes disc space. As you can see, the number of pixels and the bit-depth affect the required disc space. Additionally, we cannot easily change the size of the image; enlarging causes the image to look 'pixelated' or we have to use interpolation techniques to estimate the value of added pixels. Decreasing the number of pixels in the image might result in a loss of image features. There are plenty of commercial and open-source programs to work with raster images; representative software packages are Adobe Photoshop (Adobe Inc., San Jose) and GIMP (GNU Image Manipulation Program, The GIMP team, http://gimp.org).

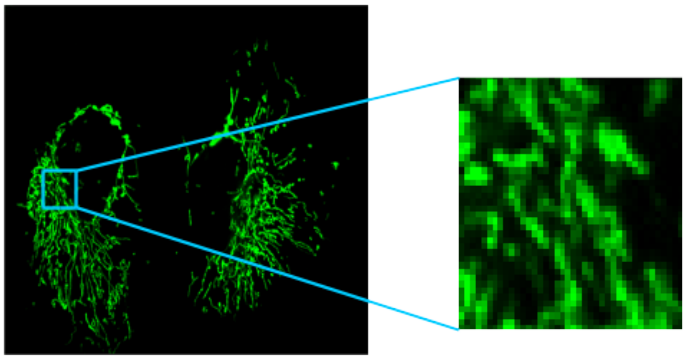

Figure 1.1: Fluorescent labeled mitochondria in HeLa cells. THis image shows that raster graphics are made up of pixels that become clearly visible when zoomed.

In contrast, vector graphics are not made up of a bitmap. Instead, they are based on geometrical shapes such as lines, curves, polygons or other shapes that are based on mathematical expressions. Therefore, these images can be scaled without a loss in image quality and typically use less disc space than raster images. Especially when the scaling of lines, text or other image content might change, it makes sense to use vector graphics (e.g. figures for a manuscript, scientific poster). Similar to the editors for raster images, there are plenty of vector graphics programs; representative exam-

ples are Adobe Illustrator (Adobe Inc., San Jose) and Inkscape (The Inkscape team, http://www.inkscape.org).

Often, programs support the conversion between raster and vector graphics. These are called *rasterization* and *vectorization*. Usually, vector graphics programs allow you to include raster-graphics without conversion.

Let's say you're done for the day and want to submit another great manuscript to Nature (Nature Publishing Group). Nature has detailed requirements on how your figures should be prepared for print (http://www.nature.com/nature/authors/gta/3c_Final_artwork.pdf, as of 08.10.2014). According to their guide, they prefer vector graphics for text and line-art. We will revisit the Nature figure requirements as we go along discussing properties of digital images!

Can you explain why journals prefer vector graphics for text and line-art?

1.1 Pixels, spatial resolution and sampling frequency

It is important to remember that our microscopes have an optical resolution[2] that defines the ability to resolve details of the specimen.

Limited by the optical resolution, the number of discrete pixels in a digital image then determines the spatial resolution. The number of pixels is usually called the sampling frequency (or sampling interval). This simply means that the spatial resolution depends on the number of pixels within given physical dimensions and that the maximum spatial resolution is limited by the optical resolution[3]. It is important to note that each pixel represents the average response of the optical system measured at a point within an area that is specified by the characteristics of our optical system.

In a typical confocal microscope (not camera based), we can choose the number of pixels with which we want to sample the acquired image. A change in the pixel number obviously changes the size of the pixel and therefore also the area of which the average response is obtained. Choosing the wrong sampling frequency is a typical pitfall in microscopy, especially when zoom controls are used. If we sample inadequately, details can be lost. Oversampling is usually not critical, we just increase the amount of data we have to handle and we have to know that we oversample. Fortunately, sampling theory tells us exactly which sampling interval (pixel size, pixel number) is required

[2]For this course, we assume that you have a basic understanding why microscopes have an optical resolution and on which parameters the optical resolution depends.

[3]This is only partially true – we can theoretically build a microscope where the detector is the limiting factor. In general, the detector is chosen to match the optical resolution; and all internal microscope components involved in generating the digital image have to be matched as well.

to optimally represent the biological specimen given our optical resolution (the famous *Nyquist theorem* states that we require a sampling interval $\geq$ twice the highest spatial frequency in our specimen and the highest frequencies we can observe are physically limited by our optical resolution).

This raises the question why our microscopes even allow us to change the sampling frequency and not always simply operate at the diffraction-limited configuration. The reason is, that there are many experimental situations where it is desirable to intentionally lower our spatial resolution. Basically, we have to make a trade-off between the required resolution and imaging speed, amount of light collected at each pixel, phototoxicity or other experimental parameters. The important thing is that we are aware of the fact that we under-sample for whatever reasons we have.

The limited number of discrete pixels in an image acquired with a microscope has serious consequences when you create figures for a manuscript. Publishers often require a certain number of pixels per cm (inch) to create high-quality figures and unfortunately, they leave it up to you to adjust your figures. However, they usually ignore the fact that you only have a limited number of pixels in your original data. One option you have is resampling your original data to change the number of pixels. Another option is to show pixels enlarged – leading to a pixelated view of your data.

1.1.1 Resampling Images

Resampling an image changes the original data; the number of pixels is either increased or decreased (enlarging, shrinking). In comparison, using the zoom tool (icon magnifying glass) just increases or decreases the size of the pixels on your screen without resampling. Similar to other programs, Fiji also allows you to change the size of the canvas (the space on which your image is drawn, see Figure 1.2. If you adjust the canvas size `[Image > Adjust > Canvas Size]`, the image is either cut (pixels outside the canvas size are cut off) or a margin is added where added pixels usually have the background color (zero value).

As you saw above, we determine the number of pixels during image acquisition according to our needs and the limits of the microscope. However, when you prepare your data for a presentation, poster or manuscript you will usually have to scale your original data. Before we discuss best practices and look at an example, we will introduce important basic terminology.

Image resolution outside microscopy

In printed images and in document scanners, resolution is often given in *DPI (Dots Per Inch)*, the number of dots within a line that spans 1 inch (2.54 cm). Computer screen, tablet or phone resolution is typically given in *PPI (Pixels Per Inch)*, the number of

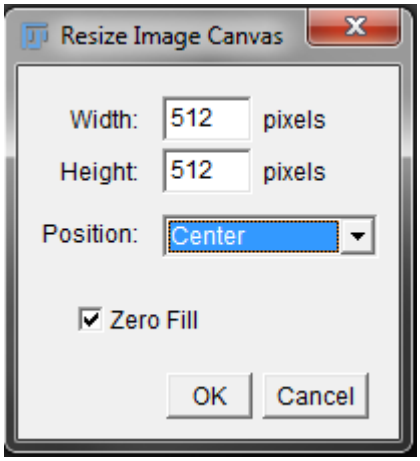

Figure 1.2: Adjust Canvas Size Dialog.

pixels within a line that spans 1 inch. Confusing, DPI is often used when PPI would be better, and even more confusing, one printer dot is usually not equal to one pixel!

As we learned in section 1.1, knowing the number of pixels is not sufficient to define the resolution – pixel size (sampling rate, distance of pixels, ..) is necessary as well. This is also true for consumer products such as digital cameras, smartphones or flatscreen TVs. Knowing that a Full HD TV is capable of displaying 1920 x 1080 pixels presents no information about the size of each pixel. An 18 mega-pixel camera that is able to generate photos with 5184 x 3456 pixels also has no resolution associated. For screens, we can calculate the PPI by dividing the number of pixels by the size of the screen. For example, a 40" Full HD TV would have 55.07 PPI, an IPhone 5 about 326 PPI (4", 1640 x 1136 pixels but two dots per pixel!), an Amazon Kindle Paperwhite 212 PPI (6", 1024 x 758).

High quality photo printing requires about 200-300 PPI. With the number of pixels of our digital camera, we can calculate the maximum size of a high-quality print. For example, an 18 mega-pixel camera print with at least 200 PPI:

> 18 mega-pixel = 5184 x 3456 pixels, divided by 200 PPI results in a print of about 65 x 43 cm (1 inch = 2.54 cm).

Publishers usually require that you submit your figures in 300-600 DPI (meaning PPI), depending on the content (e.g. see the requirements of Nature[4] or PNAS[5]). This already determines the maximum size of your (sub)figure given available pixels.

[4]http://www.nature.com/nature/authors/gta/3c_Final_artwork.pdf, as of 08.10.2014
[5]http://www.pnas.org/site/misc/digitalart.pdf, as of 04/03/2015

Exercise 1.1: Resampling Example - No Interpolation

1. Open the file resampling-test.tiff in /module1/data. This is a 20x20 pixel black-and-white (binary) image. Use the magnification tool to zoom to the maximum magnification. You should now see a one pixel wide and a two pixel wide vertical white line and a 1px diagonal line.

2. Before we perform image manipulations, we duplicate the original image for convenience `[Image > Duplicate]`.

3. Go to [Image > Adjust > Size], you should see a dialog as shown in figure 1.3.

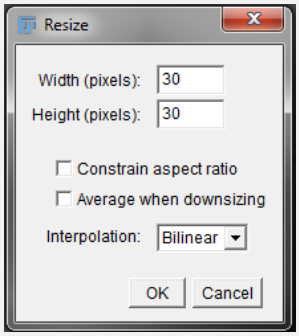

Figure 1.3: Adjust Size Dialog.

4. Perform a resize to 30 x 30 pixels (150% size), with no interpolation, and compare the result with the original figure. Use the `Line-Tool` (see fig. 1.4) to measure the width of both vertical lines. You should observe that one line was not scaled while the other was scaled to 150%.

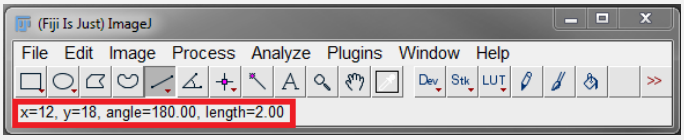

Figure 1.4: Line tool selection and values.

5. Try other values for the resizing and observe the results.

Exercise 1.2: Resampling Example - With Interpolation

1. Again, work on a duplicate of the resampling-test.tif image.
2. Adjust the size to 150% with interpolation set to 'Bilinear'. Use the `Point-Tool` and move the mouse over the image. On the bottom of the Fiji bar, you should see the mouse position in pixels and the value of the current pixel (Figure 1.5).

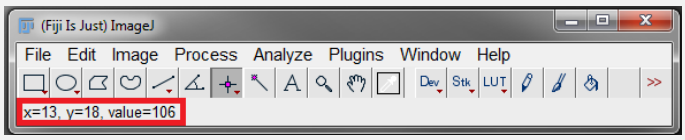

Figure 1.5: Pixel position and value.

3. While the interpolation helps to visually estimate the 150% re-sampling in the vertical and diagonal lines, you can see that the original data has been changed.
4. Try other values for the resizing and observe the results.

As you should have observed, the bilinear interpolation leads to gray values appearing in the previously black-and-white image. Fiji supports the bilinear and the bicubic interpolation. Both interpolation algorithms sample pixel values surrounding each pixel to calculate the pixel value at the given position[6].

Let us go back to our main question of making figures for a manuscript that we want to publish. PNAS recommends a minimum resolution of 300 DPI for color images and further suggests not to use interpolation when changing the size of the image. What should we do with our microscope images? Following guidelines should help you decide what to do:

- Best practice: *no image resampling ever* (e.g., see recommendations of the Journal of Cell Biology). Usually, you have enough pixels (>1000 x 1000 pixels), allowing you to choose an image size within your figure that is sufficient. For example, a 1000 x 1000 pixel image at 300 DPI results in about 130 pixels/cm – you can choose a single column of 89 mm width (Nature requirements again) to display this image. If you then adjust the size of the image in your graphics software, you will only increase/decrease the size of each pixel (like the zoom tool) without resampling.

[6]For more information, Wikipedia has entries for bilinear and bicubic interpolations.

- Never resample more than once during image processing.
- Do the resampling at the end, after all other image manipulations and quantifications have been done.
- Report any resampling procedures, including the original image size, pixel dimensions and the interpolation method.
- Most journals allow supplementary original data - a good way to present the raw, unfiltered data. Also, there is increasing interest to provide platforms that allow you to publish raw data (e.g. Figshare[7])
- Most important if you really have to resample: Convince yourself (and your labmates) that the original image and the re-sampled image convey the same information.

Fiji also has the option to directly interpolate the image to a given size and DPI (PPI!) using `[Image > Adjust > Scale To DPI]`. Using this function, you do not have to calculate the values yourself using the `[Adjust Size]` function (Fig. 1.6).

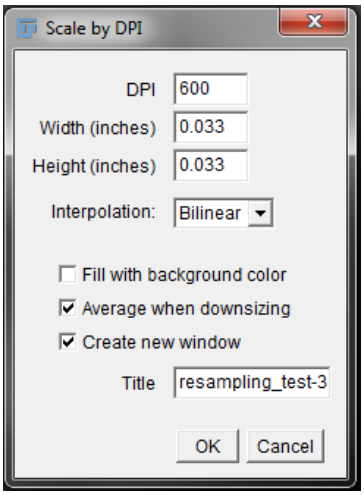

Figure 1.6: Scale to DPI dialog.

[7] www.figshare.com, an online digital repository - free to upload and free to access; as of 13.10.2014

1.2 Bit-Depth & Computer Number Representations

We already introduced the term bit-depth that describes the range of values that can be represented at each pixel. When we refer to the possible values of each pixel, we use the terms brightness or intensity in the course context. It is very likely that you already encountered the words *bit*, *bits*, (kilo-, mega-, giga-)*bytes* somewhere: 32/64-bit computers, 8/16-bit microscopy images or 50 Mbit/s internet connection speed.

A *bit* is the elementary unit of binary numbers and its possible values are 0 and 1. Electronic devices code and store information with binary numbers because binary states (0 and 1) can easily be implemented in electronics[8]. If you have two bits, you can represent 4 different states (00, 01, 10, 11). In general, if you have **n** bits, you can represent $\mathbf{2^n}$ different states. Therefore, 8 bits allow you to represent 256 (2^8) different brightness values and 16 bits already 65536 (2^{16}) different values. The higher the bit-depth of our image, the more gray-levels can be represented between black (0) and white (Maximum value). One *byte* consists of 8 bits (historic and convenience reasons). One kilobyte is a multiple of a byte, either 1000 bytes or 1024 bytes; this can be confusing and usually depends on the context. The same is true for further multiples, such as mega-, giga-, tera-, and petabytes.

Similar to choosing an appropriate pixel size (see section 1.1), we also have to make choices regarding the pixel intensity values during acquisition:

- *Bit-depth:* During image acquisition, we round the brightness values found in our samples into a certain number of levels that our chosen bit-depth can represent. For example, if we have 300 distinct brightness levels in our sample and we choose an 8-bit bit-depth, we cannot represent all 300 values (we only have 256 levels!). In this case, the 300 distinct values are rounded to the nearest level that can be represented. Pixels that have different intensities in our sample get rounded to the same values and the differences are lost. In practice, we do not worry about rounding errors during image acquisition – we usually only have to make the choice between a low bit-depth (8-bit) and a higher bit-depth (12/16-bit). This is discussed in section 1.2.1. However, rounding errors can present problems during image processing. This is discussed in later chapters.
- *Data saturation / clipping:* When you take images on a microscope, you usually set the laser output power, detector gain and detector offset for every image. The reason is that whatever bit-depth we chose, we want the brightest pixels just around the maximum value that can be represented by our bit-depth range and the darkest pixels just around zero. If we use the wrong settings and pixels are saturated/clipped, we loose information about the brightest and darkest parts of

[8] While this statement is true, it leaves out any details. However, there are plenty of resources on the web if you are really interested why and how bits are used.

our sample. This can prevent further analysis of our data! Clipping can also occur during image processing as you will see in a few minutes.

The most common image bit-depths that you can generate with microscopes and that are supported in Fiji are:

- 8-bit images that can display 256 gray levels with whole numbers (Integers).
- 16-bit images that can display 65536 different gray levels with whole numbers (Integers).

In addition, there is a 32-bit image format that allows you to display 2^{32} gray levels with real numbers. However, this format is a little bit more tricky to use as many functions in Fiji do not handle this bit-depth properly.

When you prepare your figures, bit-depth is usually not a big issue. However, you have to be careful when you export/import images using various file formats – some common file formats only support 8-bit bit-depth and you have to make sure that you convert appropriately.

Exercise 1.3: Brightness Adjustments

1. Open the file fibroblast_sim.tif in /data/chapter1. This is a 16-bit gray-scale image showing actin filaments in a cell.
2. You should note that the image looks rather dark. Fortunately, Fiji has a way to adjust the brightness and contrast of an image without altering the original data. Go to `[Image > Adjust > Brightness/Contrast...]`, you should see a dialog as shown in figure 1.7.

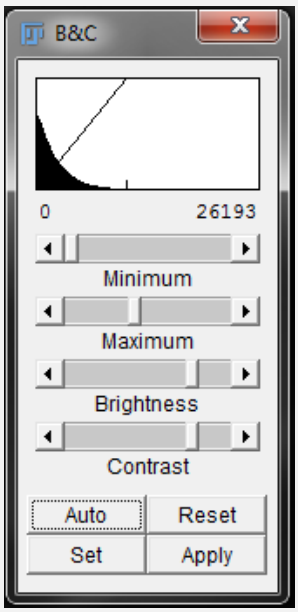

Figure 1.7: Brightness and Contrast Adjustment.

3. Click on `[Auto]`. The image gets brighter as the maximum brightness is now associated with a lower image intensity. This linear scale can be adjusted manually by changing the slider positions. `[Reset]` reverts to the original intensity scaling.
4. Play with the sliders to set the image intensity scaling.
5. Using `[Set]`, we can either enter precise minimum and maximum values or show a defined range (8-,10-,12-,15-,16-bit) and also propagate our selection to all other open images. Again, the original pixel values remain, we only change the display.

Exercise 1.4: Bit-Depth Conversion

1. Open the image beads.tif from mod1/data using `[File > Open]`. Duplicate the image with `[Image > Duplicate...]`. Choose the line selection tool and draw a line through one of the bright spheres in the image (Fig. 1.8).

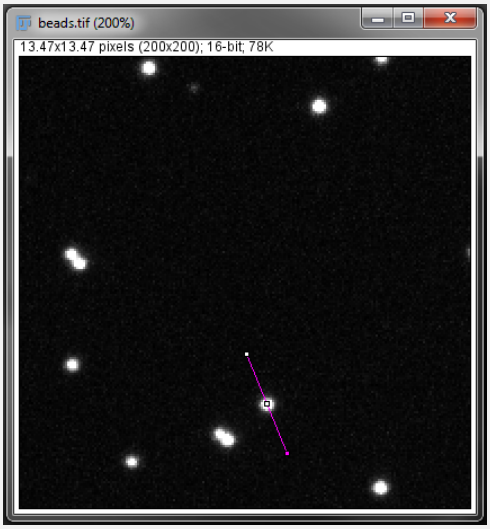

Figure 1.8: A line ROI through one of the beads.

2. In the next step, we will look at the intensity (brightness) distribution along this line. For this, do `[Analyze > Plot Profile]` (Fig. 1.9). You should observe that the gray value (y-axis) ranges from 0 to 65535 and that the curve looks cut at the upper end - this means that we have *saturated* pixels, i.e. we cannot resolve any differences between these saturated pixels although the shape of the curve would suggest intensity changes. The [Plot Profile] function allows you to list (show), save and copy the values. If you click on `[Live]`, you can change the line ROI and the plotted profile will update. Try the update by drawing a line somewhere on the background and then again through a bead. Turn the live mode off again by another click on the button.

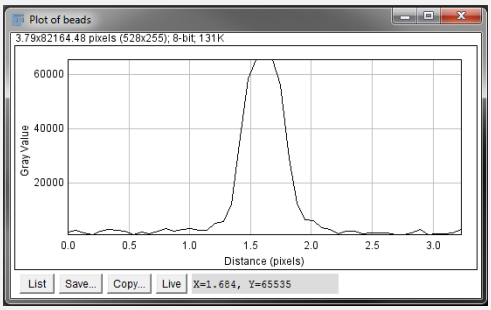

Figure 1.9: Intensity profile of the line.

3. Now, we convert the 16bit image to 8bit. First, we make sure that we scale during conversion by `[Edit > Option > Conversion]`. Then, we use `[Image > Type > 8-bit]` to convert the image. Make sure that the line ROI is still there and perform the plot profile function again. A second window pops up. Compare the plot profiles of the 8-bit and the 16-bit images. The conversion modified the brightness value (y-value). While the profiles are similar, the scaling is different. The image is scaled according to following equation:

$$I_{8bit} = \frac{I_{16bit}(x,y) - min(I_{16bit}(x,y))}{max(I_{16bit}(x,y)) - min(I_{16bit}(x,y))} * 2^8$$

with:
I_{16bit}(x,y): 16bit image
I_{8bit}(x,y): 8bit image
max(I_{16bit}(x,y)): maximum value of 16bit image
min(I_{16bit}(x,y)): minimum value of 16bit image

4. Convert the image back to 16-bit and check the intensity values again. In this case, the intensity values are not increased.

5. The conversion actually looks at the data as it is displayed. Adjust the brightness and contrast to an extreme value using `[Image > Adjust > Brightness/Contrast...]` (contrast slider to the right edge). Convert the image to 8-bit again. Look at the profile of a bead. You should see that the image only consists of 2 intensities: 0 and 255. As you saw, it is important to reset the brightness and contrast display before converting the image.

1.2.1 Choice of Bitdepth

The choice whether to use a low or high bit-depth depends on the application. Of course you could always use the maximum bit-depth available (to be on the safe side). However, this increases the amount of data you acquire. For example, going from 8-bit to 16-bit doubles the required harddisk space. In addition, image processing might be much slower as well! A typical application where a lower bit-depth is sufficient is where you have a very bright signal (signal-noise ration very good) and you just want to identify cells (vesicle, nuclei, ...) for counting or shape analysis. On the other hand, if you want to analyze bright regions as well as darker regions, or need precise intensity comparisons (e.g. for protein density measurements), a higher bit-depth is recommended.

1.3 Image Dimensions

Up to now, we just looked at 2-dimensional images, where each pixel p can be identified by its spatial coordinates, its position along both axes *p(x,y)*. In microscopy, we often deal with more dimensions, common scenarios are:

- **3D z-stacks:** A single acquisition of a 3D volume. The additional dimension is the depth of the section. Each pixel p can be identified by *p(x,y,z)*.
- **3D time-series:** A sequence of 2D image acquisitions. The additional dimension is the time of the acquisition. Each pixel p is identified by *p(x,y,t)*.
- **3D multi-channel images:** A 2D image is acquired with >1 color channel (multiple fluorophores). The additional dimension is the color. Each pixel is identified by *p(x,y,c)*.
- **4D images:** a combination of 4 already discussed dimensions: z-stacks with multiple color channels, sequence of z-stacks over time or 2D multicolor image over time.
- **5D images:** a combination of all discussed dimensions: sequence of z-stacks with multiple color channels over time. Each pixel p is identified by *p(x,y,c,z,t)*.

Fiji allows you to work with all these image types and we will discuss those methods and possible ways to visualize these data in publications in the following sections.

1.3.1 3D stacks

For historical reasons, Fiji (ImageJ) has two different structures to handle 3D data. At first, ImageJ used *stacks* to support 0-3 dimensions. This was done to z-slices or time points. To be more flexible, *hyperstacks* were introduced to support 0-5 dimensions. For compatibility reasons, both structures co-exist at the moment, but the distinction is disappearing with the hyperstack becoming the standard structure to represent multi-dimensional data. As you will see, hyperstacks are used when importing microscopy file formats. If a pixel p defined by three spatial dimensions $p(x,y,z)$, we call the pixel *voxel* (volume element).

Exercise 1.5: Viewing a 3D Stack

1. Open the file flybrain-template.tif in /mod1/data. This is an 8-bit gray-scale z-stack, showing a standard template of a fly brain. You should see that there is a slider below the image to go through individual z-sections of the stack.

2. If the image looks too bright or dark, adjust the brightness (`[Image > Adjust > Brightness/Contrast...]`).

3. Click on the start animation button left to the slider to start an automatic stepping through the sections (Fig. **??** similar to a video). Clicking on the button again, pauses the animation (button icon changes accordingly).

Figure 1.10: Start stack animation.

4. Using the stack toolbar (Fig. 1.11), you can `[Start Animation]` and `[Stop Animation]` as well.

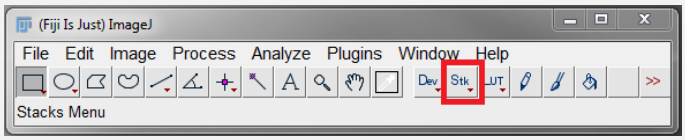

Figure 1.11: Stacks toolbar.

5. Use the `[Animation Options]` from the stack toolbar to increase the animation speed to 20 fps (frames-per-second) and loop back and forth. The animation should look much smoother now (Fig. 1.12).

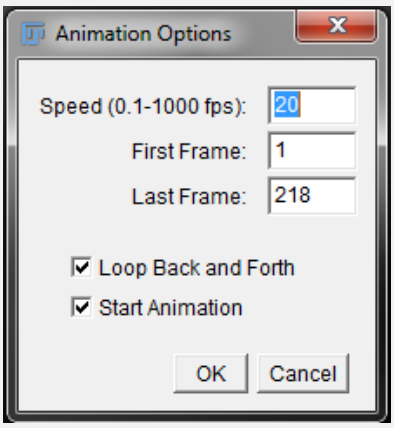

Figure 1.12: Stacks toolbar.

6. When you want to use an animation of the stack in a presentation, you can save the stack as an avi using `[File > Save As > AVI...]`.

It is possible that the image data you import has the dimensions mixed up. If the data is a stack, you can use `[Image > Properties]` to reorder channels, slices and frames. However, it is more flexible to convert the stack to a hyperstack and perform re-ordering on the hyperstack.

Exercise 1.6: Order of Dimensions

1. Open the file flybrain-template.tif in /mod1/data if it is not still open.
2. Use `[Image > Hyperstacks > Stack to Hyperstack...]` to convert the stack to a hyperstack. In our test image, the order, channels, slices and frames should be detected correctly.
3. Now, you can easily re-order the dimensions of the hyperstack using `[Image > Hyperstacks > Re-order Hyperstacks...]`. Although this does not make any sense, change the z-dimension to a time dimension. As you can see, from the user perspective, this does not change anything at the moment.

Fiji offers many ways to manipulate (hyper)stacks. You can concatenate, combine, interleave, insert or split stacks; convert images to stacks and back, remove and add slices, make substacks, create a montage or generate a stack from a montage. These functions can be found under `[Image > Stacks]` and especially `[Image > Stacks > Tools]`.

Exercise 1.7: Manipulating Stacks – Creating a Montage

A common way 3-dimensional data is presented (typically along time or z-axis) is the montage view.

1. Open the file flybrain-template.tif in /mod1/data if it is not still open.
2. Use `[Image > Stacks > Make Montage...]`. In the dialog, set `[Columns]` and `[Rows]` to 5, `[First Slice]` to 100, `[Last Slice]` to 124, change the `[Border Width]` to 1 pixel and tick `[Label Slices]` (Fig. 1.13).

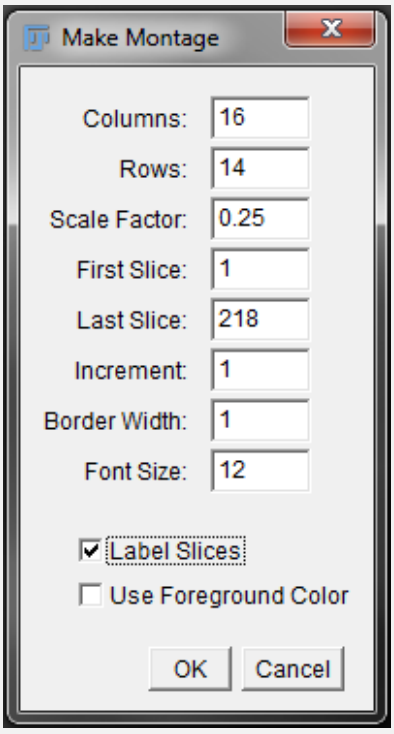

Figure 1.13: Make Montage Dialog.

3. You can see how easy it is to create a custom montage view, play with the different options, e.g. `[Increment]`.

Exercise 1.8: Manipulating Stacks – Creating an Insert

We now want to do something more complicated: let's say we want to show a detail of the flybrain, e.g. the central complex or the optic lobes. To help viewers, we want to put a little version of the complete brain in the corner of our 3D image as an overview. How would you proceed?

1. Open the file flybrain-template.tif in /mod1/data if it is not still open.
2. Use the rectangle tool to select an interesting part of the brain that you want to highlight. Duplicate the complete stack using `[Image > Duplicate]`, this will only duplicate the part you selected.
3. Use `[Image > Adjust > Size...]` to create an image with 1024 pixel width and no interpolation.
4. Go back to our original image of the fly brain and use `[Image > Scale...]` to reduce the image size to 20% (x, y) with bilinear interpolation.
5. The insert is created with `[Image > Stacks > Tools > Insert...]`. Make sure you use the detailed view of brain as `[Destination]` and the overview as `[Source]`. `[X-Location]` and `[Y-Location]` can remain 0 (Fig. 1.14).

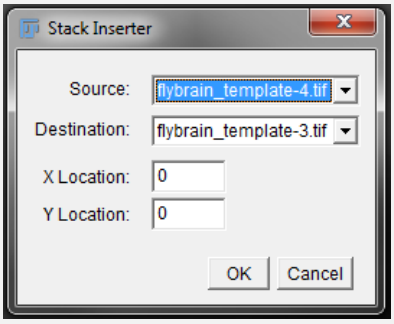

Figure 1.14: Stack Inserter Dialog.

6. Again, a very easy procedure – explore further stack operations on your own.

1.3.2 Color images

So far, we only analyzed images with one color channel (grayscale), i.e. at each spatial position, we only had one pixel with one value. In microscopy, we often encounter multi-channel images where each channel corresponds to one modality. One channel for each fluorophore intensity, a channel for a DIC image or even a channel that does not represent brightness but for example fluorophore lifetime. Outside microscopy, we also usually encounter color images, these are often RGB images that consist of red, green and blue 8-bit channels. Again, publisher demands on the color usage varies, sometimes they demand that the RGB color space you usually work with on your computer is converted to the CMYK color space that is typically used in print. Luckily, as the primary source for distributing and viewing a publication are various digital devices, conversion demands become less common.

Color models

Outside microscopy, two colorspaces are very common:

1. RGB (Red Green Blue)
   A single RGB image has three, fixed color channels: red, green and blue. Each of those have a fixed bit-depth of 8-bit, resulting in a single RGB image with 24-bit. These colors have been chosen as computer screens generate colors by mixing red, green and blue light and therefore, RGB images directly determine how much of each color to use but are device dependent (actually, RGB is based on human color perception).
2. CMYK (Cyan Magenta Yellow Key)
   A single CMYK image is based on four colors: cyan, magenta, yellow and black. The color model is primarily used in printing, but is device-dependent as well. When you want to publish a manuscript, it is very likely that you will be asked to convert your figures to the CMYK color space. This can be done with Fiji using existing Plugins, but it might be better to perform the conversion for the total final figure when all other editing has been finished. As the online versions of publications have become much more important than the actual prints, efforts to produce color-true conversions for printing become much less important.

Exercise 1.9: RGB Images

1. Open the file muscle-cell.tif in /mod1/dataṪhis image was taken from a publication in Nature Cell Biology 5, 598(2003); Cell of the Month: The vascular smooth muscle cell cytoskeleton; Mario Gimona; DOI:10.1038.ncb0703-598. This RGB image shows mouse smooth muscle cell with fluorescent labels of the cytoskeleton. Use `[Image > Show Info...]` for details about this image.

2. Use `[Image > Adjust > Brightness/Contrast]` and change the slider values. Observe that this operation affects all colors simultaneously. Reset the changes.

3. We now split the red, green and blue channels of the RGB image with `[Image > Color > Split Channels]`. Three windows appear, each showing the respective color content.

4. Let's combine these channels again with `[Image > Color > Merge Channels]`. The merge-function gives us many options to create a merged image (Fig. **??**). Do not set any options and use the same color channels.

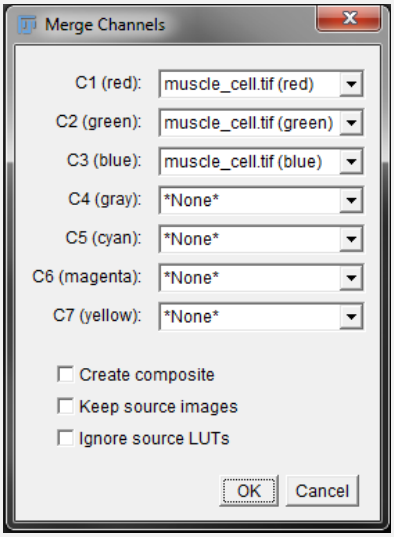

Figure 1.15: Merge Channels Dialog.

5. Now split again and merge back, but with option `[Create composite]` ticked. This creates a slider below the image, indicating that we created a three-layered stack, one layer for each color.

6. The composite image allows us to work on each channel separately. Perform `[Image > Color > Channels Tool...]`. In this dialog, you can select individual channels in composite mode or view individual channels in Color/Grayscale mode (Fig. 1.16). Try out the different settings.

Figure 1.16: Channels Tool Dialog.

7. Try to perform already know operations on just one color channel, e.g. Adjust the brightness (you can see that the little histogram changes color when you change the channel!).

Exercise 1.10: Convert RGB to CMYK – Extending Fiji with Plugins

Although Fiji has a great core functionality included, the power of this tool can only be appreciated with the many existing macros and plugins. While some plugins are already included in the Fiji distribution, you most likely will encounter some function you need that is not included – and if this is a common task it is very likely that someone already solved this problem.

1. Convert the composite image back to a RGB image with `[Image > Color > Stack to RGB]`.
2. We now want to convert the RGB image into a CMYK image for publication. However, Fiji is not able to do so and we have to extend the functionality. A quick search ('imagej convert to cmyk') shows that, fortunately, Stephan Saalfeld and Wayne Rasband wrote a plugin for this (two names you might encounter more often).
3. You can find the file RGB_to_CMYK.class in the folder mod1/data/. Installing this plugin is very easy, just drag and drop the file onto the main Fiji window. You are asked to save the file in the plugins directory of Fiji.
4. Convert the image to CMYK with `[Plugins > RGB to CMYK]`.

Lookup Tables

For a computer, an image is a matrix with the same dimensions as the image and the range of possible values at each matrix position are determined by the bit-depth. A *lookup table (LUT)* maps these values to a color/brightness that is displayed on the screen. Often, the LUT is simply called *colormap*. The default LUT is gray-scale that, in an 8-bit image, assigns black to zero and white to 255. Intermediate values are represented as gray-levels and the same idea applies to any bit-depth. The advantage using a LUT is obvious; we can create arbitrary LUTs with any coloring we want. These colored LUTs are called 'pseudo-color'. These LUTs are typically used to indicate fluorophore emission wavelengths but are sometimes changed to emphasize certain aspects of the data.

Figure 1.19 shows the LUTs available in Fiji, the figure was created by the command `[Image > Color > Display LUTs]`. Very common LUTs are *grayscale*, *spectrum*, *fire* and *physics*.

Exercise 1.11: Exploring Lookup Tables

1. Open the image beads.tif in /mod1/data. You can look at the LUT of the current image with `[Image > Color > Show LUT]` which is the standard linear grayscale LUT you have already seen.

2. Change the LUT using `[Image > Color > Edit LUT...]`. Click on the top-left dark value and change the color from black to blue, select the white entry from the bottom right and change the color to red (Fig. 1.17).

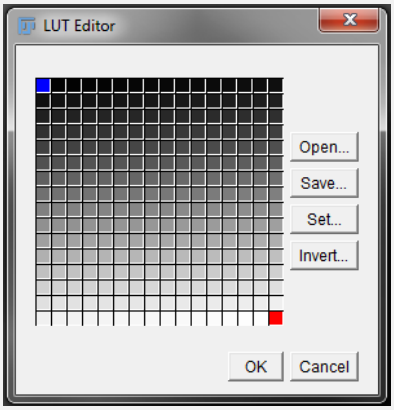

Figure 1.17: LUT Editor.

3. Look at the image. Does it look familiar? This is the HiLo LUT that is often used in microscopy to optimize parameters for acquisition (emitted light, gain, offset). You can obtain the same image by selecting the HiLo LUT in Fiji (Fig. 1.18).

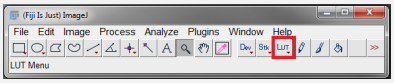

Figure 1.18: LUT Tool.

4. Close the image and open the file cell-colony.tif in /mod1/data. Change the LUT to Spectrum using `[Image > Lookup Tables > Spectrum]`.

5. Display the current LUT of the image. Try out different available LUTs and display their profile.

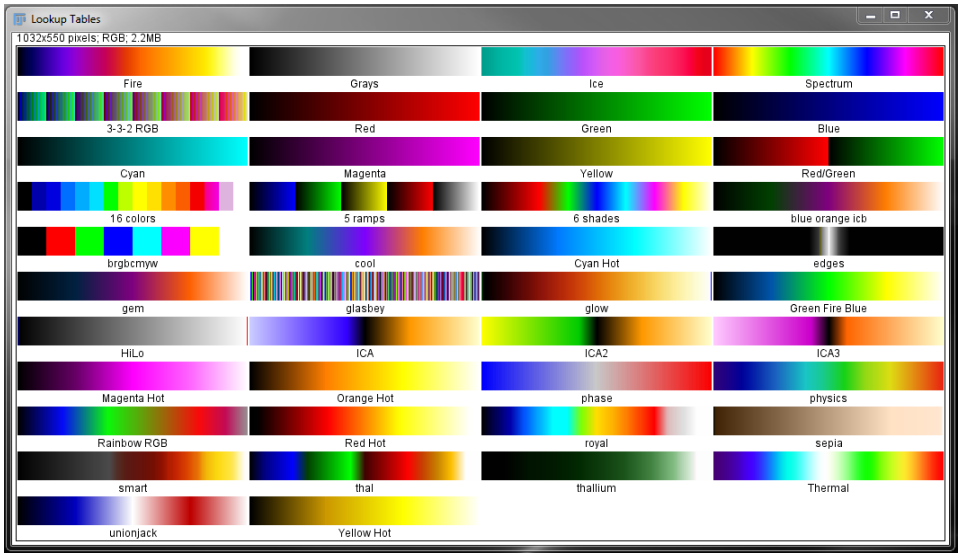

Figure 1.19: Lookup Tables in Fiji

A short note about colors in publications

There is plenty of literature around discussing the best colormaps and general rules one should apply when using colors in scientific publications. We do not want to enter this, sometimes heated, debate, as color choice in microscopy usually comes naturally based on the wavelengths. Nevertheless, we want you to get a feeling why this debate is going on and provide two examples: *Color Bias* and *Color Blindness*. Color bias refers to the problem that poor color choices can introduce a biased view of our data. For example, when we change the grayscale LUT to a spectrum LUT, relative changes of intensity can be obscured. Color blindness addresses the issue that microscopy images, especially illustrating colocalization, are often presented with red and green (all following information on color blindness is taken from Wong [2011]). However, a substantial number of people are affected by various forms of color blindness (about 8% of male Europeans). Therefore, it is very likely that you will have reviewers for your manuscripts or audience in your talk that are color blind. One option to adjust your images is simply to replace the red with magenta.

Exercise 1.12: Effects of LUT Changes

1. Open the image muscle-cell.tif in /mod1/data.
2. Simulate color-blindness with `[Image > Color > Simulate Color Blindness]`. This function only works on an RGB image.
3. Use `[Image > Color > Replace Red with Magenta]` to exchange colors and then simulate color-blindness again. Color differences are much more obvious in the second image.
4. go back to the original image of the muscle and split the channels.
5. Duplicate the window containing the blue channel information.
6. Change the LUT to spectrum and discuss the differences between the gray and spectrum LUTs.

Exercise 1.13: Calibration Bars

A calibration bar is typically added to a figure when intensity comparisons are made. A calibration bar indicates which color corresponds to which brightness. This is especially important when you use LUTs with more than one color or when you use a single color LUT where the change in intensity is not linear ('gamma').

1. Open the image fibroblast-sim.tif in /mod1/data.
2. Adjust Brightness/Contrast and select a LUT you like.
3. Add a calibration bar with [Analyze > Tools > Calibration Bar...]. In the dialog, choose a location, fill color, label color, number of labels, asf. (Fig. 1.20). Note that this calibration bar shows the brightness settings you just applied and that you cannot change those now.

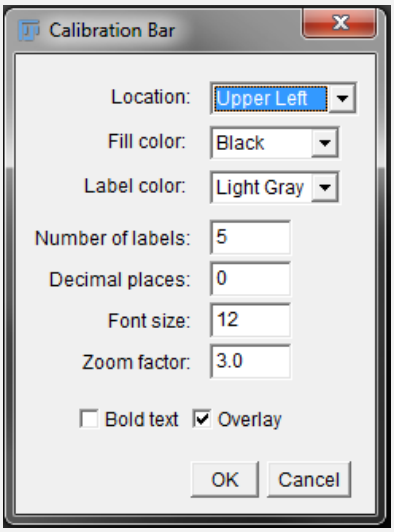

Figure 1.20: Calibration Bar Dialog.

1.3.3 4D/5D stacks

It is common that the data you acquire on microscopes has more than 3 dimensions. In this section, we will first discuss how you work with these data before we will discuss the problem of presenting this data in a figure.

With the hyperstack data format, representation of 4D and 5D data is very similar. Basically, sliders are added to the image window. However, you have to keep the order of dimensions in mind. Common dimension order is XYCZT, however this order might be different when you import data – doublecheck when importing!

Exercise 1.14: Working with 5D Data

To illustrate working with 5D data, we will import a sequence of files that are labeled with _t000_z000_c000 and increasing numbering. The sequence of files has been generated from the standard Fiji sample Mitosis `[File > Open Samples > Mitosis (26 MB, 5D stack)]`. This data shows Drosophila S2 cell expressing GFP-Aurora B and mCherry-tubulin fusion protein undergoing mitosis (Courtesy of Eric Griffis).

1. Go to `[Plugins > Bio-Formats > Bio-Formats Importer]` and select the first image in the folder /mod1/data/mitosis.

2. The Bio-Formats import options dialog shows up, make sure to select [Group files with similar names].

3. In the dialog, you can choose how to stitch the files (Fig. 1.21). We select Pattern. This parses the file based on: mitosis_t0<01-51>_z00<1-5>_c00<1-2>.tif. <> denote the range of the import, e.g. we could only select one channel, or any other substack.

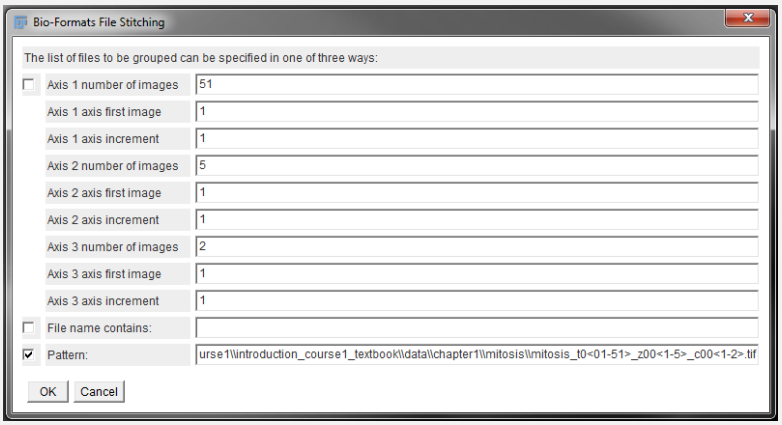

Figure 1.21: Bio-Formats File Stitching Dialog.

4. The imported image shows three sliders: channel (2), z-position(5), time-stamps(51). Browse through the image, adjust Brightness if neces-

sary. On thing to notice is that we lost information about the z-spacing, distance between time-stamps or channel colors.

1.3.4 Visualizing Multidimensional Data

You already worked with the most basic visualization of multidimensional data that only displays two dimensions at a time. In this view, you can browse through the dimensions by adjusting the sliders. Another view we already discussed is the image montage. Both of these visualizations are also valid options to present the data in a figure. In general, visualizing multi-dimensional data is non-trivial, as we are constrained by the physical limitations of our visual sense. In this section, we will look at some common visualization techniques for multi-dimensional microscopy data that are useful to highlight certain aspects of your data in a figure.

Exercise 1.15: Projection (Dimensionality reduction)

The first visualization we will explore is the projection. In a projection, data is summarized along one axis (dimension). A typical case is the maximum intensity projection of the z-axis of a 3D stack, resulting in a 2D image where each pixel represents the maximum value that was found in this *(x,y)* position along z - which is then used for the manuscript. In Fiji, you can choose between the minimum, average or maximum intensity projection and the sum, standard deviation or median of each pixel along z. Remember that you can swap dimensions with `[Image > Hyperstack > Re-order Hyperstack]`.

1. Open the image GMR-10A12-AE-01.tif from /mod1/data. This image shows the expression patterns of a GAL4 line, displayed on a standardized fly brain template (credits belong to the Rubin-lab, JFRC (Arnim Jennet) and The Virtual Fly Brain). Try to estimate the expression pattern (green) by going through z using the slider.

2. In this case, we might decide a z-projection is helpful to quickly determine whether the expression pattern is of interest. We can perform the projection with `[Image > Stacks > Z Project...]`, choosing Max Intensity (Fig. 1.22).

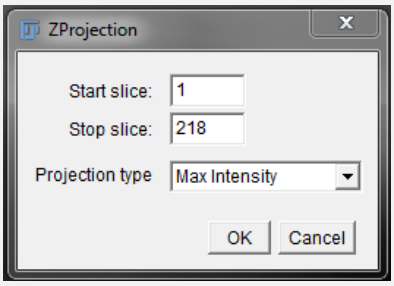

Figure 1.22: Z Projection Dialog.

3. Explore various projections on the stack created from the file sequence in mod1/data/mitosis/.

Exercise 1.16: Orthogonal View

Another very common visualization is the orthogonal view. In this view, two additional windows are created that are linked to the current slider position of the 2D view. These windows show the XZ and YZ position. This visualization can be useful if you want to present the intensity profiles in 3D data while presenting overview images for the orientation of the reader at the same time.

1. Open the stack GMR-10A12-AE-01.tif from mod1/data/ if it is not already open. Show the orthogonal view with `[Image > Stacks > Orthogonal Views]`. Try scrolling through the axes to get a feeling for this view.

Exercise 1.17: Color Coding

One option to add a third dimension on a 2-dimensional plot is to somehow code the information; e.g. using different colors for z-depth or time. This can be useful when the data is structured along on axis, e.g. different layers of cells within a tissue, axons growing into other tissue parts or vesicles moving around over time.

1. Open the image fake-tracks.tif from /mod1/data (another Fiji sample image). We use a fake file to better illustrate how the color coding works, but you can apply it to any 3D data for further testing. Perform [Image > Hyperstacks > Temporal-Color Code] and select a LUT of your choice (Fig. 1.23).

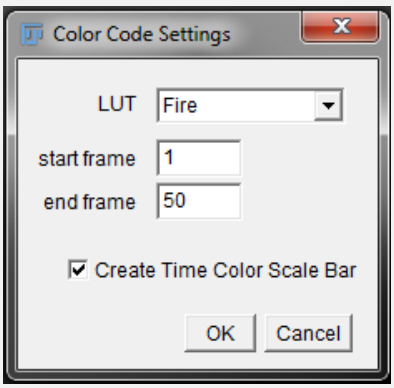

Figure 1.23: Temporal Color Code Dialog.

2. Compare the color-coded image with the original time-series.

Exercise 1.18: Generating a Kymograph Plot

A Kymograph is an visualization to present a dynamic process in a single image where movements along a line are plotted for all time-frames in a stack (x-t plot). Therefore, it is also a way to deruce dimensionality. They are common to show cellular components moving along a some path (e.g. mitochondria moving along an axon or cells migrating in reference to a body axis during development).

1. Open the stack axon-mitos.tif from /mod1/data (Image by Arun Akon-

dadi, Rugarli Lab, University of Cologne). Adjust Brightness. This image shows mitochondria moving along an axon.

2. This image has a problem: the axon was not stained itself, but we can hopefully reconstruct the axon path by the positions of the moving mitochondria. For this, perform a maximum projection over time.

3. The maximum-intensity image helps a lot to estimate the axon position in the image. Right-click on the [Line-Tool] to change the Straight Line to a Segmented Line (Fig. 1.24). Use the segmented line to trace the axon path. A double-click ends the line.

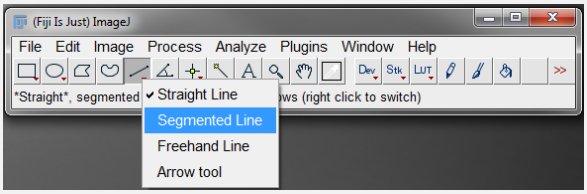

Figure 1.24: Segmented Line Tool.

4. Go to the original stack and do `[Edit > Selection > Restore Selection]`. This restores the line we just selected on the maximum projection on the stack.

5. Use `[Image > Stack > Reslice]` to generate the Kymograph. For display reasons, you can also invert the image with `[Edit > Invert]` and Adjust the Brightness.

6. If the line was placed correctly, you should see something similar to Fig. 1.25. The kymograph helps to distinguish stationary mitochondria and we can sometimes even distinguish anterograde from retrograde movement (in reference to soma).

Figure 1.25: Example Kymograph.

Exercise 1.19: 3D View

Instead of trying to visualize our data in 2 dimensions, we can also visualize in 3D using the 3D viewer in Fiji. For visualizations in 3D, several display options are common: Volume, Orthoslice, and Surface. While this choice is obviously not available for printed figures, online publishing of supplementary videos can be a good option to present your data.

1. Open the stack GMR-10A12-AE-01.tif from /mod1/data if it is not already open. Select the 3D viewer with [Plugins > 3D Viewer]. In the options dialog (Fig. 1.26), select the image and display as a volume.

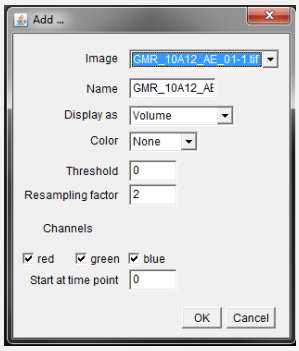

Figure 1.26: 3D Viewer Dialog.

2. Depending on your hardware, this might take a while to display. Try to navigate the 3D view with your mouse.

3. Let us try to generate a movie that you can save as an avi file. Use the 3D viewer menu to create a simple 360 degree rotation with `[View > Record 360 degree rotation]`. This generate a stack that you can now save as a movie using `[File > Save As > AVI]`. In this dialog, you can set the compression as well as the frame rate (how fast the movie is displayed). For our purposes, we set the compression to uncompressed and the Frame Rate to 15 fps (frames-per-second). Movie generation for journals can also be a tricky thing as they often impose strict size limits and requires specific formats. You often need to adjust the movie size as well as the compression algorithm to adhere to these requirements and the choice of both can be complicated. However, you can usually accept some compression artefacts as movies are often not considered raw data but more of a nice additional visualization. Still, make sure that you

adhere to scientific principles and tell specifically how you treated your data to generate the movie.

4. Further instructions on the 3D viewer can be found at: http://3dviewer.neurofly.de/; the website of the developers (as of 28.10.2014).

1.4 File Formats

File formats are an important, but dry topic. Every time you share data with colleagues or during the manuscript submission process, you have to work with various formats. This section discusses some basics of formats, focusing on formats suitable for microscopy data (and more general image file formats).

An image file consists of two main parts:

1. *Header* – The header contains additional information about your image, such as image type, dimensions, bit-depth, pixel size, microscope settings and general file information. The data in the header is often called *metadata*. Parts of the metadata are as important as the pixel data as they are necessary to interpret the image data (e.g. pixel size, bit-depth).

2. *Content* – The content of an image file consists of the actual pixel data, the matrix with numerical values.

The way the header and the pixel data are organized is determined by the file format. Many general image file formats exists, common are JPEG, PNG, TIFF or GIF. These formats are read by many different software tools across platforms and allow quick previews in all major operating systems. They are therefore very suitable to view and share image data. However, there are some problems with these file formats when we want to use them to store our scientific image data:

- The format *metadata* might not include essential information about our microscopy data (e.g. pixel size).

- *Bit-depth* choice can be limited (e.g. GIF only allows 8-bit).

- *Multi-dimensional information* might not be supported and extra-dimensions lost when converted (limits apply to all standard formats).

- The format uses *lossy compression* to shrink file size (e.g. JPEG).

- Sometimes, limits on the *maximum number of pixels* in a plane can prevent use for microscopy data.

Compression

Compression is used to reduce image file sizes. In general, compression methods can be divided into two types: *loss-less* and *lossy* compression. As the names indicate, the original pixel data cannot be reconstructed if lossy compression is employed (information is lost) and the original data can be reconstructed in lossless compression. A typical example of lossless compression is used in PNG. In this format, compression is achieved by shrinking redundant parts. JPEG and GIF are usually lossy formats. In addition to shrinking redundant parts, these formats also interpolate parts of the image to decrease file size even more.

> Lossy compression should be avoided to maintain scientific integrity!
> (Unless you know exactly what you are doing)

Limitations of general image file formats led many microscope and image processing software manufacturers to develop their own image file formats (LIF, SCN – Leica, LSM, ZVI – Zeiss, ND2 – Nikon, ...). These formats store all important metadata information and the original pixel data (only loss-less compression used). However, the drawback is that you typically need proprietary software to view these files and access all metadata and that another software you want to use for analysis or display might not be able to read the original data file.

> The originally acquired files are the most trustworthy – this is your original, raw data
> You should always keep these files with appropriate backup solutions[9].

As you can see, this is a mess for researchers. On the one hand, you want to maintain scientific integrity and adhere to standards concerning your data, but on the other hand, you want to share your data and have easy access for further processing.

Fortunately, the LOCI Bioformats plugin[10] can import most proprietary file formats and extract most of their metadata in Fiji. There is also significant effort by the Open Microscopy Environment (OME) to provide a standardized file format OME-TIFF that includes rich metadata descriptions suitable for microscopy. This is an ongoing process and if you use this format to store and share data, you have to make sure that all important metadata is actually included.

[9] It is also very likely that the funding agency supporting your research has rules how you treat your original data!

[10] http://www.openmicroscopy.org/site/support/bio-formats5/ (as of 24/10/2014)

Journals usually accept common image file formats (if compressed with loss-less compression). I would recommend using a vector graphics program for final figure preparation (think about text/line art) with microscopy images embedded (that have been prepared by e.g. Fiji).

Exercise 1.20: Using Bioformats

Let's import a microscope specific file format into Fiji.

1. Open the image sted-confocal.lif in /mod1/data by drag/drop or with `[Plugins > Bio-Formats > Bio-Formats Importer]`.

2. In the next dialog, you can select various import options (Fig. 1.27). Make sure that you load the data into a Hyperstack and tick the Display metadata option.

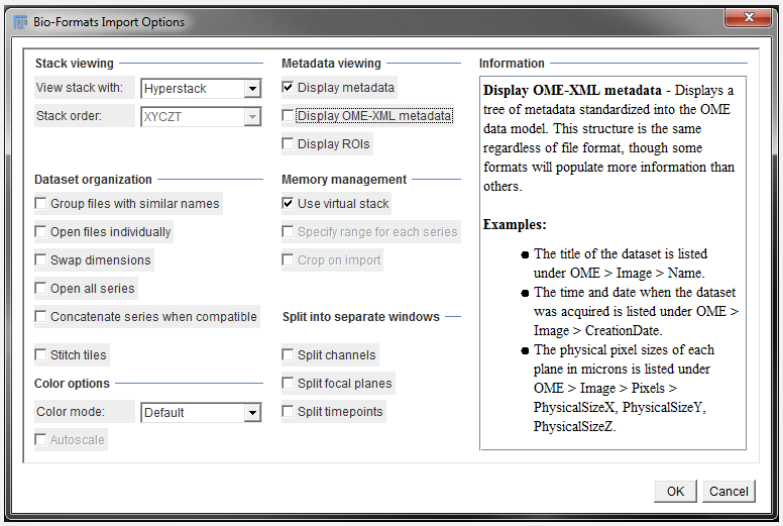

Figure 1.27: Bioformats Import Dialog.

3. The next window allows you to select a series. Several microscope image formats are actually libraries of files. In this case, you should see three different files. Open one of the series.

4. Two windows are opened. One showing the image, the other showing the metadata information. Scroll through the information and find the Dimension Length, Number of Elements and Unit. Use these values to calculate the pixel size.

Exercise 1.21: Scale Bars

Let's explore one final thing you usually have to include in your microscopy data: a scale bar that indicates the size of each pixel.

1. Open the image sted-confocal.lif in /mod1/data by drag/drop or with `[Plugins > Bio-Formats > Bio-Formats Importer]`.

2. We now want to add a scale bar to our image. In this case, Fiji already knows the pixel size - let us check whether our previous calculations were correct. Go to `[Analyze > Set Scale...]`. This dialog should show you how many pixels are in one micrometer (Fig. 1.28).

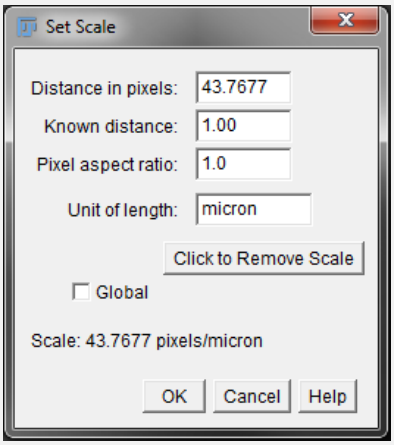

Figure 1.28: Set Scale Dialog.

3. In case you have an image where information about the scale is visible in the image itself, you can then measure the length with a line and include this information in the dialog. Clicking on Global helps if you take one image of a micro-scale and want to use this information in other images you took with the same settings (e.g. on a small Lab-Microscope).

4. Let's add a scale bar now. Use [Analyze > Tools > Scale Bar]. Similar to the calibration bar, the dialog lets you adjust various visual parameters (Fig. 1.29).

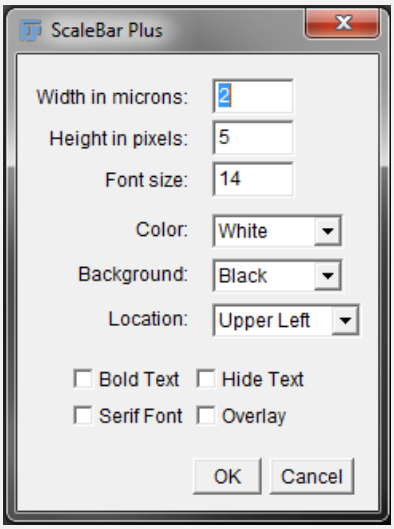

Figure 1.29: Scalebar Dialog.

1.5 Figure publishing plugins for Fiji

If you want to generate your figures completely in Fiji, there are great (published) plugins available that help you with this task (http://fiji.sc/2013-11-04_-_Plugins_for_making_scientific_Figures. as of 25/02/2015). The FigureJ plugin lets you specify a figure canvas on which you can place various images with labels and scale bars (http://imagejdocu.tudor.lu/doku.php?id=plugin:utilities:figu as of 25/02/2015).

Module 2

Measuring intensities

An image contains one type of information: the intensities of the pixels in the image. In this chapter, we will show you how to work with this intensity information in a quantitative way. We will look at intensity distributions and how to perform measurements based on the intensity distributions in regions of the image (e.g. a cell).

2.1 Histograms

Each pixel in an image has an intensity (a pixel value). The intensity histogram of an image is a plot that shows the distribution of pixel counts over a range of intensity values, typically the bit-depth of the image or the range between the minimum and maximum values within the image. The histogram can be used to examine the signal and background levels (useful when you want to identify objects of interest).

Exercise 2.22: Working with histograms

1. Open the image hela-cells.tif in /mod2/data. This is another standard Fiji sample image showing HeLa cells. Lysosomes are red, mitochondria green and the nucleus blue.
2. We can show the histogram with `[Analyze > Histogram]`. The histogram is displayed in a new window (Fig. 2.1). The x-axis shows the pixel value and the y-axis the count. Note that the histogram refers to the channel that was active when the command was called.

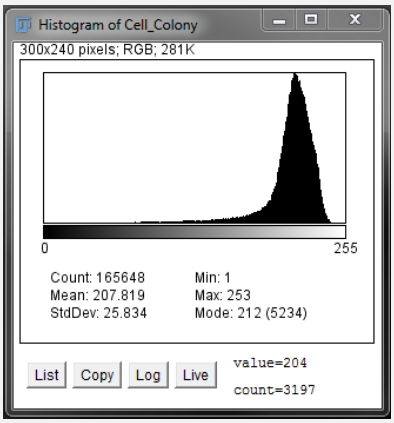

Figure 2.1: Histogram.

3. Obtain the highest and lowest intensities from the histogram. What does the histogram range tell you?

4. Click on [list] to obtain a list with values and counts. [Log] displays the same histogram on a log-scale.

5. Click on [live] in the histogram dialog and change the channel. Observe the changes in the histogram and note the color changes in the depicted colormap bar.

2.2 ROIs and the ROI Manager

We now introduce a very powerful concept that is widely used for many types of analyses - the region of interest. Many operations can be applied to parts of the image when a region is selected by *ROI* (region-of-interest) tools. The shape of the ROI is arbitrary; it can consist of a geometric shape, such as a circle or a line, but can be any selection of pixels. You already used ROIs when you selected a line or a point! Next, we will cover some basic ROI operations.

Exercise 2.23: Working with ROIs

One of the operations you can perform on a ROI is cropping:

1. Open any image. Duplicate the image. Select a region by a rectangu-

lar ROI. Then, perform `[Image > Crop]` on the duplicate. Close the cropped image.

2. Duplicate the original image again. Use the freehand selection to create a ROI on the duplicate. Crop the image. Note that the minimum and maximum values of your ROI have been used to determine a rectangular selection for the cropping.

ROIs can be used to create image masks. An image masks is a binary image that define which parts of the image to analyze or measure.

1. Open any image. Duplicate the image. Create a few circular ROIs (press <shift> during creation to keep multiple ROIs).

2. Use `[Edit > Clear Outside]` and then `[Edit > Fill]` to generate a mask image (this should look similar to fig. 2.2. The 'Clear' command fills the respective area with the background color, 'fill' with the foreground color.

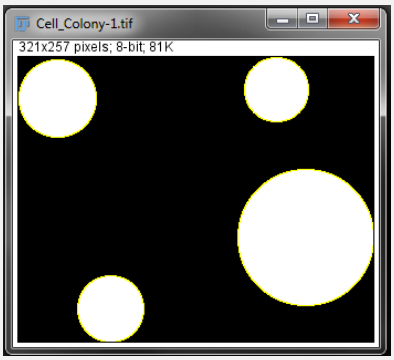

Figure 2.2: Example binary mask.

3. Undo what you have done with `[Edit > Undo]`. You will notice that, in contrast to other programs, Fiji only allows to undo the most recent operation. This is done to minimize memory consumption. `[File > Revert]` allows you to go back to the last saved state.

4. Go back to the original image. Use the command `[Edit > Selection > Create Mask]` to directly generate a mask image.

Operations are usually performed on the selected ROIs and on the whole image if no selections exist.

1. Open any image. Duplicate the image. Select any ROI you like.

2. Observe that `[Edit > Invert]` only affects the selected pixels. Undo the inversion.

Finally, there are several operations that work on a ROI itself without changing pixel values `[Edit > Selection > ...]`, see fig. 2.3. Let us explore a few of those options.

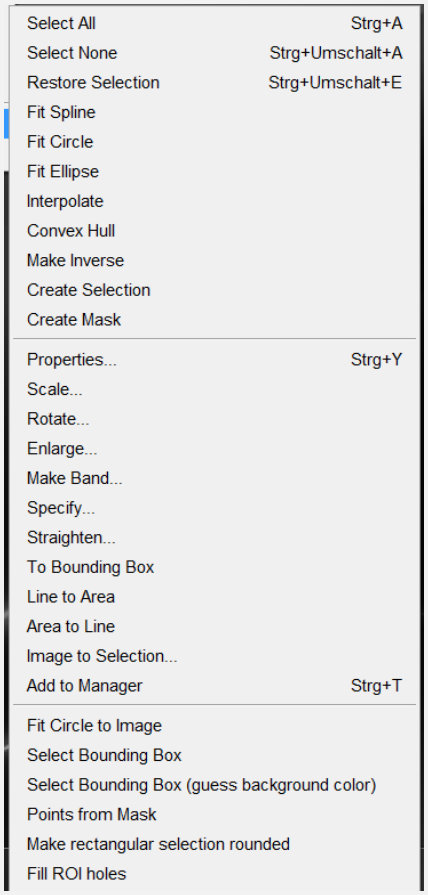

Figure 2.3: Selection options.

1. Open an image, duplicate and draw a freehand selection with an outline that includes inner parts that are not selected. Fit a spline `[Fit Spline]`. This creates a smoothed version of the selection where individual points can now be dragged around and the shape can be changed. This can be useful to correct the outline of a shape (e.g. a worm or a cell).

2. Use `[To Bounding Box]`. This creates a rectangular region that just fits over the selection (this is the same as the crop area). Undo the last operation.

3. Use `[Convex Hull]`. This creates an outline of the selection, where a straight line between every pair of points within the ROI is also within the ROI (Definition of a convex object in Euclidean space). This can be useful if you want to measure the extent of an object with an irregular shape. Undo the last operation.

4. Explore the operations `[Scale]`, `[Make Inverse]`, `[Enlarge]` and `[Rotate]`.

5. Each ROI has properties that can be displayed and changed with `[Properties]`. Especially useful when you work with multiple ROIs (and the ROI Manager, see below), can be ROI names (Fig. 2.4). If you need the ROI coordinates outside Fiji, you can list the coordinates of the ROI as well.

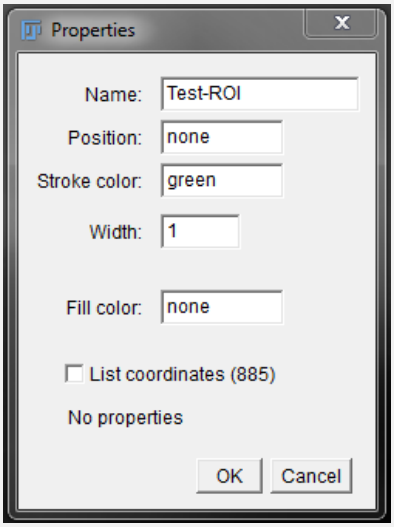

Figure 2.4: ROI properties.

Managing multiple ROIs

Fiji has an ROI manager to assist when you work with multiple ROIs. Open the ROI manager with `[Analyze > Tools > ROI Manager...]`. The manager allows you to add, delete and modify existing ROIs, save and load ROI coordinates, and change their display properties (Fig. 2.5).

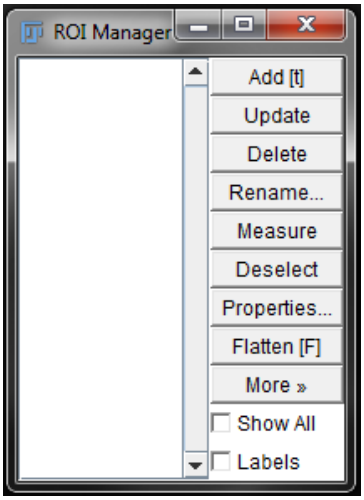

Figure 2.5: ROI manager.

Exercise 2.24: Working with the ROI Manager

1. Open the image hela-cells.tif in /mod2/data.
2. Open the ROI Manager with `[Analyze > Tools > ROI Manager...]`.
3. For the following analysis, we will only work on the bottom-most cell. Select the blue channel showing the nucleus. Use the ROI tools to create an accurate outline of the nucleus and add the selected region to the ROI manager. Rename the ROI to 'Nucleus'.
4. Select the blue channel showing the mitochondria. In this channel, we can estimate the cell outline - create the outline and add the selected region to the ROI manager, rename ROI to 'Cell'.
5. Let's say the task we want to solve is get the area of the cell, excluding the nucleus, for our further analysis. We now explore an option using the ROI manager, we will later explore another way using masks and image math. Select 'Nucleus' and 'Cell ROIs simultaneously. Then go to the [More»] button and select the XOR operation. *XOR* is the *exclusive or* (exclusive disjunction). This is a logical operation that we apply to the ROIs. The XOR function returns the area where both inputs differ, i.e. not overlap. As the 'nucleus' ROI is located within the 'Cell' ROI, this subtracts the nucleus from the cell. Add the resulting ROI to the ROI Manager and rename to 'Cytoplasm'.
6. Select all ROIs and save them on disk - we will use these ROIs for measurements.

2.3 Measurements

The easiest way to read pixel values is moving the mouse pointer over individual pixels and reading their intensity value in the main Fiji bar. The histogram allows us to get the distribution of pixel intensities. Fiji has the function `[Analyze > Measure]` to obtain statistical information about pixel values within a ROI. `[Analyze > Set Measurements...]` allows us to select the parameters we want to measure. For example, following parameters can be obtained (Fig. **??**):

Area Area of selection in square pixels. If the image is calibrated, Area is displayed in according units.

Standard Deviation Standard deviation of intensity values.

Min and Max Gray Value Minimum and maximum intensities in selection.

Center of mass Intensity-weighted spatial average.

Bounding rectangle Bounding box, smallest rectangle enclosing selection.

Area fraction The percentage of highlighted pixels, or the percentage of non-zero pixels.

Mean gray value Mean intensity in selection.

Centroid The average of all x and y coordinates in the selection.

Perimeter The length of the outside boundary of the selection.

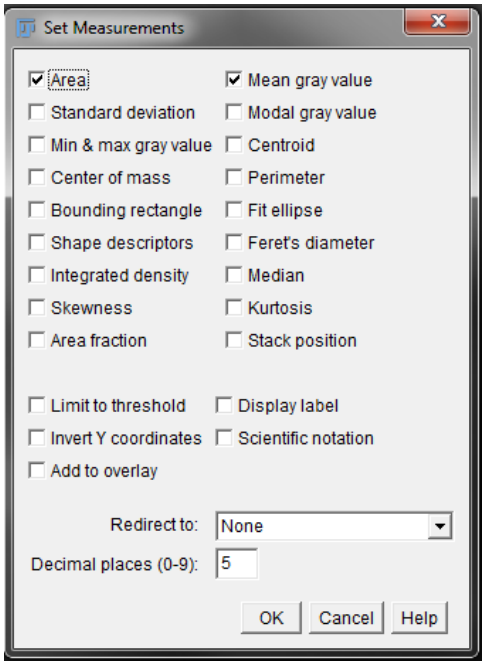

Figure 2.6: Set Measurements Dialog.

Exercise 2.25: Using the ROI Manager

Now, we are going to combine ROIs (and the ROI Manager) with measurements – you will see how powerful this already gets!

1. Open the image hela-cells.tif in /mod2/data. Open the ROI Manager and load the previously generated ROI data.

2. Measure the area and average intensity of the whole cell, the nucleus and the cytoplasm in the green channel and compare the results (Fig. 2.7).

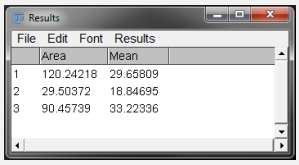

Figure 2.7: Results window showing measurements.

3. Create a ROI in the background of the image (no cell), select the green channel and measure the average intensity again. Name the ROI 'Background' and update the saved ROI file; we will need this for further measurements.

2.4 Changing pixel intensities

We now have seen various times that an image is represented as a matrix and that we can operate on individual pixels. In this section, we explore basic operations that allow us to modify the values of individual pixels in a ROI or the whole image. As you will see, you have to be careful with these operations, especially when you want to measure and/or compare intensities. However, they can be extremely useful to identify interesting regions of your images or enhance details.

2.4.1 Arithmetic

As an image is a matrix of numbers, we can do usual math on those pixels. We will go through one example where this might be useful and one example to show that things can go wrong if we are not careful.

Exercise 2.26: Subtracting background levels

A common task is the subtraction of the background levels of our images to make them comparable (background levels might vary). Subtracting a constant is the most basic background subtraction possible.

1. Open the image hela-cells.tif in /mod2/data. Open the ROI Manager and load the previously generated ROI data that includes the background ROI.
2. Measure the average intensity in the background and note the value.
3. Select the overall image `[Edit > Selection > Select All]` in the green channel.
4. Subtract the average background level from the green channel, using `[Process > Math > Subtract]`. You can also measure the backgrounds of the other color channels and subtract those as well.

Exercise 2.27: Bit-depth/Format Problems

A common task is the subtraction of the background levels of our images

to make them comparable (background levels might vary). Subtracting a constant is the most basic background subtraction possible.

1. Open the image hela-cells.tif in /mod2/data. Duplicate the green channel.
2. Look at the histogram of the green channel.
3. Now, multiply the image by 2.5 - this increases the brightness of the image.
4. Generate the histogram again and compare the number of saturated values. As you can see, if the resulting value of a pixel is clipped if we get a value outside our bit-depth. If the result is a real number but our format only allows whole numbers, the resulting value would be rounded. Of course, this was a rather stupid example, but this is a common mistake when you perform more complicated calculations on your images.
5. if you want, you can convert your image to an appropriate format and compare results.
6. You can also try to perform the previous invert-example without the conversion to 32-bit.

2.4.2 Doing math with two images

In the previous section, we operated with a single image and constant numeric values. Likewise, we can perform computations that involve two images. Assuming that both images have the same dimensions, we can perform calculations on each pair of pixels at the same position. For example, this type of operation can be used to subtract an image showing only (non-uniform) background from the image showing background + data or we can compare two images by obtaining their difference.

Exercise 2.28: Background subtraction

In this example, we are exploring a common method to subtract background information from a time-series. In the given example, we have moving bacteria that were imaged with phase contrast microscopy. The key idea for the

background subtraction is that on each pixel, most of the time no bacteria is visible. We therefore create an average image and subtract this image from the original images to increase contrast of moving bacteria.

1. Open the image bacteria-tracks.tif in /mod2/data. This image was create from supplementary video 1 of the publication: Rosser et al. (2013) Novel Methods for Analysing Bacterial Tracks Reveal Persistence in Rhodobacter sphaeroides, PLOS Computational Biology, DOI: 10.1371/journal.pcbi.1003276. Please note that this was not the original data, it seems that the supplementary movie was compressed!
2. Perform a Z-projection to create an average image.
3. Use `[Process > Image Calculator...]` to subtract the average image from each image in the original time-series and display the difference in a new window.
4. Do you see any differences?

Not surprisingly, the authors of the original publication used exactly this approach as the first step in their image processing.

Exercise 2.29: Math on Masks

Before, we used the ROI Manager XOR function to obtain the cytoplasm ROI. We can obtain the same result using image masks and image calculations (actually this is likely happening behind the scenes anyway!).

1. Open the image hela-cells.tif in /mod2/data. Open the ROI Manager and load the previously generated ROI data that includes the background ROI.
2. Duplicate just the green channel.
3. Select the Nucleus ROI and create a mask (look at the previous exercise if you forgot how to proceed). Duplicate the mask.
4. Now, select the Cell ROI and create another mask.
5. you should now have two image masks (black-and-white images, see fig. 2.8).

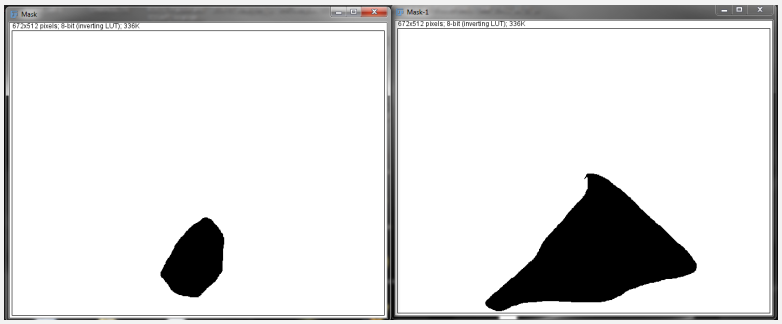

Figure 2.8: Image Masks.

6. Calculate the XOR function using the binary data of both images (`[Process > Image Calculator...]`).
7. Compare the results with the cytoplasm ROI.
8. Using `[Edit > Create Selection]`, you can create a selection from the mask.

2.5 Contrast enhancements

You might already have experience with contrast enhancing of digital images, since most photo-editing software/apps have this function. Low contrast images have small differences in tones and objects are difficult to observe. If the contrast is too high, the tone difference is very high, the picture is 'over-exposed'. Contrast can be adjusted to optimize the appearance of the image. Contrast enhancement primarily changes the LUT, leaving the original data unaffected (it is affected if you apply the changes). Contrast enhancement should be used with care as pixel values are altered and intensity calculations become useless. Therefore, you should avoid contrast enhancements when you analyze intensities (this also applies to the figures that make your point!). Even if you treat all images the same way, you have to make sure that you avoid clipping and that results are not distorted by contrast enhancement.

Exercise 2.30: Contrast Enhancement and Image Manipulation

Let's assume you want to show gel data in your manuscript. After performing following steps, can you discuss why contrast enhancement might be consid-

ered fraud?

1. Open the image gel.tif in mod2/data and duplicate the image (another Fiji sample image).

2. Adjust Contrast and Brightness and compare images. What is the problem with the contrast-enhanced image (Fig. 2.9)?

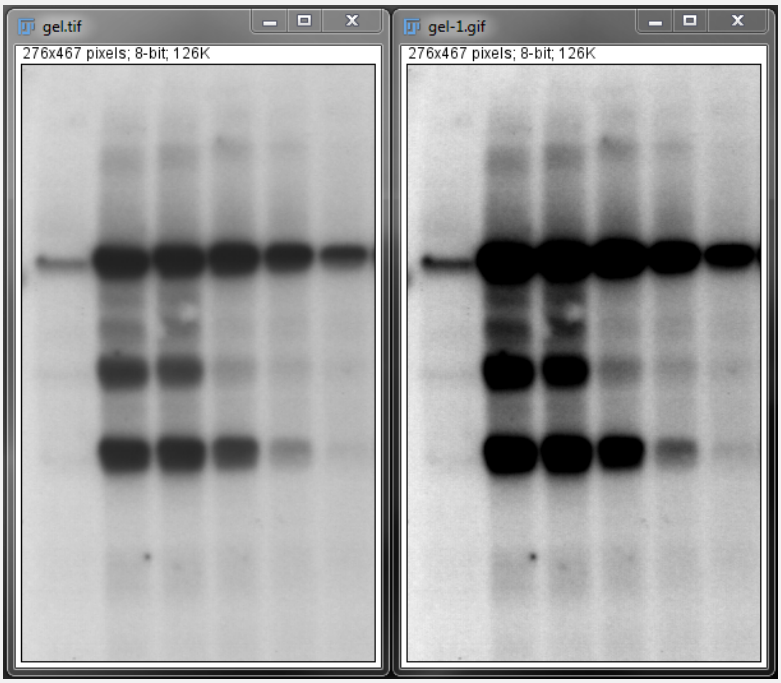

Figure 2.9: Contrast enhancement. The left gel plot shows the original data. Contrast enhancement in the right gel plot shows that you interpretation of this gel might be different.

2.5.1 Histogram Normalization

With normalization, pixel values are normalized according to the minimum and the maximum pixel values in the image and bit-depth. If the minimum pixel value is *pmin* and the maximum is *pmax* in an 8-bit image, then normalization is done as follows:

$$NewPixelValue = \frac{(OriginalPixelValue - pmin)}{(pmax - pmin)} * 255$$

Exercise 2.31: Histogram Normalization

1. Open the image hela-cells.tif in /mod2/data and duplicate the green channel. Obtain a histogram.
2. Use `[Process > Enhance Contrast]` on the duplicate. Set saturated pixels to 0, tick normalize and not equalize (Fig. 2.10) and obtain a histogram afterwards.

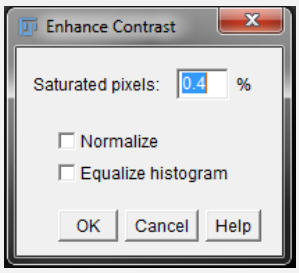

Figure 2.10: Enhance Contrast Dialog.

3. Compare the histograms.
4. Leave the images open; you can also explore different settings.

2.5.2 Histogram Equalization

Equalization converts pixel values so that the values are distributed evenly within the dynamic range.

Exercise 2.32: Histogram Equalization

1. Go to the original image showing the green channel.
2. Use `[Process > Enhance Contrast]` on the duplicate. Set saturated pixels to 0, tick equalize, not normalize and obtain a histogram.
3. Compare the histograms.

2.5.3 Local Histogram Equalization

Histogram equalization could also be performed on a local basis. This becomes powerful, as more local details could be contrast enhanced. In Fiji, you can use local histogram equalization with `[Process > Enhance Local Contrast (CLAHE)]`.

2.6 Colocalization

Module 3

Working with objects - Generalized image analysis workflows

In the module 'Measuring Intensities', you explored how you can manually define a region of interest and perform intensity measurements on this region. While this is already a commonly found approach, the manual labor can be very tedious or impossible with given resources. Another aspect that is often neglected is the possible (and likely) introduction of bias - outside actual clinical studies, a double blind approach is very uncommon. This module will present all components of a generalized image analysis workflow:

1. Obtain raw image data. In this course, we assume that raw image data already exists and that this image data is suitable for the necessary analysis.

2. Image visualization and preprocessing. This typically involves simple pixel operations and multiple image filters.

3. Identify objects of interest (image segmentation). The preprocessed image is used to generate pixel label maps; in a simple case, this is based on a binary image (black = background, white = objects of interest).

4. Manipulate objects. The detected objects are modified and enhanced. For example, this can involve cleaning object boundaries, removing holes, splitting or merging objects.

5. Perform measurements. In the last step, you convert the finally identified objects into numbers for further (statistical) analysis. For example, the derived numbers can be based on intensity, shape, texture or object relationships.

3.1 Image preprocessing

We now generalize this concept of pixel operations by looking at an image as a discrete function. For a 2D image, $f(x, y)$ maps from R^2 to R, giving the intensity at position (x, y). In this view, any pixel operation o can be expressed as $g(x, y) = o(f(x, y))$, where $g(x, y)$ is the modified image. The operations we already know are either simple arithmetic- or histogram-based methods.

Another type of very powerful operations is the *image filter*. Often, image preprocessing involves the (sequential) application of various image filters. Image filters are local operations where the *neighborhood* (surrounding pixels) of each pixel determines the new intensity value. The pixel neighborhood is also called the *kernel*. These filters are mainly used to smooth the image (suppress high frequencies) or enhance edges (suppress low frequencies).

Image filters can be performed on the spatial domain (the image you know with x/y/z axes) or on the frequency domain (Fourier transformed image). For simplicity, we completely skip the Fourier transform, frequency domain filtering and the mathematical reasoning why it can make sense to transform an image into the frequency domain, perform (linear) filtering, and transform the image back to the spatial domain. Furthermore, we will restrict our explanations and examples to 2 dimensional (discrete) images.

3.1.1 Convolution, Correlation & Kernels

Convolution is the mathematical operation behind many of the image filters we will discuss. As described, the basic idea is that a neighborhood of pixels, a window with finite size and shape, is placed on top of each pixel and that the new pixel value is determined by the weighted sum of all pixels within this neighborhood. This window with the weights is called the *kernel*.

Again, we look at the image as a matrix of pixels and do the same for the kernel, the neighborhood with the weights. We then perform the weighted sum of all pixels within the kernel and repeat for every pixel in our image (Fig. 3.1).

If we want to convolve a pixel at position (x, y) of an image F with the rectangular kernel K, centered around (x, y), with width ω, we can write:

$$[F * K](x, y) = \sum_{i=-\omega}^{i=\omega} \sum_{j=-\omega}^{j=\omega} K(i, j) \cdot F(x - i, y - j)$$

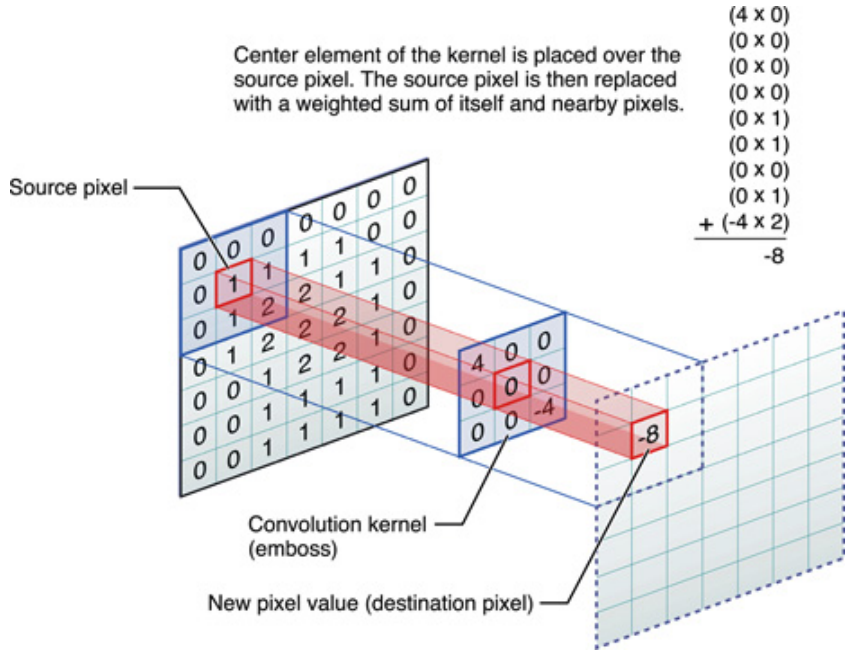

Figure 3.1: Convolution example. A single location of a 2D convolution is shown. The image was taken from The Mac Developer Library [2011]

For better illustration, let us perform the convolution on a small example, we average an image showing a vertical white line on black background with a 3x3 smoothing kernel (this operation is explained below):

$$\begin{bmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \begin{bmatrix} 0 & 57 & 57 & 57 & 0 \\ 0 & 85 & 85 & 85 & 0 \\ 0 & 85 & 85 & 85 & 0 \\ 0 & 85 & 85 & 85 & 0 \\ 0 & 57 & 57 & 57 & 0 \end{bmatrix}$$

Figure 3.2 looks at a few steps of the convolution to illustrate how the resulting image is calculated.

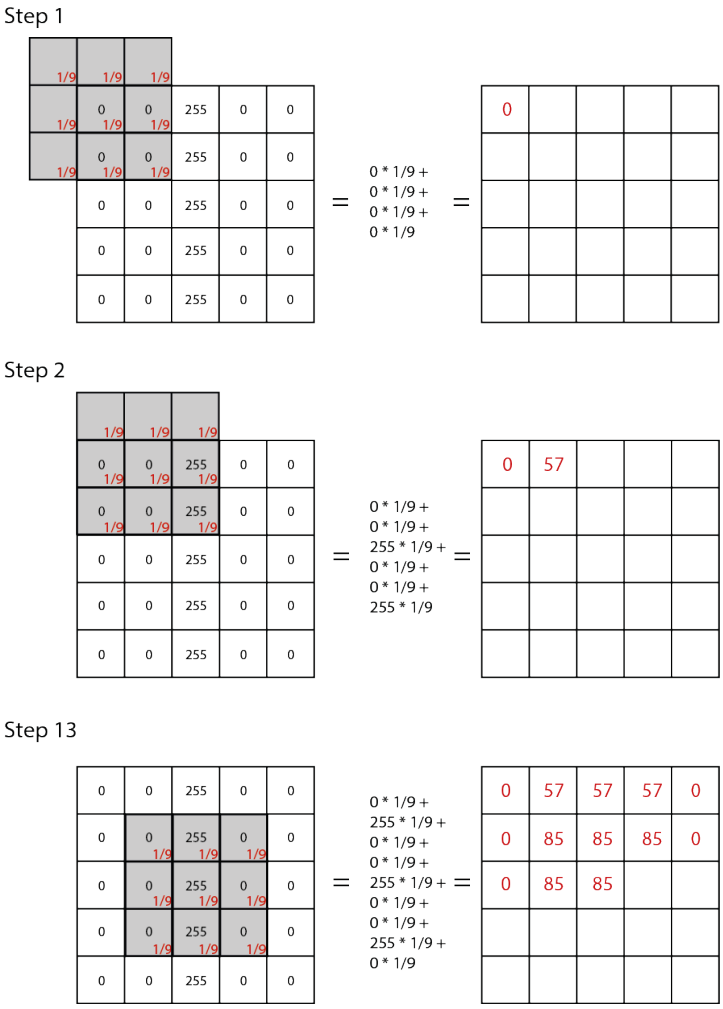

Figure 3.2: Example steps of the convolution

Exercise 3.33: Working with Image Filters

Let us compare the previous example with the operation performed in Fiji.

1. Generate an 8-bit, 5x5 pixel black image with a white vertical line as shown in the convolution example. Duplicate and zoom in to observe individual pixels and check whether the intensity values of the black pixels are zero and the values of the white pixels are 255.

2. Go to `[Process > Filters > Convolve...]`. Change the default kernel in the Convolver dialog to (Fig. 3.3):

1	1	1
1	1	1
1	1	1

Note that we did not enter 1/9; the values are automatically divided by the number of elements in the kernel if 'Normalize Kernel' is ticked. Select 'Preview' if you directly want to observe the effects of your chosen kernel.

Convolution and Correlation

A convolution is a correlation with the kernel rotated by 180 degrees. This makes no difference if the filter is symmetric (e.g. Gaussian). The reflection of the kernel has a few important effects. One is that the convolution is associative ($f * g * h = f * (g * h)$) while the correlation is not; i.e. the associativity allows you to pre-convolve several filters into a single filter that gets applied to your image. The correlation describes the comparison between the kernel and the pixel neighborhood and is, for example, used in template matching as associativity is not important for this task. Also, the convolution is a multiplication in the frequency domain and it can be faster to perform the filtering in the frequency domain (depending on image and kernel size).

Exercise 3.34: The point spread function (PSF)

The image you obtain with a microscope is blurred by the optics of the microscope itself. The point spread function describes the impulse response of the microscope, i.e. the 3D diffraction pattern resulting from a single point. In microscopes (non-coherent systems), the image formation process is linear, which means that the imaging of many objects produces a result that is the sum of individually imaged objects. Therefore, image formation in a microscope can be seen as a convolution of the true object with the PSF. Estimating the PSF of a microscope then allows image deconvolution – a process where the true image is estimated by trying to reverse the effects of the convolution. The problem is that deconvolution is an ill-posed problem – no solution, or no unique solution might exist and noise can strongly influence a solution. Computing intense deconvolution algorithms are used to approximate the true image; a PSF for deconvolution can be obtained by measuring sub-resolution beads with know radii or by theoretical estimation using parameters of the optical system.
In this exercise, we use an artificial ground-truth image and convolve this image with a single slice of a cropped and low bit-depth version of a measured PSF.

1. Open the images artificial-groundtruth.tif and PSF-LeicaSP8-63x-Ar488.tif in /mod3/data. The image shows a cropped, 8 bit, x/y slice through a 3D PSF taken at the center (focal plane).

2. Convolve the groundtruth image with the PSF by loading the file PSF-LeicaSP8-63x-Ar488.txt in the convolver dialog.

3. Compare the groundtruth image with the convolved image. This is what

happens every time you take an image with the microscope! (Although this is only a 2D, simplified example).

3.1.2 Linear Filters

Convolution is the mathematical operation behind linear filters; and in the following, we will look at some of the most common linear filters available.

Averaging filter

An averaging operation (also called smoothing, arithmetic mean or box-car filter) is used to suppress noise in an image. The size of the structures that are suppressed is dependent of the kernel size. Small fluorescence fluctuations will be suppressed by a small filter; increasing the filter size, larger structures will be reduced in size and smoothed into the surrounding regions. Manually varying the filter size allows you to decide the optimal scale of the smoothing for your analysis.

This filter was already shown in the convolution demonstration and the first exercise, the kernel (without normalization to the number of elements) is:

1	1	1
1	1	1
1	1	1

Exercise 3.35: Averaging images

1. Open the image microtubuli.tif in /mod3/data. Zoom into the image and observe the fluorescence fluctuations along the microtubuli.
2. Preview the smoothing filter with a kernel size of 3x3 (see above), using `[Process > Filters > Convolve...]`. Then increase the kernel size to 5x5 and 7x7 and compare the results (duplicate the images!). What happens when the size of the kernel becomes larger? Can you enter a kernel size of 2x2?

3. The averaging filter has its own direct command in Fiji. Go to `[Process > Filters > Mean...]`. You can choose a radius and optional preview in the next window. A radius can be given instead of size in X and Y as the kernel mask is circular – the neighborhood can be defined arbitrarily! Circular kernels are common as pixels the outermost pixels have the same distance to the pixel of interest (Circular kernels can be shown with `[Process > Filters > Show Circular Masks...]`). Try different radii to obtain the best smoothing to suppress small fluorescence fluctuations.

4. To show the effect of the filter more directly, you can subtract the smoothed from the original image.

3.1.3 Gradient filters

Image features with high spatial frequencies (edges) are those where intensity varies greatly over short image distances (e.g. neighboring pixels). These changes over distance can be seen as a derivative (like velocity is the derivative of position over time). For our images, we use the *gradient* as a generalization of the derivative of a function in one dimension (like velocity), to a function in several dimensions (like our 2D images). For our 2D images $I(x, y)$, we can write the gradient as:

$$\nabla I(x, y) = \left(\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y} \right)$$

A common use for gradient filters is edge detection. A large gradient value of a pixel might indicate an edge and edges could then be traced along those pixels (perpendicular to gradient direction). For example, edge detection can be very useful if your structures of interest vary in absolute intensity but are always distinctive against their direct surroundings.

Find Edge (Sobel Filter)

The find edge filter is used to create an image which emphasizes edges. It uses an approximation of the gradient to create two 3x3 kernels which are convolved with the original image I (kernels are an approximation of the derivative in horizontal, respective vertical direction) to create two output images J_x, J_y:

$$J_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I, J_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

The gradient magnitude image J is then calculated with:

$$J = \sqrt{J_x^2 + J_y^2}$$

Sharpen (Laplacian Filter)

The sharpen filter is also often used to emphasize changes in pixel intensities. In this case, the 3x3 kernel is approximated by the sum of the second derivatives:

−1	−1	−1
−1	12	−1
−1	−1	−1

Exercise 3.36: Gradient Filters

1. Open the image microtubuli.tif in /mod3/data. Zoom into the image and observe the fluorescence fluctuations along the microtubuli. Remember to duplicate the images before testing any processing.
2. At first, test the sharpen filter with `[Process > Sharpen]` and optionally using the kernel in the convolution dialog.
3. The find edge filter can be tested directly using `[Process > Find Edges]`. If you want to explore the filter a bit more, you can also generate both filtered images, convert those to 32 bit and perform the calculations by yourself. Are the results identical?
4. What happens to the noise if you use the sharpen filter?
5. What happens if you first use a smoothing filter and then the edge detection filter?

Gaussian filters

A very important linear filter that smooths an image and reduces noise is the *gaussian filter*. In difference to the mean filter, pixels in the neighborhood are weighted with their distance to the pixel of interest. The weights are obtained by a gaussian function where

the width σ of the gaussian function $g(x, y)$determines the kernel size. A is a scaling factor used for normalization:

$$g(x, y) = Ae^{\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

If you want to smooth your image, it is always a good idea to not only explore the averaging filter but also the gaussian filter. Again, varying σ, you can determine the optimal filter size for your application.

Difference of Gaussian filter (DoG)

Let's say you want to suppress small structures but also large structures and focus on detecting structures within a certain size range. In this case, you could combine two gaussian filters: one with a small σ to suppress small structures and one with a large σ to suppress large structures as well. When you then subtract the second filtered image from the first filtered image, the image that is left contains information between both smoothing scales.

Exercise 3.37: Gaussian Filters

1. Open the image microtubuli.tif in /mod3/data. Zoom into the image and observe the fluorescence fluctuations along the microtubuli. Remember to duplicate the images before testing any processing.
2. The gaussian filter can be tested using `[Process > Filters > Gaussian Blur...]`.
3. Try to find an optimal value for σ.
4. Implement a DoG filter with σ values of 1 and 10.

Exercise 3.38: Unsharp Mask Filter

Let's illustrate the power of combining filters and image math with another example. In this case, we will replicate the unsharp mask filter. This is a very common filter available in many software packages, e.g. for photo editing, but has to be used very carefully for quantitative scientific analysis.

1. Open the image microtubuli.tif in /mod3/data. Duplicate.
2. Use a gaussian filter with a σ of 2.
3. Multiply the gaussian filtered image with a weight w of 0.8.
4. Subtract the weighted image from the original image.
5. Divide the resulting image by 0.2.
6. Last, compare your results to the results of `[Process > Filters > Unsharp Mask...]`.

3.1.4 Nonlinear Filters

In addition to linear filters which can be implemented with a convolution, we can also use nonlinear filters to enhance our images. We will discuss one particular class of such filters (rank filters), focusing on the median filter. This filter or its variants can be very powerful when you want to remove noise but maintain other details of the image or you want to be less susceptible to outliers.

Median, maximum and minimum filters

For the median, maximum and minimum filters, pixels in the neighborhood are sampled and then sorted by their intensity values. Let's assume you have an 3x3 image part surrounding the pixel with value 100, such as:

0	95	105
0	100	110
0	0	90

The sorted list would look like: $0, 0, 0, 0, 90, 95, 100, 105, 110$. The median filter would update the pixel value from 100 to 90. The mean filter would update to a value of 56. If this pixel is part of an edge, the median filter would be better to maintain this edge. The maximum and minimum filters use the same sorted list but update the pixel with either the highest or the lowest value.

Exercise 3.39: Median Filter

1. Open the image microtubuli.tif in /mod3/data. Before we do any further processing, we add some Salt-and-Pepper noise using `[Process > Noise > Salt and Pepper]`. This sets random pixels to the lowest or highest values.
2. Duplicate the image with added noise and perform the mean filter to suppress the just added noise (find a suitable radius).
3. Now, use the median filter `[Process > Filters > Median]` with the same radius.
4. Which filter works better?

Exercise 3.40: Background subtraction and batch processing

To make this exercise more interesting, we first establish the method we want to use and then apply it to many files automatically. If a certain task is performed multiple times, it is called *batch processing*.

1. Open the image xu2015-adipocytes.tif in /mod3/data. This image was provided as an example image by Elaine Xu (March 2015, Bruening lab, MPI Stoffwechselforschung). The image shows adipocytes and is a good example to illustrate uneven illumination and how to improve the image with background subtraction.
2. At first, we explore a method to subtract the background. For this, we first perform a Gaussian blur on a duplicate image with a large σ of 20. This creates a very blurry image that already shows that we picked up the uneven illumination. You can look at a line profile in the original and the blurred image.
3. Now we subtract the original image from the blurred image. Adjust brightness/contrast to confirm that background was subtracted. As the gaussian blur is a lowpass-filter, the subtraction of this lowpass image can be considered as a (pseudo) highpass filtering.
4. Fortunately, Fiji already has a function embedded that performs a similar operation for background subtraction, called a rolling ball background subtraction (Same idea: somehow calculate an average for every

pixel. However, the actual calculations differ).

5. We now want to perform the background subtraction on all image files in a certain folder. For this, we *record* our actions and then use the recorded sequence on each of those files.

6. Close all images except the original xu2015-adipocytes.tif.

7. To start the recording, perform `[Plugins > Macros > Record...]`. A Recorder window pops up (Fig. 3.4). Make sure that 'Macro' is selected in Record and nothing else.

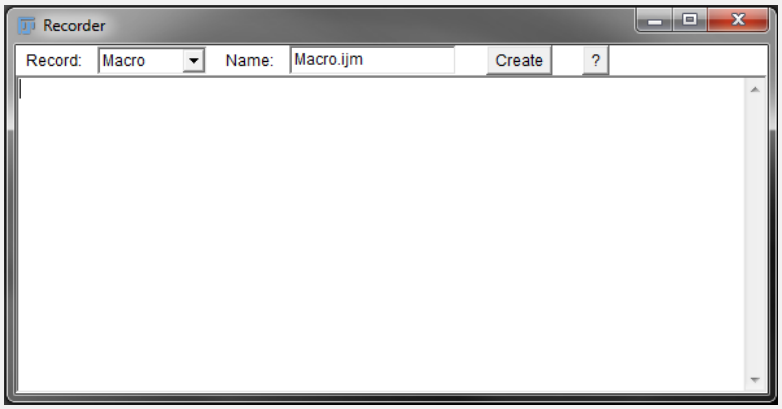

Figure 3.4: Recorder window.

8. Do `[Process > Subtract Background...]`, set the rolling ball radius to 20 (Fig. 3.5). After you clicked on 'ok', the operation gets performed and the recorder should show following line: `run("Subtract Background...", "rolling=20");`.

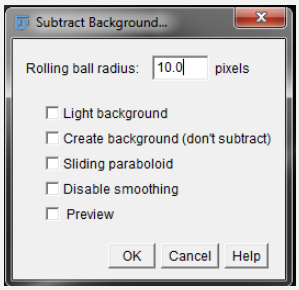

Figure 3.5: Subtract background window.

9. Select the line and copy (Strg+C). Go to `[Process > Batch > Macro...]`. In the batch processing window, select the input folder /batch-files in /mod3/data. This folder contains 5 identical adipocyte

images with different names. Also, choose an output folder, make sure that this output folder exists. Paste the line of code into the window (Fig. 3.6). and click on 'Process'. Check the chosen output folder for the results.

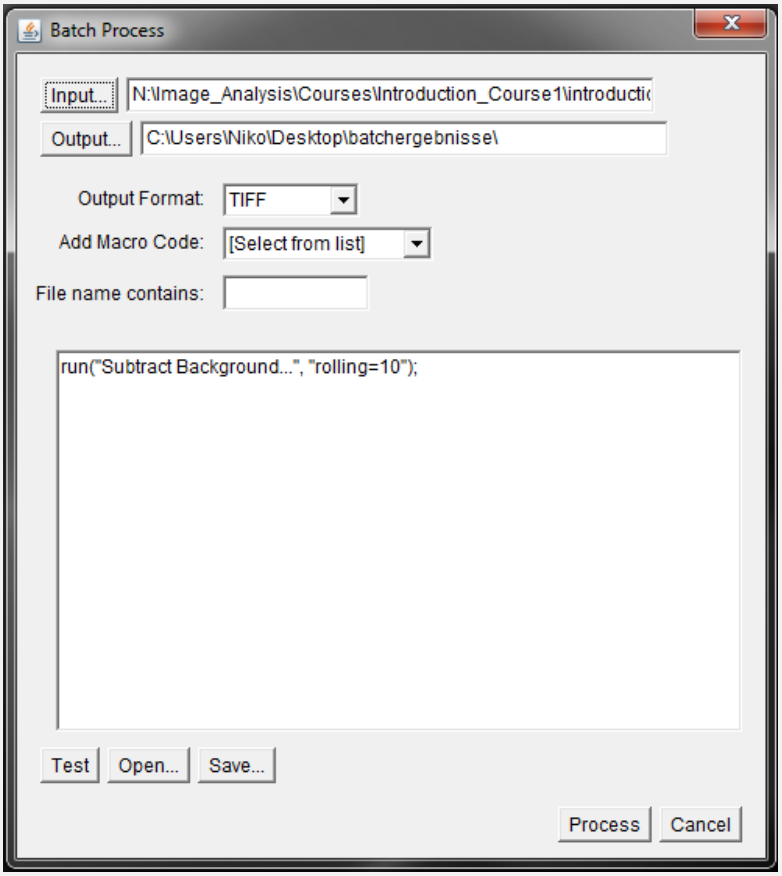

Figure 3.6: Subtract background window.

When you try to perform a batch processing where a function requires explicit file names the batch processor will fail because it does not know that these need to be replaced (e.g. when the sequence Duplicate Image, Gaussian, Image Calculator is used). In this case, we have to create a macro where individual file names get replaced. Unfortunately, this requires a little bit of very basic understanding of programming principles (we need variables, for-loop and a function). As this is outside the scope of this module, we will not discuss those concepts (however, they are important for more complex automated workflows).

3.2 Image segmentation

Image segmentation is a process where every pixel in an image obtains a label, such that pixels with the same label share characteristics. Typically, this is done to detect *objects* or structures of interest. Sometimes, these objects are called *connected components*. Basically, this is what you have already done using regions of interest to mark the pixels belonging to a nucleus or cytoplasm. This can be called manual segmentation and works if there is not too much data to analyze. Overall, it is preferable to automate the process of segmentation; not only because it is faster, but also because it should lead to better reproducibility and lower bias of the results.

3.2.1 Thresholding

A very simple and easy to understand image segmentation method is called *thresholding*. In this method, each pixel belongs to one of two groups: background or foreground (black or white, object or not object). For images obtained with fluorescence microscopy, this is easy to understand: we typically have dark background with light objects. Thresholding image $f(x, y)$ with a thresholt T then leads to extraction of the objects in image $g(x, y)$:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{otherwise} \end{cases}$$

Image $g(x, y)$ is a binary image, only consisting of black and white pixels.

Exercise 3.41: Thresholding with contrast adjustment

Let us first have a look at the effects of thresholding using brightness/contrast adjustments.

1. Open the image cell-colony.tif in /mod3/data. Open the brightness/contrast adjustment window. Observe the slope of the line that shows the change in (displayed) intensity between the minimum and maximum pixel values in the histogram (Fig. 3.7, red line).

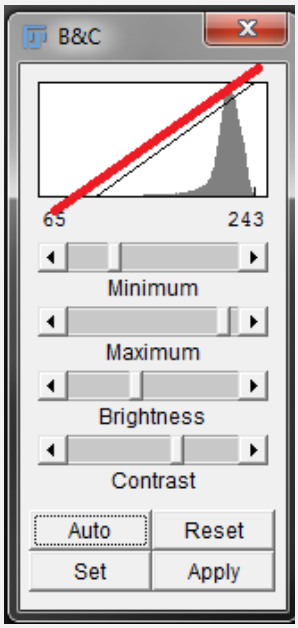

Figure 3.7: Brightness and Contrast adjustment.

2. Now, set the contrast to the maximum value and observe that the slope of the line changes until the line appears nearly vertical. The image is a black and white image now, everything to the left of the line is black, everything right of the line is white (the majority of the pixels as seen in the histogram).

3. If you now change the brightness, you can move this vertical line around and observe that the number of pixels identified as white (black) changes.

Exercise 3.42: Manual thresholding

Instead of adjusting the contrast, Fiji has a specialized function to adjust the threshold `[Image > Adjust > Threshold...]`.

1. Open the image cell-colony.tif in /mod3/data if it is not already open.

2. Do `[Image > Adjust > Threshold...]` to open the threshold adjustment window (Fig. 3.8). In this window, you can set the intensity for the threshold. Furthermore, you can select whether you have a dark or light background to determine whether you are interested in light or

dark objects. Note that you can select a window for your threshold, by setting the lower and upper values to a value outside the extremes. This is called a *density slice* and can be used to detect objects that fall within a certain intensity range.

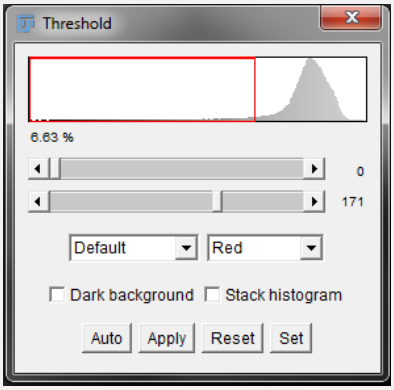

Figure 3.8: Adjust threshold window.

3. Change values and play with different settings until you think you found an optimal threshold. How difficult is it to obtain an optimal threshold, i.e. how sensitive is the result to your chosen value?

Manually thresholding an image is already a powerful way to detect objects of interest. But what should you do if you cannot apply the same threshold value for all images you want to analyze? Of course you could manually choose a value for each image. However, this can introduce a bias into your results. A better option would be to automatically determine the optimal threshold value for each image, using the same algorithm. Fiji already has a variety of algorithms that you can choose from (Fig. 3.9):

- Huang
- Intermodes
- IsoData
- Li
- MaxEntropy
- Mean
- MinError
- Minimum
- Moments

- Otsu
- Percentile
- RenyiEntropy
- Shanbag
- Triangle
- Yen

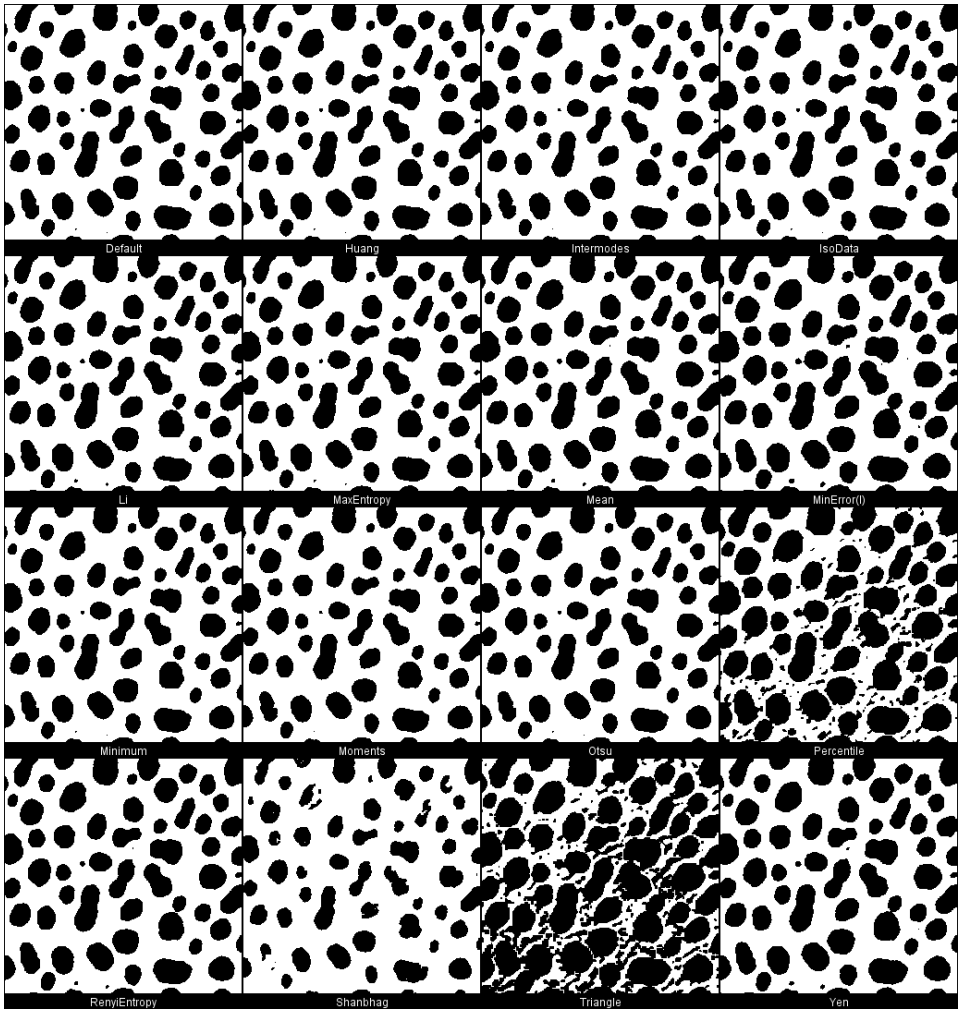

Figure 3.9: Results of all available thresholding methods, using [Image > Adjust > Auto Threshold], Try All.

Choosing an algorithm can theoretically be done by looking at the details of the algorithm and determining best which method suits your data. In practice, however, you can simply try all and choose the one that suits your purposes best.

Let us have a look at one of the thresholding methods, *Otsu's method*. This algorithm makes the assumption that two classes of pixels are in an image, with each class having a distribution of intensity values. The algorithm calculates the optimum threshold that separates these two classes so that the variance of pixels within each class is minimal. This algorithm is very common, easy to understand and implement.

Exercise 3.43: Automated thresholding

1. Open the image hela-cells.tif in /mod3/data. In this task, we first want to automatically identify the nuclei. Select the appropriate channel and duplicate.
2. Test all available thresholding methods. You can see that most method already produce very nice results.
3. In the next step, we want to identify the lysosomes (red). Try to threshold the appropriate channel with Otsu's method. Are the results good enough?
4. As you can see, the algorithm already works relatively well, but is not good enough for our purposes. Can you think of a reason why the method fails in some parts of the cells?
5. After an analysis, we come to the conclusion that background signal is too high in parts of the cell. Therefore, we now first use a background subtraction (e.g. with a rolling ball radius of 10 pixels) before we threshold with Otsu's method. Do the results improve?
6. Finally,try to use the green channel to identify the cells' cytoplasm. Again, it seems that an image filter might improve the results. Try a gaussian filter with varying sigma values to improve the results. At the end, the result we obtain is not perfect but might be sufficient for our needs. You will see in section about binary images how we could still improve on this thresholding.

3.2.2 Binary Images

Thresholding into two pixel classes leads to a binary image. In this section, we will learn how operations on binary images can be used to process an image to facilitate further analysis.

Morphological operations

Morphological operations are very similar to image filters, only that in this case, logical operations are performed on the image. Similar to image filters, a small *structuring element* (kernel in image filters) is moved over the image pixel by pixel and logical operations are performed between the image and the structuring element. In this section, we will discuss the basic morphological operations.

Erosion

Erosion of a binary image I with an structuring element S changes a foreground pixel (white) to background (black) if at least one of the neighbor pixels (defined by S) is a background pixel. Erosion is used to shrink or thin an object (foreground) in a binary image; only pixels that were only surrounded by foreground pixels remain (Fig. 3.10). In Fiji, the structuring element is a 3x3 pixel square.

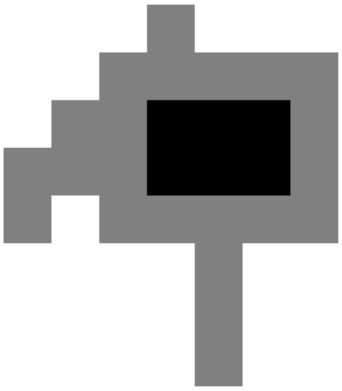

Figure 3.10: Erosion. Gray pixels indicate the original image. Black pixels remain after erosion with a 3x3 square. For display reasons, black pixels indicate foreground, white pixels background.

Dilation

In contrast to erosion, dilation grows or thickens an object in a binary image. A background pixel is changed to foreground if at least one pixel in the neighborhood is a foreground pixel (Fig. 3.11).

Opening and Closing

The combination of dilation and erosion can be very useful to remove artefacts that remain in an image after thresholding. *Opening* is an erosion followed by a dilation

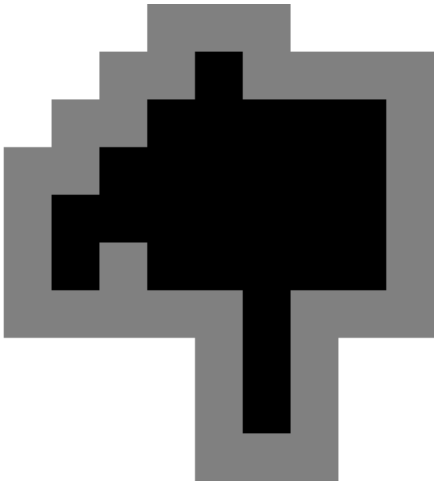

Figure 3.11: Dilation. Gray pixels indicate the dilated object. Black pixels are the original object.

and used to smooth objects, break thin connections and remove protrusions. *Closing* is a dilation followed by an erosion and used to smooth objects, join closely neighbored objects and fill small holes.

Exercise 3.44: Morphological operations

1. Open the image thresholded-nuclei.tif in /mod3/data. This image shows real thresholded nuclei with artificially introduced thresholding errors.
2. Perform erosion, dilation, opening and closing in `[Process > Binary]` and observe the changes of individual nuclei. Make sure you understand where the different operations enhance the image and where they make analysis more difficult.
3. Instead of using opening or closing, try to manually erode 4-5 times and then dilate 4-5 times again; perform the same operations reversed.
4. Try the command `[Process > Binary > Fill Holes]`. This operation finds background areas that are completely surrounded by foreground and sets all those pixels to foreground.

3.2.3 Skeleton Analysis

A skeleton analysis can be very useful if you are interested in the branching of a structure (neuronal arborizations, mitochondrial network, ..). Basically, your object becomes thinned (eroded) until only a single (center)line remains to describe the objects structure.

Exercise 3.45: Skeleton analysis

1. Open the image drosophila-ddac-neuron.tif in /mod3/data. In this task, you are going to analyze the arborization pattern of this, already thresholded, neuron.
2. Duplicate the image, do `[Plugins > Skeleton > Skeletonize]`. Make sure that the skeleton is white against a black background (you might need to invert the image). Now, edit the LUT so that the white pixels appear yellow, magenta or any other light color.
3. Look at the original thresholded image and add the skeletonized image as an overlay (with zero transparent). Analyze the skeletonization results with the overlay.
4. Perform `[Plugins > Skeleton > Analyze Skeleton]` on the skeletonized image. The Analyze Skeleton window allows you to prune the skeleton before analysis to get rid of small branches either by their length or intensity. Tick 'Show detailed info' and click on 'OK' (Fig. 3.12).

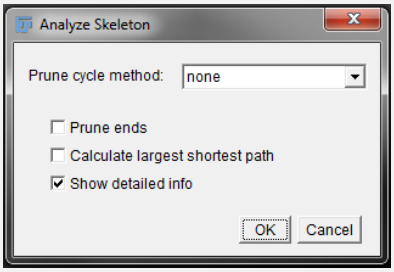

Figure 3.12: Analyze skeleton window.

5. Three windows show up: Detailed information about each branch, summary results with branch statistics, and an image showing the skeleton with branch points and end points in different colors. You can try out different prune methods and compare the results.

3.2.4 Watershed transform

Watershed segmentation is used to separate touching objects, i.e. they appear as a single object in the binary image. The algorithm first calculates an Euclidean distance map (Fig. 3.13), where the distance of each pixel to the nearest foreground pixel is calculated. The remaining black objects (peaks after the distance transform, centers of the original binary objects) are then dilated until the edge of the original object is found or the edge touches a region of another dilating object. This approach works best if the objects are 'round-ish' and have small overlap.

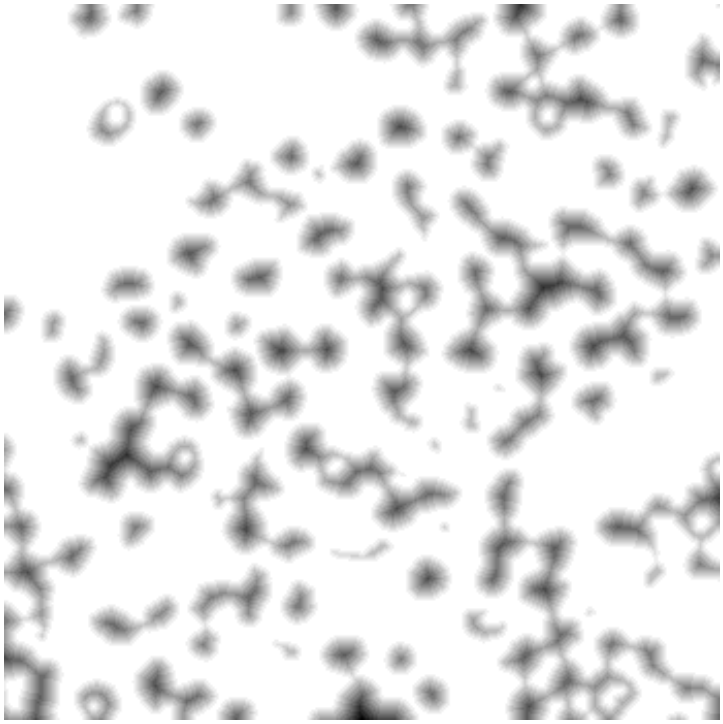

Figure 3.13: Distance transform

Exercise 3.46: Watershed transform

1. Open the image bunch-of-nuclei.tif in /data/chapter4. Use the gauss-filter and thresholding to generate a binary image with the nuclei detected. Duplicate.
2. Use `[Process > Binary > Watershed]` on the binary image. Observe

where the algorithm splits objects.

3.3 Analyzing objects

Fiji has a very powerful method to obtain measurements from segmented objects which is called 'Analyze Particles' (Fig. 3.14). The windows allows you to select detected objects by size and by circularity, excluding objects at the edges of the image and produces several result outputs. It can add detected objects as individual ROIs to the ROI manager and perform calculations by performing the 'Measure' method on each detected object.

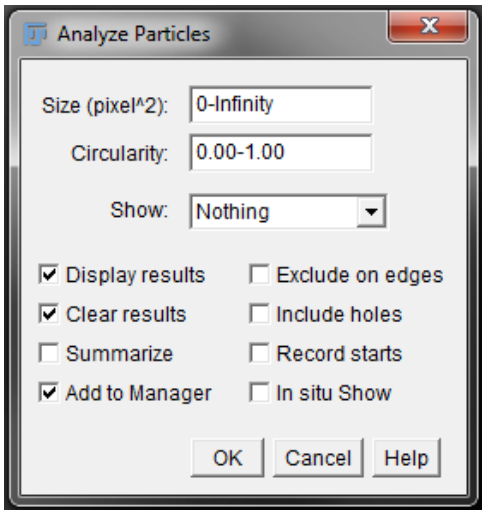

Figure 3.14: Analyze Particles Window

Exercise 3.47: Analyze Particles

1. Open the image bunch-of-nuclei.tif in /mod3/data and use a sequence of filtering, thresholding and binary operations to identify the nuclei. Remember that this image does not provide an easy perfect result on purpose!
2. Set the measurements you want to perform `[Analyze > Set`

`Measurements...]`. For this example, we want at least measure the area and the mean gray value.

3. Perform `[Analyze > Analyze Particles...]`. Do not select objects by size or circularity, show the outlines and tick 'Display results', 'Clear results', 'Add to Manager' and 'Summarize'.

4. Several windows should show up. A summary report indicating the number of detected objects, and several average statistics; A results window showing the selected measurements for each detected object; an outline image indicating the object number for each object; the ROI Manager with each object as an individual ROI. Why is the mean intensity value 255 for each object? The measurement was performed on the thresholded binary image. While thresholding does not change the shape of an object, the intensity values are obviously not maintained. In order to perform the measurements on the original image, Redirect the measurements to the original image in 'Set Measurements...'.

5. Explore the functions of the particle-analyzer method and try to select objects in a way that only small/large or round objects are measured.

3.3.1 Working with Plugins: Trainable Segmentation

Machine learning methods can be used to let the computer learn what you think as object and background. Fiji has a Trainable Weka Segmentation plugin. Using this plugin, you can manually mark pixels as signal or background and let the plugin 'learn' the differences between these two classes of pixels. To achieve this differences learning, the plugin uses many features of the pixels to come up with a model that categorizes the pixels into classes.

Exercise 3.48: Trainable Weka Segmentation

1. Open the image tissue-adipocytes.tif in /mod3/data if it is not already open.

2. Go to `[Plugins > Segmentation > Trainable Weka Segmentation]`. A window pops up that shows the image and several buttons (Fig. 3.15).

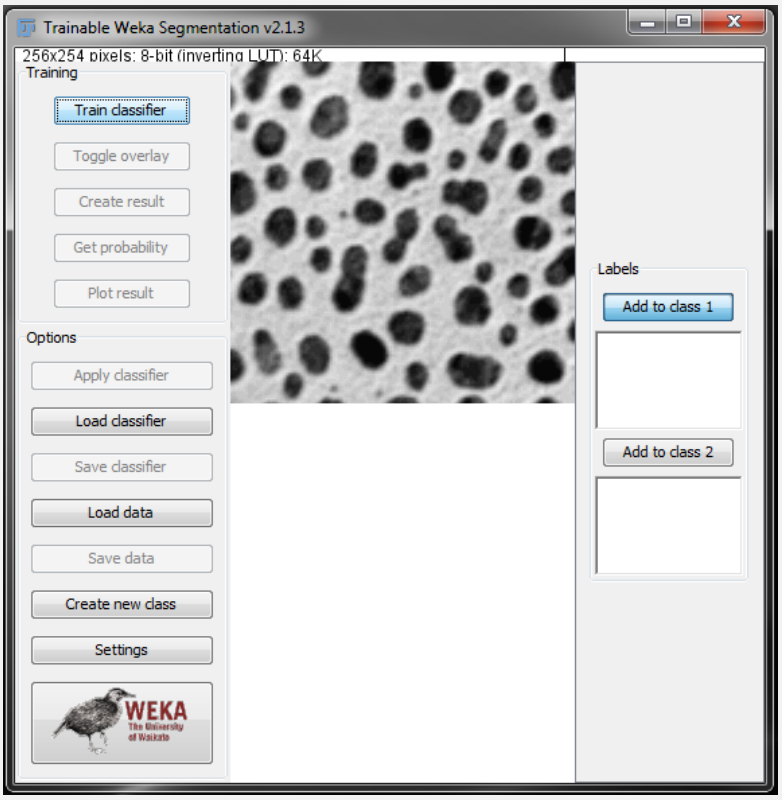

Figure 3.15: Trainable segmentation window.

3. Use the freehand line tool and draw on an image object you want to detect. Add this selection to class 1. Repeat this step and then add two background selections to class 2.

4. Click on 'Train classifier'. You should now see what the method thinks is background and foreground. If you click on 'Create result' a two color image is generated that can easily be converted into a binary image.

You have now seen the most basic usage of this plugin. For example, you could use more than two classes or apply a learned classifier to more than one file.

Bibliography

M Rossner and K M Yamada. What's in the picture? the temptation of image manipulation. *Journal of Cell Biology*, 166:11–15, 2004.

The Mac Developer Library. Performing convolution operations. https://developer.apple.com/library/mac/documentation/Performance/Conceptual/vImage/Convolution 2011. [Online; as of 19/11/2014].

B Wong. Points of view: Color blindness. *Nature Methods*, 8:441, 2011.