Author Information

Christian "Tischi" Tischer

e-mails: christian.tischer@embl.de, tischitischer@gmail.com

Recommended literature on bioimage analysis

- http://www.imaging-git.com/olympus-website-bioimage-data-analysis

...

Inspection of the numerical content of images

An image essentially is an array of numbers with some metadata. For scientific image analysis it is very important to constantly inspect the numeric content of our images, for instance to check whether the image was acquired properly, or whether a mathematical operation such as background subtraction had the desired effect.

Activity: Image inspection

Let's open an image and explore different tools to inspect the numbers in this image. We start by inspection an 8-bit image, where the numbers range from 0 to 255 (2^8-1); we'll explore different bit depths later.

- Open image "../image-inspection/B.tif" [File > Open]

Mouse over

Simply move with the mouse over the image; the intensity will be shown in ImageJ's menu bar.

Pixel inspection tool

- Menu bar: [Px]

Intensity line profile

- Menu bar: Select the line profile tool
- [Analyze > Plot Profile]

Histogram

- [Analyze > Histogram]

Lookup tables (LUTs)

LUTs assign a certain color to each numerical value. Intensity differences are best seen using a grayscale LUT. Choosing the LUT color similar to the emission color of the imaged fluorophore can also make sense. LUTs with multiple colors (e.g., "Fire" in ImageJ) are good for simultaneously seening very dim and very bright images. Finally, LUTs where only the lowest and highest value have a certain color are useful for microscopy, e.g. to indicate saturated pixels.

Activity: Adjust Brightness & Contrast

While the colors in a given LUT are fixed, one can change how these colors are mapped onto the numbers in the image.

- Open image "../image-inspection/B.tif" [File > Open]
- Change the LUT settings:
  - [Image > Adjust > Brightness/Contrast]

Note that this does not change the numbers but only the appearance on your screen.

Important: Don't press [Apply] as this will in fact change the pixel values.

Activity: Explore different LUTs

- Open image "../image-inspection/B.tif" [File > Open]
- Explore different LUTs [Image > Lookup Tables], e.g.
  - Grays
  - HiLo
    - Red: highest, Blue: lowest
    - Important note: "highest" and "lowest" depend on your Brightness&Contrast settings!
  - Fire

Important numerical properties of microscopy images

Background (offset)

Dynamic range

Saturation

Practical activity: Image content inspection

In this activity we will open several images and find out which "issues" they have. The aim is to assign each image to one of the following issues:

- no problem
- high background
- too low background
- low dynamic range
- too much saturation

Use below workflow to inspect the images:

- Open "../image-inspection/A.tif" [File > Open]
- Also open B.tif, C.tif, D.tif, E.tif
- Use below methods to inspect the images and find their "issues"
    - Adjust the display [Image > Adjust > Brightness/Contrast]
    - Examine gray values in whole image [Analyze > Histogram]
    - Analyze gray values along a line [Analyze > Plot Profile]

Additional tasks:

- Lets find five or more different ways to identify saturated pixels in an image

Image bit depths

Images can have different bit depths. Let's start by exploring some of the limitations of the 8-bit image that we were dealing with until now.

Activity: Exploring the limitations of an 8-bit image

- Open image "../image-inspection/B.tif" [File > Open]
- Adding numbers:
    - Copy the original image [Image > Duplicate]
    - Add 500 to each pixel in the image [Process > Math > Add]
    - Inspect the gray values!
- Subtracting numbers:
    - Copy the original image [Image > Duplicate]
    - Subtract 100 from each pixel in the image [Process > Math > Subtract]
    - Inspect the gray values!
- Dividing numbers:
    - Copy the image [Image > Duplicate]
    - Divide each pixel in the image by 2 [Process > Math > Divide]
    - Inspect the gray values!

Obviously this is not what we want since it is all wrong :-).

Activity: Exploring properties of floating point images

- Open image "../image-inspection/B.tif" [File > Open]
- Duplicate the image and already name "32-bit" [Image > Duplicate]
- Convert to 32-bit floating point [Image > Type > 32 bit]
- Inspect the gray values! Did they change after the conversion to 32 bit?
- Now let's repeat above activity

Much better, right?!

Image bit depths in ImageJ

- 8-bit
  - integers from 0-255 (2^8-1)
- 16-bit
  - integers from 0-65535 (2^16-1)
- 32-bit floating point
  - can have negative numbers, such as -1
  - can have non-integer numbers, such as 1.5 or -3.2
  - this format is generally recommended as soon as you do any kind of mathematical operations on your images
  - disadvantage: needs more memory and disk space

Although ImageJ does not support it, your images could also have been acquired with cameras of different bit depth such as 12 or 14 bit.

Image bit depth conversions

Image bit depth conversion is something that you should generally avoid, but sometimes you can't; either because you need to save disk space or because certain operations or plugins only work with certain bit depths. Let's thus explore now what happens if you do convert between different bit depths.

Activity: Conversion from 8-bit to 32-bit floating point

- Open image "../image-inspection/B.tif" [File > Open]
- Duplicate the image and rename it to "32-bit" [Image > Duplicate]
- Convert to 32-bit floating point [Image > Type > 32 bit]
- Inspect the gray values! Did they change after the conversion?

Activity: Conversion from 16-bit to 32-bit floating point

- Open image "../image-format-conversion/16bit.tif" [File > Open]
- Duplicate the image and rename it to "32-bit" [Image > Duplicate]
- Convert to 32-bit floating point [Image > Type > 32 bit]
- Inspect the gray values! Did they change after the conversion?

Activity: 16-bit to 8-bit conversion

OK! Now comes the **tricky part**, where several projects were going very wrong in the past!

- Open "../image-format-conversion/16bit.tif" [File > Open]
- Inspect the gray values: What are the minimum and maximum? Note them down.
- Adjust the display such that it looks nice [Image > Adjust > Brightness/Contrast]
- Convert to 8-bit [Image > Type > 8bit]
- Inspect the gray values again: What are the minimum and maximum now?

Hopefully you are **shocked** that we all got different results! How can this be?

Discussion: How to convert 16-bit to 8-bit

- 0, 65535 => 0, 255
  - Preserves intensities but looses dynamic range
- min, max => 0, 255
  - Maximizes dynamic range, but looses intensity information
- minLUT, maxLUT => 0, 255
  - Leave it up to the user!
  - This is what ImageJ is doing!

Image format conversion

Unfortunately there are many different image formats and since not all software can open all formats you most likely will have to sometimes save your images in different formats. It is of utmost importance that you check what happens to the numerical content of your images when you are doing this! So let's practice!

Activity: Save an image in different formats and inspect how this affects its numerical content and file size

- Open “../image-format-conversion/16bit.tif” [File > Open]
  - Adjust the display such that you actually see something [Image > Adjust > Brightness/Contrast]
- Save as **Jpeg** using different levels of compression (quality)
  - Adjust Jpeg quality (0-100) to 10 [Edit > Option > Input/Output]
  - save as "quality_10.jpg" [File > Save As > Jpeg]
  - repeat for Jpeg qualities 75, and 100
- Save as **png** [File > Save As > PNG]
- Adjust the display such that the image **appears saturated** [Image > Adjust > Brightness/Contrast]
  - Save as "saturated.jpg" [File > Save As > Jpeg]
  - Save as "saturated.png" [File > Save As > PNG]

Now lets go to the folder where you saved the images and check their file size! And, even more important, lets reopen them and check what happened to their gray values!

Image intensity measurements

Mean intensity and sum intensity

=> PowerPoint presentation.

The biophysical meaning of intensities in fluorescence microscopy images

=> PowerPoint presentation.

Practical activity: Manual intensity measurements

- Open “../bit-conversion/16bit.tif”
- Draw a region around a nucleus, e.g. using ImageJ's Polygon Selection
  - Since we will do sum intensity measurements with proper background subtraction you should draw this region rather generously not to miss any intensities!

- Add region to ROI manager [Analyze > Tools > ROI Manager > Add)
- Name the region “nucleus_1” [Analyze > Tools > ROI Manager > Rename]
- Repeat above steps for a background ROI and another nucleus
- Select measurements [Analyze > Set Measurements]:
  - ✓ Mean gray value
  - ✓ Area
  - ✓ Integrated density
    - This measures the sum intensity; in fact it will output two values; the "good" one is the **RawIntDen**, which really simply adds up the gray values in the measurement ROI.
- Select all regions and measure [ROI Manager > Measure]

Now we need to do the proper background subtraction for the two nuclei ROIs, using below formula:

```
Sum_BgCorr = RawIntDen - Area * Mean_Background
```

In words, we subtract for each pixel in the ROI the mean value of the background.

Intensity measurements and their interpretation, with local background

Intensity measurements are a **very tricky business**, not because they are technically difficult, but because one can make many mistakes in the interpretation of the numbers. This very easily leads to wrong scientific conclusions!

Activity: Intensity measurements with local background subtraction

Let's first open the images:

- Open all images in this folder "../dna-damage-synthetic-data/"

We pretent that these are **widefield microscopy** images of one nucleus where a GFP-tagged DNA damage repair enzyme is diffusing around. In some of the images a well controlled laser cut was induced and thus the DNA repair enzyme binds the damage site. Some images say "Treated" in their title. The idea is that the scientist added a drug and wanted to find out if this drug enhances or diminishes the binding of the DNA repair enzyme.

Examine with line profile

Let's first look at the images using an intensity line profile and discuss what we see.

Measure fraction of protein bound to damage site

Ok, now let's try to really measure a bio-physically meaningful number. This is generally challenging and one really has to think about it!

In this case, assuming

- this is widefield microscopy, and
- the unbound molecule is fast diffusing
- the laser cut had the exact same effect in all experiments

..it probably makes sense to divide the sum intensity of the bound protein by the sum intensity in the nucleus; i.e. total_bound / total_available.

To do this we need to measure:

- Mean intensity outside the nucleus (mean_bg)
- Mean intensity next to damage site inside the nucleus (mean_nucleus_diffusive)
- Sum intensity of nucleus and area of corresponding ROI (sum_nucleus, area_nucleus_ROI)
- Sum intensity of damage site and area of corresponding ROI (sum_damage, area_damage_ROI)

Now we need to compute:

- total_signal_nucleus = sum_nucleus - area_nucleus_ROI * mean_bg
- total_signal_damage = sum_damage - area_damage_ROI * mean_nucleus_diffusive

And finally:

- fraction_bound_to_damage = total_signal_damage / total_signal_nucleus

Hard work, right? And many options to make little mistakes, thus we only should preform intensity measurements with utmost care!

Discussion points

- How do our observations relate to this: https://en.wikipedia.org/wiki/Binding_constant
- Divide by the length (and or width) of the laser damage cut?
    - length makes sense
    - width probably not
- What about computing the mean intensity in the nucleus next to the damage site?
    - In principle attractive, because for a KD we need the concentration of the unbound protein.
    - For a confocal image this can make sense if there are not many substructures in the nucleus (like nucleoli). Basically, if the concentration of the protein is homgeneous in the region where we measure the mean (you have to think in 3-D; PSF!) this mean intensity gives information about the concentration of unbound protein; however as soon as there is a lot of structure in the signal it is not clear that it helps
    - For a widefield image it is kind of the same argument, however the 3D shape of the measured region is infinitely big! One can see that one measures a larger amount of protein in the center than at the edge of the nucleus.
- What changes if we assume that this is **one confocal slice** rather than a widefield image?

Image segmentation

https://en.wikipedia.org/wiki/Image_segmentation says: In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

Applications of image segmentation in biology

- object counting
- object localization measurements
- object shape measurements
- object intensity measurements
- in general: object **feature** measurements

In general, image segmentation typically is a two step process, where you

1. identify all pixels that potentially belong to an object
2. group pixels together belonging to one object (as you typically have several objects in one image).

Activity: Manual global thresholding followed by "particle analysis"

In fluorescence microscopy, image segmentation often is easy, because the objects of interest are simply brighter than the "background".

Let's try:

- Configure image segmentation settings [Process > Binary > Options]:
  - ✓ Black Background
- Open image: "../image-inspection/B.tif" [File > Open]
- Manually adjust a threshold value [Image > Adjust > Threshold]
  - You may press [Apply] but you do not have to; it also works with the "red" overlay.
- Perform a "connected component analysis" [Analyze > Analyze Particles]
  - Other wordings are: "object detection", "particle analysis"
- Run it again and explore the different options of the "Particle Analyzer"

Discussion

- Single threshold vs. 'gating'
- Object size and shape filtering
- Different types of object representations (pros and cons)

The signal to noise (S/N) ratio

At the microscope, especially setting up a live cell experiment where you want to avoid photo-bleaching of your fluorophore, you typically wonder: "How good does the image need to be in order for me to be able to still segment the objects?"

A very important concept in this regard is the signal to noise ratio (S/N), which, in my humble opinion, is often confused with the much less important signal to background ratio (S/B).

Let's have a look and try to segment nuclei of different intensities:

- Open "../signal-to-noise/hb2-mCherry.tif" [File > Open]
- Now try to threshold the nuclei [Image > Adjust > Threshold]
  - You see that this is easy for the bright ones but does not really work for the very dim ones (you may have to adjust the LUT settings [Image > Adjust > Brightness/Contrast] to even see the dark ones).

Let's now try to quantify why it is difficult to segment the dark nuclei by measuring their S/N.

- [Analyze > Set Measurements]:
  - ✓ Mean gray value
  - ✓ Standard deviation
- Draw an ROI inside a dim nuclues of your choice, e.g. using the “Oval Selection”
- Save that ROI [Analyze > Tools > ROI Manager > Add]
  - ...and give it a good name [Analyze > Tools > ROI Manager > Rename]
- Now also draw and save an ROI in the background right next to the nucleus
- Select both regions and measure them [ROI Manager > Measure]

Now let's apply below formulas to measure the S/N and S/B:

- S/N = (Mean_Nucleus - Mean_Background) / Sdev_Background

- S/B = Mean_Nucleus / Mean_Background

Discussion

As mentioned, although sometimes used, I don't understand the use of S/B. For S/N however it is very clear that if you are getting as low as two, you start getting into trouble in terms of being able to still segment this object.

Image filtering (convolution)

Image filtering is a very wide field, where mostly one replaces the intensity of each pixel by some mathematical function of its neighbors. The most simple example is probably the 3x3 mean filter, where each pixel is replaced by the mean value in a 3x3 neighborhood (i.e. inclucing the pixel itself and its 8 neighbors).

Activity: Manually compute a 3x3 mean and a 3x3 median filter

Mean and median can be almost the same, but, depening on the data, also be very different; thus let's compare them. In fact, the median filter is quite important for local background subtraction, while the mean filter is less useful in this context (see below).

Let's compute a 3x3 mean and a 3x3 median filter for the central pixel in below examples and compare the results.

10	11	10	13	12
13	1000	10	11	14
21	15	**11**	13	10
14	13	12	11	10
11	11	10	13	12

What do you get for mean and median?

10	11	10	1	1
13	12	10	1	1
21	15	**11**	0	0
14	13	12	0	0
11	11	10	1	0

And what do you get here for mean and median?

Discussion

From above examples, it should have become clear why a median filter is called both:

- robust to outliers
- edge preserving

Activity: Segmentation of noisy images with the help of filtering (smoothing)

- Open: "../signal-to-noise/noisy-nuclei.tif" [File > Open]

- Try to threshold the image [Image > Adjust > Threshold]
    - This doesn't really work, right?
- Let's try to smooth the image first, e.g. using
    - A 11x11 median filter [Process > Filters > Median]
        - Radius: 5 (2*5 + 1 = 11)
- Now, let's segment the nuclei, using:
    - [Image > Adjust > Threshold]
    - [Analyze > Analyze Particles]

Image convolution

The 3x3 mean filter that we applied above can also be expressed in terms on a "convolution", with a "convolution kernel" as show below:

Kernel:

1/9	1/9	1/9
1/9	**1/9**	1/9
1/9	1/9	1/9

Original image:

15	11	5
14	**13**	12
11	11	10

Intermediate step:

15 * 1/9	11 * 1/9	5 * 1/9
14 * 1/9	**13** * 1/9	12 * 1/9
11 * 1/9	11 * 1/9	10 * 1/9

Sum is 11.33333

Convolved image:

?	?	?
?	**11.333**	?
?	?	?

Basically, you multiply each pixel in the original image with the number that is written in the kernel and then you replace the center pixel with the sum of all pixels.

Activity: Try the effect of different convolution kernels

- Open any image of your choice
- [Process > Filters > Convolve]

Intensity measurements with automated local background subtraction

Above we already practiced above how to subtract a local background manually; let's try to automated this. This is important for image batch analysis or when the local background is uneven and not easy to subtract manually.

In biological fluorescence microscopy one often wants to detect locally bright objects such as vesicular structures on top of a non-uniform background fluorescence, e.g. from unbound cytoplasmic protein. There are different methods to remove such 'background' fluorescence from the image, e.g.:

- corrected_image = image - mean_filter(image, radius)
- corrected_image = image - median_filter(image, radius)
- corrected_image = image - morphological_opening(image, radius) = top_hat(image, radius)
- corrected_image = IJs "RollingBall" Algorithm
- ...

In all methods the *radius* parameter should be "quite a bit larger" than the radius of the largest locally bright structure that you want to measure (why that is becomes clear when we discuss the methods in detail).

Local background subtraction using a median filter

What we'll do here is to duplicate the image and apply median filter to remove the locally bright spots. Then we subtract the median filtered image from the original image:

- [File>Open..] "../dna-damage-synthetic-data/Damaged.tif"
- [Image>Rename..] 'Title=original'
- [Image>Duplicate] 'Title=median'
- Select the 'median' image and [Process>Filters>Median] 'radius=5'
- [Process>Image Calculator] 'Image1 = original' 'Operation = Subtract' 'Image2 = median'
  - ✓ Create new window
  - ✓ 32-bit output
- [Image>Rename] "median_subtraction"

Local background subtraction using a top-hat filter

A morphological opening filter is applied to the image and subtracted from the original. The morphological opening is defined as the dilation of the erosion if the image. Alltogether this reads: top_hat(image) = image - dilation(erosion(image))

- [File>Open..] "../dna-damage-synthetic-data/Damaged.tif"
- [Image>Rename..] 'Title=original'
- [Image>Duplicate] 'Title=opened'
- [Process>Filters>Minimum..] 'radius=5'
- [Process>Filters>Maximum..] 'radius=5'
- [Process>Image Calculator] 'original' 'Subtract' 'opened'
  - ✓ Create new window

- [ ] 32-bit output
    - Note: By construction, the 'opened' image is always lower than the original; thus we cannot get negative pixels.
- [Image>Rename] 'Tophat.tif'

Local background subtraction using IJs "Subtract Background"

- [File > Open..] "../dna-damage-synthetic-data/Damaged.tif"
- [Process > Subtract Background..]
    - radius=5

This seems to implement a 'rolling ball' background estimation (=> Whiteboard). I don't understand the mathematical algorithm how to compute this, but based on the code that i saw it seems not so simple (see also here and here).

Discussion: Comparison of the different BG subtraction methods

- Difference between median-subtraction and top-hat:
    - top-hat underestimates background in presence of noise
    - median overestimates background in presence of "holes"
    - median does not work well along curved egdes
- Local background subtraction should only be used for (small) isolated objects
    - e.g., may fail if used to find the background intensity in an image full of cells

Segmentation with uneven background

If there is a strong uneven background in your image segmenting the objects with just one threshold will not work.

Ways to combat this challenge are:

- Local background subtraction
    - If possible, this is the best method because it also corrects the intensities in your image in the right way.
- Local tresholding
    - Works, but does not correct intensities.
- Edge enhancement combined with 'fill holes' (not shown)
    - Also works, but also in fact even alters your intensities in a bad way.

Example data:

- ../data_new/uneven-background/blobs-with-background.tif
- ../data/workflow_autophagosomes/autophagosomes_raw.tif

Activity: Try different local background subtraction methods

- Try different **local background subtraction** methods (see here for more detailed examples)
    - Subtract a **median** filtered version of the image from the original
    - Subtract a morphological **opening** of the image from the original (i.e. do a **top-hat** filter)

    - Use ImageJ's "Subtract background" method
- After local background subtraction, segment the objects with a global intensity threshold and connected component analysis
- ImageJ commands:
    - [Process > Filters > Mean]
    - [Process > Filters > Median]
    - [Process > Filters > Minimum]
    - [Process > Filters > Maximum]
    - [Process > Image Calculator]
    - [Process > Subtract Background]
    - [Image > Adjust > Threshold]
    - [Analyze > Analyze Particles]

Automated global thresholding

Sometimes, if you have many images to analyse, you may need automated methods that find the threshold for you. There are many good methods, but it is dangerous to apply them, and you always need to check if it worked!

Workflow:

- Apply automated thresholding

Example data within Fiji:

- [File > Open Samples > Blobs]
    - [Image > Lookup Tables > Invert LUT]
- [File > Open Samples > Hela cells]
    - [Image > Color > Split Channels]
- [File > New Image]
    - [Process > Noise > Add Noise]

Fiji commands:

- [Image > Adjust > Auto Threshold]

Discussion

- Many automated thresholding methods always find a threshold, even if there is only noise.

Automated local tresholding

Automated local thresholding is another method to segment objects in the prescence of a locally varying background.

Workflow:

- Threshold using a local thresholding algorithm
- Connected component analysis
- Fiji commands:
    - [Image > Adjust > Auto Local Threshold]
        - Documentation: http://imagej.net/Auto_Local_Threshold
    - [Analyze > Analyze Particles]

Spot detection using Difference of Gaussian

The Difference of Gaussian (DoG) is a very popluar method for object detection when there is uneven background or also an uneven object brightness distribution.

Workflow:

- Blur image with a small Gaussian (about the size of the objects)
- Blur image with a large Gaussian (about two-three times the object size)
- Subtract the large blur from the small blur; this is the DoG!
- Threshold the resulting image to find the object centers
- Fiji commands:
    - [Process > Filters > Gaussian]
    - [Process > Image Calculator]
    - [Analyze > Analyze Particles] or [Process > Find Maxima]

Discussion:

- The object shape is not preserved with this method
    - Workflow:
        - Find object centers using DoG
        - Find object volumes growing from the object centers

Object manipulation

Once you found your objects you often want to split touching object or measure only in certain parts of the object or just close-by the object.

Example data:

- [File > Open Samples > Blobs]; [Image > Lookup Tables > Invert LUT]
- [File > Open Samples > HeLa Cells]; [Color > Split Channels]
- [File > Open Samples > Fluorescent Cells]; [Color > Split Channels]

Convert the image to a binary image (s.a.) and then manipluate the object's shape using below methods.

Object growing and shrinking and boundary creation

Workflow:

- Example data:
    - Simply draw some objects yourself
- Dilate (grow, 'maximum') and erode (shrink, 'minimum') the binary image to change the object size
- In addition, by subtracting the eroded image from the original one can generate outlines
- Fiji commands:
    - [File > New Image...]
        - Use Fiji's drawing tools to generate some objects of choice
    - [Process > Filters > Minimum]
    - [Process > Filters > Maximum]
    - [Image > Duplicate]
    - [Process > Image Calculator]
        - Use this to subtract images from each other

Application examples:

- Measure the amount of protein X in the nuclear envelope
    - Data given: Confocal slice with staining for DAPI and protein X

Distance transform

The distance transform is a powerful tool for many image analysis tasks, also in biology. Using the distance transform you can

- measure distances between different objects (from different fluorescence channels)
- select specific regions within of or close by objects
- split objects based on a shape criterium (see below)

Workflow for measuring distances to nearest object in another channel:

- Example data: One object in Ch0, many objects in Ch1
  - ../data-new/distance-transform-applications/distances-between-objects.tif
- Split stack into individual channels
- Distance transform Ch0 => DistTrafoCh0
- Measure mean intensity of Ch1 objects in DistTrafoCh0
- Fiji commands:
  - [Image > Stacks > Stack to Images]
  - [Process > Binary > Distance Map]
  - [Image > Rename]
  - [Analyze > Set Measurements]: [X] Mean gray value; Redirect to 'DistTrafoCh0'
  - [Analyze > Analyze Particles]: [x] Display Results

Workflow for finding circle centers:

- Example data:
  - ../data-new/distance-transform-applications/hollow-tubes.tif
- Compute distance transform
- Find local maxima
- Fiji commands:
  - [Process > Binary > Distance Map]
  - [Process > Find Maxima]

Further reading:

- http://imagej.net/Local_Thickness

Object splitting

Often, objects that are very close are idenftied as one object in the connected component analysis. The distance-transform based watershed algorithm is the most commonly way to deal with this.

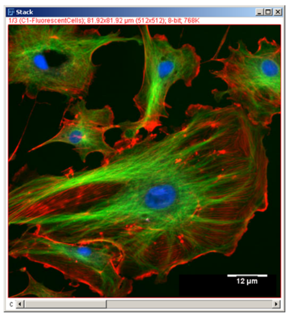
Workflow:

- Subject the binary image to a (distance-transform based) watershed algorithm
- Fiji commands:
  - [Process > Binary > Watershed]

Comments:

- E.g., CellProfiler offers a number of interesting choices for object splitting, which are not only shape but also intensity-based

Handling multi-color images and adding a scale bar


Stack
1/3 (C1-FluorescentCells); 81.92x81.92 µm (512x512); 8-bit; 768K
12 µm

Aim: make the images look like the image above!

- **[File>Import>Image Sequence] '../multi-color/'** (*opens all images in one folder; **Mac**: just click on the folder; **Win**: click on one of the files*)
- **[Image>Color>Make Composite] 'Display Mode = Composite'** (*converts to an image type that is good for colors*)
- Use the **c** slider at bottom of image to select a channel and then change its color via **[Image>Lookup Tables]**
- **[Image>Properties] 'Unit of length = um' 'Pixel width = 0.16' 'Pixel height = 0.16'** (*changes the scaling to physical distances*)
- Add scale bar: **[Analyze>Tools>Scale Bar..] 'Overlay = Check'**
- Save image twice:
  - **[File>Save As>Tiff]**
  - **[File>Save As>Jpeg]**
- open both images in **PowerPoint** and compare
- reopen both in **Fiji** and compare

What is better for saving? Tiff or Jpeg?

Tip for comparing different images

When comparing images it is a good idea to ensure that the LUT settings are the same for all images. This can be achieved with **[Image>Adjust>Brightness/Contrast..]** clicking **[Set]** and choosing **'Propagate to all other open images = Check'**.

Colocalisation: Pixel-, distance- and object-based

Literature: A guided tour into subcellular colocalization analysis in light microscopy. S. B O LT E & F. P. C O R D E L I È R E S Journal of Microscopy, Vol. 224, 2006, pp. 213–232.

General considerations

- Use tetraspec beads to check your microscope
- You have to chose your point of view: overlap of ch1 with ch2 vs. overlap of ch2 with ch1
- You can measure pixel- or object-based overlap or some distance criterium (e.g., centroid- or boundary-based)
- Diffraction limit depends on wavelength
- Due to the diffraction limit everything can appear to colocalise with ER or tubulin
- The cytoplasm in a cell can be quite small, so proteins might colocalise just by chance
- You should make only comparative statements sich as: In condition XY the colocalisation increases
- Create synthetic (simulated) test images to check if your analysis does what you want

Segmentation: Local background subtraction and thresholding

First we segment the images, i.e. we make objects pixels 255 and background pixels 0.

- **[File>Open..] '"../colocalization/stain1.tif"**
- **[Image>Rename..] 'Title=original'**
- **[Image>Duplicate] 'Title=median'**
- **[Process>Filters>Median..] 'radius=20'**
  - 3D: [Process>Filters>Median 3D..]
- **[Process>Image Calculator] 'Image1 = original' 'Operation = Subtract' 'Image2 = median' 'Create new window=Check'**
- **[Image>Adjust>Threshold..] 'lower th=30' 'upper th=255'** (*you don't have to press [Apply] now, the actual 'applying' of the threshold will happen in the next step*)
- **[Analyze>Analyze Particles..] 'Size = 5-Infinity' 'Pixel units = Check' 'Show = Masks' [OK]** (*selecting 'Show = Masks' generates a binary image in which only 'connected components' of at least 5 pixels are kept, i.e. 'noise' is filtered*)
  - for 3D use [Analyze>3D Objects Counter]
- **[File>Save As..] '"../colocalization/stain1_segmented.tif"**
- You also have to do this "stain2.tif"...

Area, object and distance-based colocalization

First we load the data and compute the overlap image:

- **[File>Open..] '"../colocalization/stain1_segmented.tif"**
- **[File>Open..] '"../colocalization/stain2_segmented.tif"**
- **[Process>Image Calculator] 'stain1_segmented.tif' 'AND' 'stain2_segmented.tif'**
- **[Image>Rename..] 'Title = overlap'**

Now, you can compute an area-based, object-based or distance-based colocalisation:

- **[Analyze>Set Measurements..] 'Centroid = Check' 'Display label = Check'** *('Display label = check': this will associate measurements with the name of the image (and ROI) on which they were computed.)*
- For all three images, i.e. **'stain1_segmented.tif', 'stain2_segmented.tif', 'overlap'**, run the following command :
  - **[Analyze>Analyze Particles] 'Display Results = Check' 'Summarize =Check' 'Display Labels**
    - for 3D use [Analyze>3D Objects Counter]
    - choose particle selection criteria that make sense for your project...
- Object based colocalization: **Count** (e.g., divide overlap by stain1)
- Area based colocalization: **Area** (e.g., divide overlap by stain1)
- Distance based colocalization: you have to write some code to find particles that are next to each other in the Results tables

The meaning of intensities in confocal and widefield microscopy

todo: put images here

Intensity-based quantifications of BFA-induced Golgi disassembly (confocal)

Data

- 3D confocal stacks
- ...

Challenges

- Intensities depend on evertyhing!

Detector offest (background) subtraction

Measure and subtract later

Formula for sum intensities: Mean_Background * Area

Subtract from whole image

32-bit conversion necessary to be truly independent on measurement area.

Accuracy of background subtraction

Subtracting the wrong background can lead to false biological interpretations of your data. There will always be a small error in the background subtraction. In order to figure out how much this will influence your sum-intensity measurements you can do the following calculation:

...

Issues

What to do if you have many data sets? Can I subtract the same background from all?

Maximum intensity in 3-D maximum intensity projection

Maximum value in a 3-D maximum intensity projection is proportional to the highest local density of observed fluorophores, where local corresponds to the confocal point spread function (~200x200x800 nm^3).

Workflow

- Maximum projection
- Manual background subtraction
- Draw ROI
- Measure
- Easy to compute readout for the maximal local concentration in a 3-D data set.

Normalisation strategies

In a time-lapse experiment one could use the intial maximal local concentration in each cell and then monitor the

Pro

Con

- Only very local readout of whole cell
-
    -
    - Measure the average maximal local concentration in a set of control cells and divide by this value

Sum

Intensity-based quantifications of H2B-mCherry during the cell-cycle (wide-field)

Mean intensity

Maximum intensity

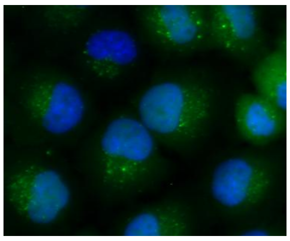
Workflow: Autophagosome quantification

In the following you will learn all the ingredients to perform a very typical cell biology image analysis workflow.

Your working directory is: **'../data/workflow_autophagosomes'**. This folder contains the two main input files **'autophagosomes_raw.tif'** and **'nuclei_raw.tif'**. There are also **a lot of other files** stored, some of which you will generate during the course. You will **overwrite** most of those files **with your own results**!

If you messed up or missed a step there also is a sub-folder **'teacher'**, form which you can load all files that you need during the practicals :-)

What to measure?



- **[File>Open..] 'bothChannels.jpeg'**
- Number of spots per cell
  - autophagosome initiation
  - not a 3-D image, but you could squeeze your cells...
- Size of spots
  - diffraction limit!
- Average intensity of one spot
  - diffraction limit!
  - how large should i draw my ROI?
- Sum intensity of one spot
  - autophagosome maturation
  - technical considerations:
    - microscope settings change intensities
    - total expression level changes intensities
    - good to divide by total cell intensity => fraction of protein localized to one autophagosome
    - local background correction needed for spot intensity
- Total cell intensity
  - proportional to total amount of protein (in wide-field microscopy)
- Mean intensity of each cell
  - not clear what the biological meaning in the wide-field images is.

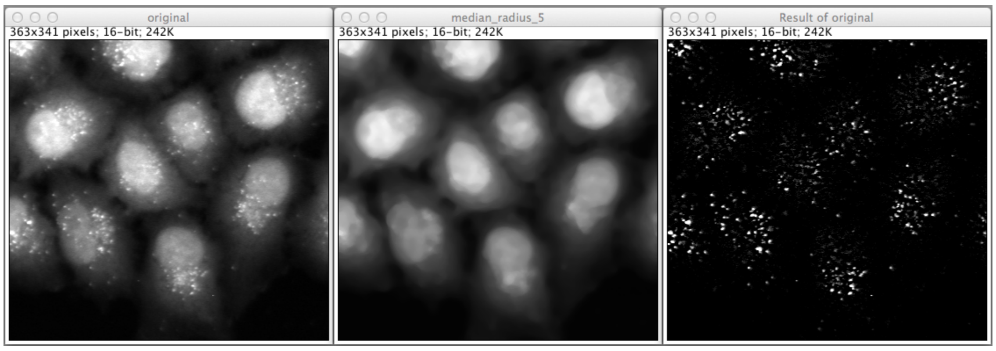
Local background subtraction

In biological fluorescence microscopy one often wants to detect locally bright objects such as vesicular structures on top of a non-uniform background fluorescence, e.g. from unbound cytoplasmic protein. There are different methods to remove such 'background' fluorescence from the image:

- corrected_image = image - mean_filter(image, radius)
- corrected_image = image - median_filter(image, radius)
- corrected_image = image - morphological_opening(image, radius) = top_hat(image, radius)
- corrected_image = IJs "RollingBall" Algorithm
- *someone knowing other methods?*

In all methods the *radius* parameter should be "quite a bit larger" than the radius of the largest locally bright structure that you want to measure (why that is becomes clear when we discuss the methods in detail).

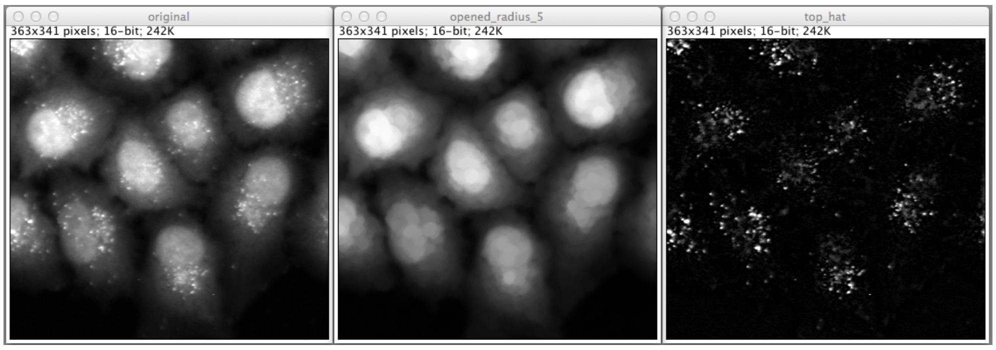
Local background subtraction using a median filter


original
363x341 pixels; 16-bit; 242K
median_radius_5
363x341 pixels; 16-bit; 242K
Result of original
363x341 pixels; 16-bit; 242K

Duplicate image and apply median filter to remove the locally bright spots. Then subtract the median filtered image from the raw image and save the result for later use.

- **[File>Open..] 'autophagosomes_raw.tif'**
- **[Image>Rename..] 'Title=original'**
- **[Image>Duplicate] 'Title=median'**
- Select the 'median' image and **[Process>Filters>Median] 'radius=5'**
- **[Process>Image Calculator] 'Image1 = original' 'Operation = Subtract' 'Image2 = median' 'Create new window=Check' '32-bit output=Uncheck'**
- **[File>Save] 'spots_median.tif'**

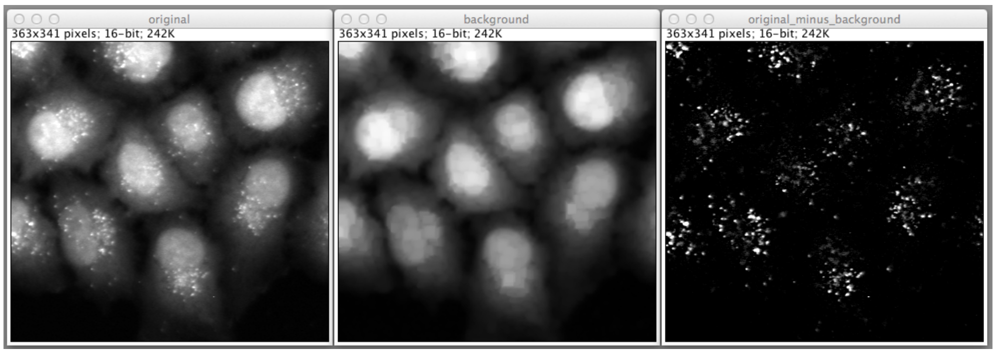
Local background subtraction using a top-hat filter


original
363x341 pixels; 16-bit; 242K
opened_radius_5
363x341 pixels; 16-bit; 242K
top_hat
363x341 pixels; 16-bit; 242K

A morphological opening filter is applied to the image and subtracted from the original. The morphological opening is defined as the dilation of the erosion if the image. Alltogether this reads: top_hat(image) = image - dilation(erosion(image))

- **[File>Open..] 'autophagosomes_raw.tif'**
- **[Image>Rename..] 'Title=original'**
- **[Image>Duplicate] 'Title=opened'**
- **[Process>Filters>Minimum..] 'radius=5'**
- **[Process>Filters>Maximum..] 'radius=5'**
- **[Process>Image Calculator] 'original' 'Subtract' 'opened' 'Create new window=Check' '32-bit output=Uncheck'**
- **[File>Save] 'spots_tophat.tif'**

Local background subtraction using IJs "Subtract Background"


original
363x341 pixels; 16-bit; 242K
background
363x341 pixels; 16-bit; 242K
original_minus_background
363x341 pixels; 16-bit; 242K

- **[File>Open..] 'autophagosomes_raw.tif'**
- **[Process>Subtract Background..] 'radius=5'**

This seems to implement a 'rolling ball' background estimation (=> Whiteboard). I don't understand the mathematical algorithm how to compute this, but based on the code that i saw it seems not so simple (see also here and here).

Comparison of the different BG subtraction methods

Whiteboard session:

- Difference between median-subtraction and top-hat:
  - top-hat underestimates background in presence of noise
  - median overestimates background in presence of "holes"
- Local background subtraction should only be used for (small) isolated objects
  - e.g., may fail if used to find the background intensity in an image full of cells
- Using the Spheroids image in 3D_Segmentation one can d emonstrate that the top-hat (dramatically) underestimates the background in noisy images

Further enhancing of spots using a Laplacian of Gaussian filter (optional)

Above local background subtraction methods already helped a lot to enhance the spots; however in some cases there might still be some patchy locally bright regions left that are not corresponding to "real" spots. The reason is that the local background subtraction methods cannot distinguish locall bright elongated from locally bright round objects. Convolution of the image with a Laplacian of Gaussian filter can help to further enhance spots of a certain size.

- ...

Spot detection using 'Find Maxima'

ImageJ's 'Find Maxima' considers a pixel a maximum if its intensity is higher - by the 'Noise tolerance' - than neighboring pixels; see Topographic prominence.

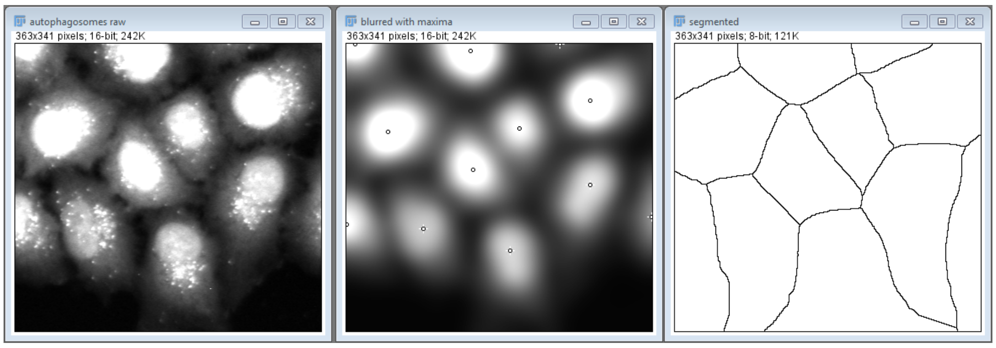
- **[File>Open] 'spots_median.tif'**
- **Draw a line ROI** across some of the spots and **[Analyze>Plot Profile]**
  - Check how many gray values the spots 'stand out'; use this value as 'Noise tolerance' in the next step
- **[Process>Find Maxima..] 'Noise tolerance=20' 'Output type=Single Points'**
  - check **'Preview point selection'** and explore different 'Noise tolerance' values!
- **[Process>Math>Divide..] 'Value=255'**
  - => spot pixel will have a value of 1 (better for counting them later..)
  - The image will appear white; you have to adjust the contrast to see the dots **[Image>Adjust>Brightness/Contrast]**
- **[File>Save] 'spots_points.tif'**

Exercise: Explore influence of local background removal

- **[File>Open..] 'autophagosomes_raw.tif'**
- **[Process>Filters>Gaussian Blur..] 'sigma=15'** (this removes the spots)
- **[Process>Find Maxima..] 'Noise tolerance=100'**

As you can see there are maxima detected only due to the cellular background. If you do the same using 'spots_median.tif' - where the background was removed - as input image there should be no maxima (for this 'Noise tolerance').

Cell detection using seeded watershed


autophagosomes raw
363x341 pixels; 16-bit; 242K
blurred with maxima
363x341 pixels; 16-bit; 242K
segmented
363x341 pixels; 8-bit; 121K

The seeded watershed algorithm implemented in ImageJ's 'Find Maxima' first finds local intensity maxima as starting ('seed') points. From these seed points it performs a 'region growing' algorithm, using the intensity information in the image to draw dividing lines at dim parts of the image.

- **[File>Open] 'autophagosomes_raw.tif'**
- **[Process>Filters>Gaussian Blur..] 'sigma=10'** (*removes features that 'distract' from the overall cell shape*)
- **[Process>Find Maxima..] 'Noise tolerance=50' 'output=Segmented Particles'** (*choosing 'Segmented Particles' invokes the Wathershed algorihm*)
- **[File>Save] 'cells_bw.tif'**

Explanation for this use-case: Look at the image after blurring it. Then imagine it starts raining. Now imagine where water running down from the different hills (bright pixels) would meet. Those are the dividing lines.

Exercise

Try the same leaving out the 'Gaussian Blur' step. Can you get it to work?

Generate cell 'objects' that can be used for measurements

We run the 'Particle Analyzer' to convert the binary cell image into 'objects' (i.e., regions of interest = ROIs). This will be handy later for cell-based measurements.

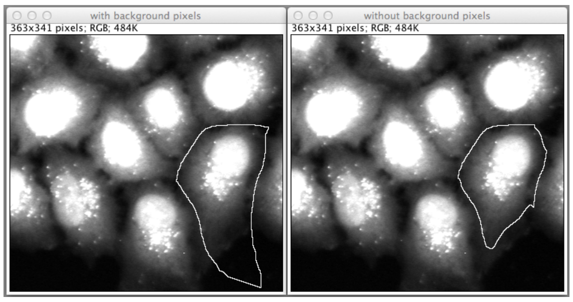
- **[File>Open] 'cells_bw.tif'**
- **[Analyze>Analyze Particles..] 'Exclude on edges = Check' 'Add to Manager = Check'**
- **[File>Open] '../data_course/autophagosomes_raw.tif'**
- Click on the ROIs in the ROI Manager to see them overlayed on the raw data.

Exercise: Exclude cells based on their shape

1. Explore all the different output options of the 'Particle Analyzer'!
2. Experiment with different particle exclusion criteria. For example:

- Close 'ROI Manager' and select image: 'cells_bw.tif'
- **[Analyze>Analyze Particles..] 'Exclude on edges = Do Not Check' 'Add to Manager=Check'**
  - Try: '**Size = 15000-Infinity**' and '**Circularity = 0.00-1.00**'
  - Or try: '**Size = 0-Infinity**' and '**Circularity = 0.60-1.00**'
    - Click **[Help]** to figure out what 'Circularity' is.

Improved cell detection by excluding background pixels


with background pixels
363x341 pixels; RGB; 484K
without background pixels
363x341 pixels; RGB; 484K

The problem of the seeded watershed algorithm is that the 'grows into the background' (see image). To avoid this one has to threshold the cells and combine this with the results of the watershed:

- **[File>Open] 'autophagosomes_raw.tif'**
- **[Process>Filters>Gaussian Blur..] 'sigma=5'** (*just to get rid of some noise*)
- **[Image>Adjust>Threshold..] 'lower th=230' 'upper th=Max' [Apply]**
- **[Image>Rename..] 'foreground'** (*all background pixels are zero*)
- **[File>Open] 'cells_bw.tif'** (*this is the image that we got from the watershed*)
- **[Process>Image Calculator..] 'foreground' 'AND' 'cells_bw.tif' [OK]** (*we combine both...*)
- **[File>Save] 'cells_bw_improved.tif'** (*...and get an image where we removed the background pixels and still have dividing lines between the cells*)
- **[Analyze>Analyze Particles..] 'Size=100-Infinity' 'Exclude on edges=Check' 'Add to Manager=Check'** (*simply finds the cell objects*)
- **[ROI Manager>More>>Save..] 'cells_improved.zip'**

Measure spots per cell

ImageJ can measure lots of features. To have our readout more "to-the-point" we will first only select a small subset:

- **[Analyze>Set Measurements..] 'Area = Check' 'Integrated Density = Check' 'Mean gray value = Check'**

To measure how many 'spots' (vesicular structures) we have in each cell, we simply measure in each cell ROI the integrated intensity in the 'spots_points.tif' image, where each pixel marking a spot has the value 1 (that is a very typical trick in image analysis :-).

- **[File>Open] 'spots_points.tif'** (*you may see nothing on this image...do you know why?*)
- **[Image>Adjust>Brightness/Contrast..] [Auto]**
- **[Analyze>Tools>ROI Manager..]**
- **[ROI Manager>More>>Open..] 'cells.zip'**
- **[ROI Manager>Measure]**
- Click on Results Table and **[File>Save] 'spot_count.csv'**

The 'RawIntDen' value is the spot count. How many spots did you find?

Exercise

Manually draw a region on the image, add it to the ROI Manager **[ROI Manager>Add]** and measure the number of spots in this region.

Manual background subtraction on whole image

If we want to measure total cell intensity in a biophysically meaningful way we have to set the image background intensity to zero. Since cells can grow dense it can be difficult or even impossible to find the correct background value in one image. Thus, sometimes one has to manually subtract a fixed background value from the image.

(=> Whiteboard session: why background subtraction; why convert to 32-bit.)

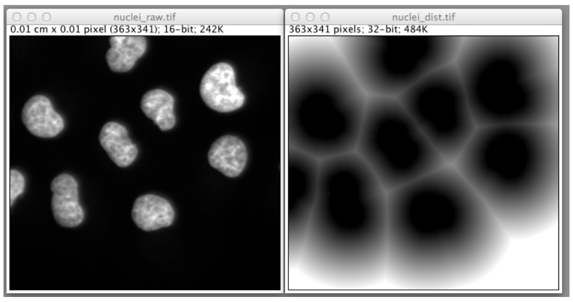
- **[File>Open..] 'autophagosomes_raw.tif'**
- **[Image>Adjust>Brightness/Contrast] 'Minimum=190' 'Maximum=230'**
- Measure background intensity: draw a little ROI in the background and **[Analyze>Measure..]**
- **[Image>Type>32-bit]**
- **[Process>Math>Subtract..] 'Value=194'** (*Possible pitfall: if you had a ROI on the image the value was only subtracted in this ROI*)
- **[File>Save As>Tiff..] 'autophagosomes_bgcorr.tif'**
- Measure background intensity again using **[Edit>Selection>Restore Selection..]** to get the same ROI (*should be zero now*)

Exercise

Do the same but leave out the 32-bit conversion step. Now measure the intensity in the background after correction! What happens?

Compute nuclear distance map

(=> Whiteboard session on Distance Transform)


nuclei_raw.tif
0.01 cm x 0.01 pixel (363x341); 16-bit; 242K
nuclei_dist.tif
363x341 pixels; 32-bit; 484K

Quite often in biology one wants to know how far a certain structure is away from another (e.g. endocytosis: vesicles from plasma membrane). Such distances often can be quite easily measured using the 'Distance Transform'.

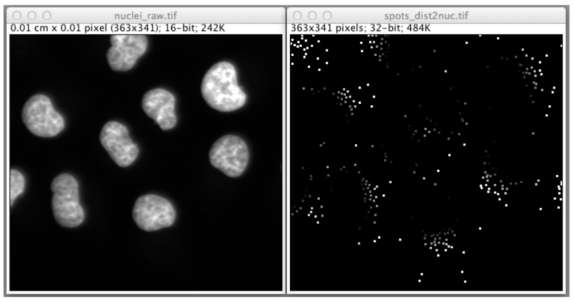
- **[File>Open..] 'nuclei_raw.tif'**
- **[Image>Adjust>Threshold..] 'lower = 500' 'upper = Maximum' [Apply]**
- **[File>Save As>Tiff..] 'nuclei_bw.tif'**
- **[Edit>Invert]** (*check: nuclei pixels should now be 0 and background 255; needed for Distance Map*)
- **[Process>Binary>Options..] 'EDM output=32-bit'** (*enables distances >255*)
- **[Process>Binary>Distance Map]** (*pixel values are now distances to nearest nucleus*)
- **[File>Save As>Tiff..] 'nuclei_dist.tif'**

Exercise

...

Use nuclear distance map on detected spots

In order to measure the distance of each previously detected spot to the nucleus we (almost) simply multiply the distance map with the spot image. The only problem we have is that a zero in the final image could mean: (i) there was no spot or (ii) there was a spot but its distance to the nucleus was zero. To distinguish these cases we will set non-spot pixels to NaN (Not a Number) before we do the multiplication


nuclei_raw.tif
0.01 cm x 0.01 pixel (363x341); 16-bit; 242K
spots_dist2nuc.tif
363x341 pixels; 32-bit; 484K

- **[File>Open..] 'spots_points.tif'** (*pixel values: 1 = spot; 0 = no spot*)

Set non spot pixels to NaN (Not a Number):

- **[Image>Type>32-bit]** (*necessary to enable NaN values*)
- **[Image>Adjust>Threshold..] [Set] 'lower=0.5' 'upper=1' [Apply]**
  - When asked: **Check 'Set background pixels to NaN'**

Multiply spot image with distance image:

- **[File>Open..] 'nuclei_dist.tif'** (*pixel values: distances to nearest nucleus*)
- **[Process>Image Calculator..] 'Image1 = nuclei_dist.tif' 'Operation = Multiply' 'Image2 = spots_points.tif' '32-bit result = Check'**
  - *the value of each pixel is now distance of the spot to nearest nucleus or NaN if there was no spot*
- **[File>Save As>Tiff..] 'spots_dist2nuc.tif'**

Measure intensity inside autophagosomes

(=> Whiteboard session on intensity measurements in diffraction limited objects in the presence of local background (unbound protein))

Often one wants quantify the intensity of objects as it reports the amount of bound labelled protein. Here, we use the 'spots_median.tif' image, where the cytoplasmic background has already been subtracted. In order to restrict the intensity measurement to the region of the spots, we use the 'spots_point.tif' image, where the center of each spot has the value 1 and the other pixels are 0. In order to measure the whole spot intensity we will dilate this image and then multiply (mask) onto the 'spots_median.tif' image (for the masking we need to set pixels outside spots to NaN (Not a Number)).

- **[File>Open..] 'spots_points.tif'** (*pixel values: 1 = spot; 0 = no spot*)
- **[Process>Filters>Maximum..] 'radius=2'** (*enlarge spots to include all fluorescence*)
- **[File>Open..] 'spots_median.tif'** (*pixel values: background corrected autophagosome intensities*)
- **[Process>Image Calculator..] 'Image1 = spots_points.tif' 'Operation = Multiply' 'Image2 = spots_median.tif' '32-bit result = Check'**
  - *pixel values: inside spots: background corrected spot intensity; outside spots: NaN*
- **[File>Save As>Tiff..] 'spots_intensity.tif'**

Perform all kinds of cell based measurements

Once we have the cell ROIs we can measure many cell-based features, simply loading different input images for the measurement:

- **[Analyze>Tools>ROI Manager..]**
- **[ROI Manager>More>>Open..] 'cells.zip'** or 'cells_improved.zip'
- **[File>Open] 'spots_points.tif', 'autophagosomes_bgcorr.tif', 'spots_intensity.tif', and 'spots_dist2nuc.tif'** (*if you did not generate all these images yourself you can load them from the 'teacher' folder*)
- For each of above input images:
  - Select the input image (i.e., click on it once)
  - **[ROI Manager>Measure]**
  - Click on the Results Table and **[File>Save] '*a good name*.csv'**
  - Close Results Table! (*otherwise the next measurement values will be appended in case you want to measure another image*)

Exercise

Try to remember the biological interpretation of these measurements. Think about ratios of any of the measured numbers.

Automation using the IJ macro language

In order to apply the previous measurements to more than one image, we will now automate all the step that we did previously. For this we will use the ImageJ Macro language (http://rsb.info.nih.gov/ij/developer/macro/macros.html) and ImageJ's inbuilt Macro Recorder [Plugins>Macros>Record].

Create a macro for automated cell detection

First clean up things, e.g. by restarting ImageJ, and then turn on the Macro Recorder [Plugins>Macros>Record..]

Execute below commands for automated cell detection (these are the same commands we used earlier):

- [File>Open] 'autophagosomes_raw.tif'
- [Process>Filters>Gaussian Blur..] 'sigma=10'
- [Process>Find Maxima..] 'Noise tolerance=100' 'Output type=Segmented Particles'
- [Analyze>Analyze Particles..] 'Size=100-Infinity' 'Exclude on edges=Check' 'Add to Manager=Check'
- In the Macro Recorder click [Create].
  - The 'Script Editor' with your macro will appear.
- Close all windows (also the ROI Manager).
- In the Script Editor click [Run].
  - Did it work? If yes: great! Maybe your first code!

- Script Editor: [File>Save As..] '.../CellDetection.ijm'

Advanced task

Look at the macro for the improved cell detection with exclusion of background pixels:

- [File>Open..] '.../teacher/CellDetection2.ijm' Try to understand the code. Look up commands that you don't know on http://rsb.info.nih.gov/ij/developer/macro/functions.html

Create a macro for automated spot detection

First clean up things, e.g. by simply restarting ImageJ, and turn on the Macro Recorder [Plugins>Macros>Record..]

Execute below commands for automated spot detection (same commands we used earlier):

- [File>Open] 'autophagosomes_raw.tif'
- [Image>Rename..] 'Title=raw'
- [Image>Duplicate] 'Title=median_bg'
- Select 'median_bg' image and [Process>Filters>Median] 'radius=5'
- [Process>Image Calculator] 'raw' 'Subtract' 'median_bg' 'Create new window=Check' '32-bit output=Uncheck'
- [Process>Find Maxima..] 'Noise tolerance=100' 'Output type=Single Points'
- [Process>Math>Divide..] 'Value=255'
- [Image>Rename..] 'spots_points.tif'
- Script Editor: [File>Save As..] '.../SpotDetection.ijm'
- Script Editor: [Run]

Combine cell- and spot-detection macros and add the spot-per-cell measurement

Open both macros (of course, if you have them both open already you can skip this step) and then simply code and paste the SpotDetection code below the CellDetection code.

- [File>Open] 'CellDetection.ijm' (or 'CellDetection2.ijm')
- [File>Open] 'SpotDetection.ijm'
- Script Editor: Copy code of SpotDetection below CellDetection
- Script Editor: [File>Save] 'CellAndSpotDetection.ijm'
- Script Editor: [Run]

Now we need to add the spot-per-cell counting by recording the appropriate macro commands:

- Close the Macro Recorder and restart it: [Plugins>Macros>Record..]
- Click on the 'spots_points.tif' image
- Roi Manager: [Measure]
- Results table: [File>Save As..] 'spot_count.csv'
- Macro Recorder: [Create]
- Script Editor: Copy your new code at end of 'CellAndSpotDetection.ijm'
- Script Editor: [File>Save] 'CellAndSpotDetection.ijm'
- Script Editor: [Run]

Exercise

Run the code twice in a row without closing any windows. Confirm that both the content of the ROI Manager and the content of the Results table get messed up. What happens?

Close all windows in a macro (help yourself!)

We saw above that we need to automatically close Results Table and ROI Manager before running our macro. Also it would be good to close all image windows. In this case macro recording not always helps (at least on the teachers Mac for a specific version of ImageJ and Fiji). Try to find the right commands using any of the following approaches:

- Have some images, Results Table and ROI Manager open (e.g. run CellAndSpotDetection.ijm) and record yourself closing them all [Plugins>Macros>Record..]; [Create] the macro and try if it works!
- Google: 'imagej macro close results table'
- Google: 'imagej macro close roi manager'
- Google: 'imagej macro close all images'

What did you find?

If you found nothing useful you can see what I found:

- [File>Open..] '../teacher/closeEverything.ijm'

Adding a function to close all windows

Above we already found the commands to close all windows. Here we will see how to neatly pack them into a "function" such they can be called conveniently from every point of your macro.

- [File>Open..] '../teacher/CellAndSpotDetection_Improved.ijm'
- Script Editor: [Run]
  - run it again: as you see now it works even twice!

=> Discussion of the code with help of teacher.

Adding code to select an arbitrary image

Right now our 'CellAndSpotDetection_Improved.ijm' only runs on one specific image. Let's look at a slightly modified version that makes it useable for any given input image:

- [File>Open..] '../teacher/CellAndSpotDetection_ChooseImage.ijm'
  - do you understand the code?
- [Run] macro on files in '../data_small/'
- [Run] macro on files in '../data_large/'

=> Discussion of the code with help of teacher.

Batch me if you can

Of course, we'd like to run a macro on many images. In ImageJ there is the function [Process>Batch>Macro..], which automatically applies a macro to all files in the 'Input..' folder. However, this function is very limited, for instance it cannot handle saving results tables.

As a more flexible alternative here is some code that handles the batch processing and you (more-or-less) only have to copy and paste your specific macro into it:

- [File>Open..] '../teacher/runAsBatch.ijm'

This code involves some 'real programming' that, in principle, you don't have to understand to use it. We will anyway discuss it together just to give you an idea.

(=> Discussion of the code with teacher.)

Batch cell and spot detection

Ok, let's have a look the final code, which apart from a few marked changes is just the central part of 'CellAndSpotDetection_ChooseImage.ijm' copied into 'runAsBatch.ijm':

- [File>Open..] '../teacher/CellAndSpotDetection_Batch.ijm'
- [Run] on 'data_small'
- [Run] on 'data_large'

- Examine the output tables in Excel

Exercise

Think about the following: Is this code ready to use for a project? What is missing?

Batch processing considerations

The macro 'CellAndSpotDetection_Batch.ijm' is not bad but it is also missing a few things:

- possibly: add more measurements; they could be simply added by recording and copying more code into the current script.
- essentially: saving of output images for visual quality control!
  - e.g., raw data with overlay of cell boundaries
  - e.g., raw data with overlay of detected spots
- possibly: add graphical user interface for some of the parameters:
  - e.g., minimal cell size
  - e.g., 'noise tolerance' for spot detection
  - e.g., threshold that distinguishes background from cells
- possibly: add more cell filter criteria:
  - e.g., remove cells from analysis that do not express enough protein

It is possible to add all this in ImageJ however I highly recommend to also check the free CellProfiler software. CellProfiler enables you to perform these operations without actual programming. Note that as of June 2015 CellProfiler only works for 2-D data but this may change.

Batch analysis in ImageJ

There are several ways to achieve batch analysis of many images in ImageJ:

- Putting images into an image (hyper-)stack
- Using [Process > Batch]
- Writing scripts, e.g. using the ImageJ Macro language

More information

- https://imagej.net/How_to_apply_a_common_operation_to_a_complete_directory
- https://imagej.net/Batch_Processing

Batch analysis using image stacks

If your images have the same dimensions in x,y (and z) and you want to apply the exact same operation to all of them the easiest option for batch analysis is to load all of them into one image (hyper-)stack.

Automatically counting the number of nuclei in many images

Counting the number of objects in many images is probably the most classical bio-image analysis application; thus let's start with this!

Workflow:

- Load all images into an image stack
  - [File > Import > Image Sequence]
    - '../data_new/mitocheck_movie'
- Threshold all images
  - [Image > Adjust Threshold]
    - [x] Dark background
    - [ ] Calculate threshold for each image
      - Uncheck this, otherwise some auto-thresholdin is going on!
    - [Apply]

            - Yes, do it for the whole stack!
- Set measurements, keeping tracking of the file-name
    - [Analyze > Set measurements]
        - ✓ Area
        - ✓ Display label
            - this will keep track of the filename
- Find and measure the cells in all images
    - [Analyze > Analyze Particles...]
        - ✓ Summarize
            - This will give you for each image the average results of all cells

Dealing with data distributed across different folders

Often your data might not be in one folder. For example have a look at the folder: "../data_new/data-in-subfolders" To still load them into one image stack you can use a macro that is provided in this repository:

Workflow:

- Load and run the macro
    - Drag&Drop onto Fiji: "../macros/import-from-subfolders.py"
    - Press [Run]
        - Filename contains: ".tif"
        - Choose as folder: "../data_new/data-in-subfolders/"
- You have all data in one stack and could for instance count the number of cells, as we learned above.

Comment:

- Like this, you cannot change the data and then save it back into the same folder-structure; this would require writing some macro code (see below)

Computing and saving 3-D maximum projections of many stacks

If you have 3-D data, all having the same x,y,z dimensions, you could implement batch operations using so-called "hyper-stacks"; the challenge here is to actually load the data into a hyperstack; once you have it, batch operations are easy.

Making a hyperstack from single files, using [Import Image Sequence]

Sometimes data beloning to different image stacks is distributed across single files. Here, you can see how to rearrange them into a hyper-stack after loading.

Workflow:

- Load all images
    - [File > Import Image Sequence]
        - "../data_new/mitosis-5D-single-files/"
- Make a hyperstack
    - [Image > Hyperstack > Stack to Hyperstack]
        - Determine the correct dimensions inspecting the filenames
        - Solution:
            - Order: xyczt
            - Channels: 2
            - Time-frames: 51
            - Z-Slices: 5
- Make a maximum projection of all time-frames
    - [Image > Stacks > Z Project]: Projection Type: "Max Intensity"
- Save data as individual files
    - [File > Save as > Image Sequence]

In above workflow we had to manually enter the dimensions of the data. If the filenames are very well structured those dimensions can be determined automatically as shown in the next workflow.

Making a hyperstack from single files using the Bioformats Importer

In order for this to work we need to install the very useful BioFormats plugin:

- [Help > Update]:
  - [Manage Update Sites]:
    - ✓ Java-8
    - ✓ Bio-Formats
      - [Close]
  - [Apply Changes]
- Restart Fiji

Workflow:

- Load individual files as hyperstack
  - [Plugins > Bio-Formats > Bio-Formats Importer]:
    - click on one file in the folder "../data_new/mitosis-5D-single-files/"
    - View stack with: "Hyperstack"
    - ✓ Group files with similar names
      - You'll see how it interprets your file-naming scheme
    - [OK]
- Do the processing of your choice and save the data again using [File > Save as > Image Sequence]

Image stacks

Another frequently occuring scenario is that you have 3-D data, where each file is one image stack. In the following we will see how to deal with this case. In fact, you can still do the hyperstack trick, because the Bio-Formats plugin can automatically combine data from image stacks into one hyperstack.

Workflow:

- Load many stacks into one hyperstack
  - [Plugins > Bio-Formats > Bio-Formats Importer]:
    - click on one file in folder: "../data_new/mitosis-4D-stacks/"
    - View stack with: "Hyperstack"
    - ✓ Group files with similar names
      - You'll see how it interprets your file-naming scheme
    - [OK]
- Do the processing of your choice, e.g. background subtraction
- Save the data again as individual stacks
  - [Plugins > Bio-Formats > Bio-Formats Exporter]
  - Note: the Bio-Formats Exporter has (had) a bug when doing this for multi-channel images

Batch analysis using [Process > Batch]

If your images have different sizes above tricks using an image stack will not work. In such cases the next thing you can try is the batch-processing that is built into ImageJ.

Converting various images to Tiff files

Data:

- Diverse images with different file formats and dimensions

Workflow:

- [Process > Batch > Convert]:
  - Input: "../data_new/different-file-formats/"
  - Output: you can choose :-)
  - Leave other options at default values

Display the filename on multiple images

It is good practice to have meaningful filenames, e.g., containing a particular treatment that your images were subjected to. For quick visual inspection of your data it can thus be useful to display the filename in the image.

Workflow:

- Use ImageJ's in-built batch processing, chosing one of the example macros.
  - [Process > Batch > Macro]
    - Input: "../data_new/meaningful-filenames/"
    - Output: your choice
    - Combine code from two of the example macros: "Print index and title", "Label"
      - `setFont("SansSerif", 18, "antialiased");`
      - `setColor("red");`
      - `drawString(getTitle(), 20, 30);`
    - [Process]

Don't worry for now too much about the code; we will learn more about it later.

Macro recording for putting a scale bar on multiple images

It is also very useful to have a scale bar in the images, e.g. for publication on a web-site. There is no predefined macro for adding a scale bar in Process > Batch; thus we have to record the command on our own.

Workflow:

- Record the macro command for putting a scale bar:
  - [Plugins > Macros > Record..]
    - Record: Macro (not Java)
  - Open some file, e.g. "../data_new/meaningful_filenames/Treatment_A.tif"
  - Put a scale bar:
    - [Analyze > Tools > Scale Bar..]
      - choose some parameters that you like
      - [OK]
  - Now the window should show some text, such as:
    - `run("Scale Bar...", "width=50 height=4 font=14 color=White background=None location=[Lower Right] bold");`
- Use the recorded text (copy & paste) to put a scale bar onto multiple images using [Process > Batch > Macro..] (see above)

Batch analysis with macro programming

References:

- https://imagej.nih.gov/ij/developer/macro/functions.html
- https://imagej.nih.gov/ij/developer/macro/macros.html

Even more automated counting of all cells in many images using an IJ-Macro

As our first real example of a macro we will automate above workflow "Automatically counting the number of cells in many images". Even though the trick of loading all images into one stack saved us a lot of work, we still needed to click a lot. If you want to do this workflow for many folders it surely will become boring. Thus, let's record a macro that does the job for us:

Record a macro:

- [Plugins > Macros > Record]
- Repeat all the steps from above, including opening the file!
- You should have recorded something like this:

```
//
// ../macros/CountCells.ijm
//
```

```
run("Image Sequence...", "open=/Users/tischi/Documents/imagej-courses/data_new/mitocheck-movie sort");
run("Set Measurements...", "area display redirect=None decimal=4");
setAutoThreshold("Default dark");
//run("Threshold...");
//setThreshold(29, 255);
setOption("BlackBackground", false);
run("Convert to Mask", "method=Default background=Dark");
selectWindow("mitocheck-movie");
run("Analyze Particles...", "  show=Nothing summarize stack");
```

If you now [Create] the macro and [Run] it, it should do the job.

- Note: "//" means that a line of code only is a comment
- We have to remove the "//" before the line starting with 'setTreshold', because we actually want to execute it.

Activity: Automatically save the results table

Above code does not automatically save the results table, try to add this, using macro recording.

Using variables

Some commands in our macro will be the same, but some stuff will be different for different files. It is good style to put all the things that can change at the top of the code, such that it is easy to modify. For this we need so-called "variables".

=> Interactive practical on variables: numbers, strings, adding, concatenating.

Activity:

In below code the directory with the images is already a variable (note how string-concatentation was used to paste it into the command).

Copy the code into Fiji and also **try to make the threshold a variable**.

```
// User input as variables
directory = "/Users/tischi/Documents/imagej-courses/data_new/mitocheck-movie";

// General code
run("Image Sequence...", "open=["+directory+"] sort");
run("Set Measurements...", "area display redirect=None decimal=4");
setAutoThreshold("Default dark");
setThreshold(29, 255);
setOption("BlackBackground", false);
run("Convert to Mask", "method=Default background=Dark");
run("Analyze Particles...", "  show=Nothing summarize stack");
```

Solution: '../macros/CountCells-Variables.ijm'

Making it really nice, with a graphical user interface

It is nice, not to have to type into the macro, but enter the variables with a GUI. In fact, it is not only nice, but also safer, because it prevents us from breaking the code by typing something wrong there.

Typically, I use google to find out how to do something related to programming:

Google: imagej macro get variable from user

- http://imagej.1557.x6.nabble.com/Having-a-macro-prompt-for-variable-input-td3694090.html
- getNumber("prompt", defaultValue);

=> interactive practical, getting a number via the GUI and printing it

Activity: adding another GUI element

In below code the threshold variable already has its GUI. Try to also **obtain the directory from the GUI**.

Hint:

- Google: imagej macro get directory

```
// User input
directory = "/Users/tischi/Documents/imagej-courses/data_new/mitocheck-movie";
threshold = getNumber("Enter threshold", 29);

// General code
run("Image Sequence...", "open=["+directory+"] sort");
run("Set Measurements...", "area display redirect=None decimal=4");
setAutoThreshold("Default dark");
setThreshold(threshold, 255);
setOption("BlackBackground", false);
run("Convert to Mask", "method=Default background=Dark");
run("Analyze Particles...", "  show=Nothing summarize stack");
```

Solutions:

- https://imagej.nih.gov/ij/macros/GetDirectoryDemo.txt
- getDirectory("Select a directory");
- '../macros/CountCells-GUI.ijm'

Activity: Saving the results table

Now that we have the input folder as a variable, we can automatically save the results table in a place related to this folder. Try to add this on your own.

Hints:

- Google: ImageJ Macro save results table
- http://imagej.1557.x6.nabble.com/save-results-table-as-csv-with-custom-name-td5003427.html
- Google: ImageJ Macro create new folder
- https://stackoverflow.com/questions/36144914/imagej-macro-make-new-folder-and-save-output-in-new-folder

The final touch: functions

It is very good for readability and for reusing parts of our code to pack it into small parts that belong together, so-called "functions".

Activity

Copy below code into Fiji and try to also **make a function for the thresholding**.

```
// User input
//
directory = getDirectory("Select a directory");
threshold = getNumber("Enter threshold", 29);

// Main
//

loadImagesIntoStack(directory);

// put into a function...
setAutoThreshold("Default dark");
setThreshold(threshold, 255);
setOption("BlackBackground", false);
run("Convert to Mask", "method=Default background=Dark");

measureCells();

// Functions
//
function loadImagesIntoStack(directory) {
	run("Image Sequence...", "open=["+directory+"] sort");
}

function measureCells() {
	run("Set Measurements...", "area display redirect=None decimal=4");
	run("Analyze Particles...", "  show=Nothing summarize stack");
}
```

Solution:

- ../macros/CountCells-Functions.ijm

Dealing with data in different folders using macro programming

Once your data is distributed across different folders none of the approaches above currently works (apart from the special script "../macros/import-from-subfolder.py").

Thus, we will learn how to deal with data in different folders. In programming, it is typical not to start from scratch, but modify existing code (everybody does that ;-). Thus, let's just try to understand below IJM (ImageJ-Macro) code. In fact, the way I found this code was like this:

GOOGLE: imagej macro batch process folders

CODE: https://imagej.nih.gov/ij/macros/BatchProcessFolders.txt

```
// "BatchProcessFolders"
//
// This macro batch processes all the files in a folder and any
// subfolders in that folder. In this example, it runs the Subtract
// Background command of TIFF files. For other kinds of processing,
// edit the processFile() function at the end of this macro.

   requires("1.33s");
   dir = getDirectory("Choose a Directory ");
   setBatchMode(true);
   count = 0;
   countFiles(dir);
   n = 0;
   processFiles(dir);
   //print(count+" files processed");

   function countFiles(dir) {
      list = getFileList(dir);
      for (i=0; i<list.length; i++) {
          if (endsWith(list[i], "/"))
              countFiles(""+dir+list[i]);
          else
              count++;
      }
   }

   function processFiles(dir) {
      list = getFileList(dir);
      for (i=0; i<list.length; i++) {
          if (endsWith(list[i], "/"))
              processFiles(""+dir+list[i]);
          else {
             showProgress(n++, count);
             path = dir+list[i];
             processFile(path);
          }
      }
  }

  function processFile(path) {
       if (endsWith(path, ".tif")) {
           open(path);
           //
           // CHANGE HERE: START
           //
           run("Subtract Background...", "rolling=50 white");
           save(path+"--processed.tif");
           //
           // CHANGE HERE: END
           //
           close();
      }
  }
```

Looks complicated, doesn't it?!

The really good news however is that the only part you really need to care about is the tiny bit between CHANGE HERE START and END. You can replace that part by anything that you recorded as a macro.

For example, let's try smoothing all the images:

...

Deep convolution

Since a few years "deep convolution" has revolutionized image segmentation. Mostly it is implemented as "Deep convolutional neural networks", see e.g., https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/.

However, to understand what deep convolution is all about one can also simply do it with classical operations in ImageJ. Let's see how!

- Open image "../deep-convolution/edge-of-dots/input01.tif" [File > Open]
- Open macro "../deep-convolution/edge-of-dots/egde-of-dots--deep-convolution.ijm"
  - [File > New > Script]
    - Script editor: [File > Open]
- Change the line after "// Load image" to match your path
- [Run] the macro