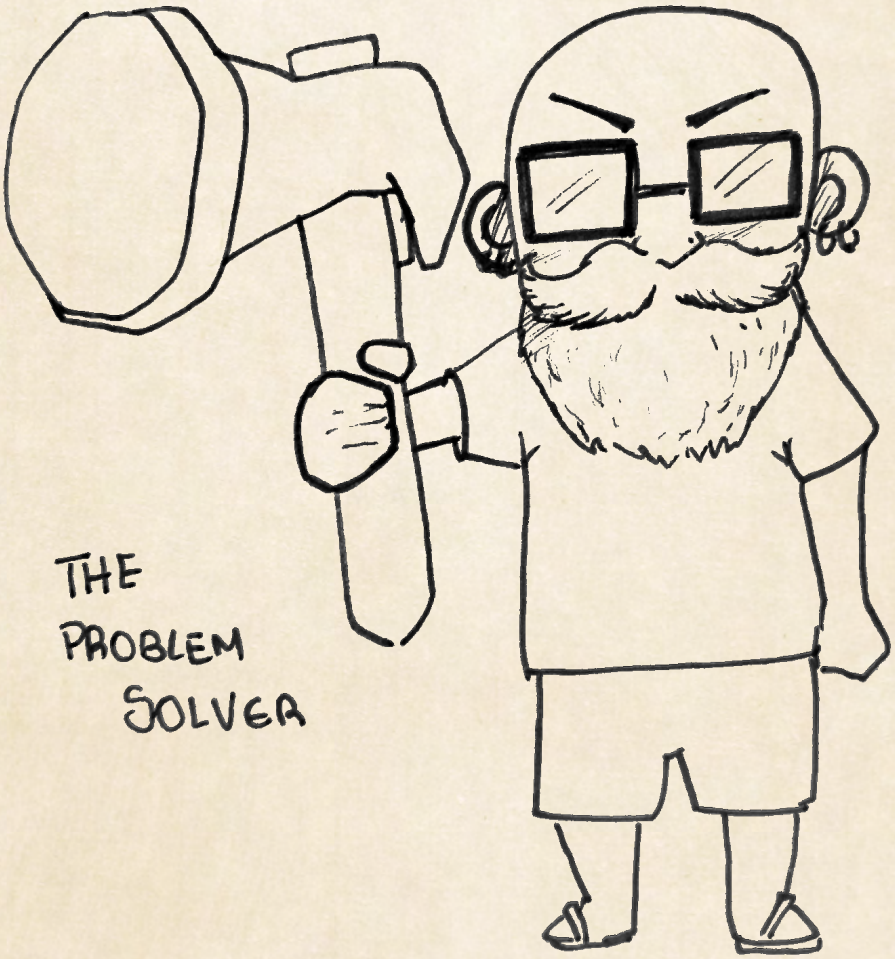
# EMBO 3D developmental imaging course

## Introduction to 3D visualisation and analysis using ImageJ




*Cartoon by Mariana Ferreira*

Fabrice P. Cordelières

July, 2022 - Oeiras

*Course material*
*available on GitHub*

# CONTENTS

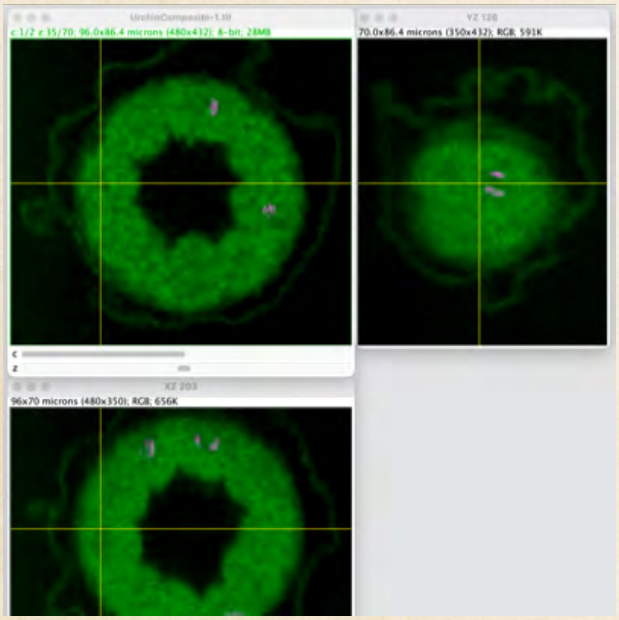
# Chapter 1: Visualize

When dealing with 3D datasets, the main difficulty is to display on a 2D medium a 3D information. Three strategies can be employed:

- Extracting part of the information
- Summarizing the full information into a 2D representation
- Generating a serie of representations that will trick our brain

## Extracting part of the information

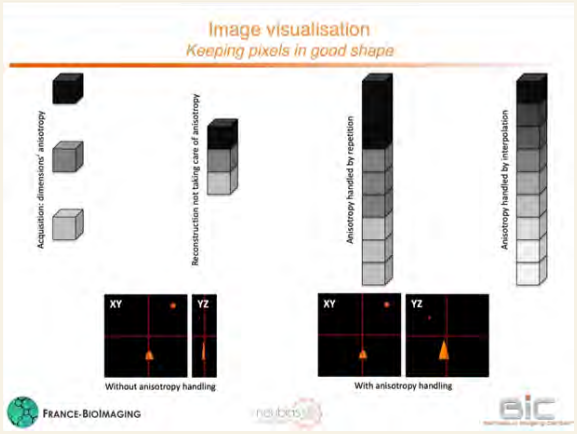
### Orthogonal views



First, let's have a little chat about vocabulary. When opening the 3D image with ImageJ, the data container is a stack (1 channel) or a hyperstack (several channels/timepoints). Both are made of several images called "slices". You can navigate the stack and duplicate an individual slice (`Image>Duplicate`) to extract "THE" image. You may also have a look at the your sample in depth, along either a vertical (XZ view) or horizontal axis (YZ view). This view can be generated using the `Image>Stacks>Orthogonal views`. To close the viewer, close one of the side views. **Warning:** this will cause both side views to be closed, so do not hesitate to duplicate the one(s) you're interested in !!!

*A word about dimensions' disparity*

As a regular optical microscope does not give isotropic resolution, the voxel (volume picture element) is not a cube but rather takes the shape of a "brick". If one wants to keep the global aspect ratio of its sample, this should be taken into account. Let's imagine a voxel size of 1x1x3µm in xyz respectively. When generating the XZ view, the pixel size is 1x3. To generate the scene, we use square pixels: keeping the aspect ratio requires the use 3 pixel in z for each pixel in x. How to deal with the data ? We only have information for 1 out of the 3 pixels ! We can replicate the intensity or interpolate the intensity between successive planes as depicted on the figure.

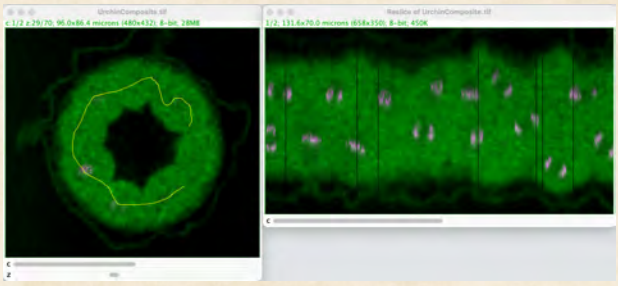



### Cutting through the volume, along "non conventionnal" directions

"How would you like your slice through the image ? Horizontal ? Vertical ? Random ?"

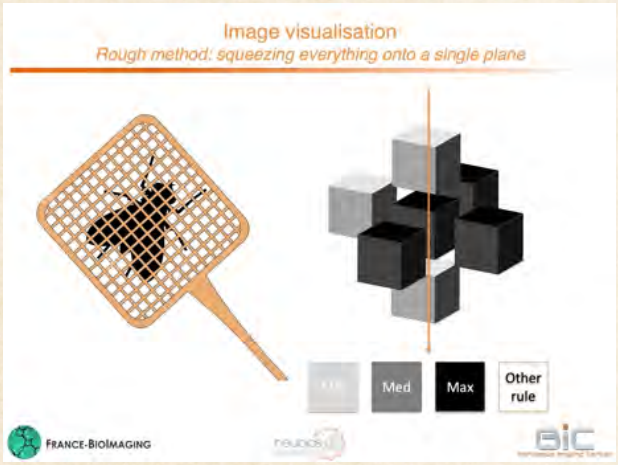
If you're looking for none of the first two, you should definitely have a look at the reslice option. This function might be found in the `Image>Stacks>Reslice` menu. Before jumping right into it, please have an image ready with a linear region of interest (ROI) set on it. If not, ImageJ will propose to generate parallele slices along the X or Y direction, starting at one of the image's borders.

In case you've taken your time and have drawn a region of interest onto the image before calling the function, ImageJ will litterally slice through the stack, along the path you've defined. This allows you to slice and unwrap the image along non linear patterns, as depicted here:



Please pay attention to the options: due to the disparity of dimensions, you should always consider enabling interpolation.

# Summarizing the full information as a 2D representation




Next option is building an image by extracting meaningful information out of the different z slices. First, a new image is created with a size that is the same as a single slice. Pixels will be filled with intensity that depends on the rule chosen by the user: one can use minimum, maximum, average etc. For each XY coordinate, all intensities along the z axis are collected. The rule is applied and the resulting intensity pushed onto the result image. Within ImageJ, this type of projection is generated using the `Image>Stacks>Z project` function.

# Generating a serie of representations to trick your brain


Using the former representation is like considering some observer looking from above the stack. As we do have a 3D stack, we could apply this principle while collecting views from all around the dataset. All we need is varying the angle along which we apply the selected rule. From ImageJ menus, use the `Image>Stacks>3D project` function. The names of the projections' types are a bit different as compared to `Z Project`: Brightest point stands for maximum intensity projection. You'll have to set the start angle for the "camera", the total rotation and steps between two generated view. Other parameters are meant for the "Nearest point" projection that takes into account the distance between objects in the stack and the viewer (to learn more, follow this link).

*Why is this tricking my brain ???*

This type of representation takes into account some psychological parameters. This is also the case in some paintings where the artists were able to get a sense of 3D on a 2D canvas: see the Golconda painting by Magritte and accompanying explanations.


## Generating anaglyphs

Using the 3D Project function, one can generate an anaglyph view of a 3D dataset. It basically relies on 2 principles: 1. Due to their positions, our eyes capture our world under two different angles. Roughly, a 15x15cm image placed at a reading distance, is observed under two orientations of +6° and -6°; 2. Using

proper coloring, an image can be masked by an adapted filter. This is what happens when using blue/red glasses and a blue/red coded image: the red image is masked by the red filter placed over the left eye: it will only access the blue image. What should we do then ???

1. Project the stack into two images 12° appart.
2. Overlay the images.
3. Adapt the colors to the glasses.

### Project the stack

As anaglyph will use 2 colors, this processing can't be done on a multi-channel stack: please extract the channel of interest using the `Image>Duplicate` menu before starting ! Use the 3D project function, with an increment of 12°, and a starting angle of 354° (360°-6°).

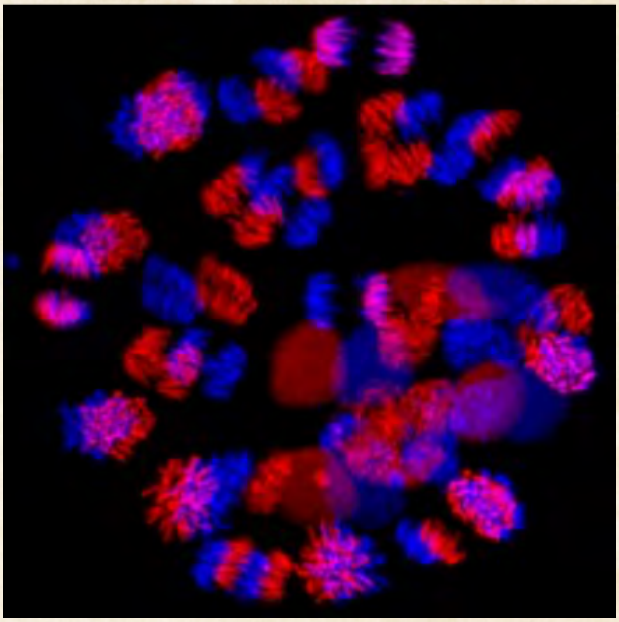
The output is a stack where the 2 images are slices, not channels: we have to switch the dimensions ! Use the `Image>Hyperstacks>Re-order hyperstack` menu to change slices into channels and channels into slices.

### Overlay the images

The two images are now defined as 2 channels. We have to change the Lookup Tables (LUT) associated to each to have the left image (first channel) colored in red and the second image in blue. Use the `Image>Color>Channels Tool` menu: choose the appropriate representation mode, Composite, (both channels are pseudo-colored and overlaid), activate the channel of interest and change its LUT using the `More menu`, picking the right LUT.

### Adapt the colors to the glasses

The result might not look great. You must make sure the displayed colors are adapted to your glasses. Please check by putting the glasses over the image, one eye at a time: only one image should be visible, the other one being filtered out. If not, activate the faulty channel and edit its LUT using `Image>Colors>Edit LUT` menu. Click on the upper-left pixel, extend the selection to all the little colored squares. Release the mouse-click: a color chooser pops-up, allowing you to define the start color (should be black). Once the box has been Oked, the end color can be selected: use the color mixer to select the one which will not be visible through your glasses.



**TADA ! Congratulations !!!**
**you've generated your first anaglyph**

# Chapter 2: Segment

Before quantifying, two operations should be performed:

- Segmentation: classify pixels into 2 categories, the background pixels and the objects' pixels. Several options are possible: partitioning the pixels based on their intensities (threshold), or using a combination of parameters (ex: "basic" machine learning).
- Connexity analysis: objects are defined as groups of connected pixels. This step will allow individualizing the objects by exploring the relationship between the objects' pixels.

## Segmentation

### Threshold

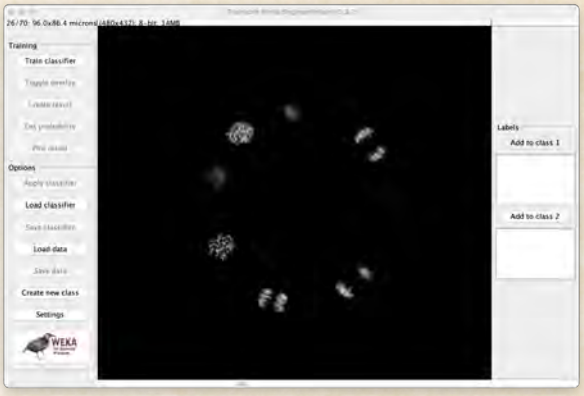
The most basic option will be using a threshold: all pixels carrying an intensity above the threshold value are considered as objects' pixels. The reference value might be either set manually or automatically. The automated methods rely on the image's histogram: a rule is applied to find the "ideal" value, depending on the histogram's shape. Want to know more ? Have a look at this link.

The threshold adjustment menu might be found here: `Image>Adjust>Threshold`. When looking to extract the overall intensity of an object, threshold might not be ideal: in case the threshold is too high, the objects will look as shrunk and therefore their intensities will be quantified over a smaller volume, leading to underestimation. Generally speaking, segmenting an object by simple thresholding and quantifying them on that really same channel might not be suitable. . .

### Random trees forest

**WARNING:** the explanations provided here may not be rigorous. My intention is to give you an idea about what is going on under the hood.

When looking at the image, intensity alone might not be a unique way to partition the background and objects' pixels. There are some textures that differentiate objects from background. Objects have outlines that we may identify. It's by combining all that information that we are able to delineate the structures of interest. We could use some algorithm to perform a similar approach. We will use the Weka plugin: `Plugins>Segmentation>Weka segmentation`. Two versions exist, depending if the processing should be performed in 2D or 3D. During the workshop, we have been using the 2D version, only for the sake of time.

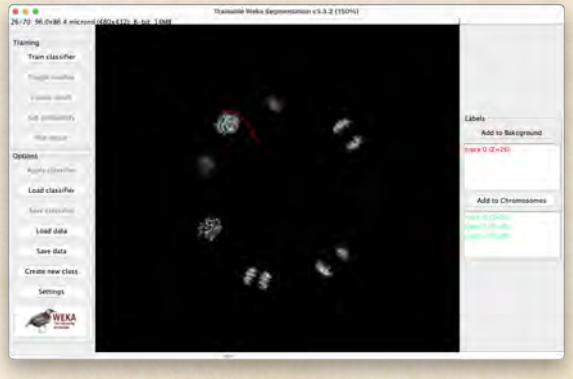



*The 2D Weka plugin's interface*

### Setting the parameters


In the interface, click on the `Settings` button: you'll have access to numerous parameters, including the type of filters to be used when building the forest (top part), the extent of the filters (minimum/maximum width) and additionnal settings about the classes. Here we will be classifying pixels into two classes: background and chromosomes. Please, rename the classes in the relevant text boxes.

### Annotating the data



Annotation is a cruicial step, where the user feeds the algorithm with sample pixels from the different categories. Using the freehand selection tool, highlight pixels from the background category and press the "background class" to add the selection. NB: to remove a selection, simply double click on the relevant line within the class' box.

### Building the random trees forest: training

Press the `Train Classifier` button on the upper left part of the interface. This step will take time during which the input image is first duplicated many times. Each duplicate will be subjected to a different filter (i.e. all filters selected within the settings' window will be applied, over all neighbourhood the user did set).


Image from Fiji website.

Decision "trees" are constructed. A tree is a serie of decisions the computer makes to get to a classification. Each branching point corresponds to a specific filtered image from where the pixel value is used to go one way or the other. The trees are build randomly, chaining decisions. Many trees are built. The classification made by each tree is confronted to the user-defined annotations. If a tree passed the test (meaning: gave the right answer), it is kept. If not, the tree is replaced by a new one.

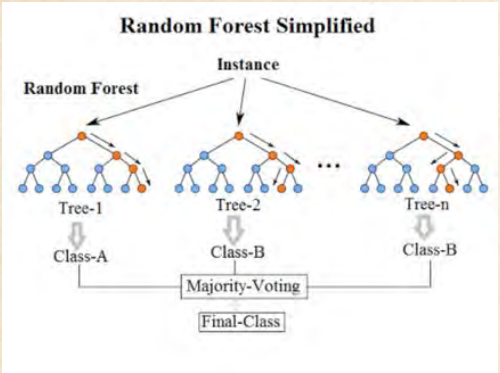
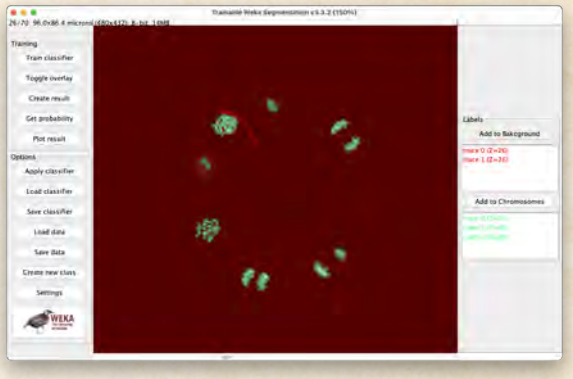



Image from Wikipedia.

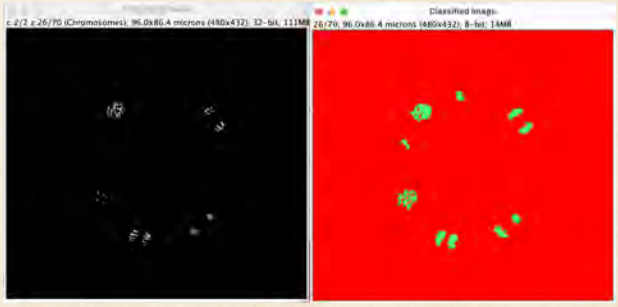
### Applying the random trees forest: classifying

Our forest is ready to classify our image ! Each pixel will be run through each tree. Each tree will make a decision. As for regular elections, the percentage of the votants for each candidate (i.e class) are determined: this will allow generating what is called the probability maps (one map per class). Anoter output can be generated: the segmentation result. Basically, the candidates with the most votes wins ! On this output image, all pixels belonging to the first category carry an intensity of 1, the ones of the second category an intensity of 2 etc.



### Generating output, re-using a classifier

To generate the probability map, press the `Get Probabilities` button; to get the segmented map, press the `Create` button.



Example output. Left: probability map, right: segmentation.

Before closing the Weka window, do not forget to save your classifier: there is a button for that ! You may reuse the same forest without having to annotate nor train the system.

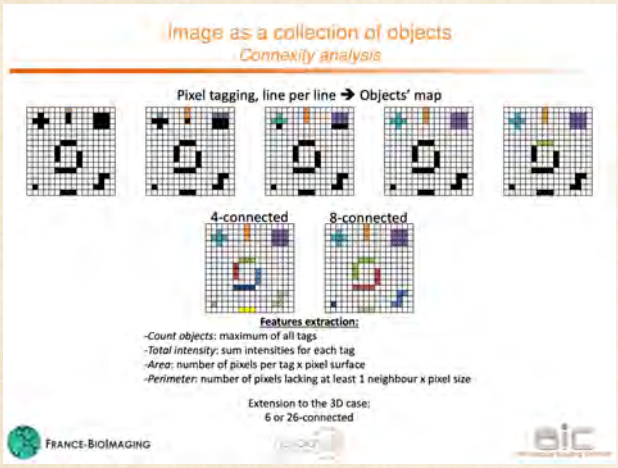
**NB:** Other plugins exist with similar purpose. LabKit, for instance, is a good alternative. It also allows easily annotating images, and is a tool of choice when it comes to preparing data for training machine learning workflows.

## Connexity analysis

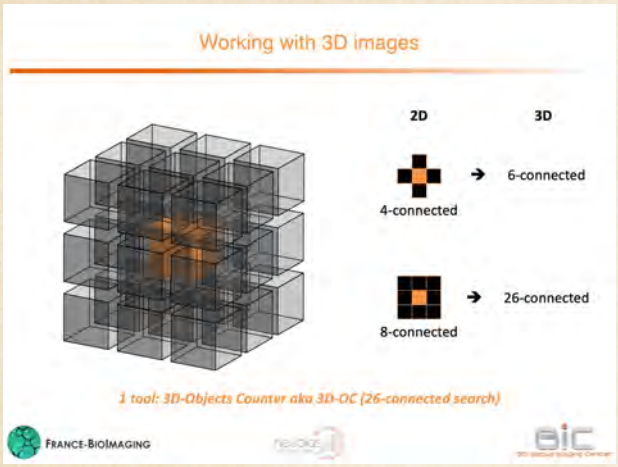
We now have partitionned the pixels into several classes. A simple threshold will be sufficient to extract the objects' pixels. A last step should be performed in order to group connected pixels into individual objects: the connexity analysis.

In short, the image will be reviewed from the upper-left corner to the lower-right corner. For each objects' pixel, the surrounding pixels (either 4 neighbours: the ones sharing a side within that pixel; or 6 neighbours: sides+diagonals) will be checked: does one of

them carries a tag ? No ? Ok, let's give that pixel a tag ! Here you go: you get the "1" as you are the first one being tagged. Now, jumping to the next pixel: that one is connected to the previous one that is tagged as belonging to the "1" family. Both are part of the same object and will therefore share the "1" tag. The stories goes on: each time an object pixel is found that has no tag and no tagged neighbour, a new tag is generated.




This is what happens when one uses `Analyze> Analyze Particles`. However, this internal function of ImageJ only works in 2D. To perform connexity analysis in 3D, the same algorithm should be applied, taking into account possible connections along the z axis. Taking into account all possible ways to connect voxels in 3D, the algorithm has to investigate 26 neibourhing voxels !



# Chapter 3: Quantify

Many plugins exist to perform 3D connexity analysis and 3D quantifications:

- **3D Objects Counter (aka 3D-OC)**, that not only counts, but also quantifies
- **3D ImageJ Suite**, containing a 3D ROI Manager and options to split/fuse objects
- **MorphoLibJ**, a complete toolbox for mathematical morphology

**NB1:** As the author of this short note wrote the 3D-OC, he chose the simplicity of promoting his own plugin over the other ones. The reader is however invited to have a look at the other plugins that either perform better on large datasets an/ord provide much more advanced features !

**NB2:** As the author of this short note is really lazy, he just copied-pasted the content of the document associated to the 3D-OC.

## How to use 3D-OC ?

### Setting the options

1. Prior to analysis, the parameters to calculate as well as some display parameters should be set. To do so, launch ***3D-OC Set Measurements***, which may be found in the `Analyze>3D-OC Set Measurements` menu. The following window appears:

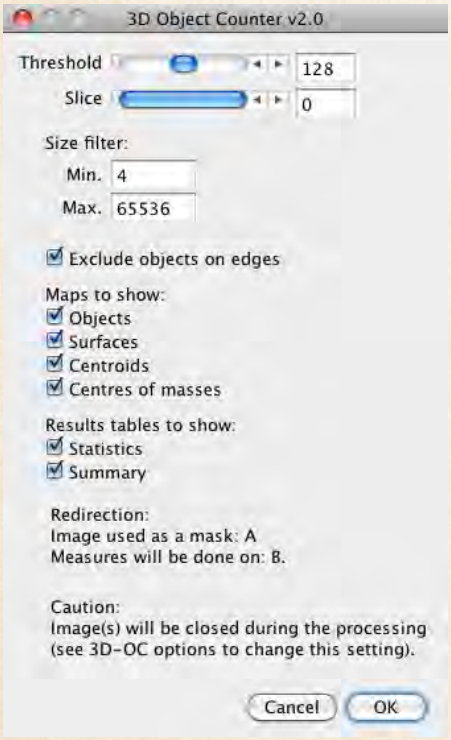

The following options are available:

- ***Parameters to calculate***: tick the parameters that should be calculated for each found object (see chapter **??** for details).
- ***Image parameters***:
  - *Close original images while processing (saves memory)*: when this option is ticked, the original image is closed at analysis startup.
  - *Show masked image (redirection required)*: if a redirection has been set (see below), ticking this box will enable the generation of a new image where only objects' pixels from the destination of redirection will be drawn. Background pixels will appear as black.
- ***Map's parameters***:
  - *Dots size*: size, in pixels, of the dots placed at the objects' centers.
  - *Font size*: size, in points, of the the object's labels, placed at the objects' centers.
  - *Show numbers*: tick this box to overlay the objects' numbers on the results images.
  - *White numbers*: tick this box to draw labels in white. If unticked, the labels will be drawn using the objects' tag value.
- ***ResultsTable parameters***:
  - *Store results within a table named after the image (macro friendly)*: the results are logged to a results table named "Results", unless this box is ticked: the table's name is then "Statistics for "+the image's title. This option makes it easier within a macro to select the corresponding results window for further processing.
- ***Redirect to***: this option allows to do the segmentation on the current image while intensity related measures will be calculated from the image chosen using the drop-down list.

2. Validate the choices by clicking on Ok: the options will be saved in the IJ_Prefs.txt file, to be used later during the analysis.

### Launching the analysis

1. Once options have been set, launch ***3D Object Counter***, which may be found in the `Analyze>3D Object Counter...` menu. Depending on the options previously set, the window that appears might take one of the two forms depicted on the figure:




- ***Threshold***: use this slider to define the limit intensity value separating the voxels in 2 populations: background voxels (intensities below the selected value) and objects' voxels.
- ***Slice***: when the plugin's interface is launched, this slider allows the user to navigate between slices.
- ***Size filter***: the two following fields will help to exclude from analysis objects which size is out of the defined range.
  - *Min.*: minimum size an object should have to get its statistics retrieved (expressed as a number of voxels).
  - *Max.*: maximum size an object should have to get its statistics retrieved (expressed as a number of voxels).
- ***Exclude objects on edges***: tick this box to exclude from analysis any object that touches at least one of the picture's edges.
- ***Maps to show***: allows to choose the kind of image outputs that should be generated. On those images, the intensity of all pixels constitutive of an object carry the same intensity: its tag. All statistics corresponding to this object will be presented on the results table on the $n^{th}$ line, n being the value of the tag. Only object which sizes' fall into the size filter are displayed. If "Exclude on edges" is ticked, the objects touching one of stack's edges will be discarded.
  - *Objects*: tick this box to create a stack where all retrieved objects are displayed.
  - *Surfaces*: tick this box to create a stack where all surface voxels (i.e. each voxel lacking at least one of its 26 neighbors) from all retrieved objects are displayed.
  - *Centroids*: tick this box to create a stack where all the geometrical centers from all retrieved objects are displayed.
  - *Centers of masses*: tick this box to create a stack where all the centers of masses from all retrieved objects are displayed.
- ***Results table to show***:
  - *Statistics*: when this box is ticked, a results table will display all statistics of the found objects. Its name will depend on the options set under the "3D-OC Set Measurements window" (see section **??**).
  - *Summary*: if ticked, a summary will be sent to the "Log" window, containing the image's name, the number of objects found, the size filter parameters and the threshold used for analysis.
- ***"Redirection" section***: in case the redirection option is on, this section recalls the image used as a mask and the image used for intensities related measures.
- ***"Caution" section***: This section acts as a reminder as it only appears if the "Close original images while processing" option has been ticked within the "3D-OC Set Measurements window" (see section **??**).

2. Validate the choices by clicking on Ok: the analysis starts, images are generated and results logged.

## Which parameters are retrieved for each object ?

- ***Number of object's voxels***: determines the number of voxels (volume picture element) constitutive of the object.
- ***Volume***: number of voxels of the object × x calibration × y calibration × z calibration. If no calibration has been attributed to the image, returns the number of object's voxels.
- ***Number of surface voxels***: determines the number of voxels lacking at least one of its 26 neighbors (i.e. in contact with the background).
- ***Surface***: determines the surface of the object, taking into account the differences of dimensions between the xy, xz and yz sides of the surface voxels.
- ***Integrated density***: sum of the intensities of all the current object's voxels.
- ***Mean of the gray values***: average of the intensities of all the current object's voxels.
- ***Standard deviation of the gray values***: standard deviation of the intensities of all the current object's voxels.
- ***Minimum gray value***: minimum intensity found within the current object.
- ***Maximum gray value***: maximum intensity found within the current object.

- ***Median of the gray values***: median of the intensities of all the current object's voxels.
- ***Mean distance to surface***: average distance from the geometric centre of the current object to the surface's pixels.
- ***Standard deviation of the distance to surface***: standard deviation of distance from the geometric centre of the current object to the surface's pixels.
- ***Median distance to surface***: median distance from the geometric centre of the current object to the surface's pixels.
- ***Centroid***: tick this box to log the 3D coordinates of the geometrical centre for each object.
- ***Centre of mass***: tick this box to log the 3D coordinates of the centre of mass for each object (each set of coordinates constitutive of an object is pondered by its intensity).
- ***Bounding box***: smallest box encompassing the object: if this box is ticked, the coordinates of the upper left corner of the box, its width, height and depth are retrieved and logged.

# Zee end


*Cartoon by Mariana Ferreira*

Thanks a lot to all of you for this fantastic course !