HOW TO USE ZOLA

Benoit Lelandais[1,2] (benoit.lelandais@pasteur.fr)

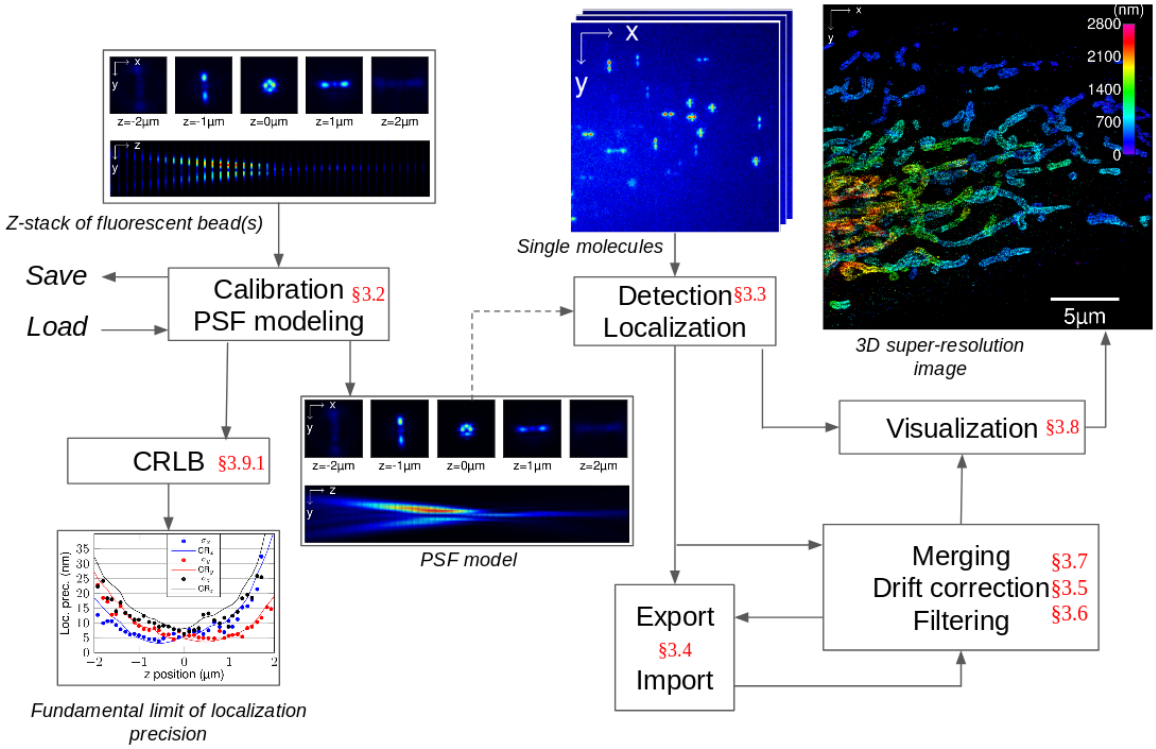
[1] Institut Pasteur, Unité Imagerie et Modélisation, 25 rue du Docteur Roux, 75015 Paris, France

[2] 2UMR 3691, CNRS; C3BI, USR 3756, IP CNRS

Contents

1 Introduction

Thank you for downloading ZOLA. ZOLA is an ImageJ/Fiji plugin that allows you to determine the PSF model corresponding to your optical system, and use this model to reconstruct 3D single molecule localization microscopy (SMLM) images. ZOLA also provides tools to correct drifts, merge consecutive localizations or render images. The following scheme presents an overview of methods featured ZOLA:



2 Requirement

2.1 GPU

Our plugin requires the 64 bit version of ImageJ or Fiji software available here: https://imagej.net/Fiji/Downloads. The calibration, localization or drift correction tools are highly computationally expensive. To increase efficiency, the computations are performed on GPU (Graphical Process Unit). Currently, these functions can be launched on Cuda compatible NVIDIA GPU exclusively.
Please, go to http://www.nvidia.fr/page/home.html and http://docs.nvidia.com/cuda/index.html to install your `NVIDIA GPU` and `Cuda-toolkit`.

2.2 Camera photon count

Zola's algorithm requires camera pixel values to be in photon counts. If this is not the case, we provide an additional tool to convert data to photon counts (see section 3.10.2).

3 ZOLA plugin

3.1 ZOLA plugin installation

Plugin installation requires 30 minutes. Follow these instructions to install the plugin:

1. Extract the *ZOLA_package.zip* file. This folder contains our *ZOLA.jar* plugin, some libraries, and sample data to test the plugin.
2. Move *ZOLA_.jar* to your *ImageJ/plugin* folder.
3. Open *ZOLA_package/JCuda libraries* folder and open the subfolder corresponding to your operating system and your Cuda version. We provide files for 7.5 and 8.0 Cuda versions. If your Cuda version is different, the libraries can be found here: http://www.jcuda.org/downloads/downloads.html. To check your Cuda version, tape `nvidia --version` in a terminal window.
4. Paste the *lib* folder into your *ImageJ* folder.
5. Copy and paste the content of the *jars* folder into your *ImageJ/jars* folder (or *ImageJ/plugin/jars* folder depending on your version of ImageJ).

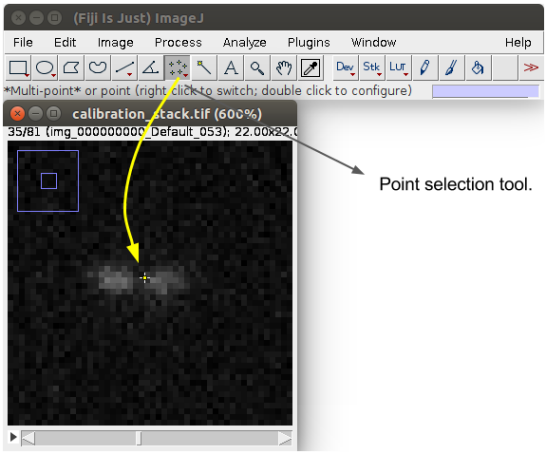
The plugin is now installed. After restarting ImageJ, you should be able to see it in the `ImageJ→plugin` menu. We provide data to test the plugin in *ZOLA_package/TEST_DATA* folder. It contains a stack of images of a fluorescent bead at different Z positions (to be used for calibration), and a stack of 10000 frames of single molecules to localize (extracted from a stack of 87959 frames).

3.2 Source code

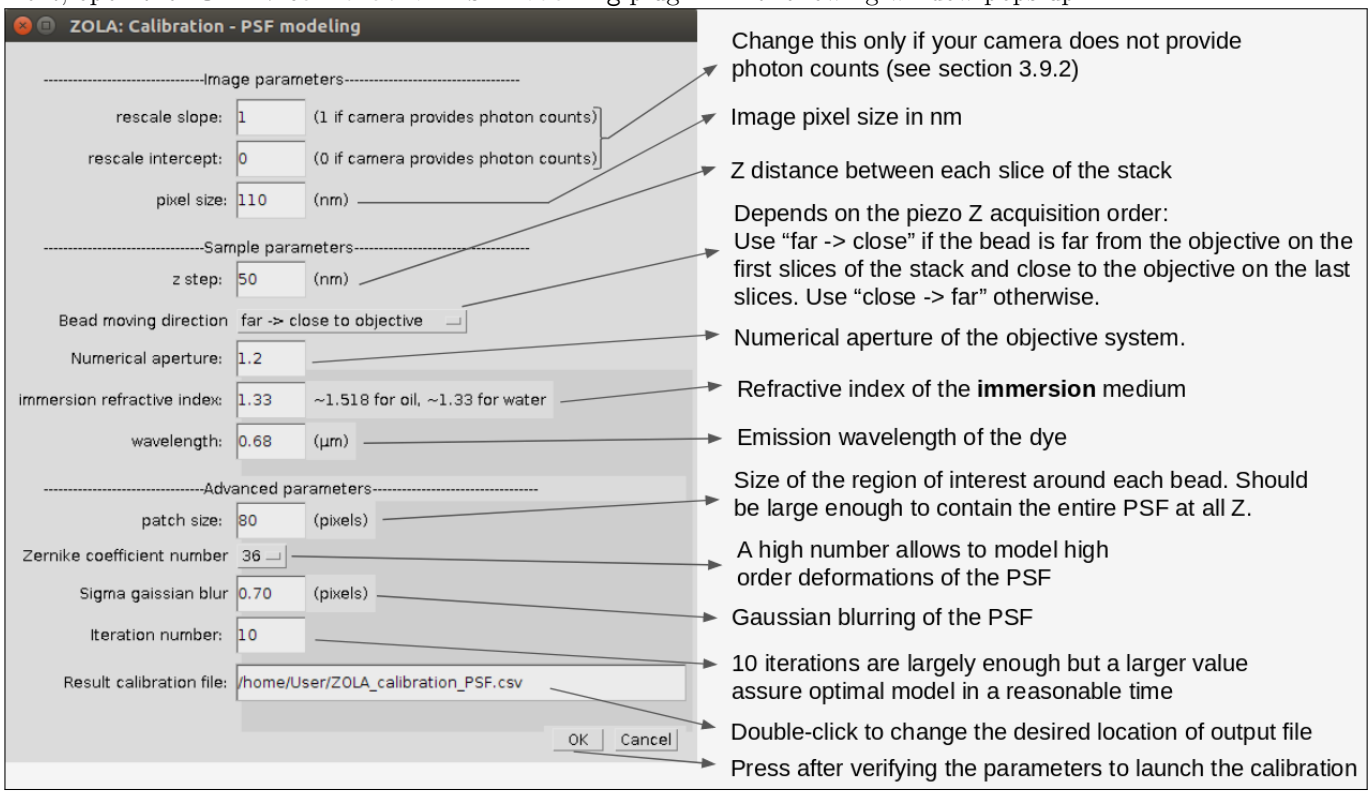
The source code of ZOLA is also available in *zola_source_code* folder. Please, check the *zola_source_code/README.md* file to compile it.

3.3 Calibration - PSF modeling

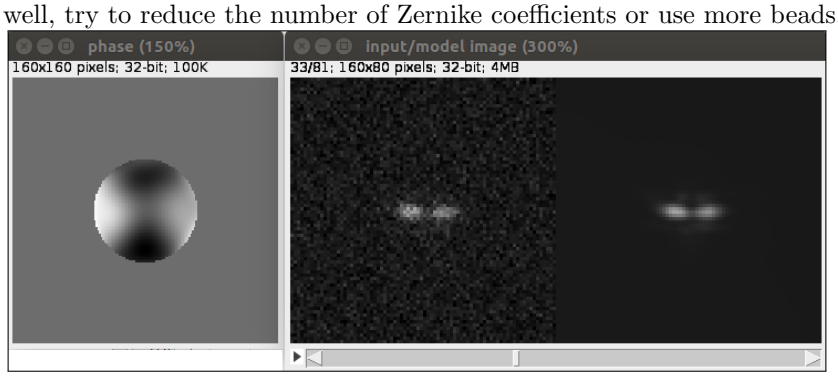
ZOLA's calibration tool allows to model the PSF from a stack of images of fluorescent beads at different Z positions. The algorithm assumes that there is no drift during the acquisition. Thus, it is recommended to acquire the stack of images fast with stable temperature in the room. Usually, the axial distance between slices is set to 50 nm and the acquisition time per slice is 50 ms. Thus, the acquisition time for a 2 µm Z range is 2 seconds.
First, open a stack of images and select one or multiple beads using the ImageJ Point selection tool:



A single bead is usually enough to accuratly model the PSF. If your image contains overlapping beads or has a low signal to noise ratio, it is still possible to model the PSF by selecting multiple beads.
Next, open the `ZOLA` → `Calibration PSF modeling` plugin. The following window pops up:



After verifying the parameters, then press `OK`. Phase retrieval begings. This can take few minutes. At the end of the phase retrieval, the plugin shows the phase and the model of the PSF as shown below. If the modeled PSF does not match the bead image well, try to reduce the number of Zernike coefficients or use more beads.

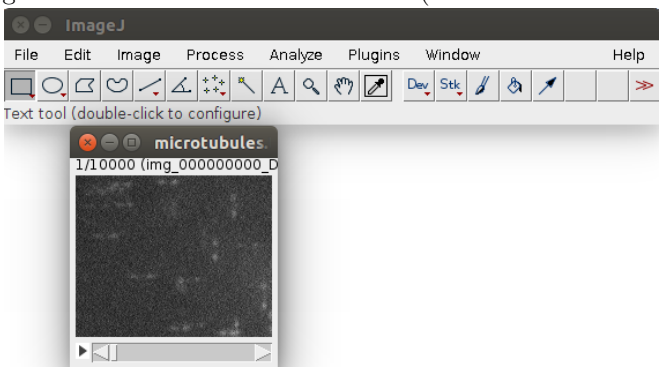


Retrieved phase Bead image PSF model result
(Use the slide to change Z)

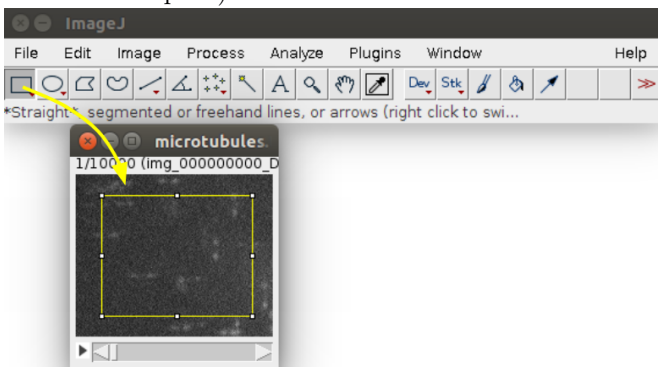
3.4 Single molecule Localization

ZOLA's localization plugin allows to detect and precisely localize single molecules in a sequence of difraction limited images as obtained in SMLM experiments. The process is the following:

First, open the sequence of images. If desired, a region of interest (ROI) can be drawn to reduce the size of the region to reconstruct as shown below (this increases the reconstruction speed).

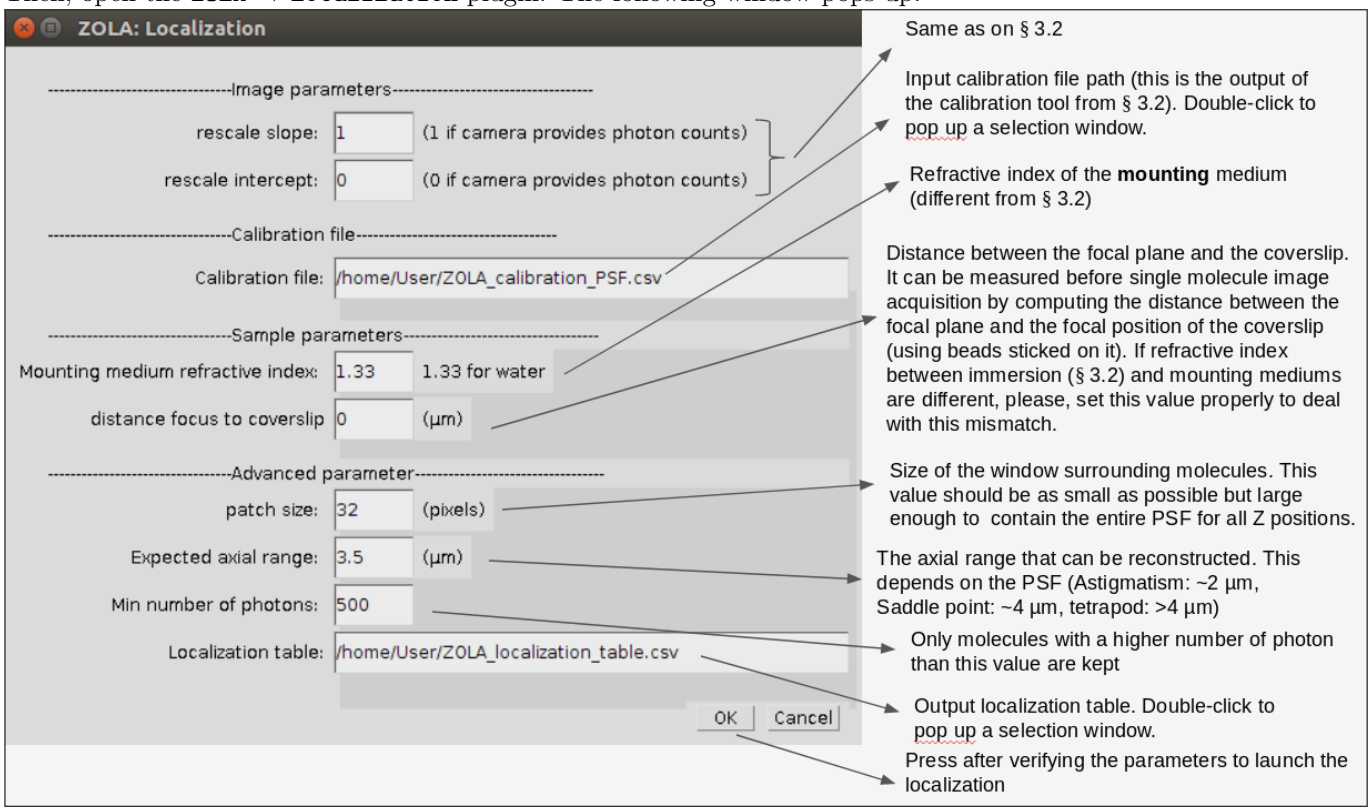


Full sequence of images

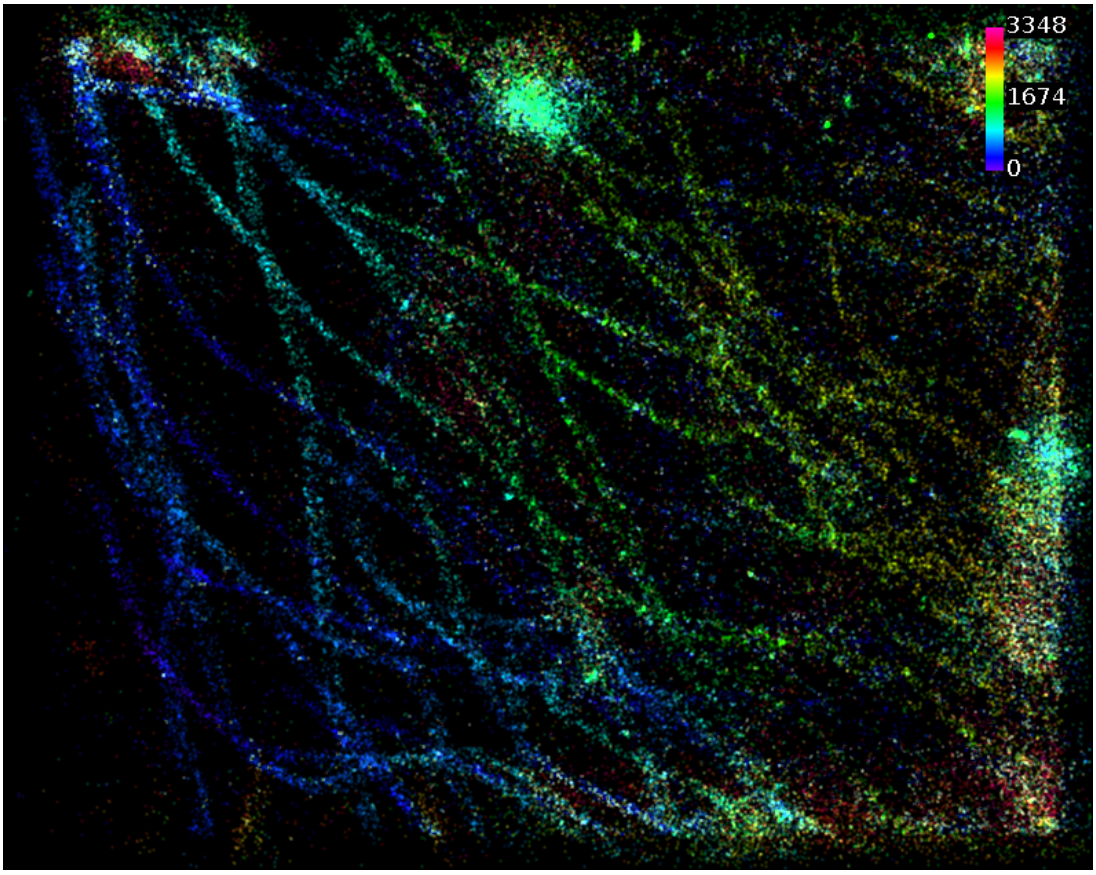


Sequence of images with ROI selection

Then, open the `ZOLA` → `Localization` plugin. The following window pops up:



The localization starts after clicking on `OK`. On test_data we provide, localization requires around 10 minutes. During localization, a colored super-resolution image (whose color encodes Z position) is updated every 30 seconds. The following figure shows the result obtained at the end of the localization. Other visualizations can be obtained using the ZOLA visualization tool (see section 3.9). Note that the quality of this image can be improved applying drift correction or filtering (see sections 3.6 and 3.7).


3348
1674
0

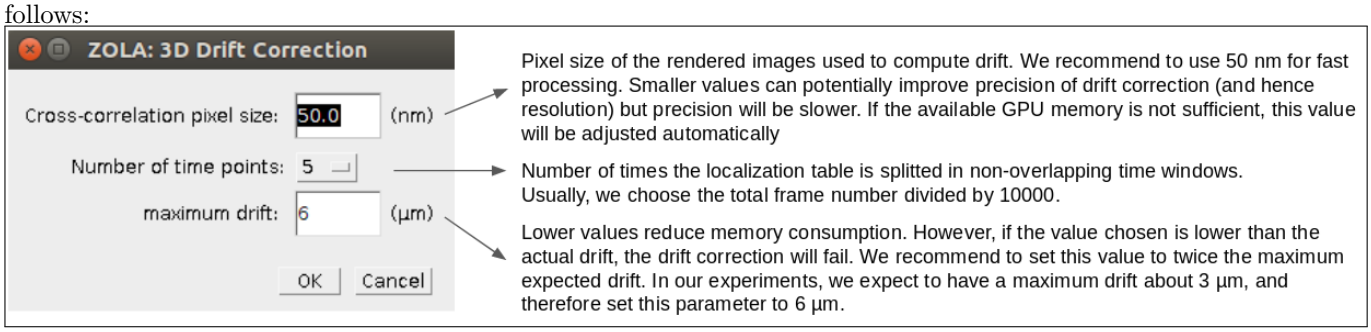
3.5 Import/Export

ZOLA allows to load a localization table in memory, or save it on disk.

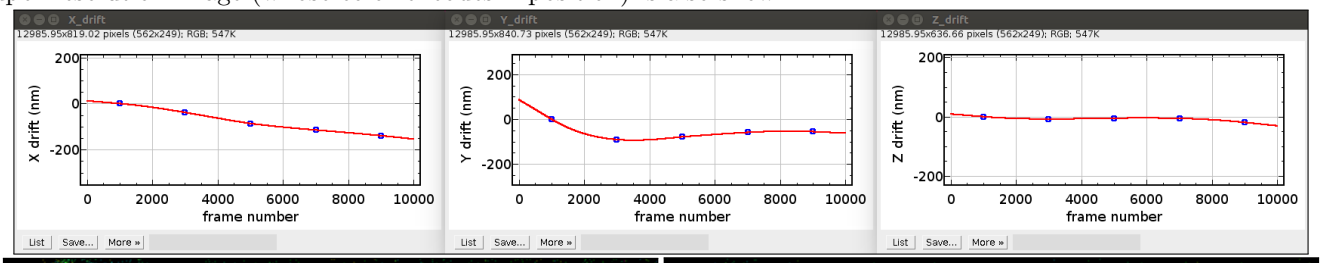
- Import: load a localization table.
- Append: Append another localization table to the existing table (useful when consecutive image sequences were processed separatly).
- Export: Save a localization table (useful to save data after post-processing, *e.g.* after drift correction, filtering or merging (see sections 3.6, 3.7 and 3.8).

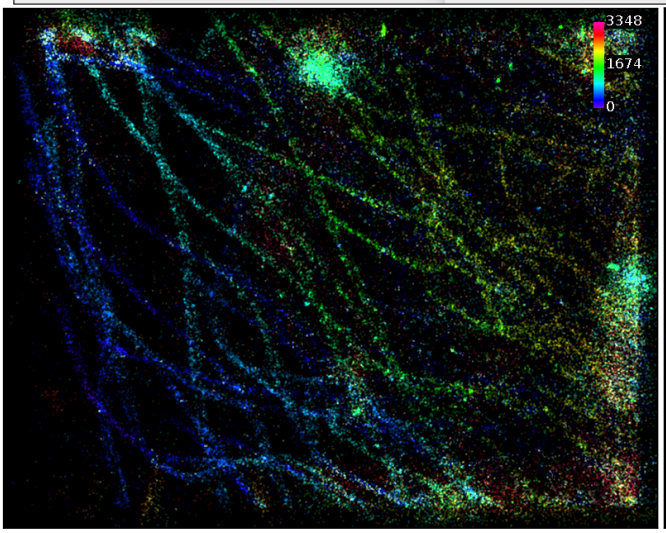
3.6 Drift correction

ZOLA features a 3D drift correction method based on redundant cross correlation. The drift correction window is as follows:

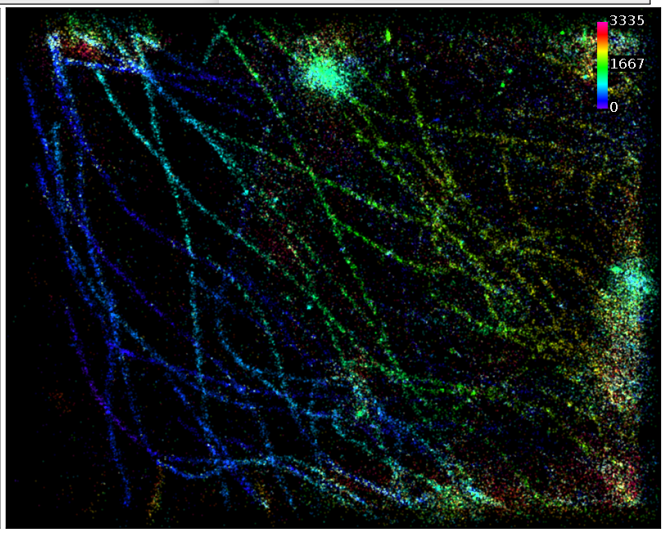


As output, the drift correction plugin shows three curves corresponding to X, Y and Z drifts. A drift-corrected super-resolution image (whose color encodes Z position) is also shown:





Super-resolution image before drift correction

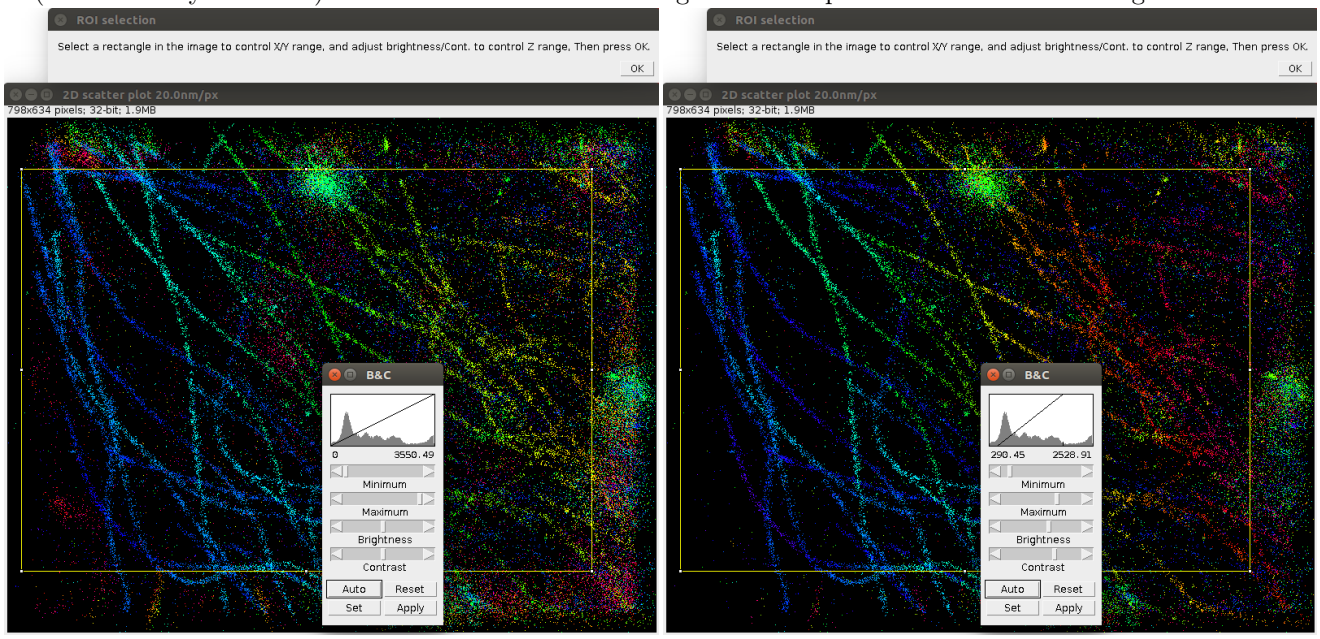


Drift-corrected super-resolution image

After applying drift correction, use `ZOLA→Export` plugin to save the localization table (see section 3.5).

3.7 Filtering

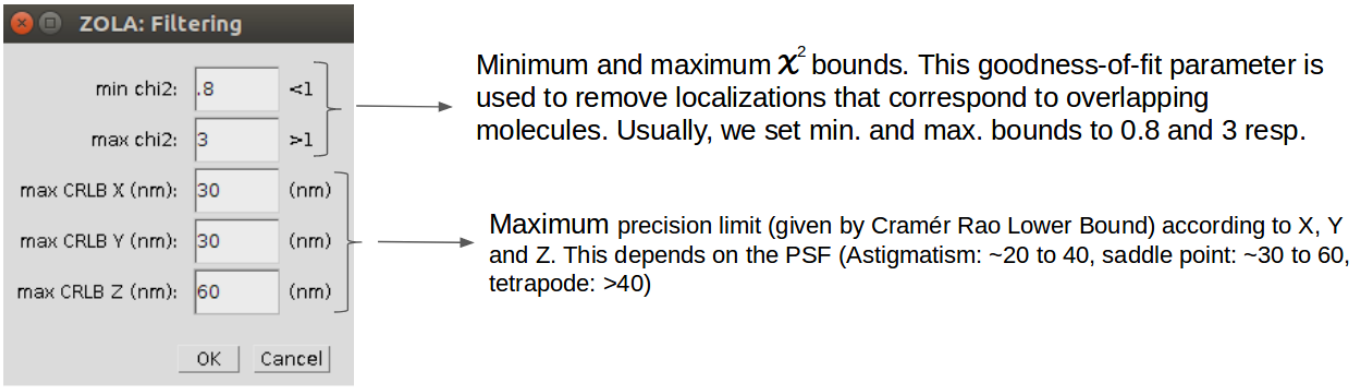
Filtering is useful to define a sub region of a 3D image, or to remove localizations with low precision or unsatisfying goodness-of-fit. First, the user selects a region of interest in the image (optional), and can use `Brightness&Contrast` tool (automatically launched) to select the desired axial range. An example is shown on the next figure:



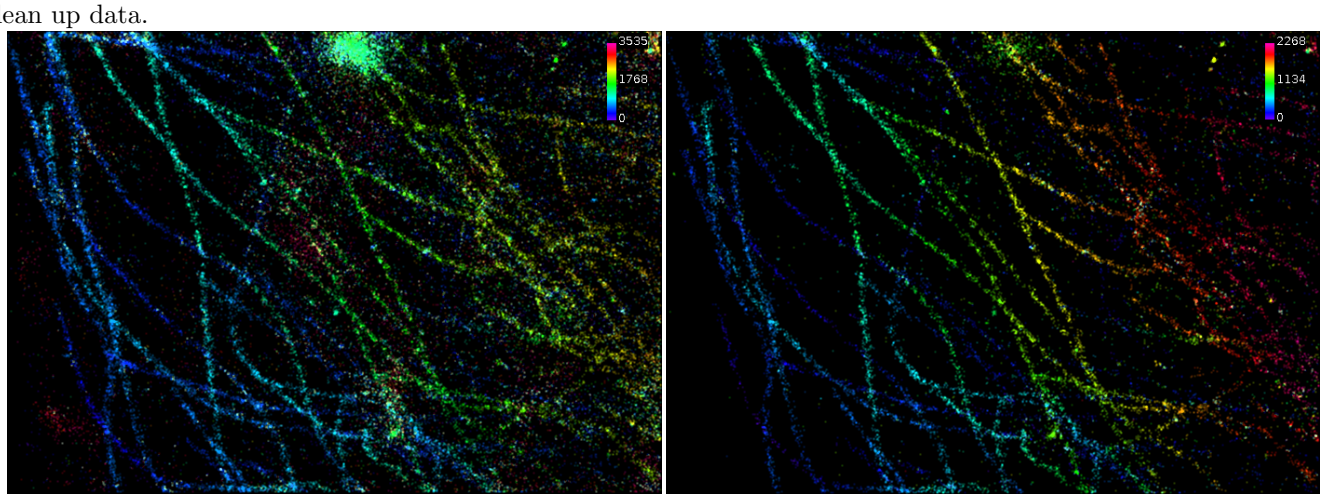
Full axial range: 3.5 µm

Clipped axial range: 2.3 µm

Then, after pressing `OK`, the following panel is opened allowing to filter localizations according to their good goodness-of-fit and their precision.



After pressing `OK`, the super-resolution resulting image pops up as shown below. As we can see, filtering allows to clean up data.



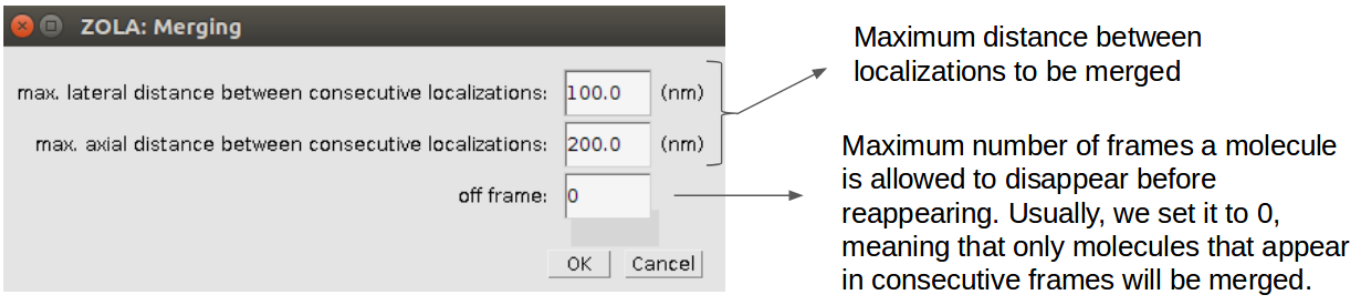
Super-resolution image before filtering

Super-resolution image after filtering

After applying merging, use `ZOLA→Export` plugin to save the localization table (see section 3.5).

3.8 Merging

This plugin is useful to merge localizations that are in close proximity in consecutive frames and are likely to originate from the same molecule. Open `ZOLA→Merging` and the following window pops up:



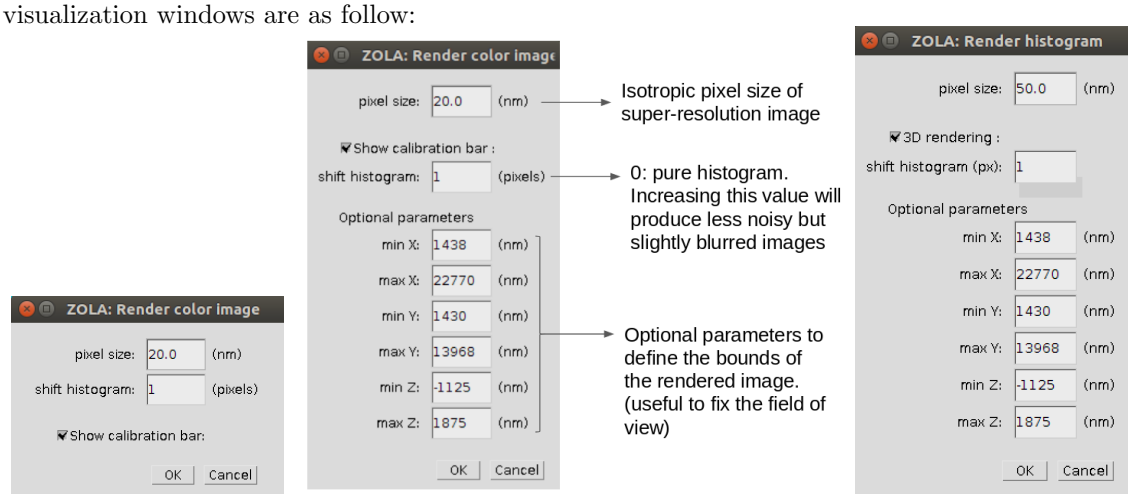
Then, click `OK` to launch the merging. As output, the plugin gives the number of localization that are merged and show the resulting super-resolution image.
After applying merging, use `ZOLA→Export` plugin to save the localization table (see section 3.5).

3.9 Visualization

ZOLA visualization plugin allows to render super-resolution images. The visualization can be launched during or after localization or after importing a localization table. Three rendering methods are proposed based on histograms:

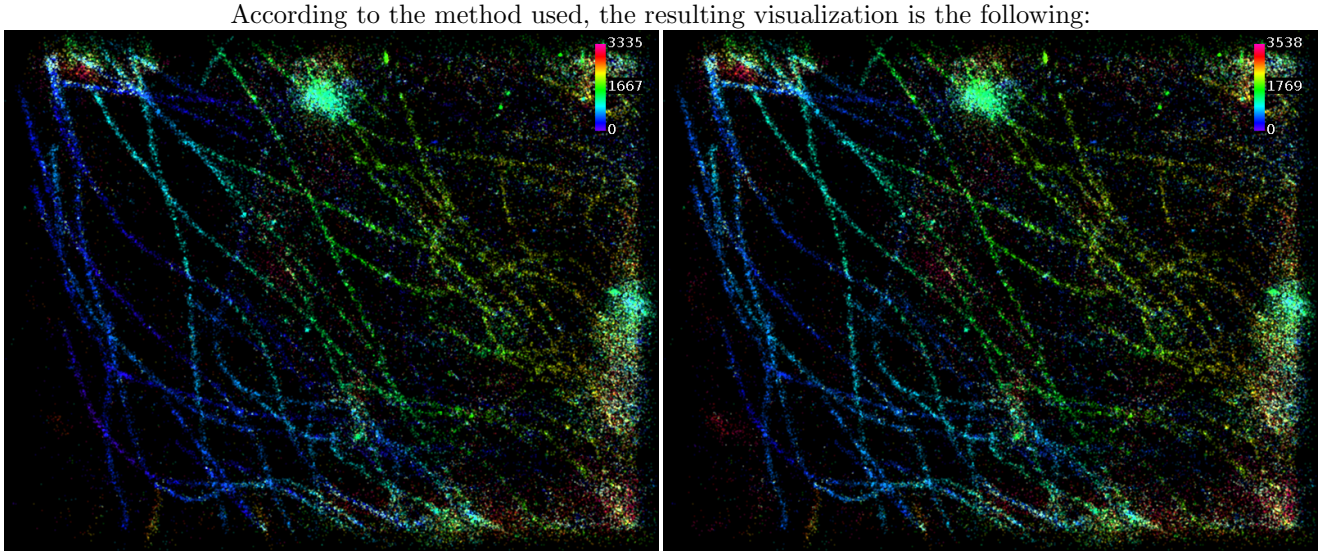
- Automatic 2D color histogram: Renders a 2D color histogram adjusting automatically the axial range and localizations that have a good precision (This method is launched automatically during localization and after import, drift correction, filtering and merging functions).
- 2D color histogram: Renders a 2D color histogram.
- 2D/3D histogram: Render a 2D or 3D histogram.

The three visualization windows are as follow:

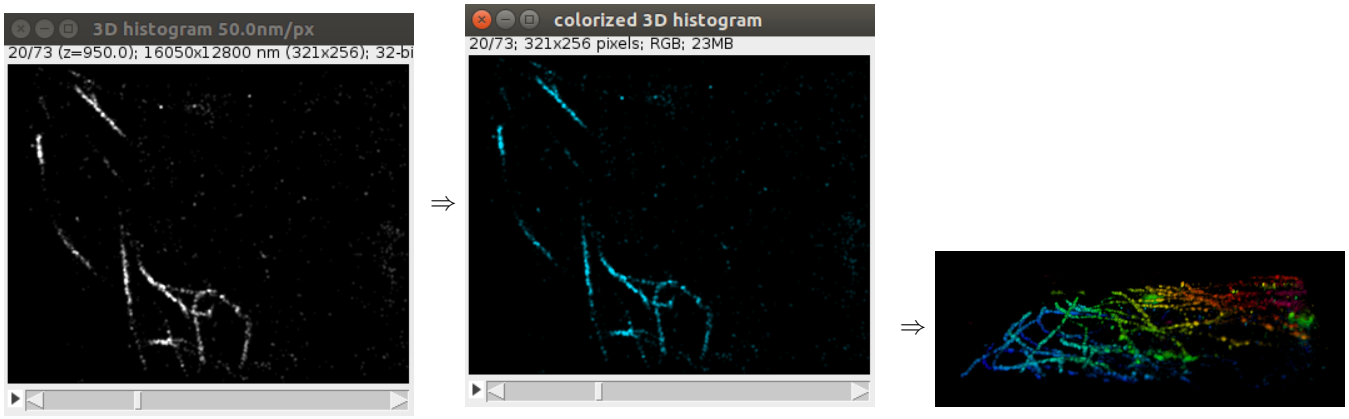


automatic color histogram Color histogram 2D/3D Histogram

According to the method used, the resulting visualization is the following:



automatic color histogram (clipped axial range) Color histogram (full axial range)

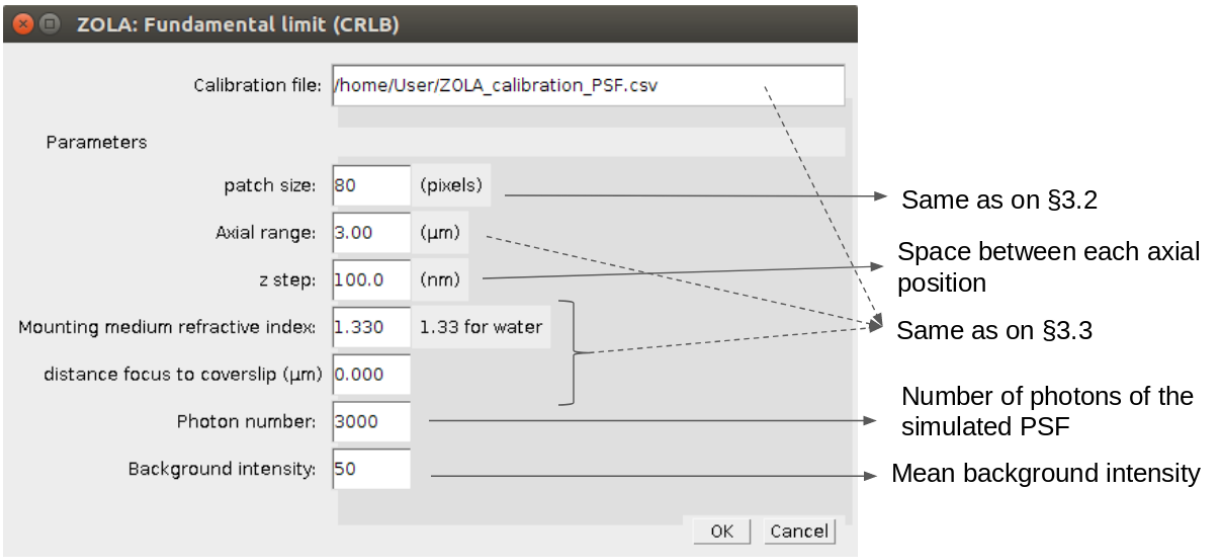


3D histogram (Z stack) | Colorized 3D histogram (Z stack) `ZOLA→additional tool→colorize opened 3D stack` | 3D visualization `3D viewer` Fiji plugin

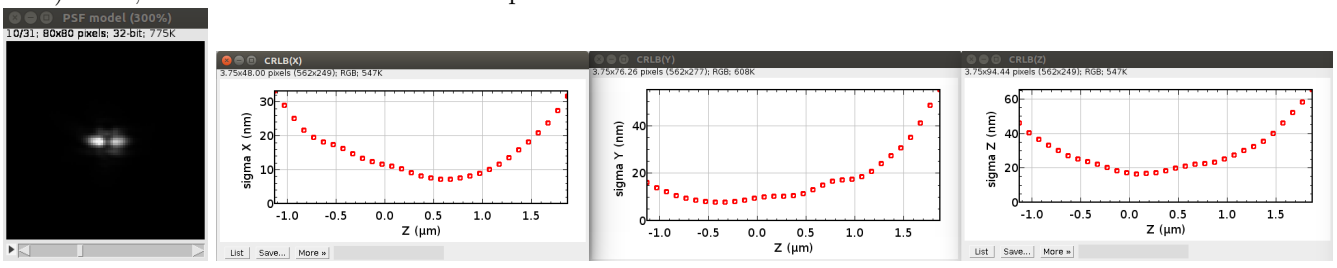
3.10 Additional tools

3.10.1 Theoretical localization precision limit: CRLB (Cramér Rao Lower Bound)

Zola's CRLB plugin allows to compute the fundamental localization precision limit for a given PSF and an expected molecule number of photon and background The CRLB of X, Y and Z is calculated as a function of axial position. Open the `ZOLA` → `CRLB` plugin. The following window pops up:



As a result, the tool shows the PSF and 3 plots showing the precision limit (the standard deviation of localization errors) for X, Y and Z as a function of axial position Z:



3.10.2 Photon conversion

ZOLA requires images expressed in photon counts. We propose here a simple conversion if it is not the case.
Do the following to check your data are in photon counts or convert them to photon counts:

1. First, acquire a sequence of 100 images of the same content (*e.g.* images of beads at a unique Z position). All images should look the same except for noise.
2. Open the stack of 100 images in ImageJ.
3. Select `Plugin` → `ZOLA`→ `additional tools`→ `Photon conversion`.

4. The tool computes two parameters for future photon conversion: `Rescale slope` and `rescale intercept`. Please, write down these values, as you will need them for calibration and localization.

If `Rescale slope`≈ 1 and `Rescale intercept`≈ 0, your data are already in photon counts.
`Rescale slope` and `rescale intercept` depend on camera settings. If the `gain` changes, you have to repeat this photon conversion.