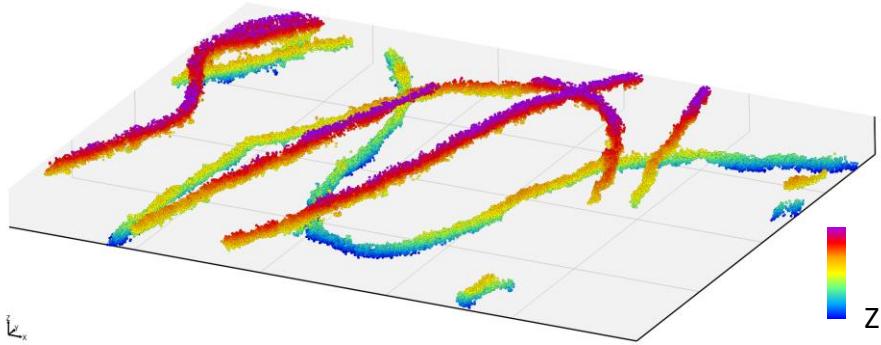


PoCA manual



Florian Levet

Quantitative Imaging of the Cell – Sibarita's team

Interdisciplinary Institute for Neuroscience

CNRS UMR 5297, University of Bordeaux

<https://github.com/flevet/PoCA>

<https://github.com/flevet/PoCA/releases>

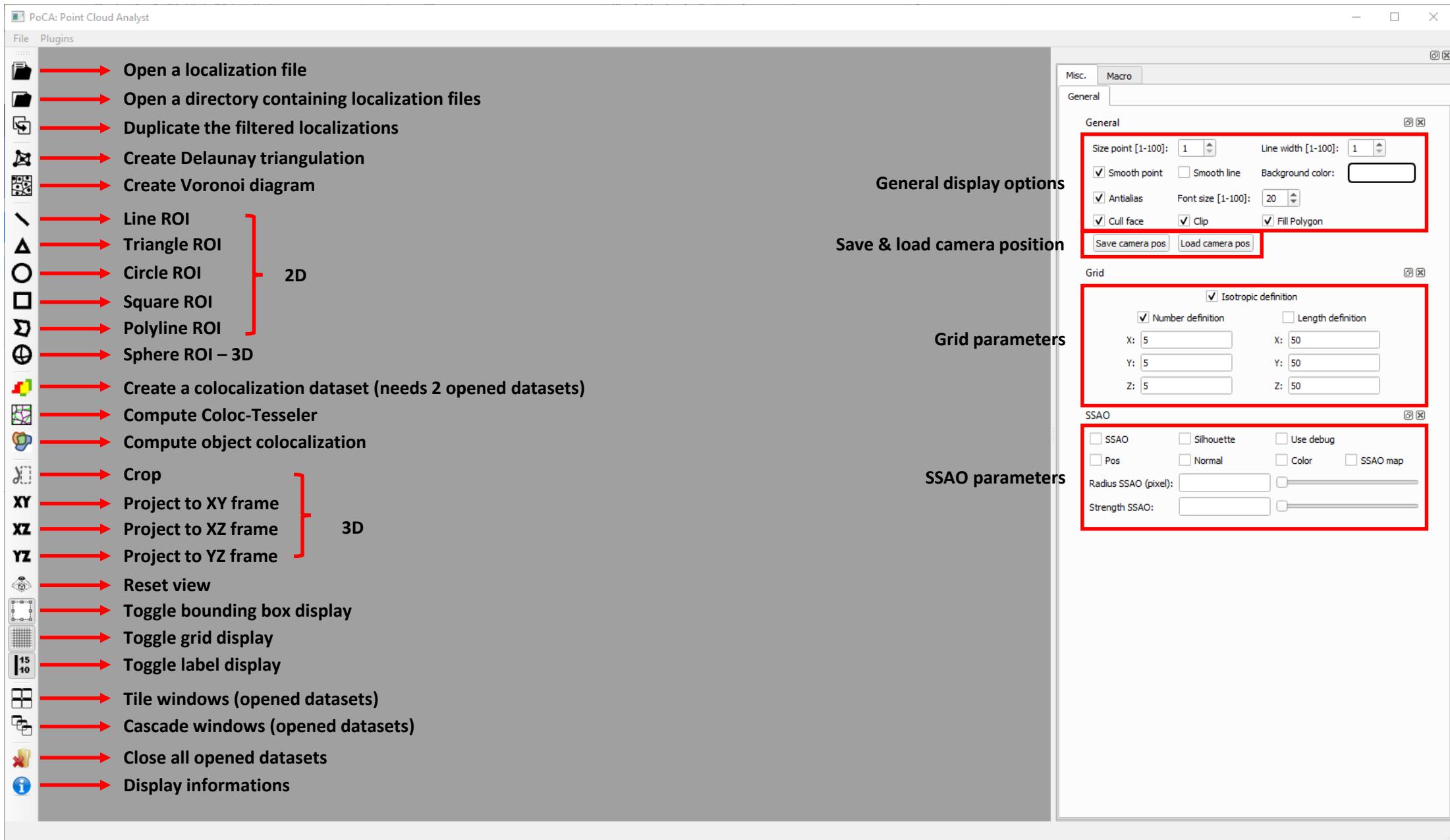
<https://github.com/flevet/PoCA/issues>



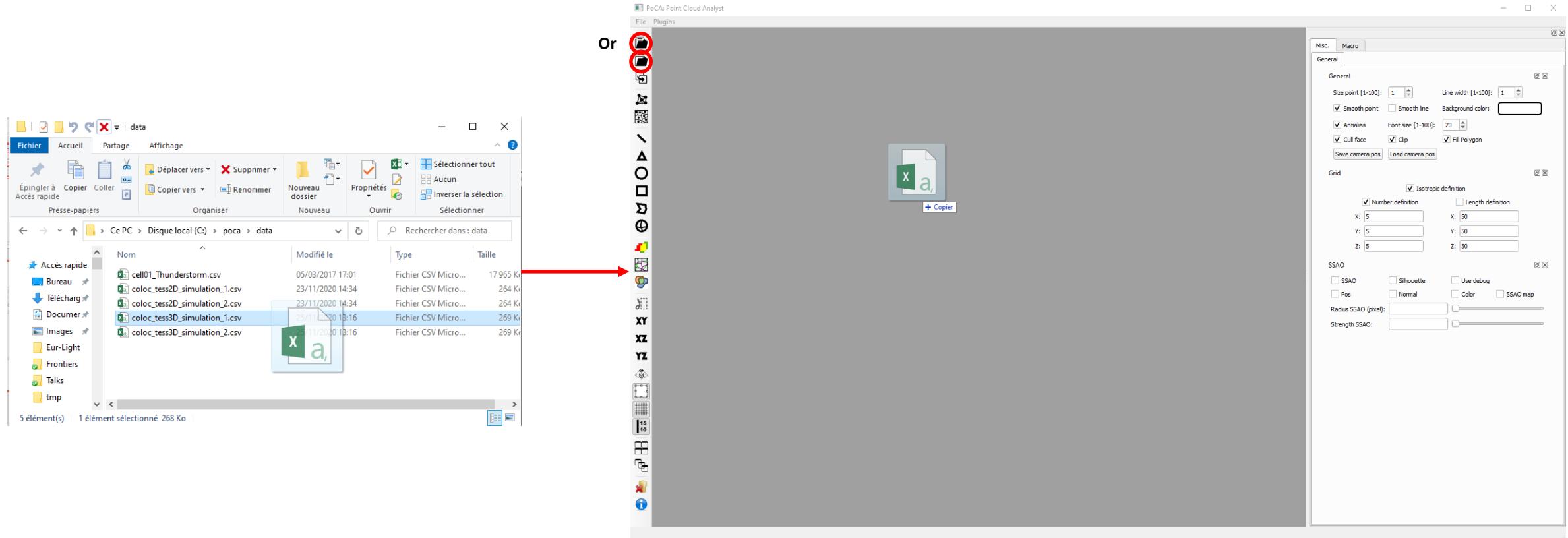
Requirements when using the Windows binaries

- If upon executing poca.exe Windows asks for dlls such as "VCRUNTIME140_1.dll", you may need to install the "microsoft visual c++ 2019 redistributable package (x64)": <https://docs.microsoft.com/en-GB/cpp/windows/latest-supported-vc-redist?view=msvc-160>
- For having access to the Voronoi 3D construction and the 3D visualization, you will need a NVidia card with the latest drivers installed as well as CUDA (10.2 for instance, may work with newest versions): <https://developer.nvidia.com/cuda-10.2-download-archive>
- For communication with Python, you will need to have installed Python 3.7.x (tested with Python 3.7.4)

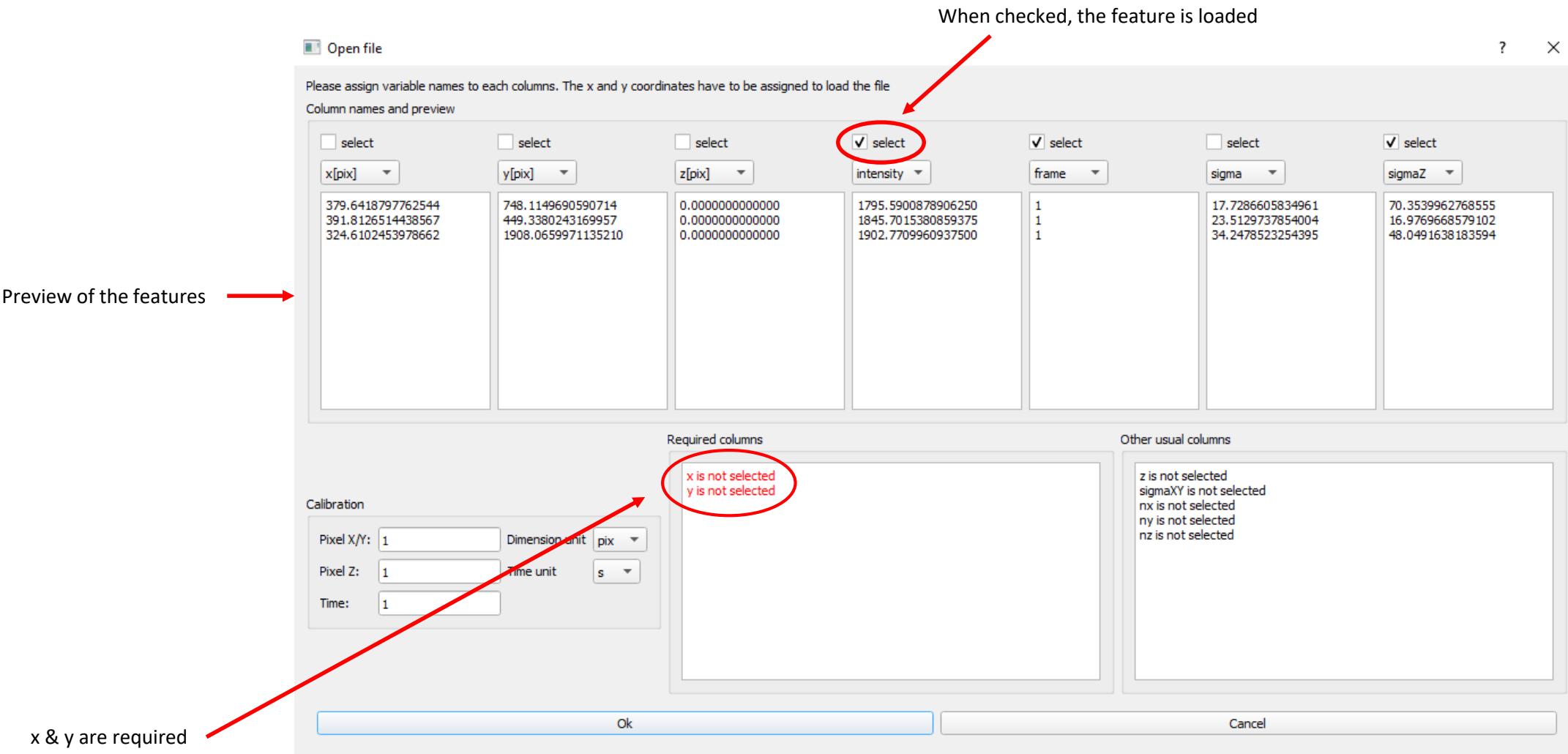
PoCA windows



Open data: use file icon (choose the file), folder icon (open all locs files in the folder) or drag/drop csv file

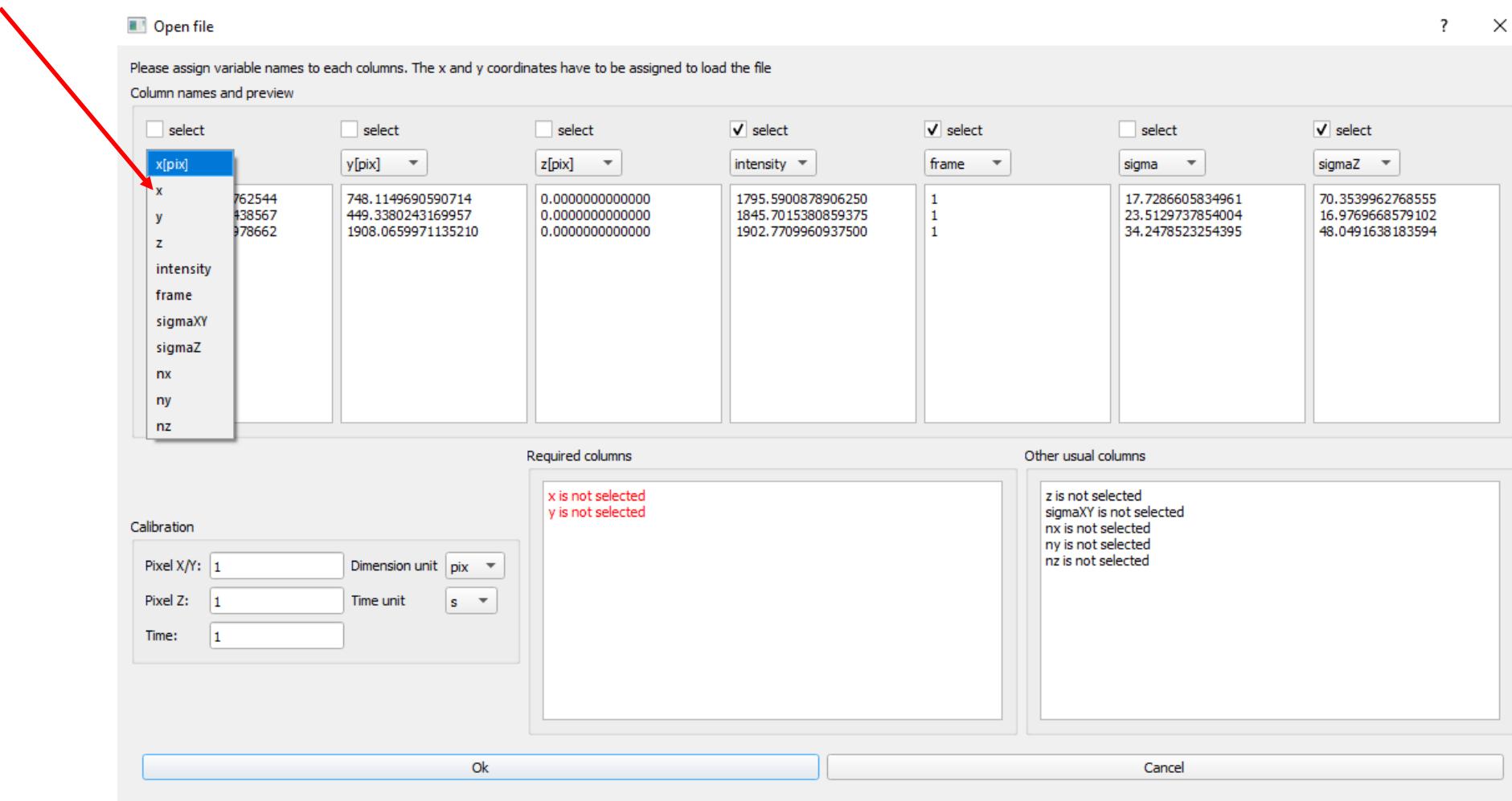


Open data: it will open a new windows where you will be able to choose the features to load



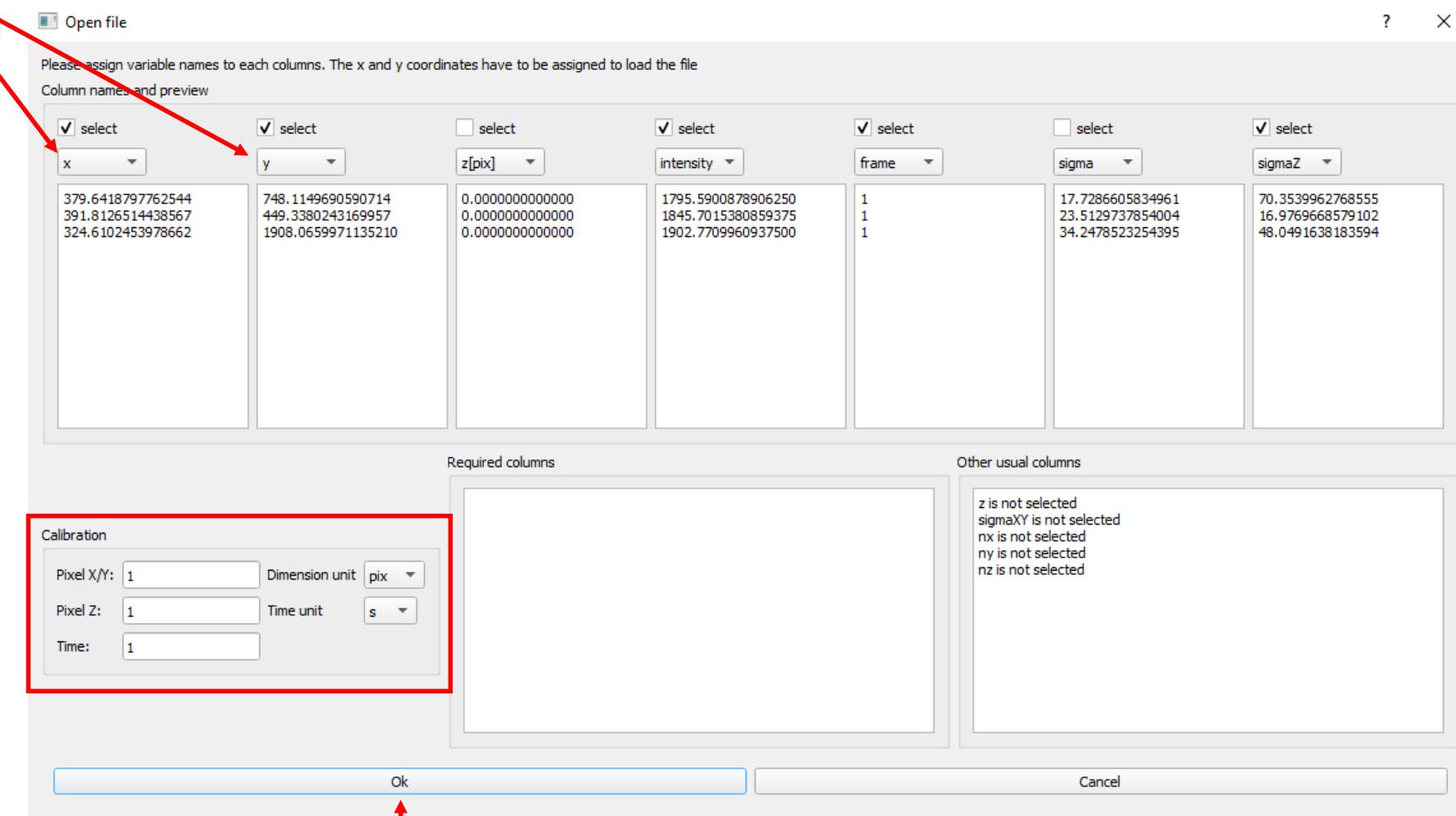
Open data: it will open a new windows where you will be able to choose the features to load

Click on the list to select x, and do the same for y



Open data: it will open a new windows where you will be able to choose the features to load

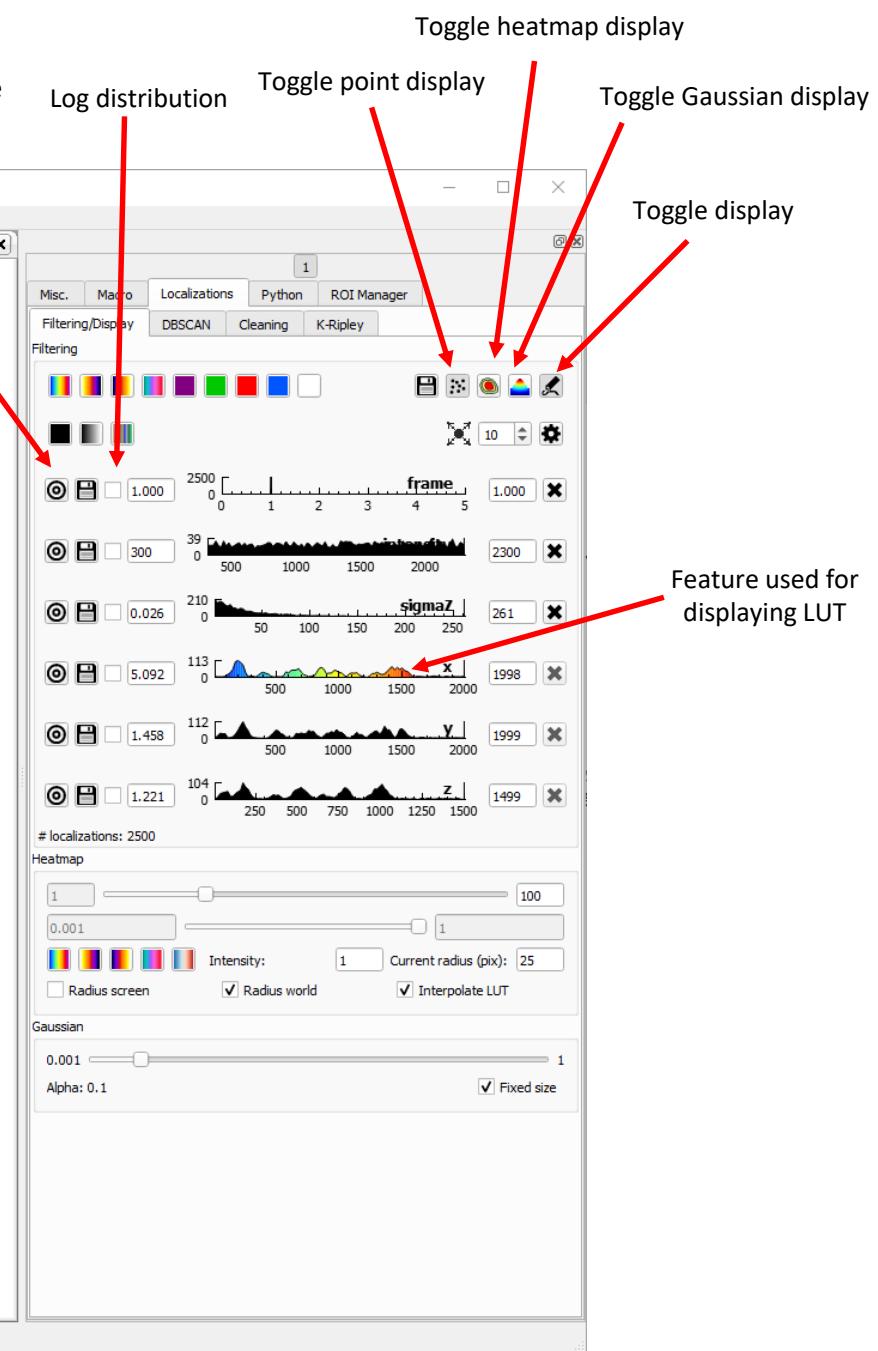
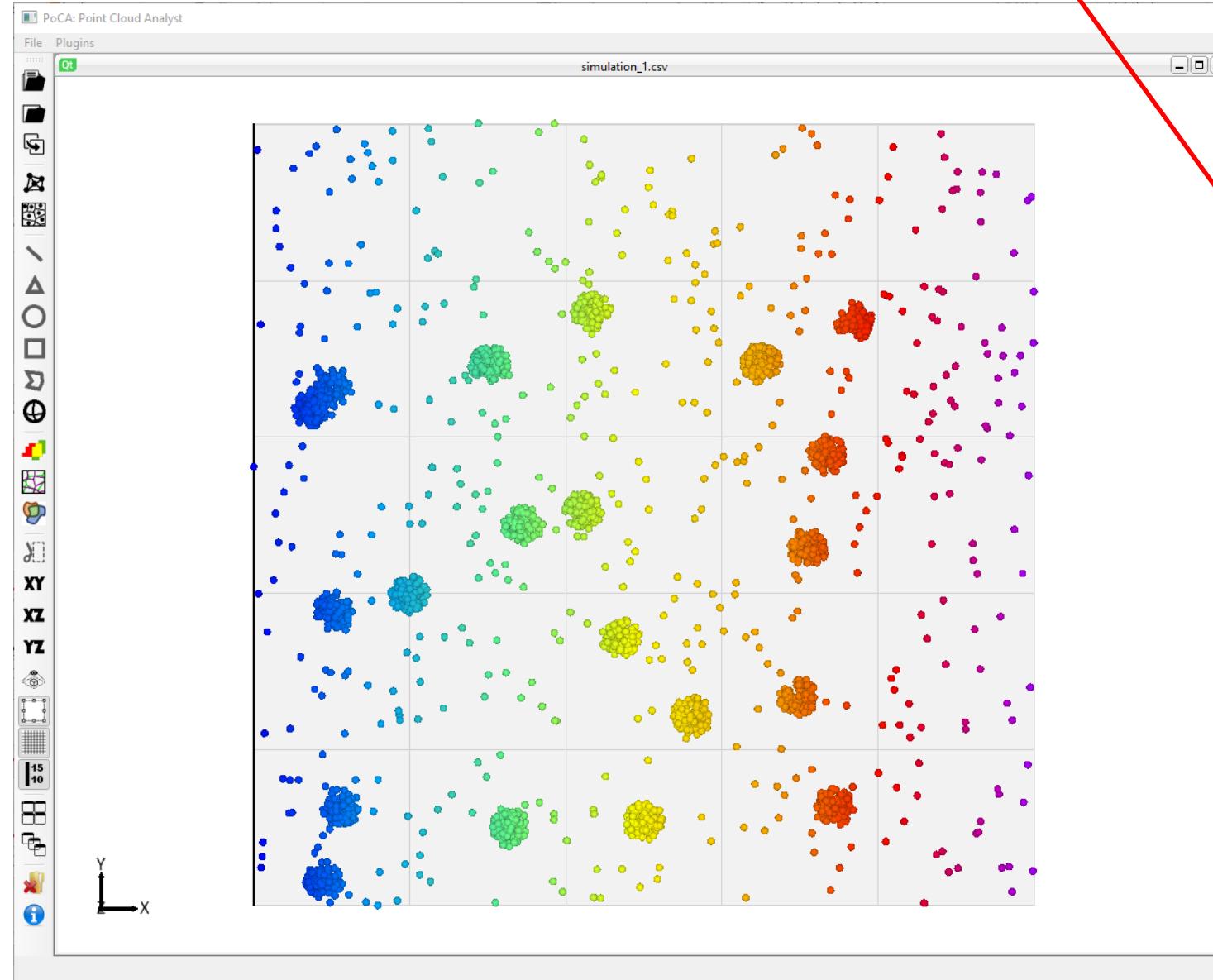
x & y are selected



You can calibrate the data

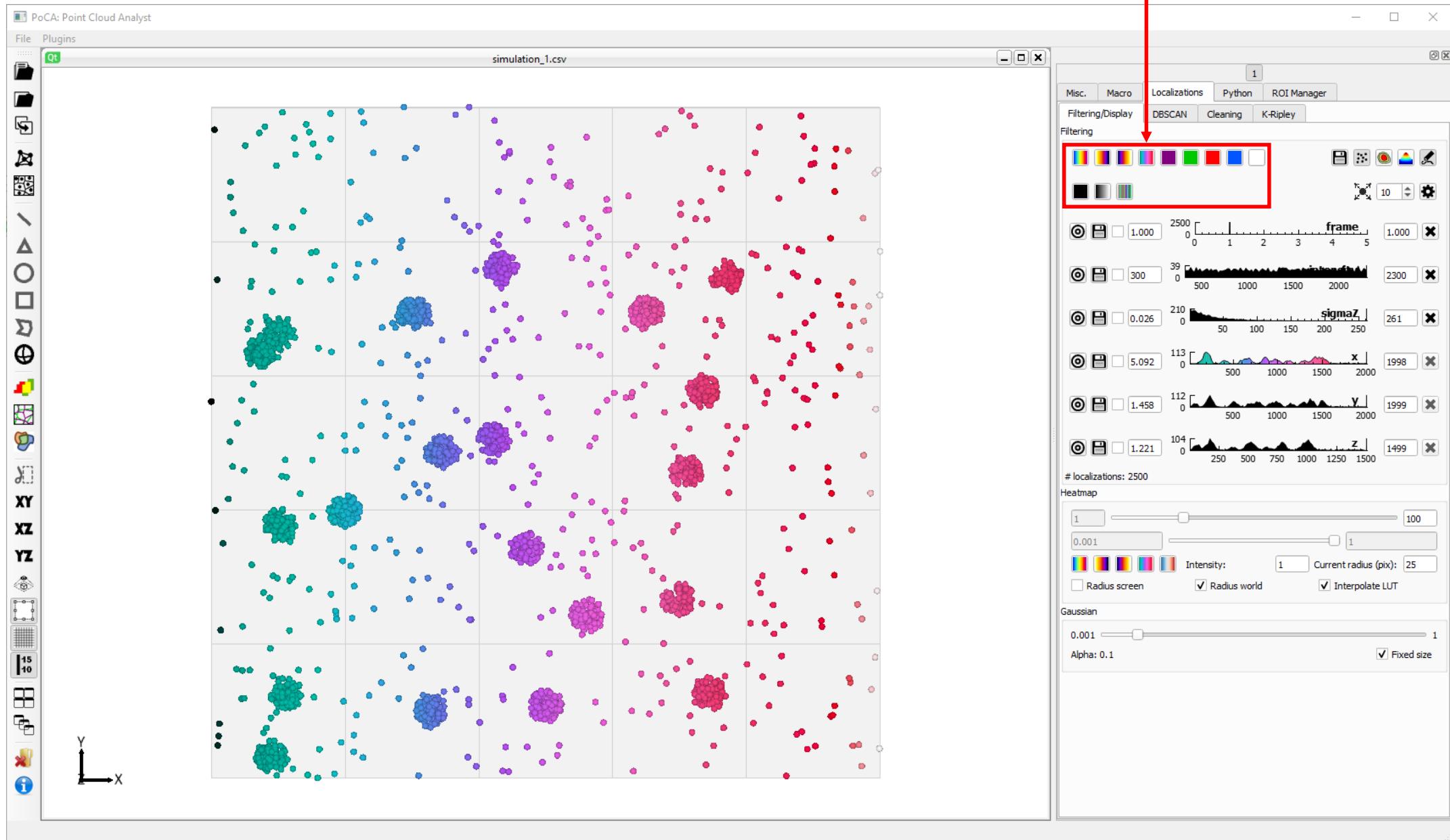
Click Ok to load the data with the selected features

Localizations tab

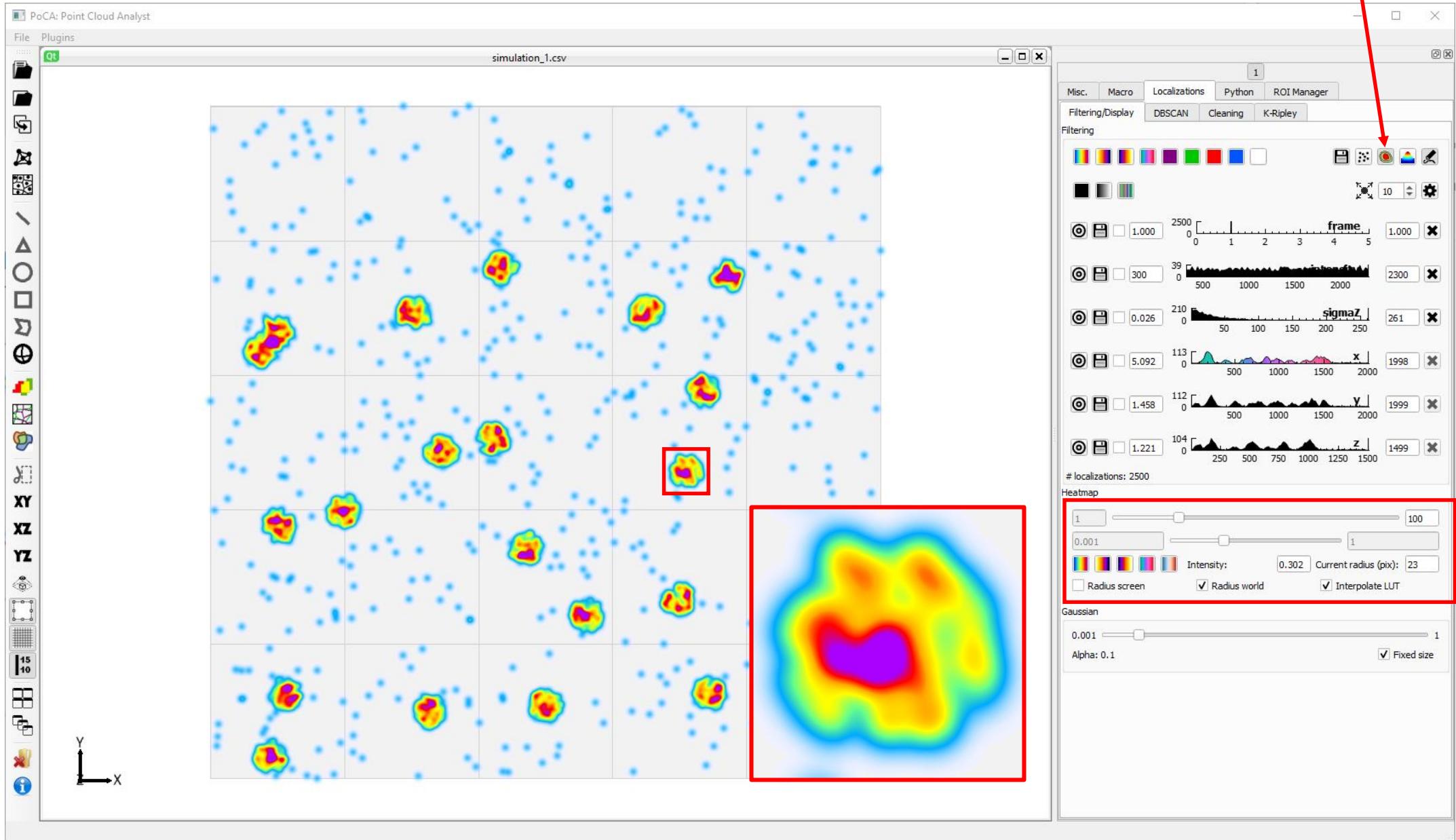


Selection of the intensity feature as LUT displaying with log distribution

Change easily which LUT is used by clicking on the icon

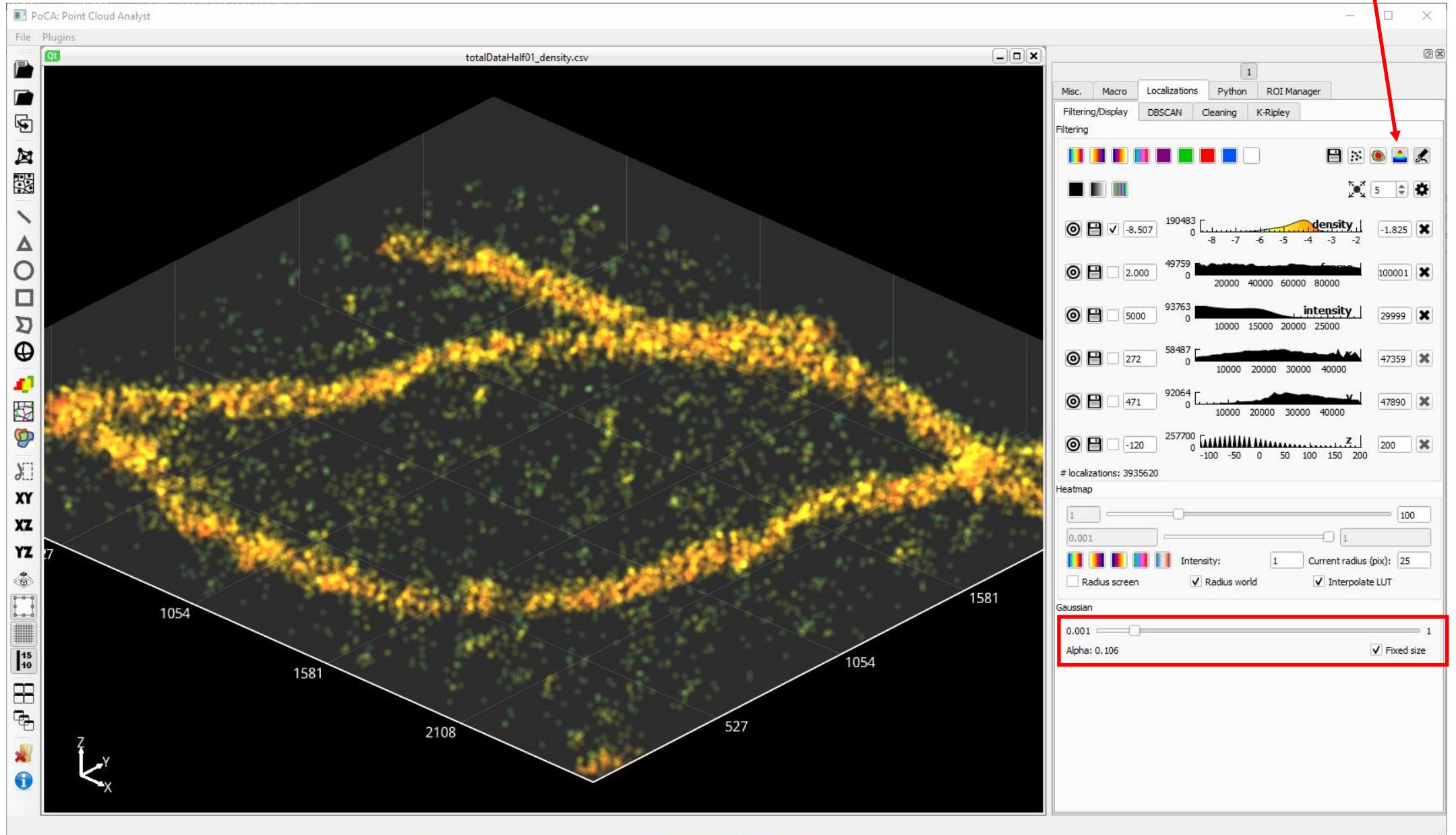


Heatmap rendering



Gaussian rendering

Works better with a black background since it requires alpha blending



parameter

3D view

left click for rotation

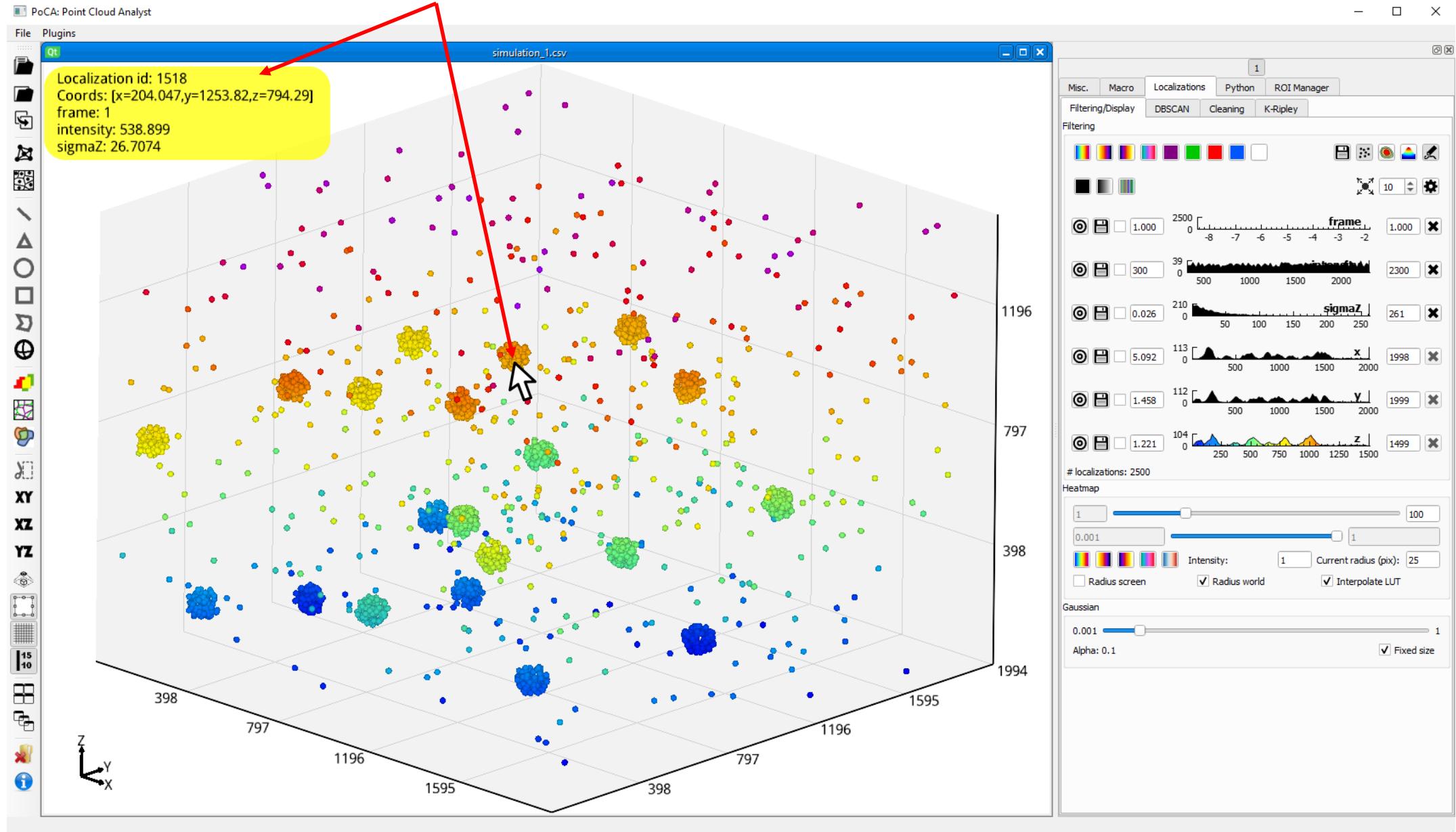
Shift + left click for zoom

Middle click for translation

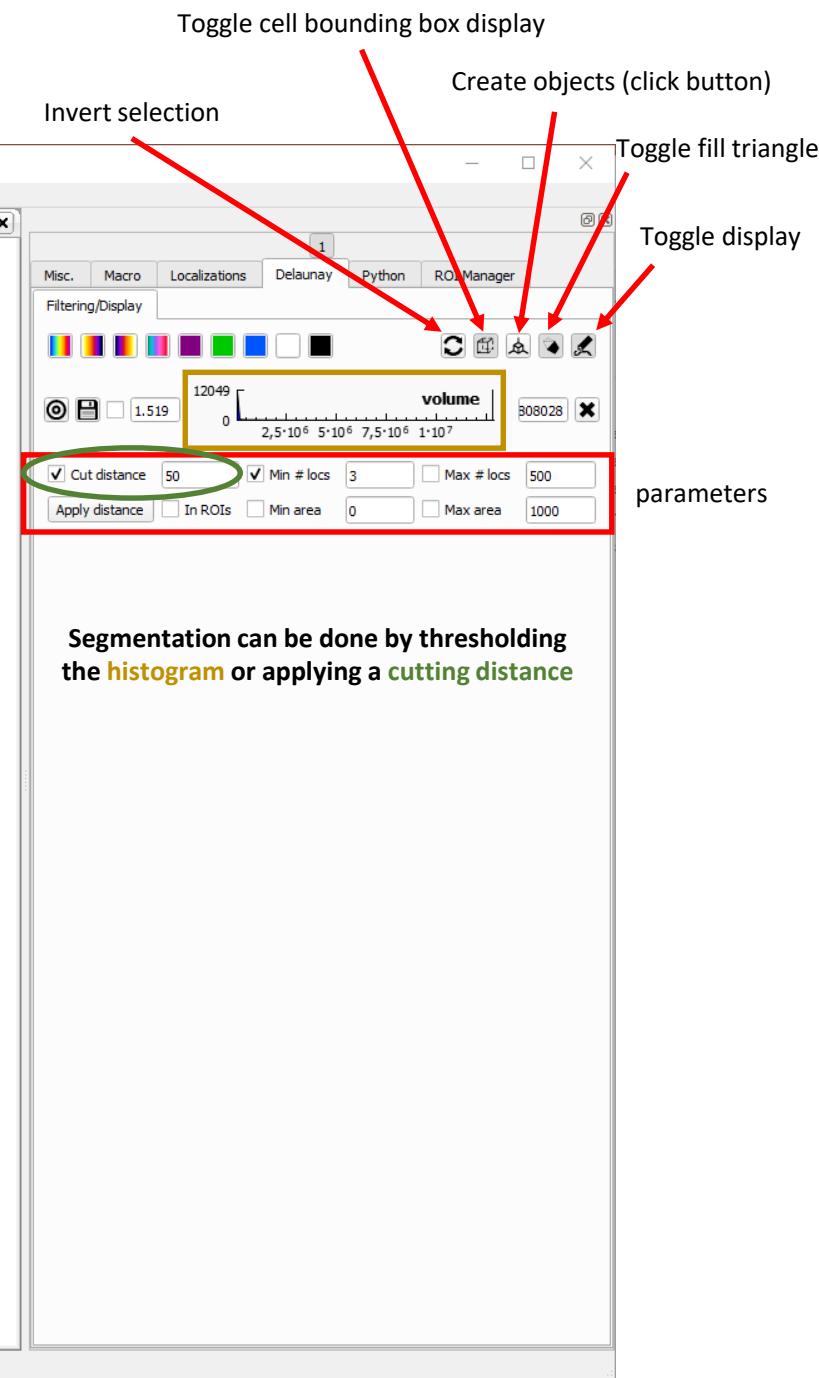
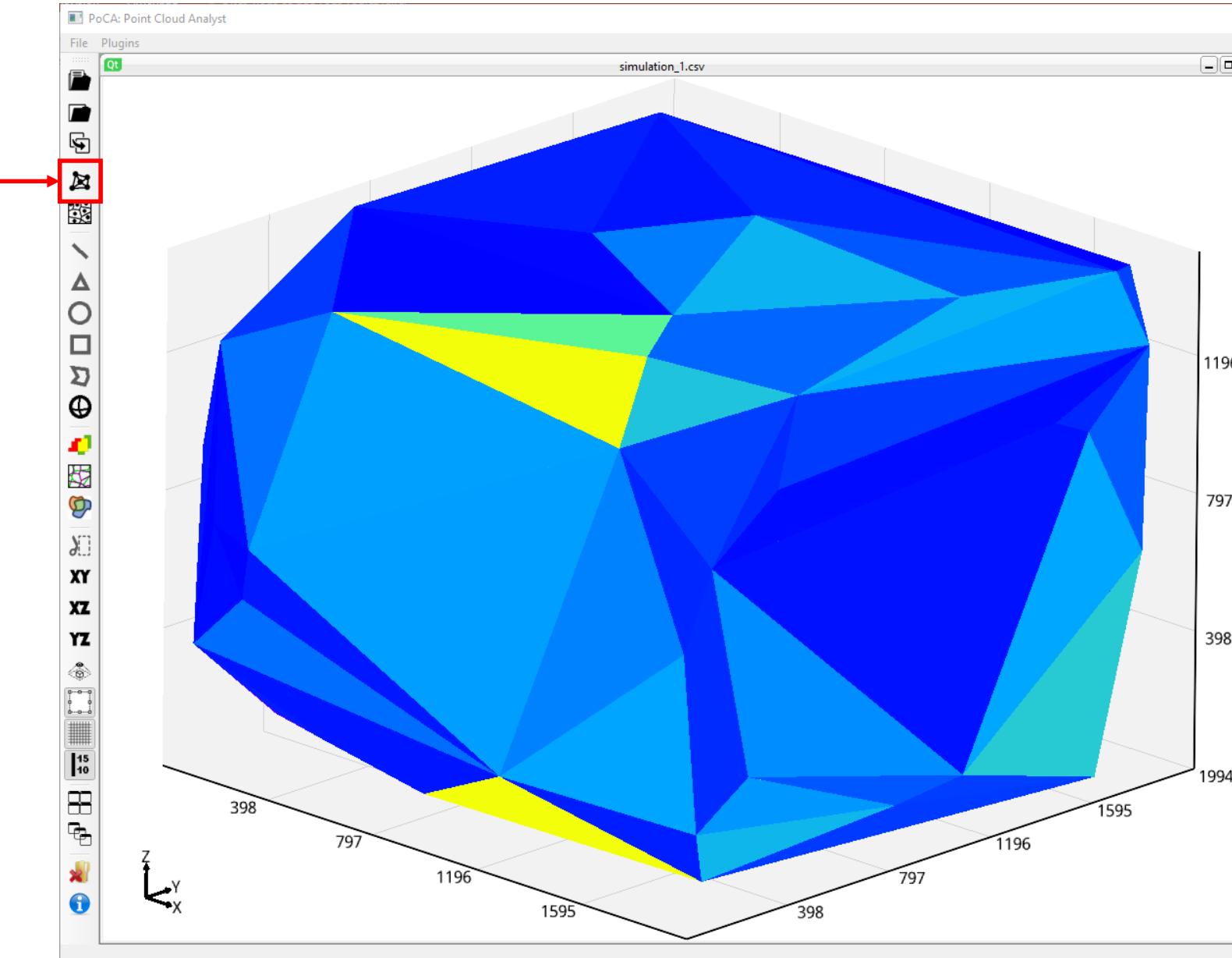
Wheel for zoom

Picking

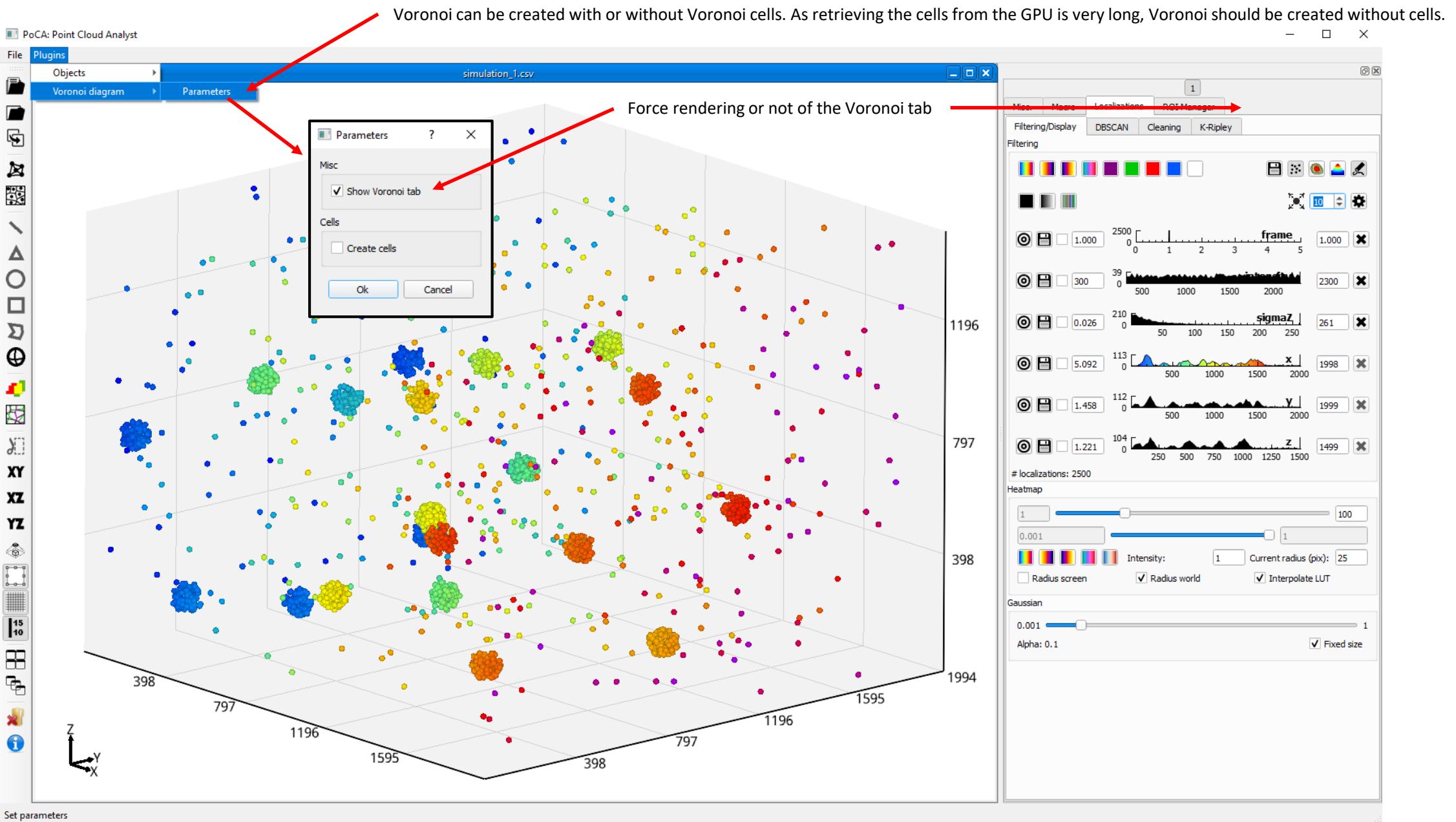
Information on the localization under the mouse icon are displayed



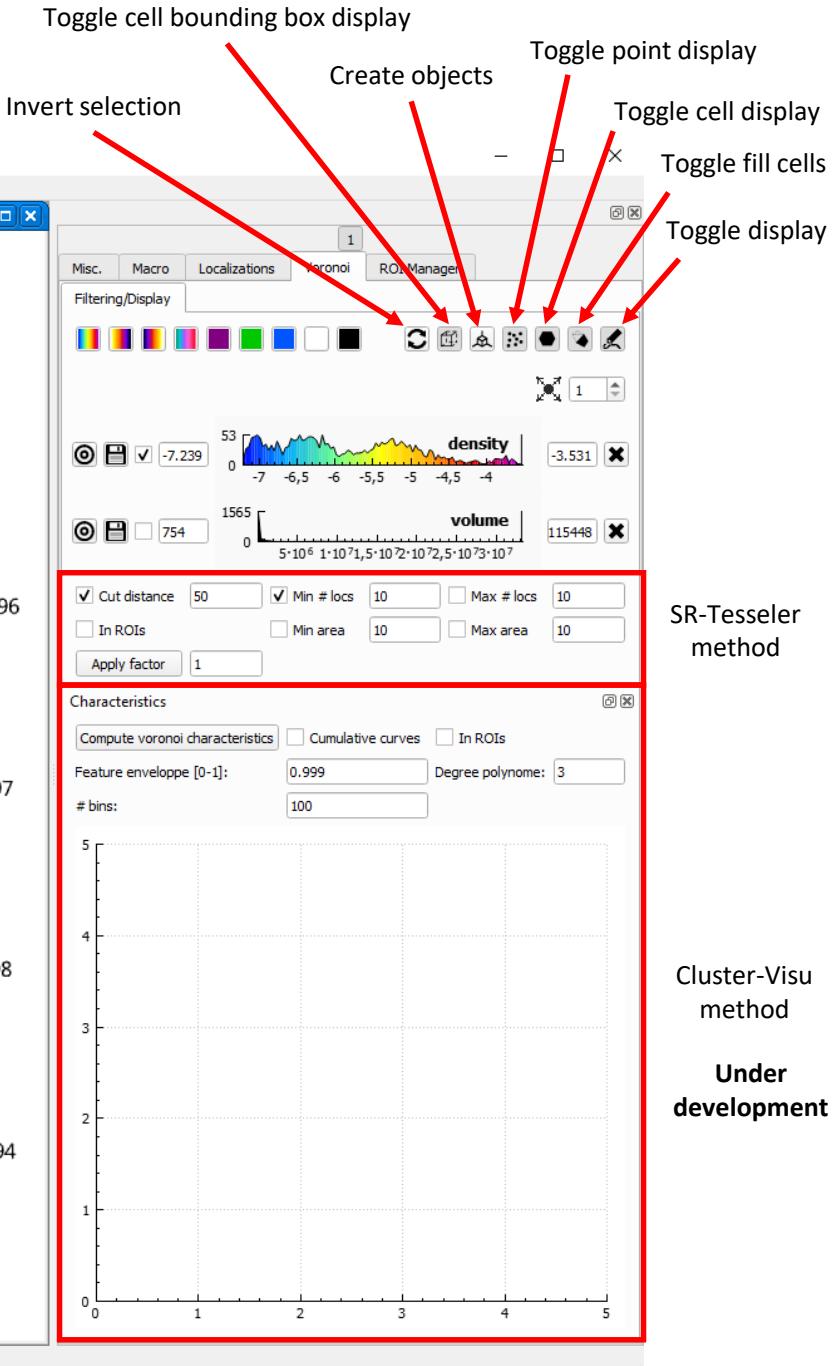
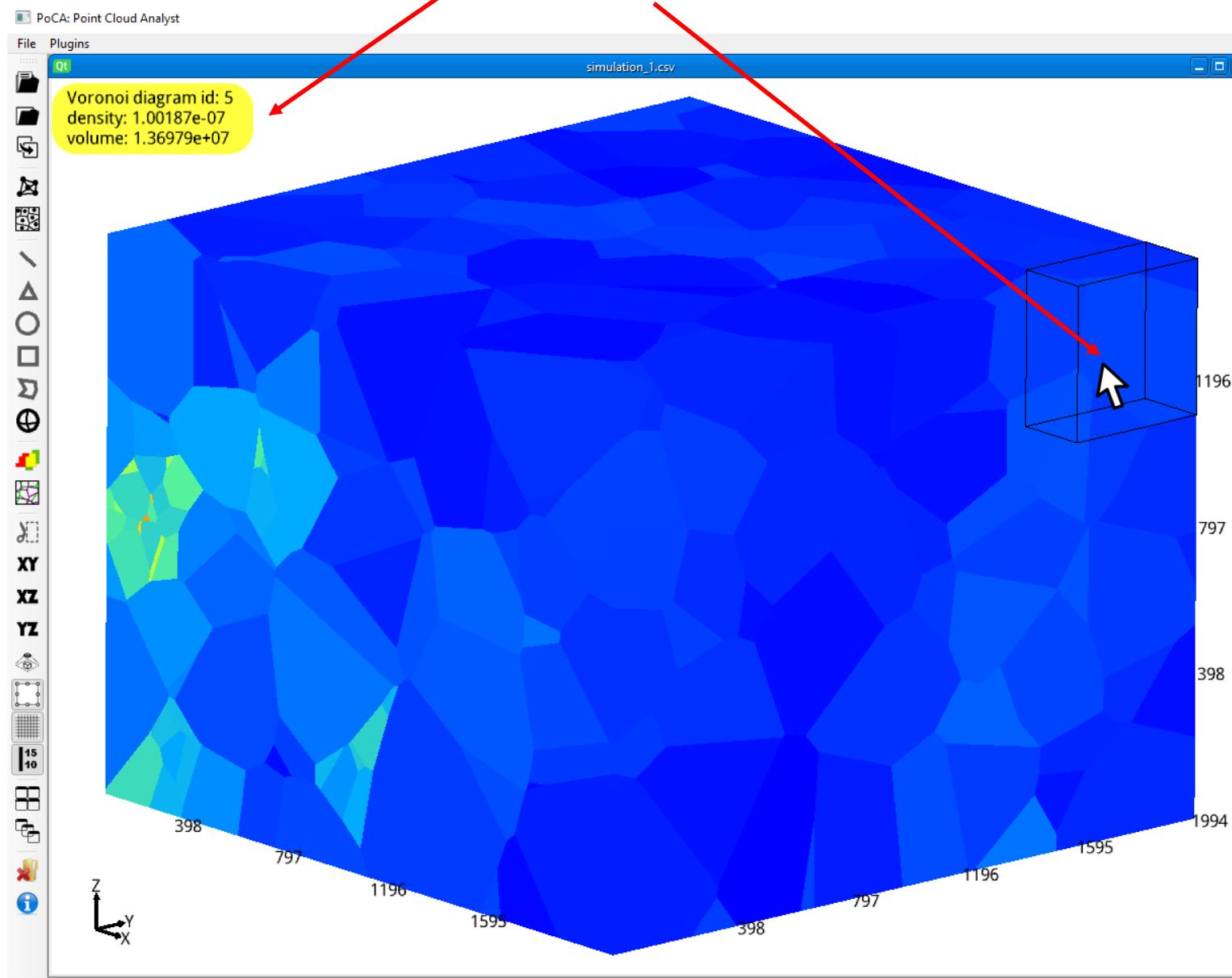
Delaunay triangulation



Voronoi creation



Voronoi created with cells



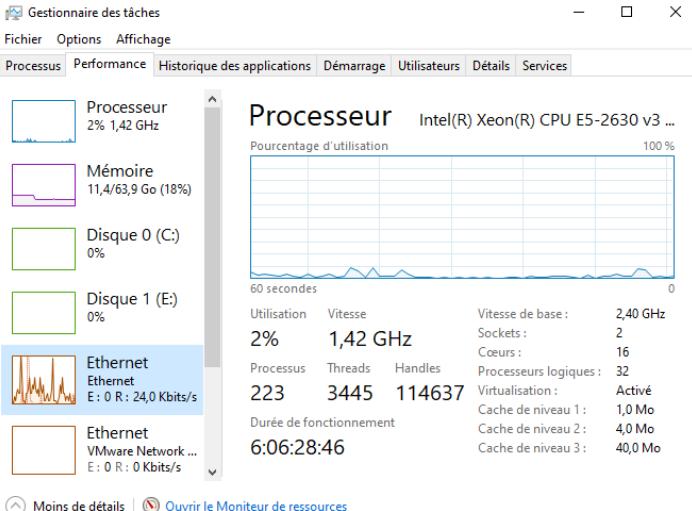
Voronoi timing

2D Voronoi on 6,000,000 localizations ≈ 40s

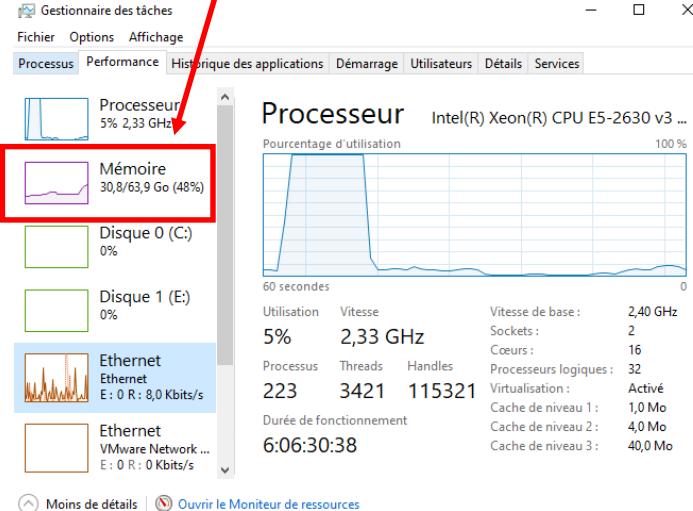
3D Voronoi on 4,000,000 localizations (without cells) ≈ 50s

3D Voronoi necessitates a Nvidia card and is very RAM demanding. If you hit 100% RAM usage, the software won't crash but the algorithm will freeze

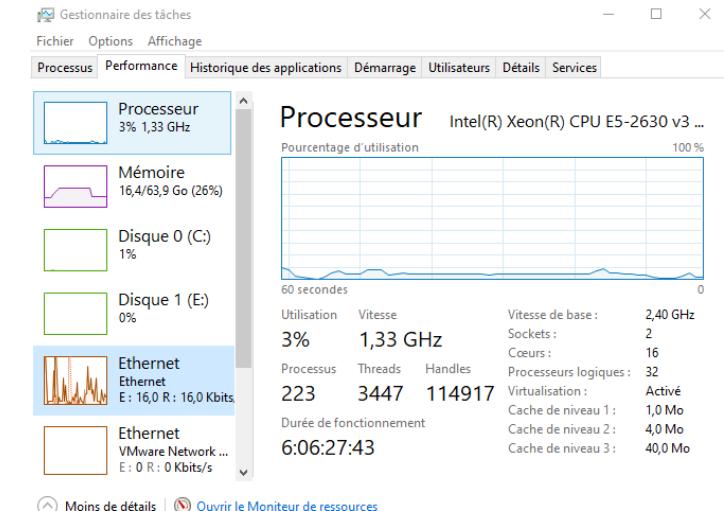
Baseline (before executing PoCA)



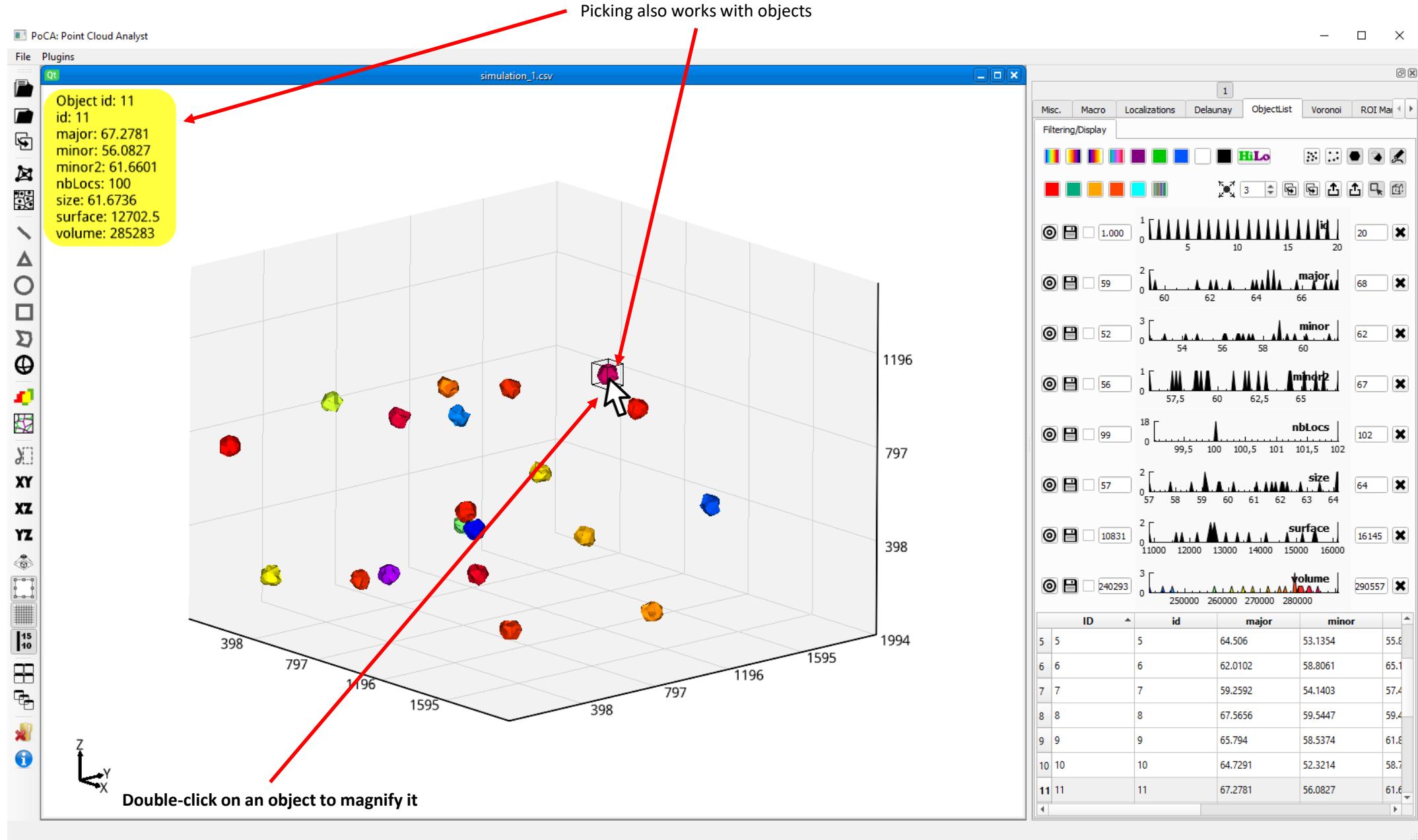
Memory peak during construction



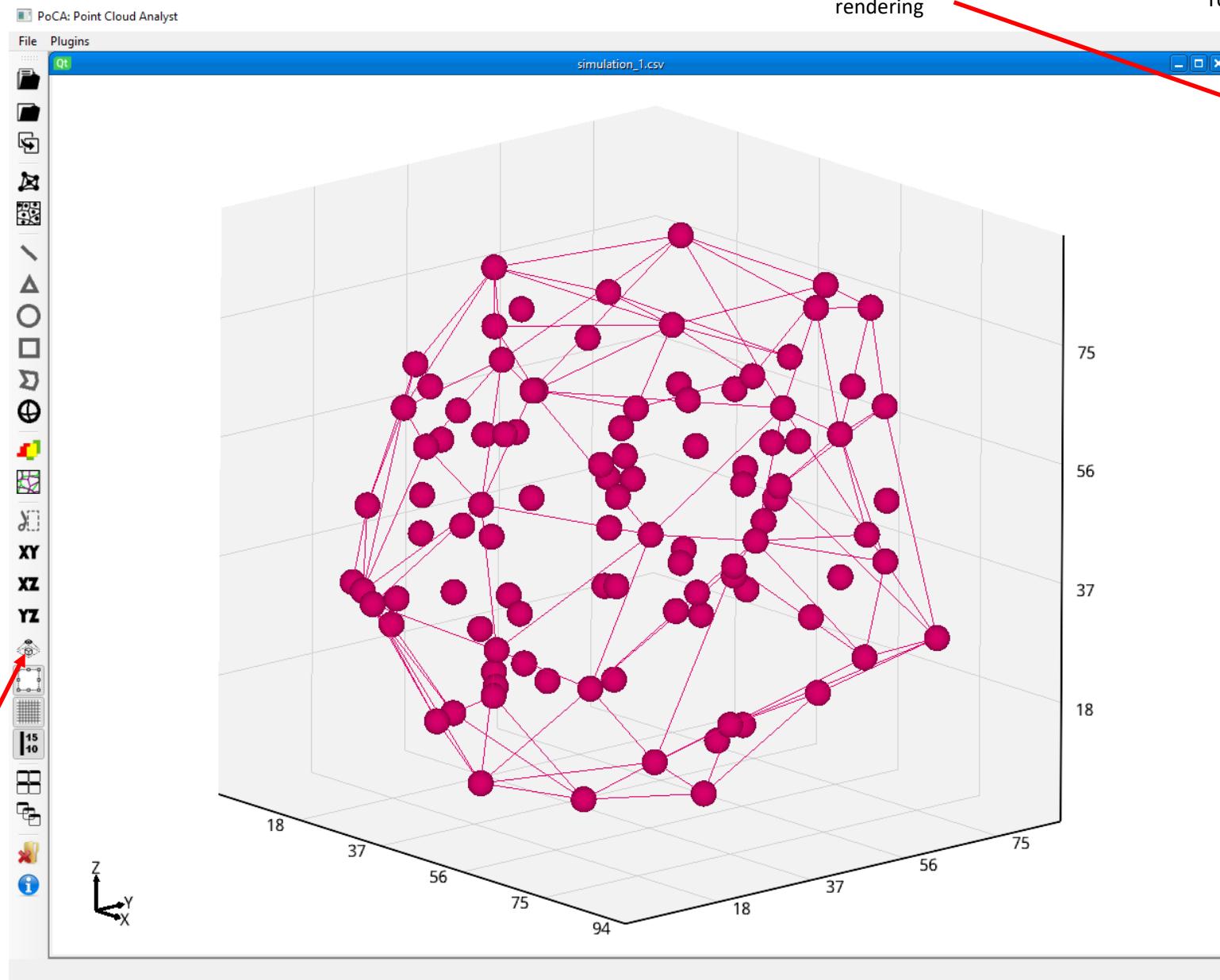
3D Voronoi on 4,000,000 locs done



ObjectList tab



ObjectList tab



Toggle surface point display Toggle surface object

Toggle point display



Toggle fill object

Toggle display

Toggle picked object bbox

Toggle picking

Save objects locs/features

Save objects stats

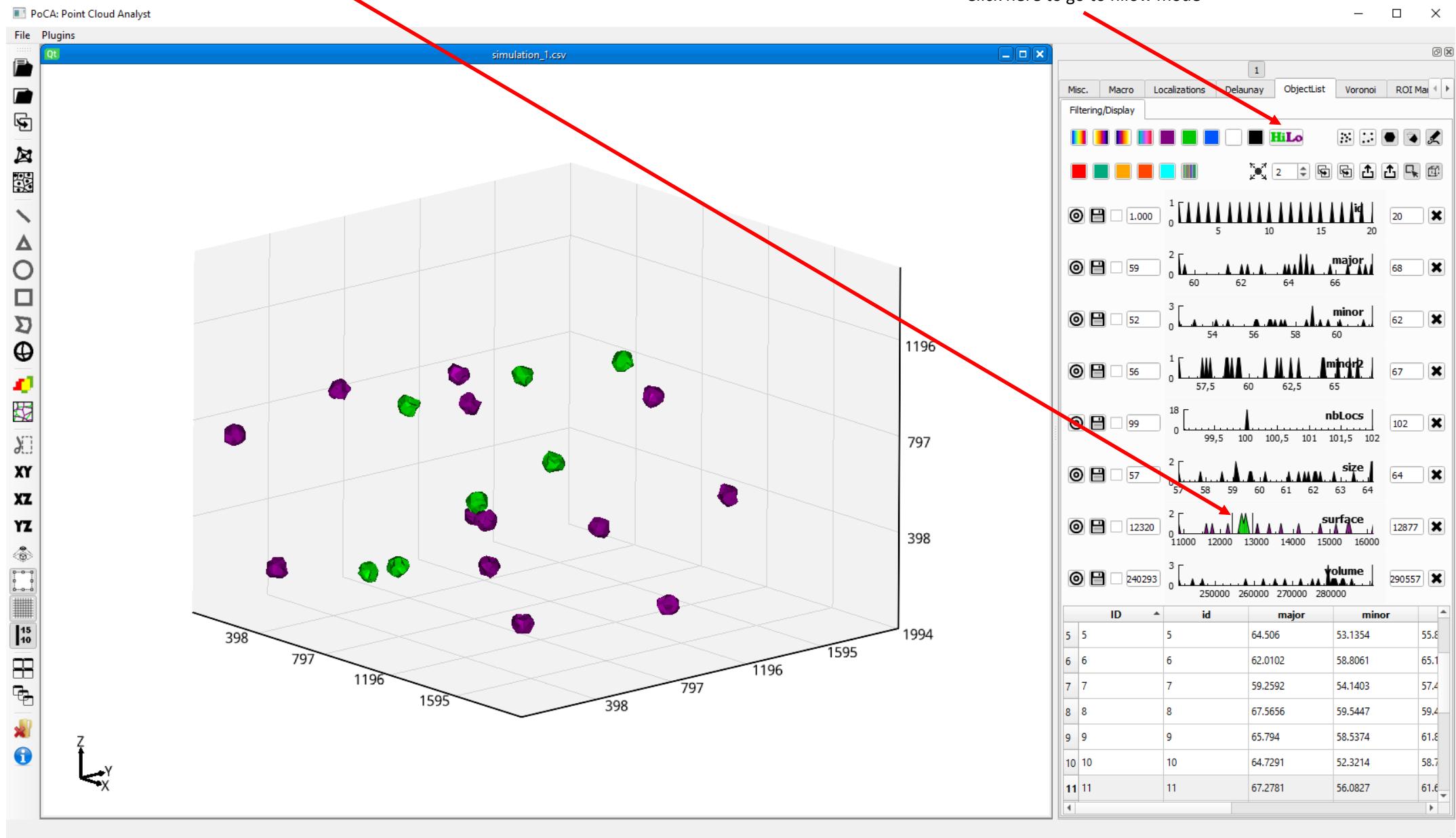
Duplicate selected objects

Duplicate centroids

ObjectList tab: hilow display

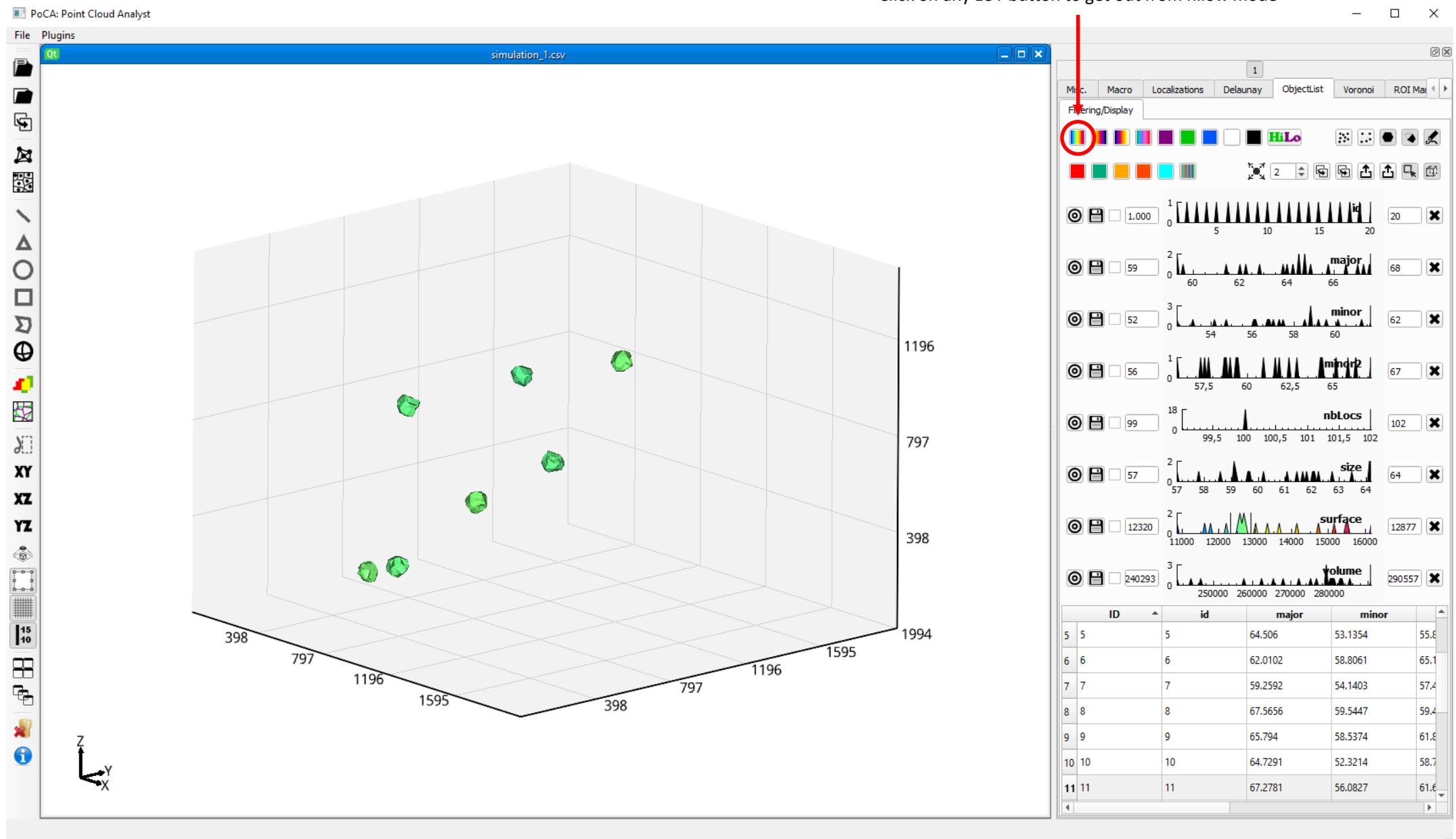
Objects in green are selected, objects in magenta are not selected

Click here to go to hilow mode



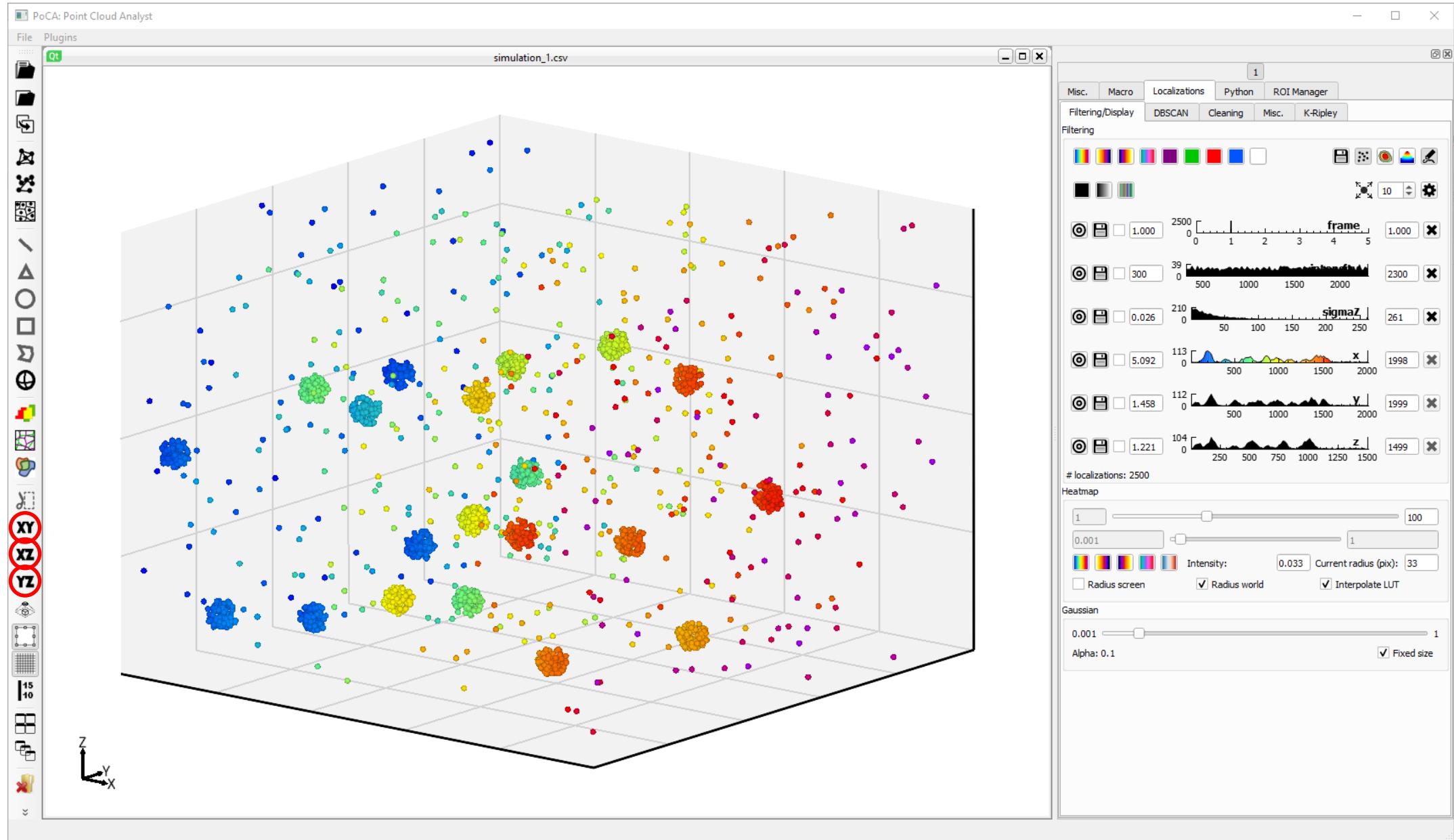
ObjectList tab: same display with normal LUT selected

Click on any LUT button to get out from hilow mode



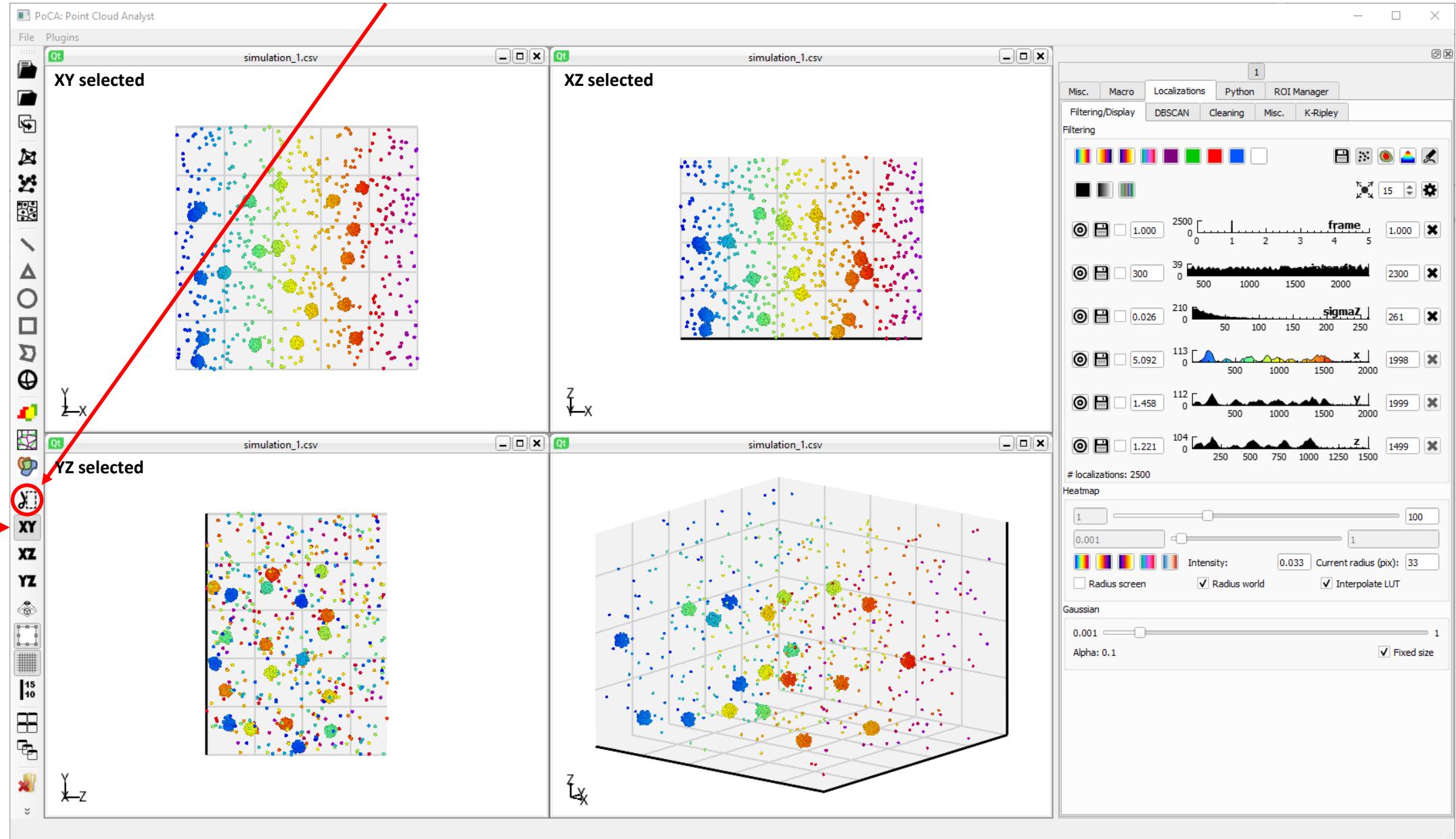
3D Viewer: crop

First, click on XY, XZ or YZ buttons to project the data to this frame



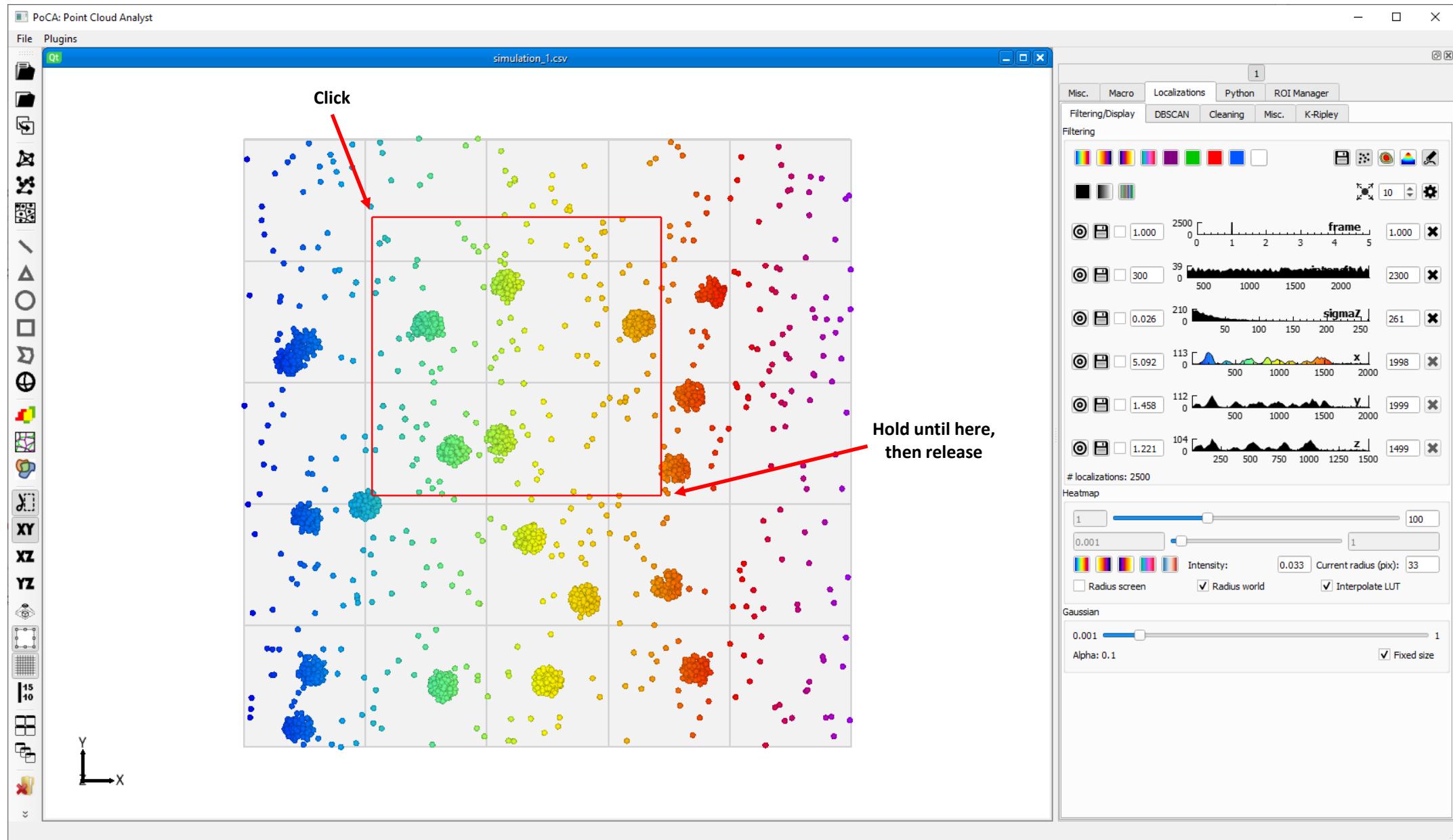
3D Viewer: crop

When one of these button is selected, the crop button becomes usable



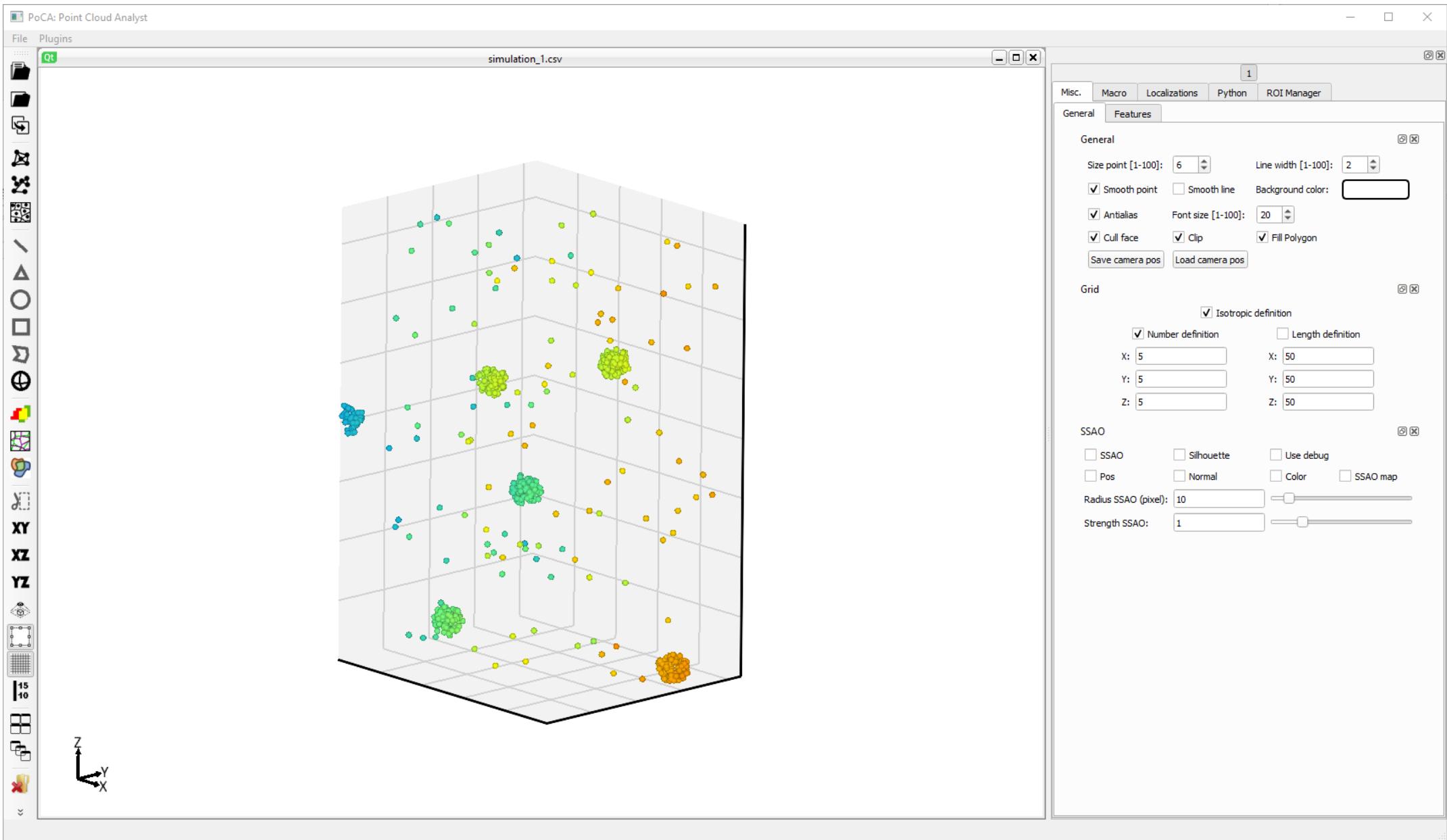
3D Viewer: crop

You can then click on the data, hold and move the mouse. When the button is released, the data will be cropped



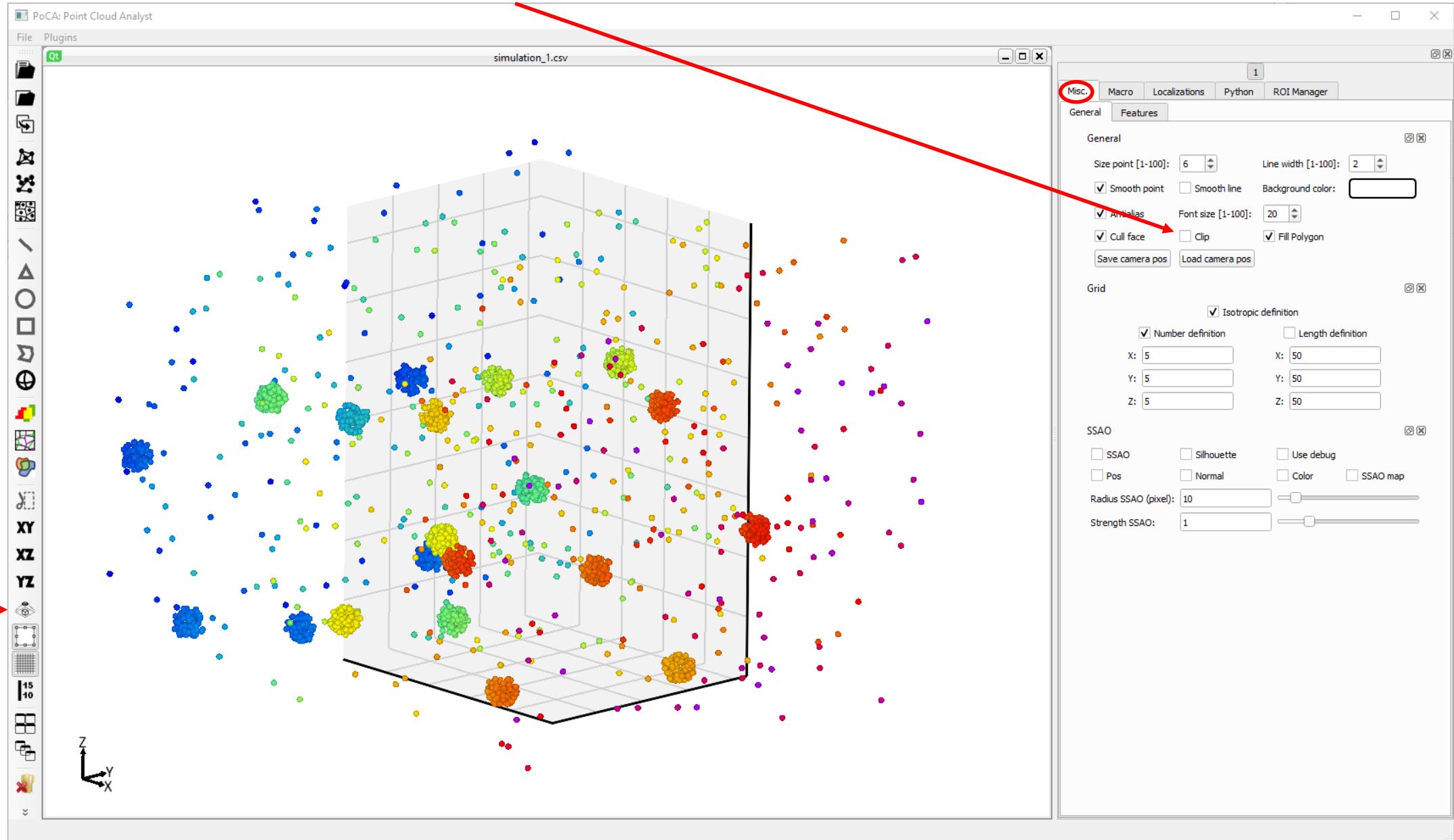
3D Viewer: crop

The [XY, XZ, YZ] button and the crop button have to be unselected to be able to rotate the data again



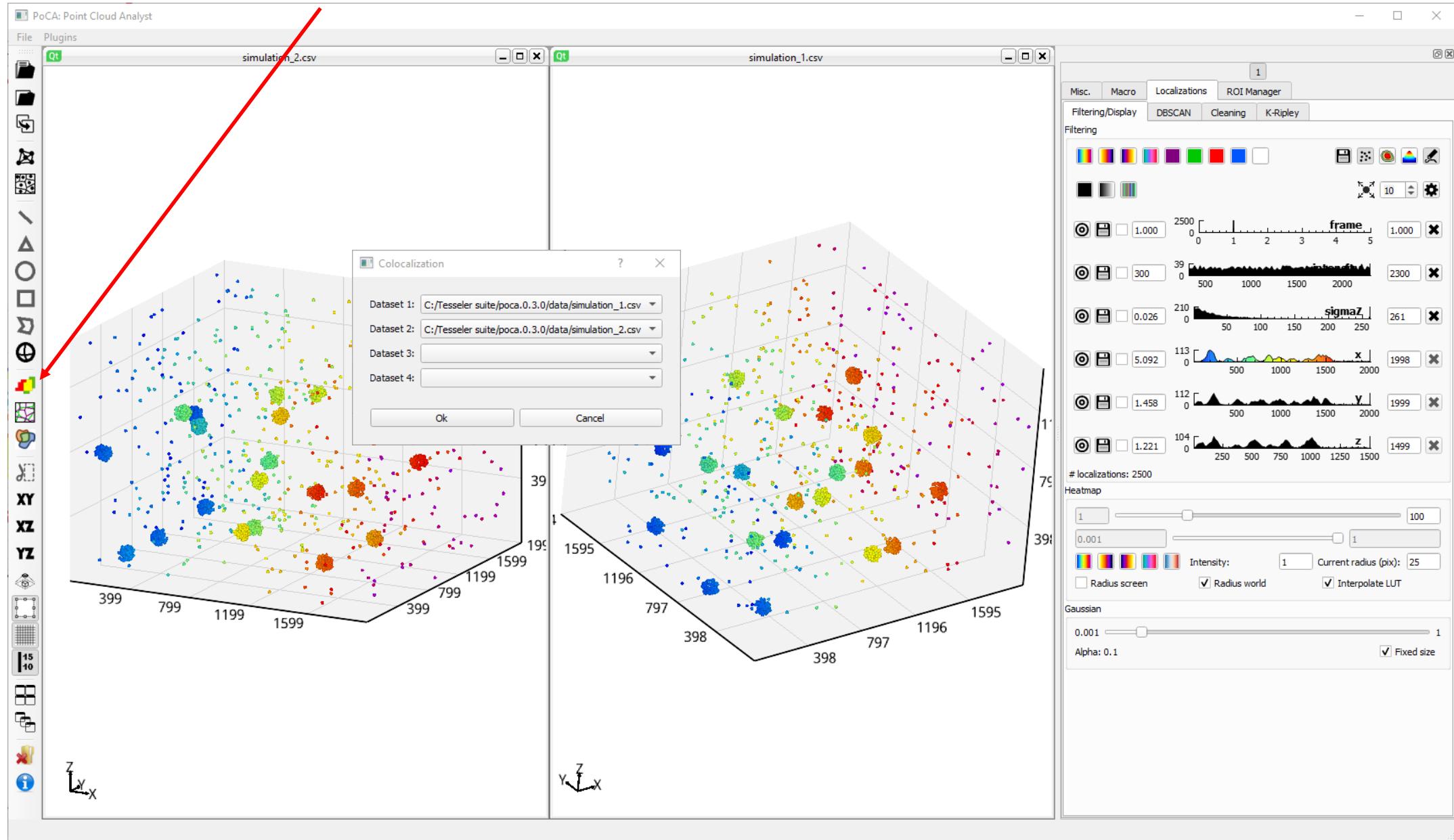
3D Viewer: crop

Crop is based on OpenGL clipping, the clip checkbox has to be checked for it to work



Colocalization object

Open 2 datasets and click this button



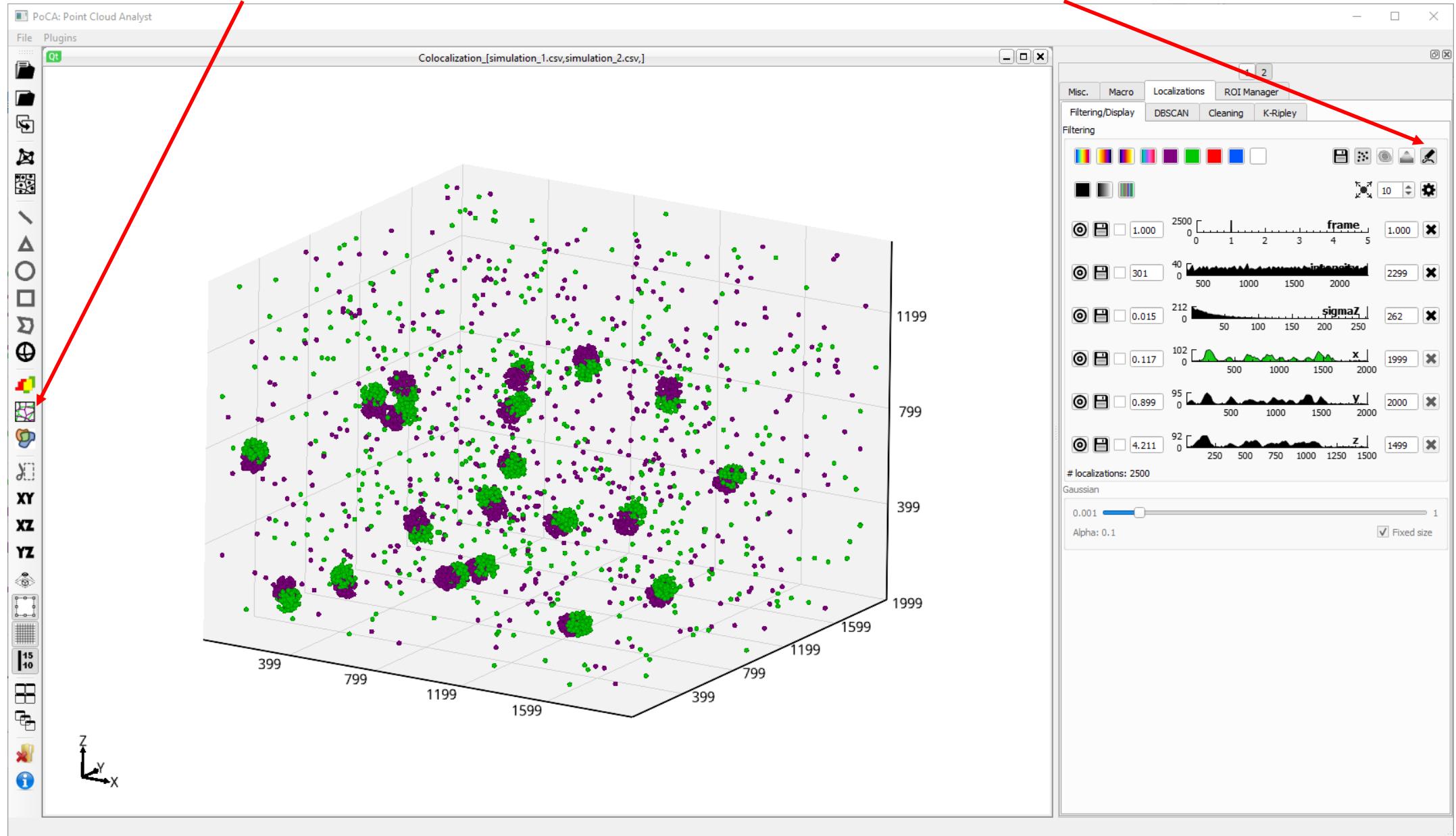
Colocalization object



Coloc-Tesseler

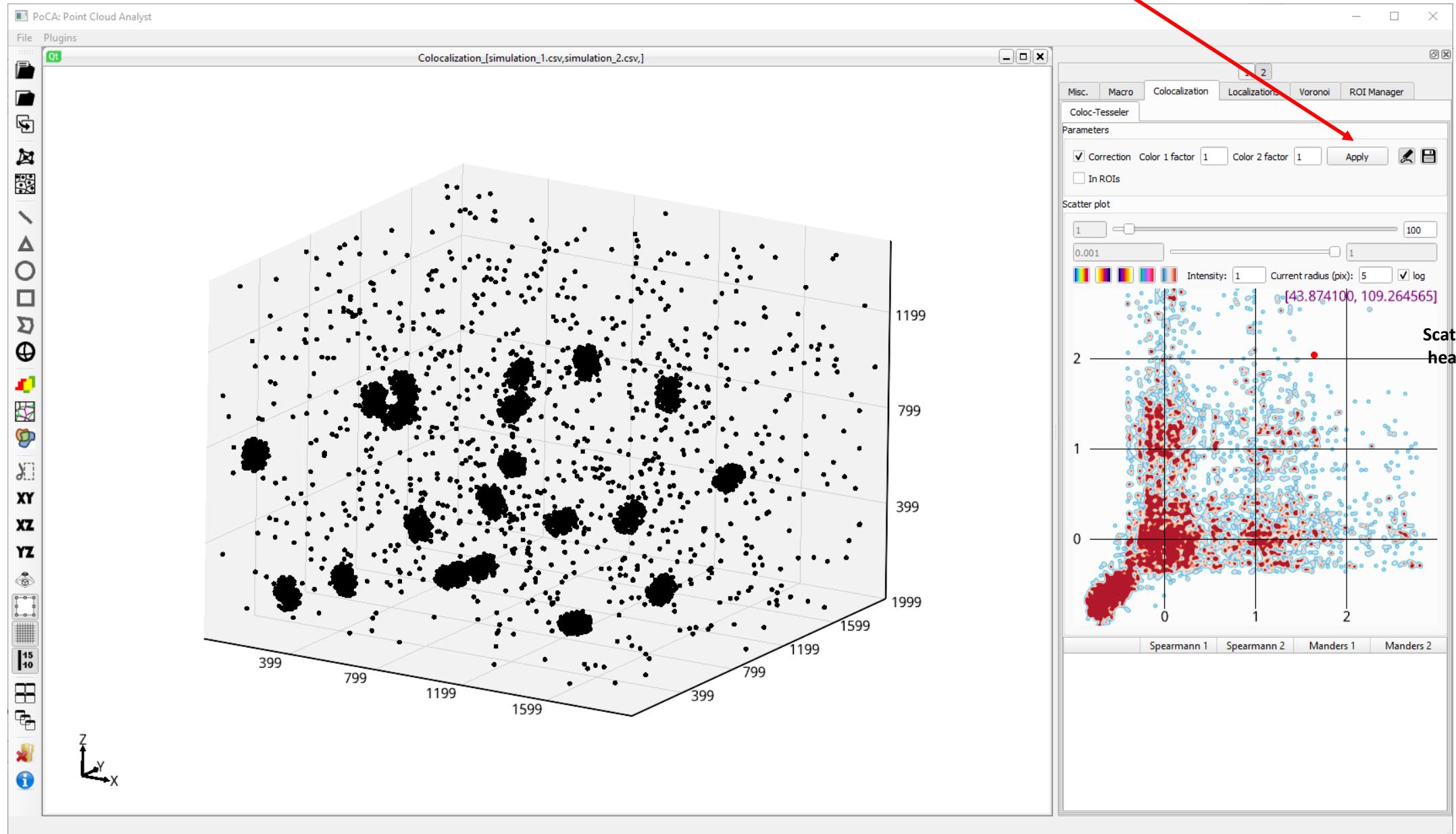
Click here

Coloc-Tesseler displays is under the localization rendering, you need to switch off for both colors



Coloc-Tesseler

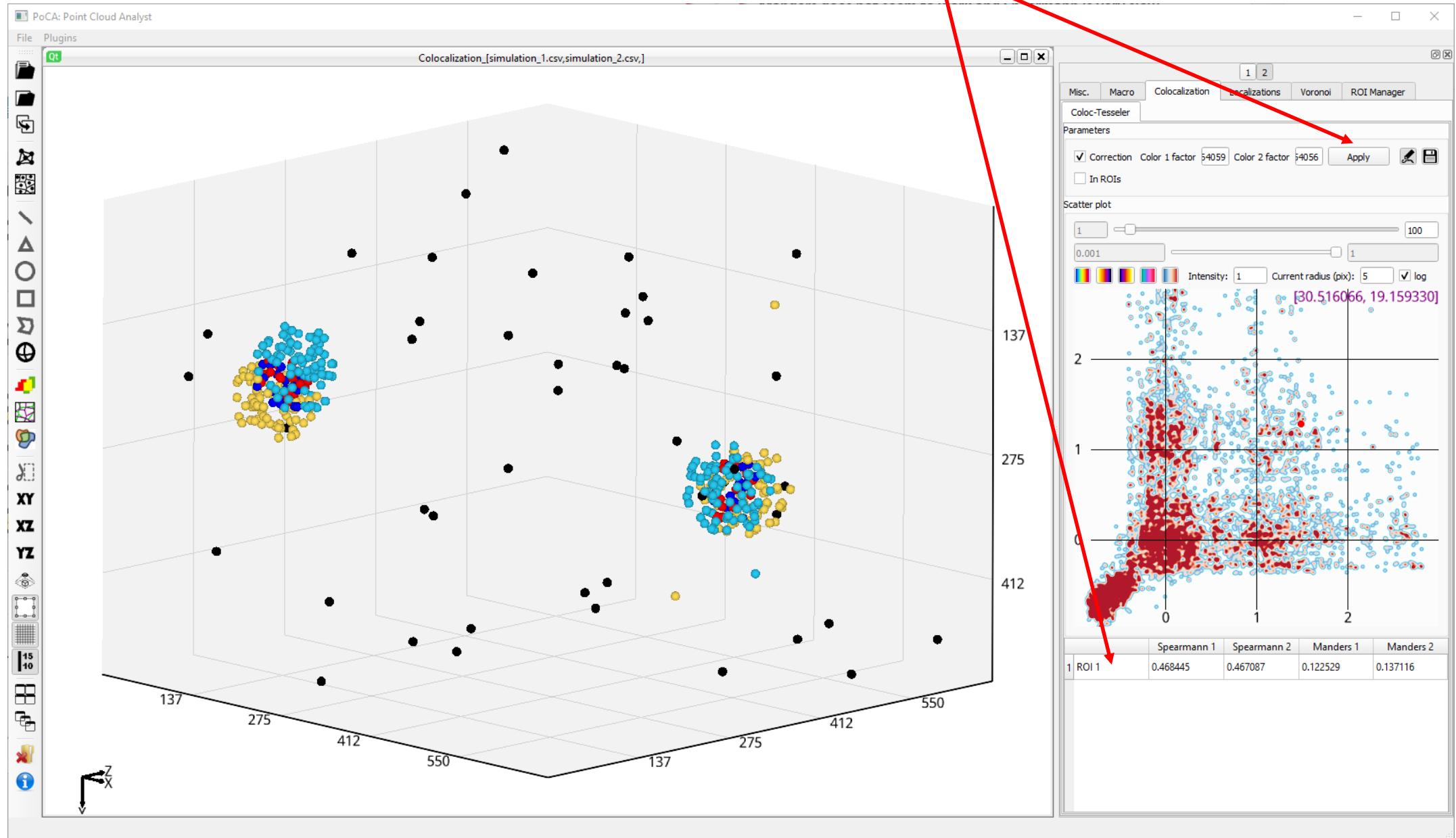
Display is black since no threshold was applied



Scatterplot uses a
heatmap display

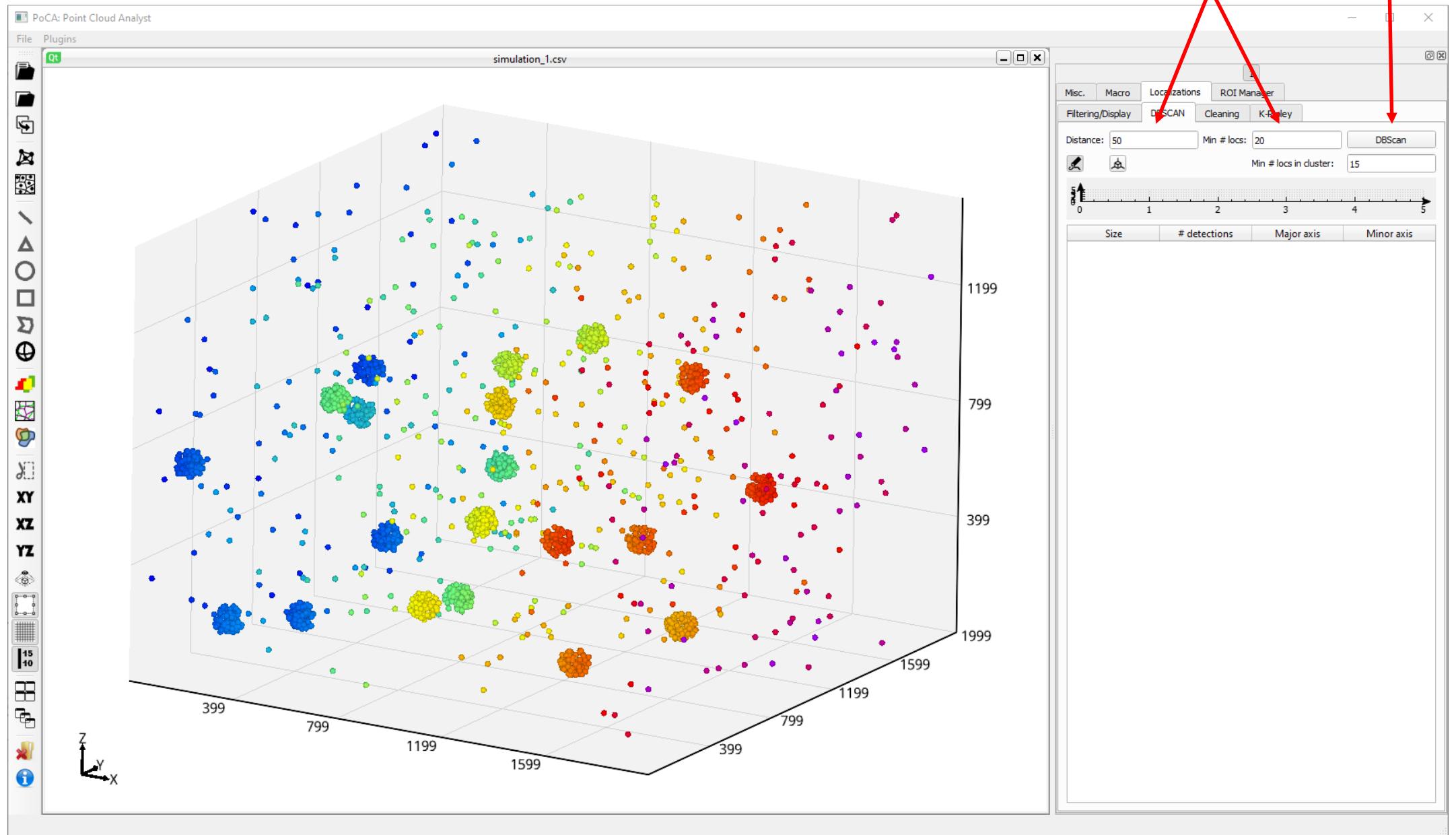
Coloc-Tesseler

Coloc-Tesseler is applied after setting the thresholds (textboxes or by clicking on the scatterplot) and clicking on « Apply »



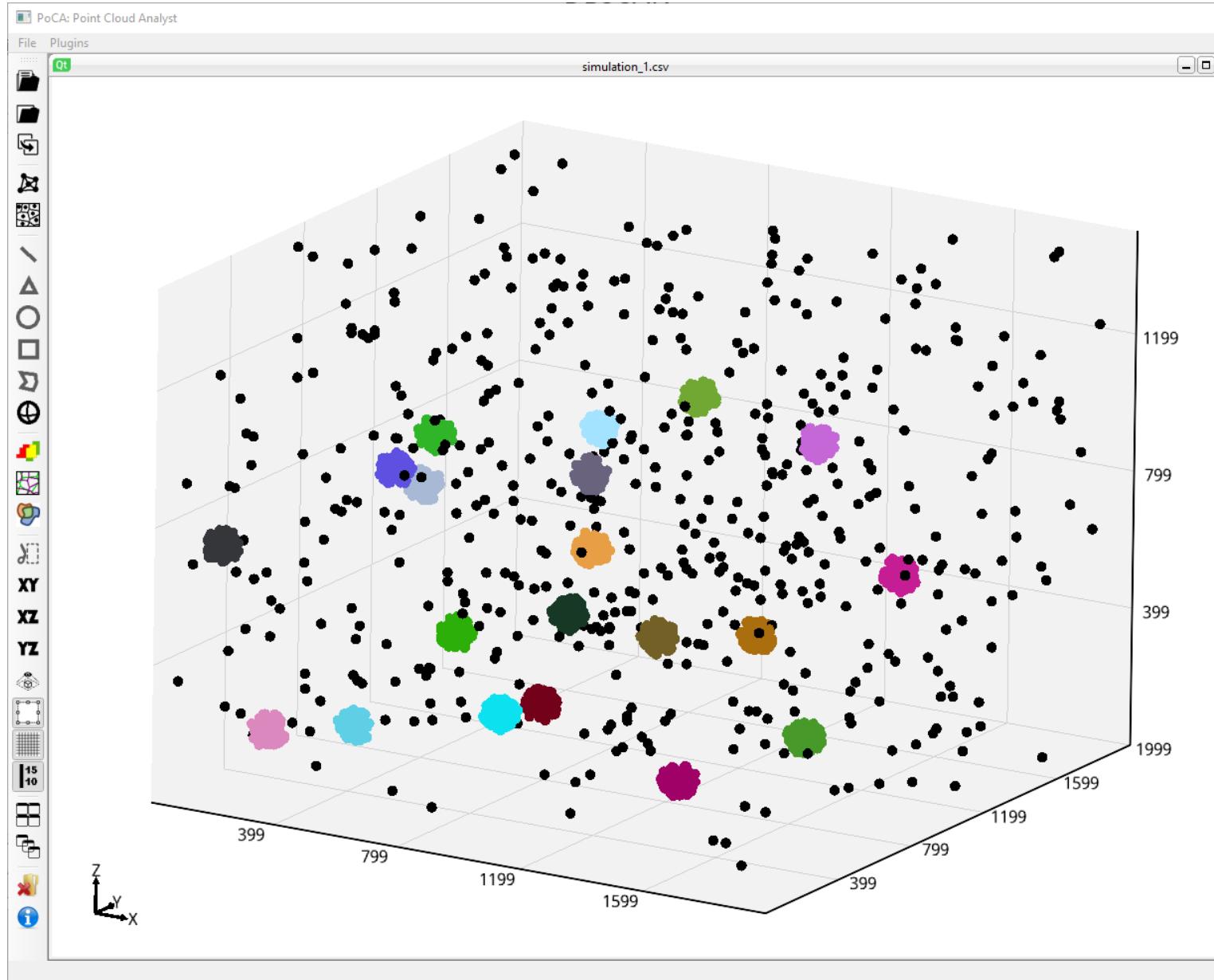
DBSCAN

In Localizations//DBSCAN



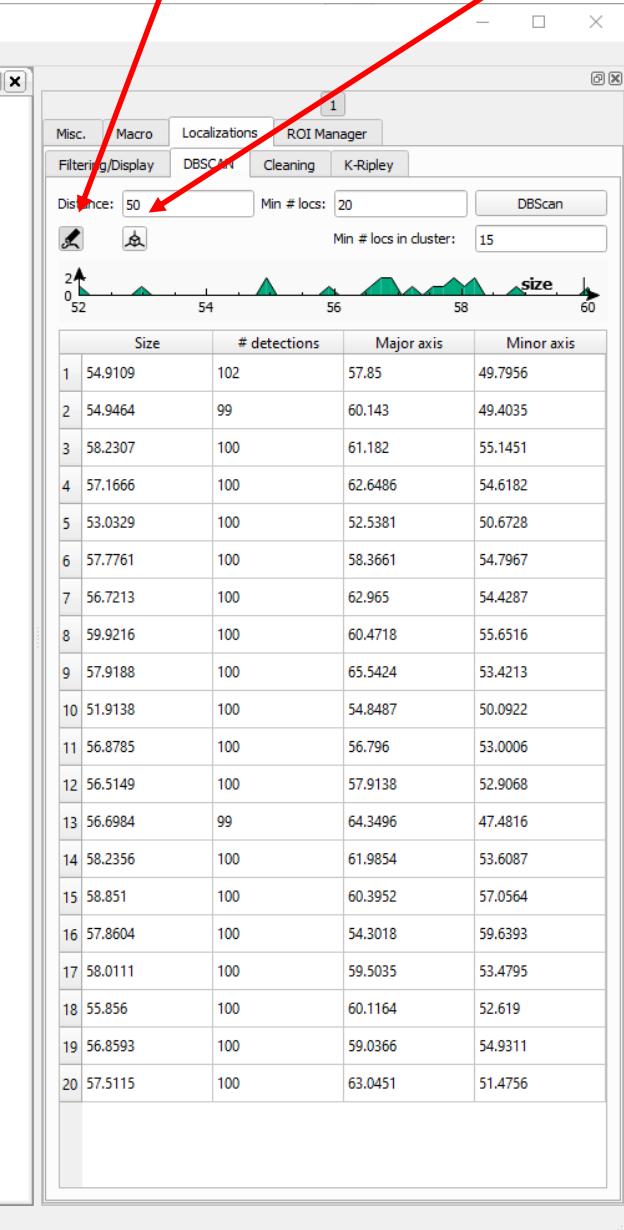
DBSCAN

DBSCAN displays is under the localization rendering, you need to switch off this one before (in Localizations\\Filtering/Display)

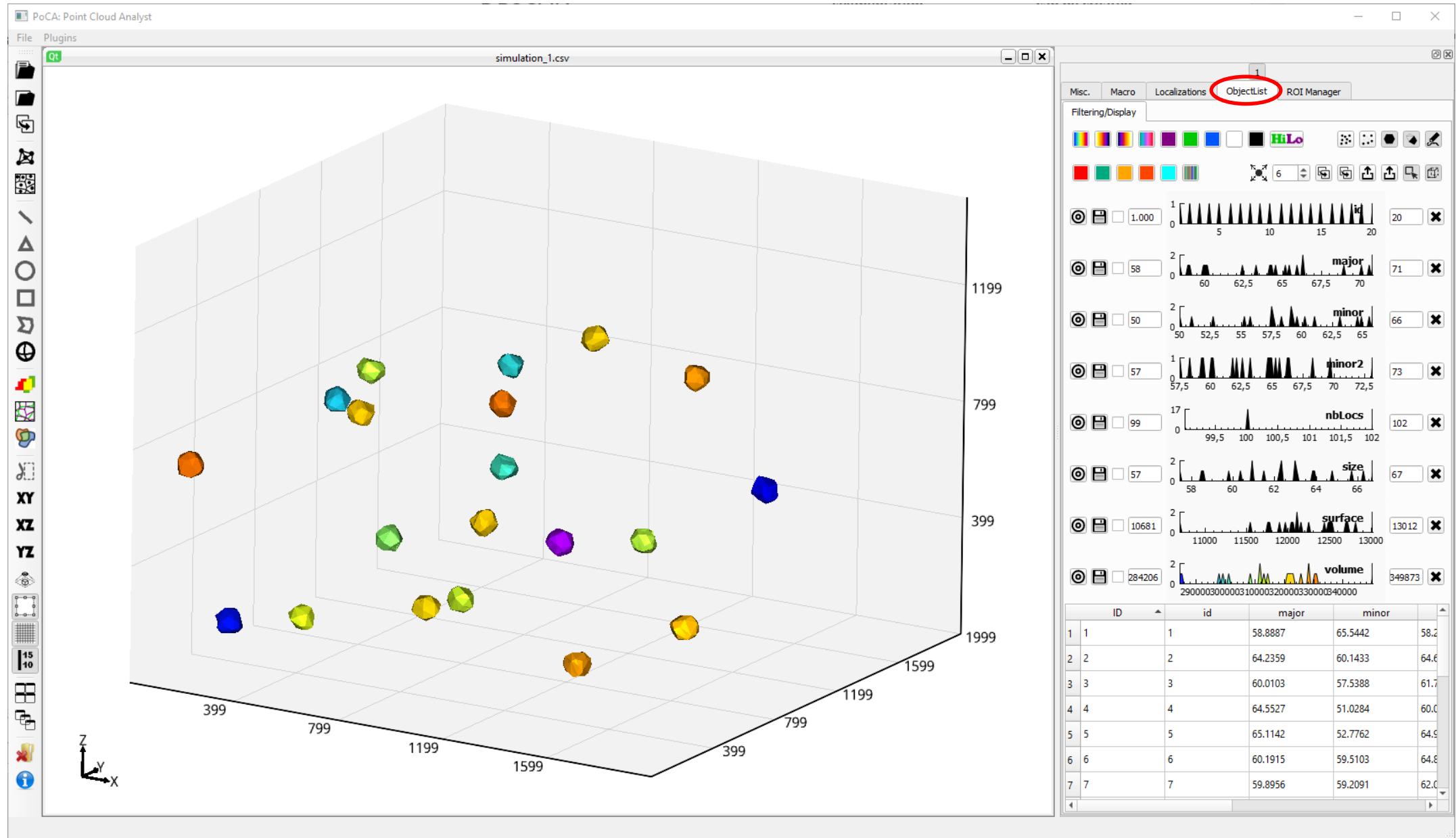


Switch DBSCAN segmentation display

PoCA's object can be created with this button

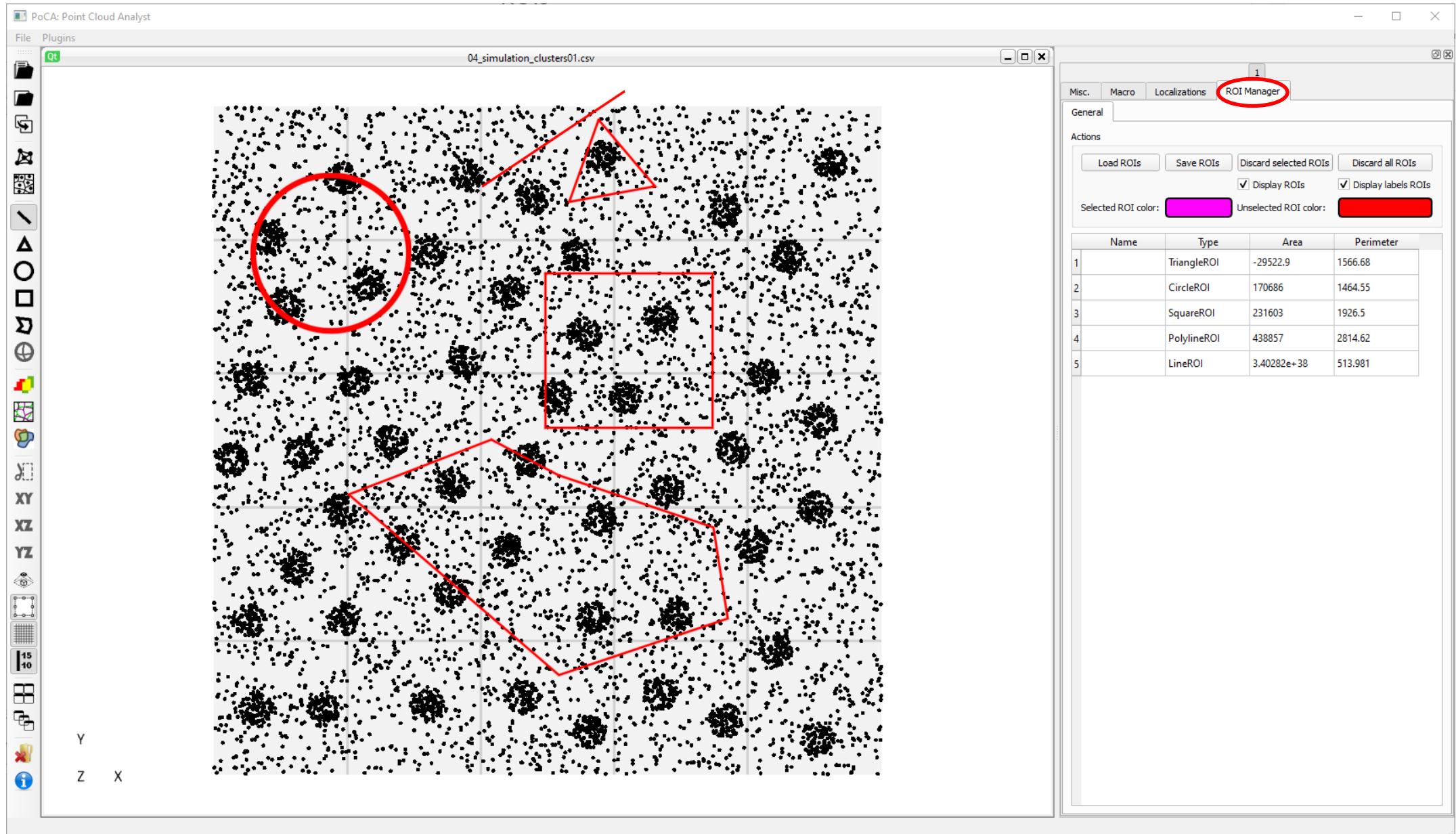


DBSCAN



ROIs – 2D

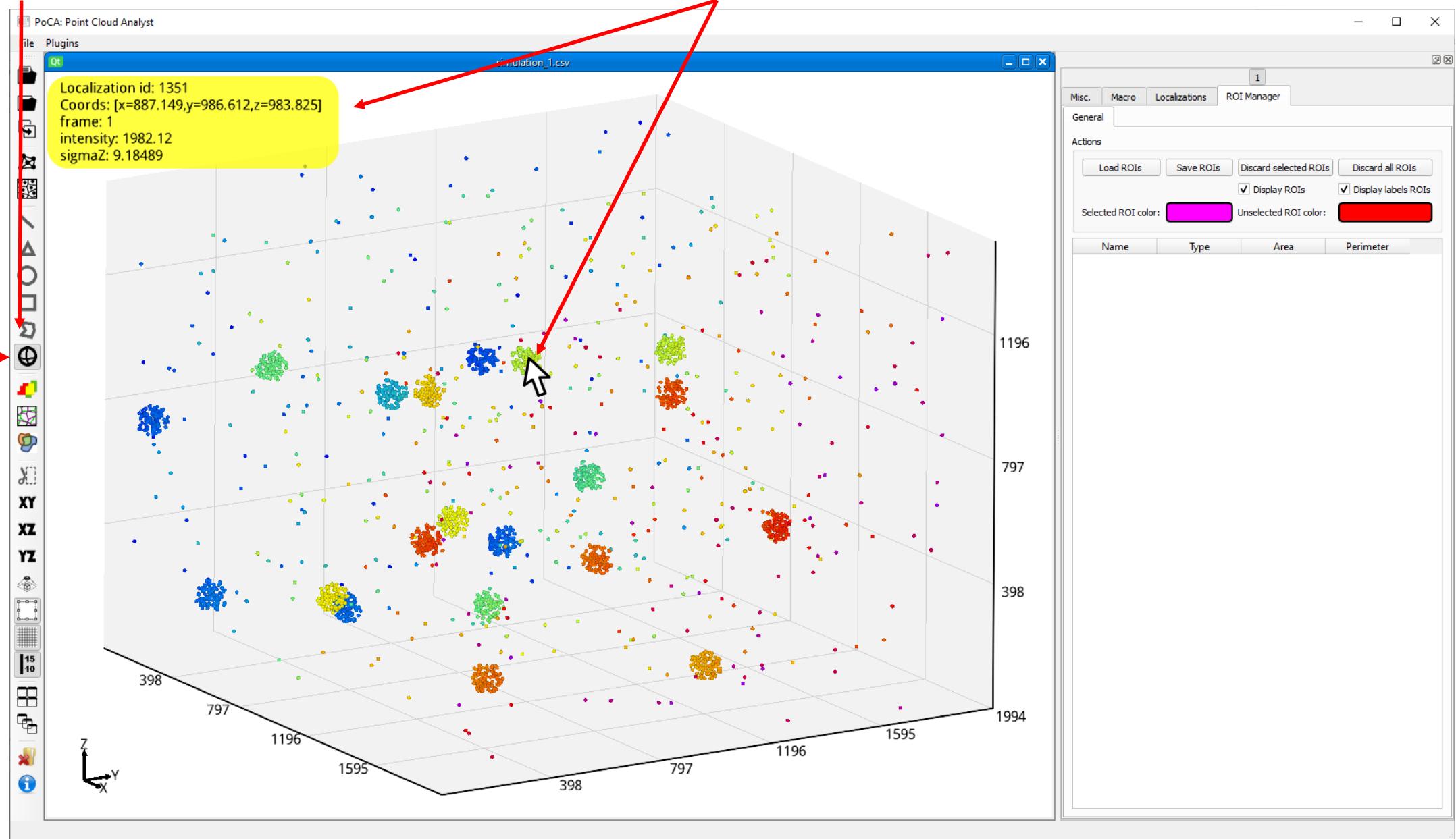
2D ROIs



ROIs – 3D

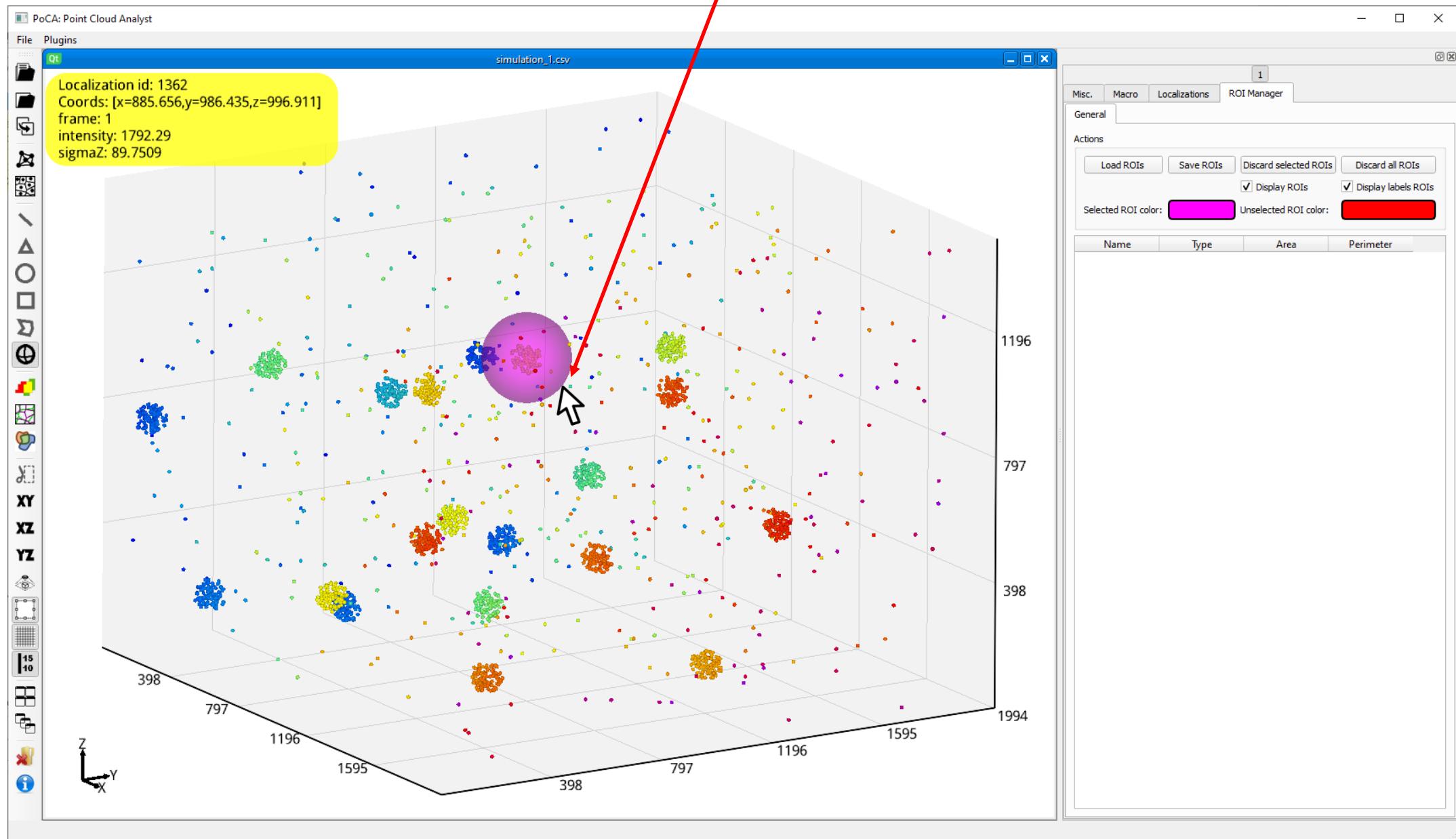
1. Select the sphere ROI button

2. Pick a localization (required step because one pixel corresponds to a line in the 3D point of view)



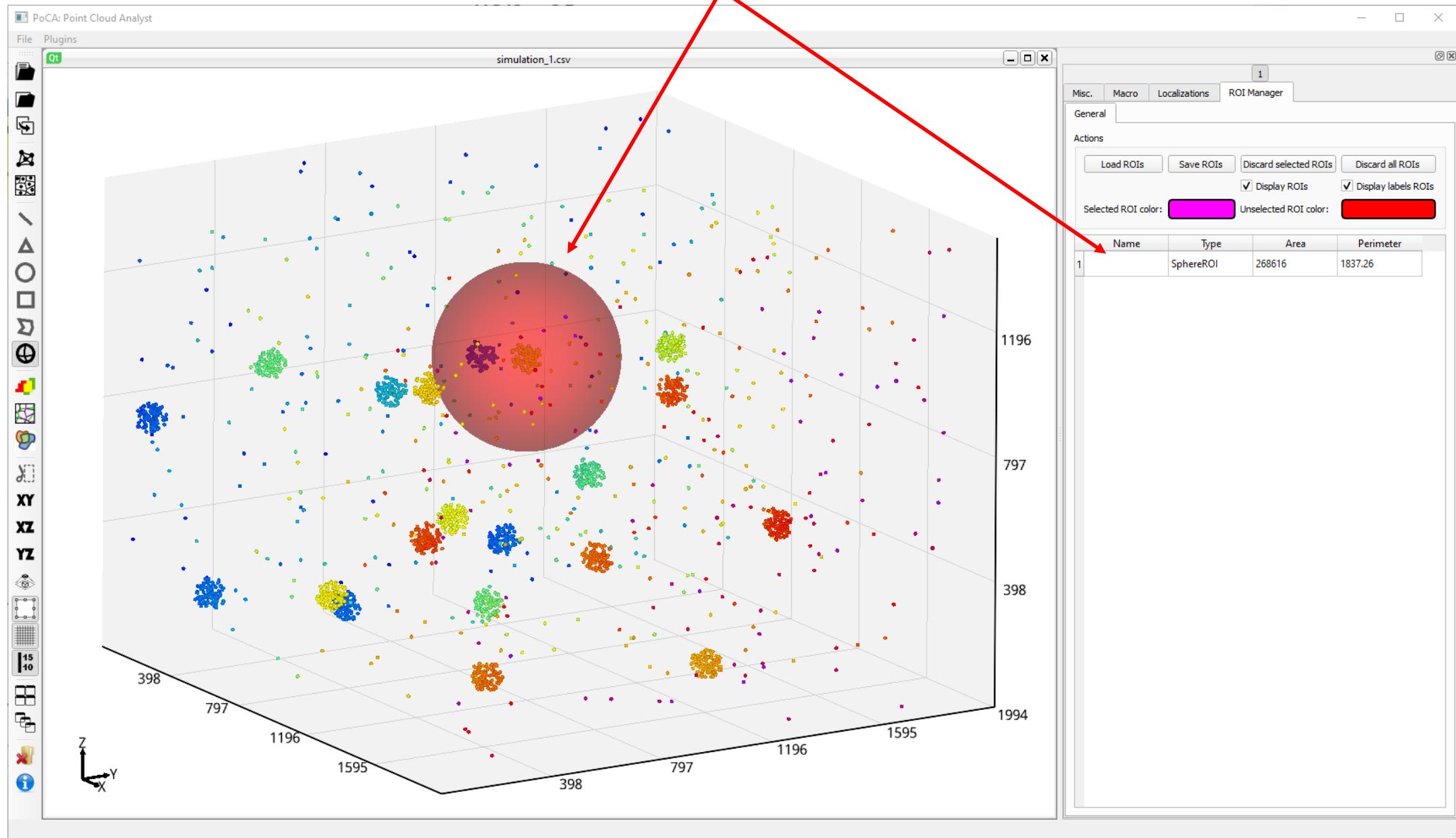
ROIs – 3D

3. Click and hold until the desired sphere ROI size



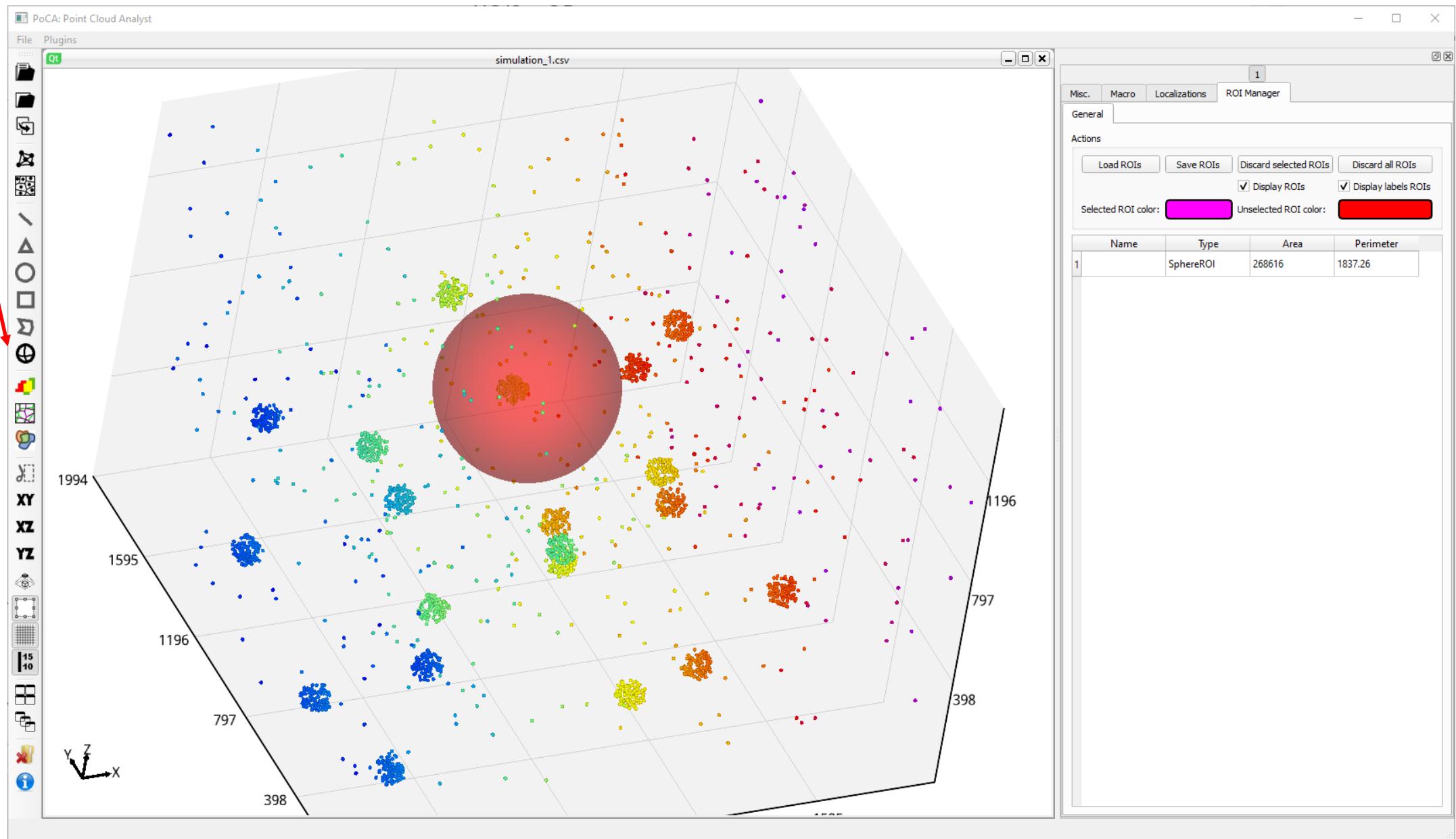
ROIs – 3D

4. Sphere ROI was created



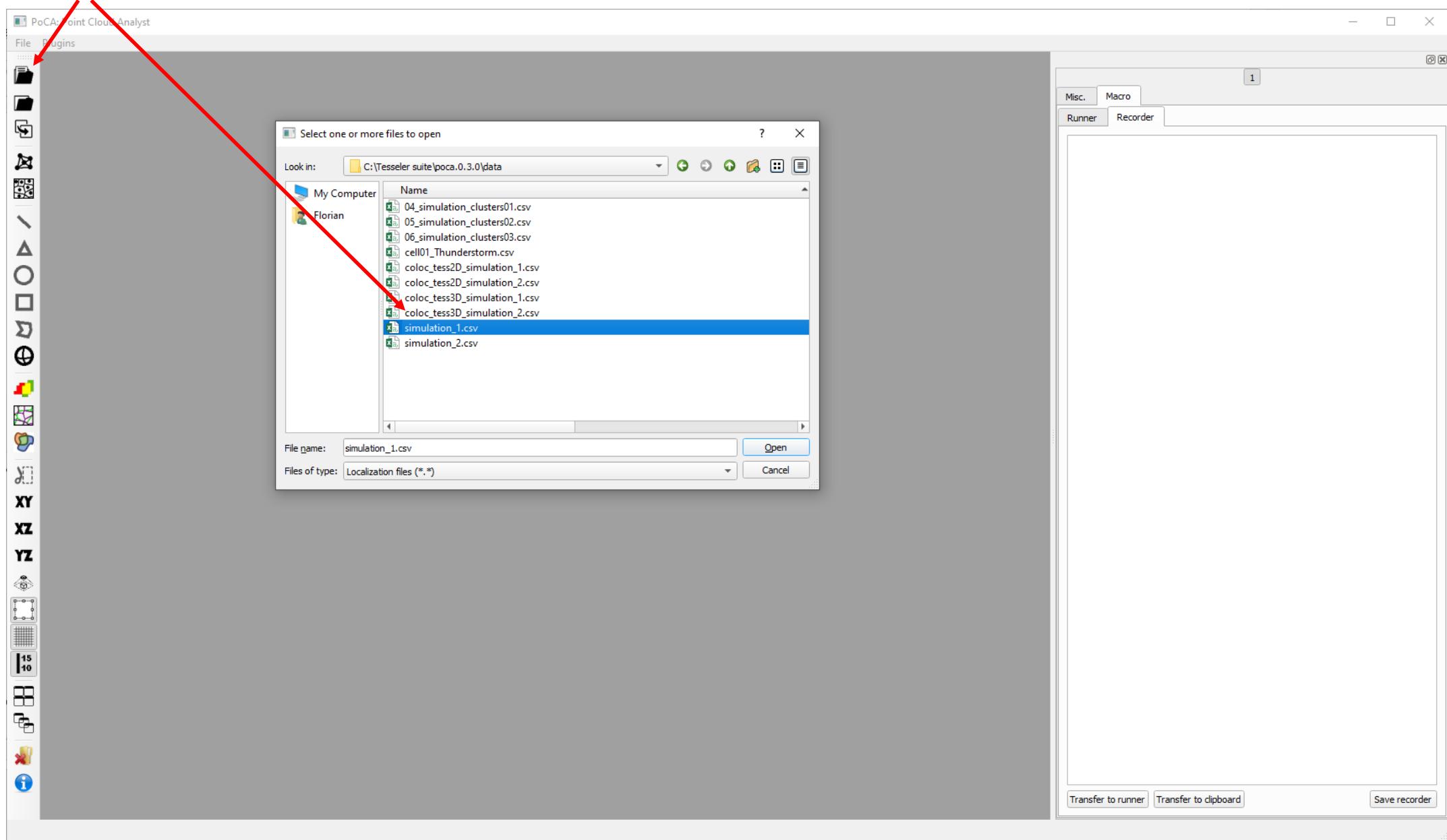
ROIs – 3D

Deselect to be able to rotate again



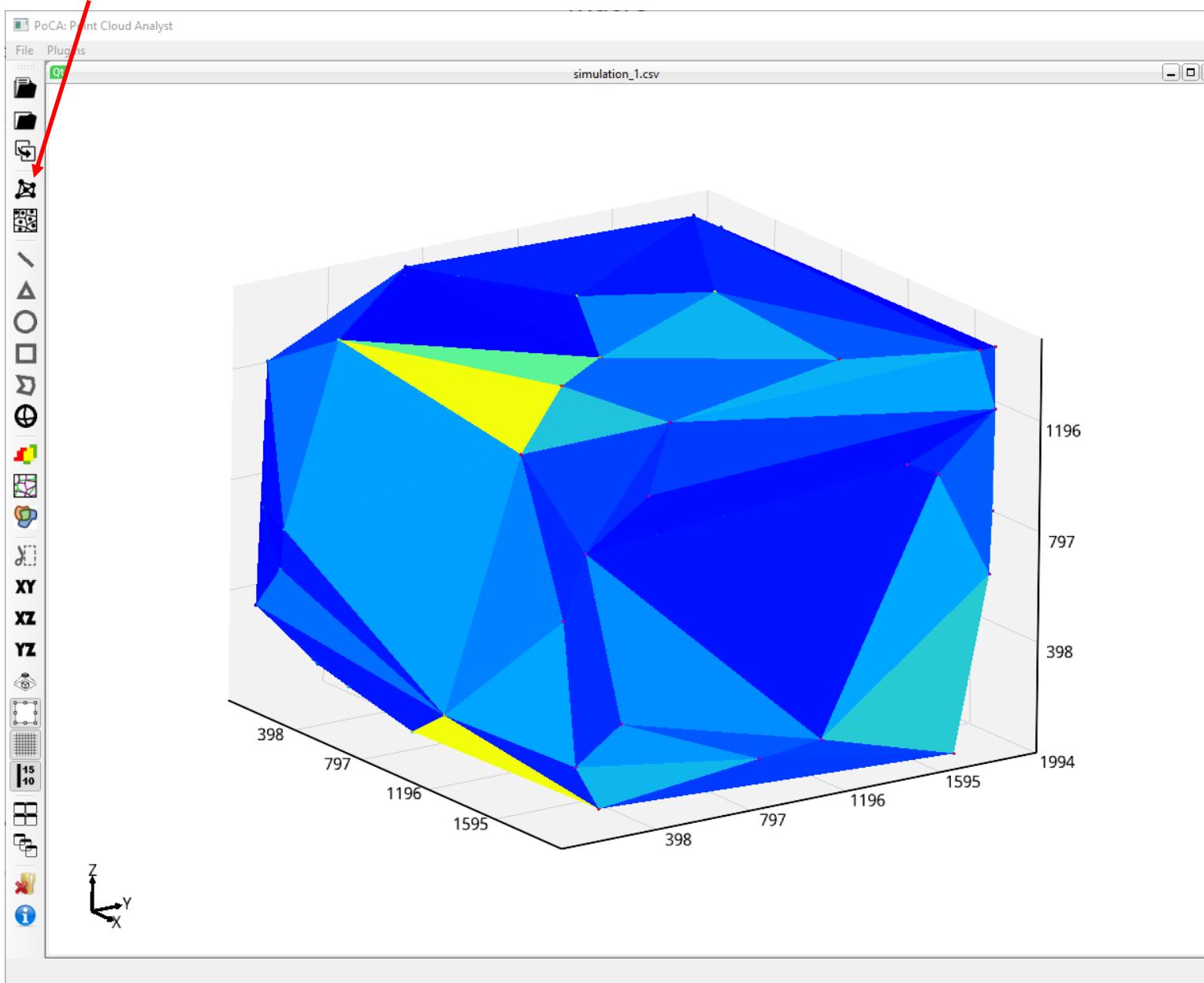
Macro

1. Open a dataset

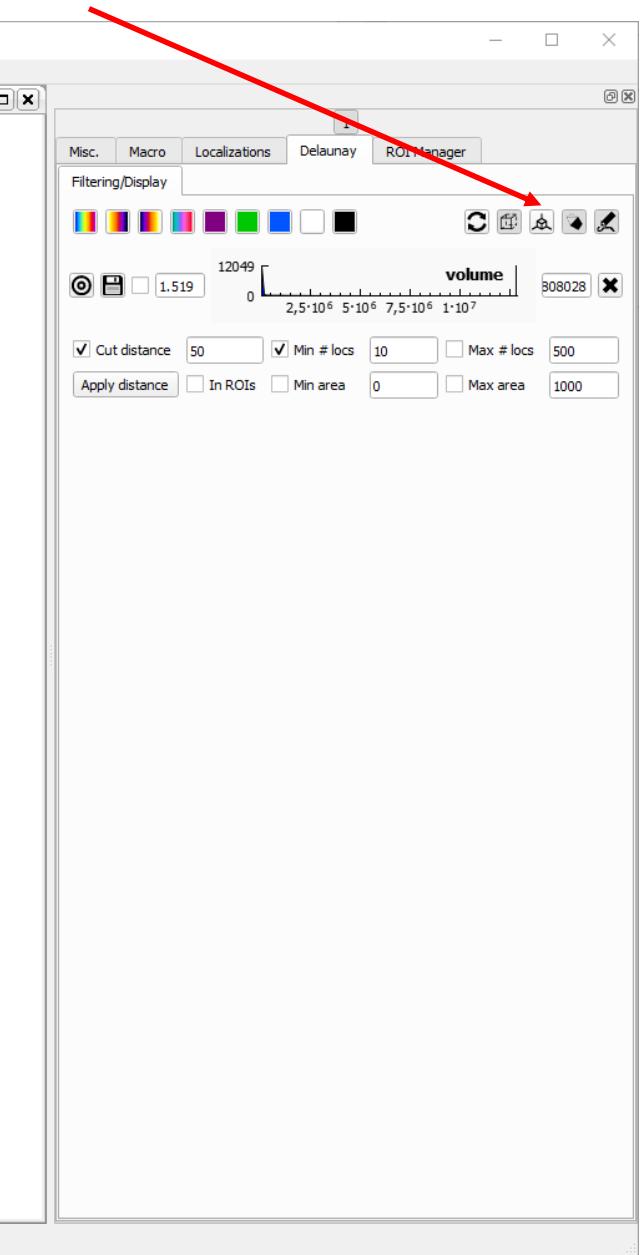


Macro

2. Create a 3D Delaunay



3. Create objects

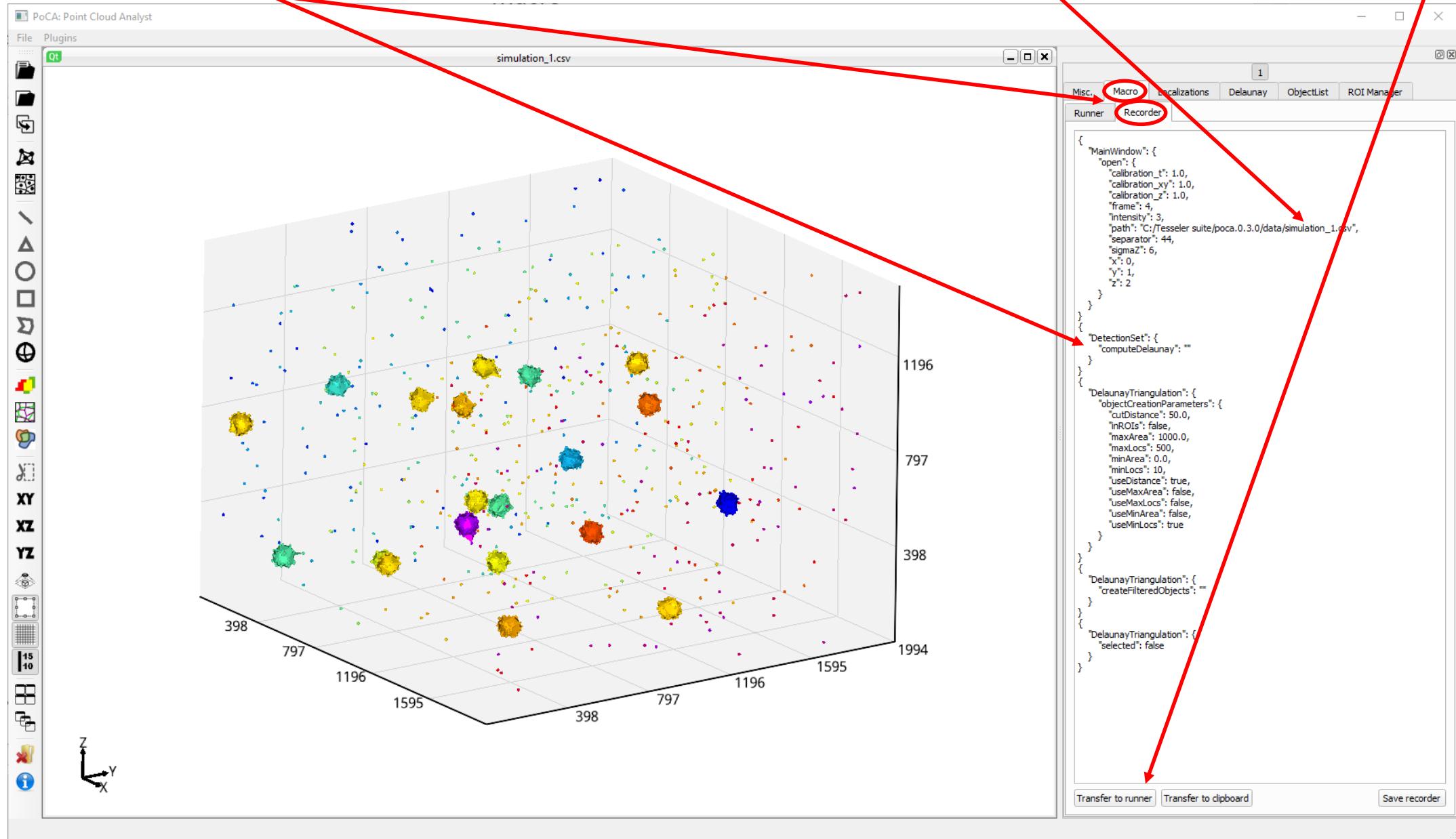


Macro

Transfer the recording to the runner

All these actions have been recorded in the Macro\\Recorder

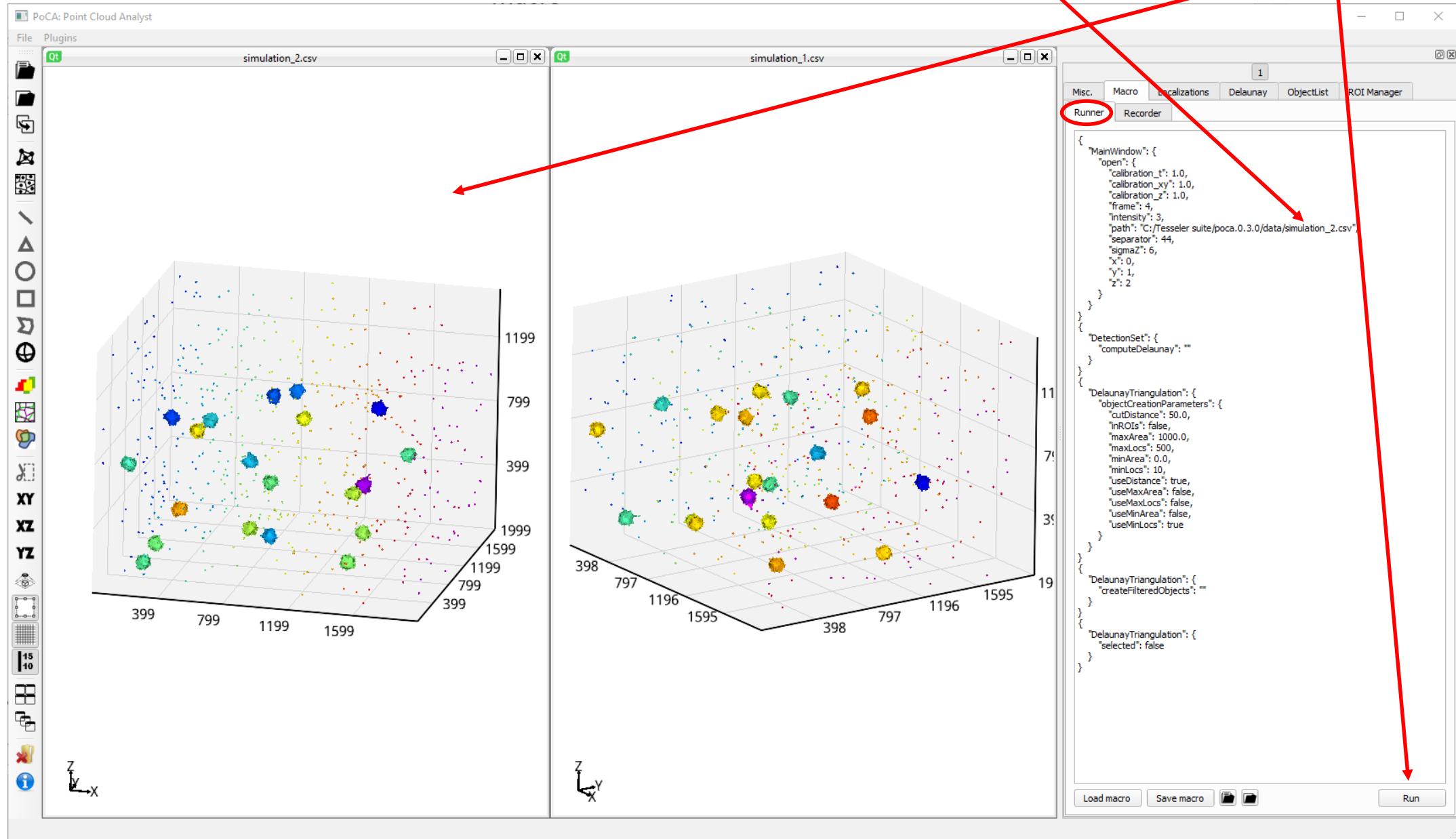
Name of the dataset was simulation_1.csv



Macro

Run macro

Change name to simulation_2.csv



Macro

The screenshot shows the PoCA software interface with two 3D point cloud windows and a Macro dialog.

- Left Window:** Titled "simulation_2.csv", it displays a 3D point cloud with colored clusters. The axes are labeled X, Y, and Z. The X-axis ranges from 399 to 1599, the Y-axis from 399 to 1199, and the Z-axis from 19 to 1199.
- Middle Window:** Titled "simulation_1.csv", it displays a 3D point cloud with colored clusters. The axes are labeled X, Y, and Z. The X-axis ranges from 398 to 1595, the Y-axis from 797 to 1196, and the Z-axis from 19 to 1199.
- Macro Dialog:** Titled "Macro", it contains a JSON configuration for a workflow. A red bracket highlights the "open" command under "MainWindow". A red arrow points from this bracket to the explanatory text below.

```
{ "MainWindow": { "open": { "calibration_t": 1.0, "calibration_xy": 1.0, "calibration_z": 1.0, "frame": 4, "intensity": 3, "path": "C:/Tesseler suite/poca.0.3.0/data/simulation_2.csv", "separator": 44, "sigmaZ": 6, "x": 0, "y": 1, "z": 2 } }, { "DetectionSet": { "computeDelaunay": "" } }, { "DelaunayTriangulation": { "objectCreationParameters": { "cutDistance": 50.0, "inROIs": false, "maxArea": 1000.0, "maxLocs": 500, "minArea": 0.0, "minLocs": 10, "useDistance": true, "useMaxArea": false, "useMaxLocs": false, "useMinArea": false, "useMinLocs": true } } }, { "DelaunayTriangulation": { "createFilteredObjects": "" } }, { "DelaunayTriangulation": { "selected": false } } }
```

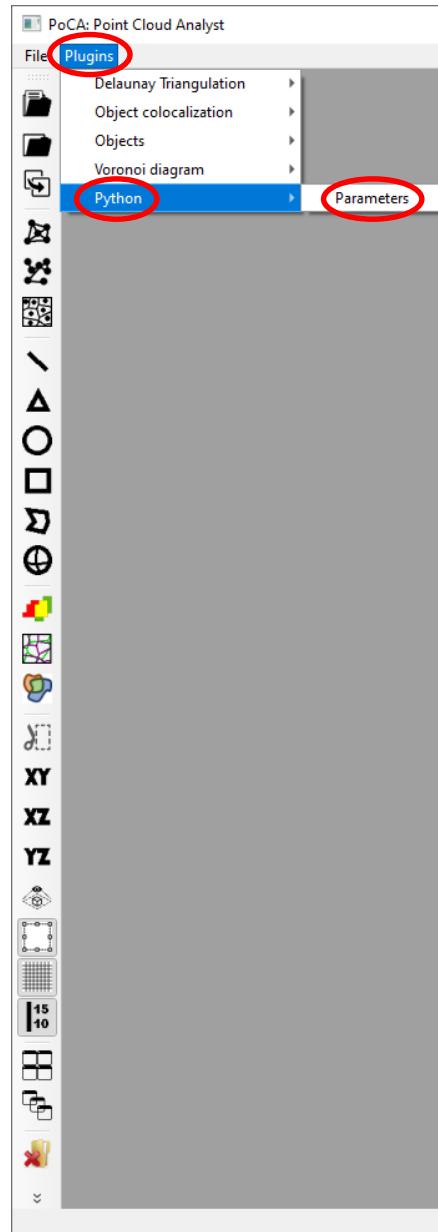
- Macro Buttons:** At the bottom of the Macro dialog, there are buttons for "Load macro", "Save macro", and "Run". A red arrow points from the "Run" button to the explanatory text below.

Require an open command as it will apply the columns' id and calibration chosen on those chosen file(s)/folder

Apply macro to a chosen file or folder

Python

1st step is to define the different paths to Python and the scripts. Go to Plugins//Python//Parameters



Use folder icons to define the path, in the example we used an environment called CAML in Anaconda

Folder to PoCA Python scripts. It is mandatory that all those scripts are in this folder

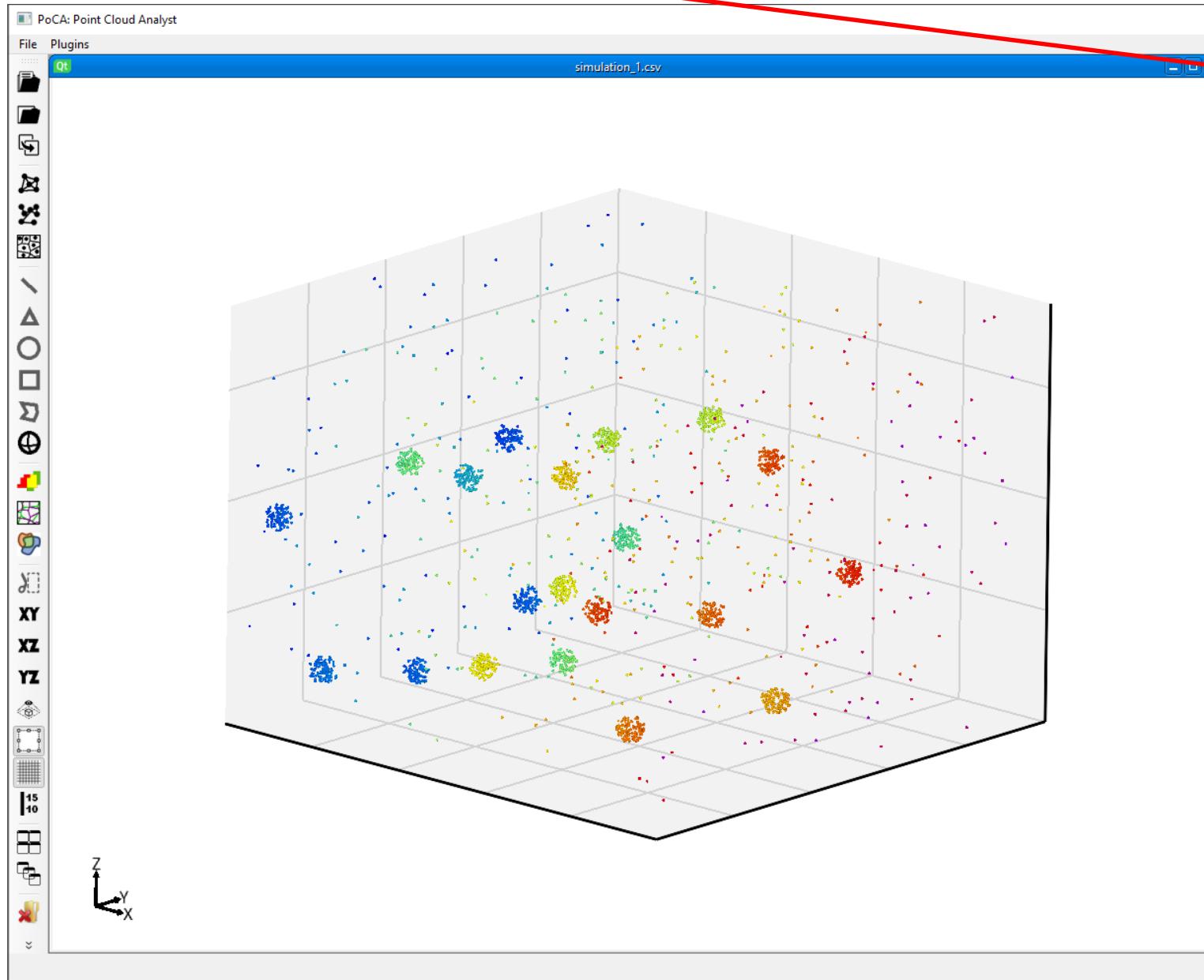
PoCA was compiled with Python 3.7.4. The Python environment used on the user's computer is therefore required to be version 3.7.4

```
        "open": {
            "calibration_t": 1.0,
            "calibration_xy": 1.0,
            "calibration_z": 1.0,
            "frame": 4,
            "intensity": 3,
            "path": "C:/DevC++/poca/bin/poca/simulation_1.csv",
            "separat": 44,
            "sigmaz": 6,
            "x": 0,
            "y": 1,
            "z": 2
        }
    }
    "DetectionSet": {
        "computeDelaunay": ""
    }
}
{
    "DelaunayTriangulation": {
        "objectCreationParameters": {
            "cutDistance": 50.0,
            "inROIs": false,
            "maxArea": 1000.0,
            "maxLocs": 500,
            "minArea": 0.0,
            "minLocs": 10,
            "useDistance": true,
            "useMaxArea": false,
            "useMaxLocs": false,
            "useMinArea": false,
            "useMinLocs": true
        }
    }
}
{
    "DelaunayTriangulation": {
        "createFilteredObjects": ""
    }
}
{
    "DelaunayTriangulation": {
        "selected": false
    }
}
```

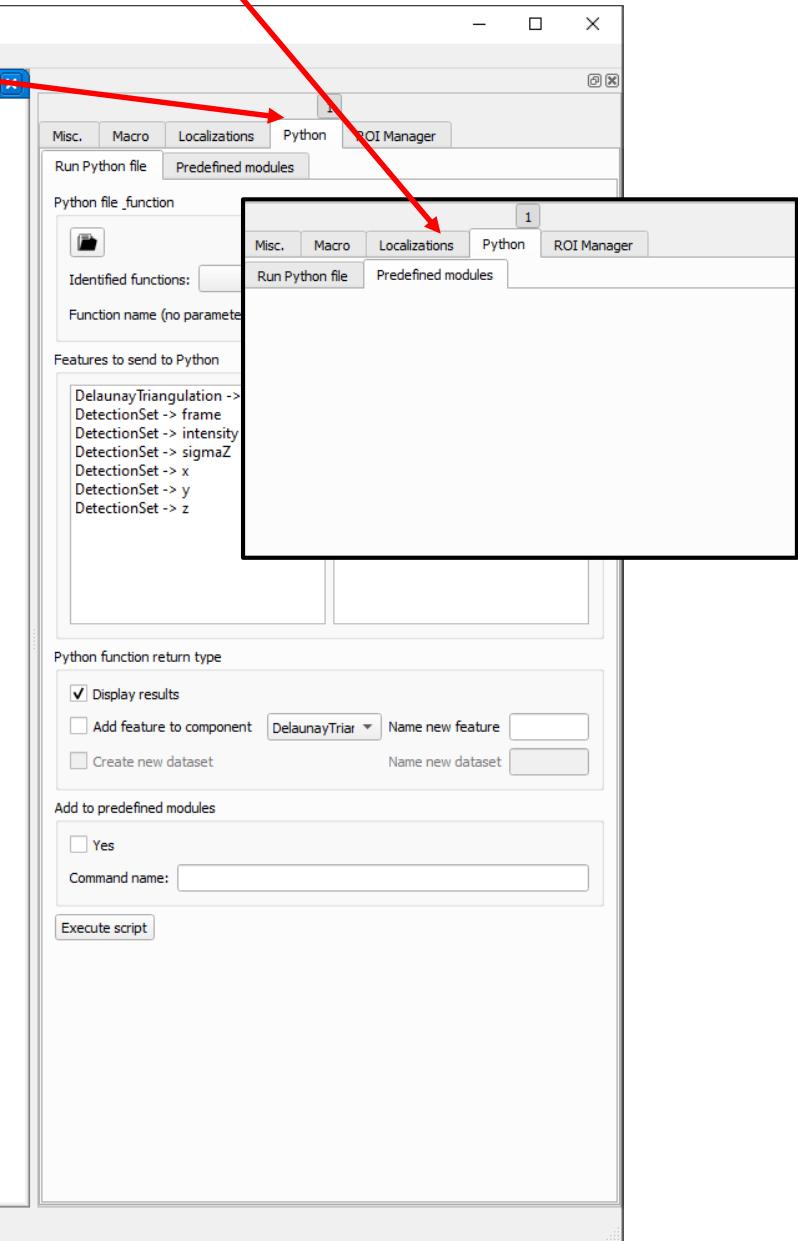
Load macro Save macro Run

Python

After loading a dataset, the Python tab is available

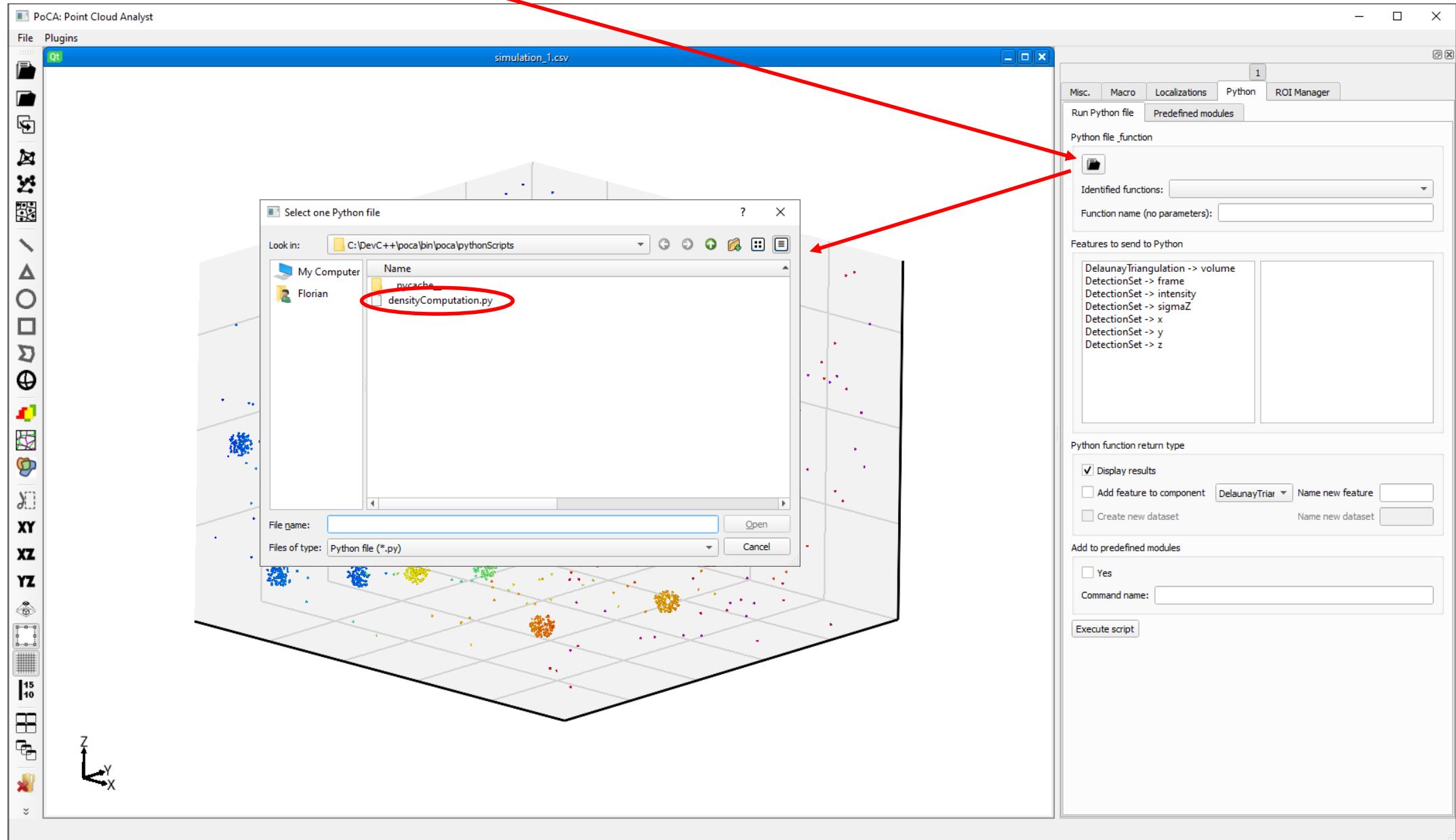


Predefined modules tab is currently empty



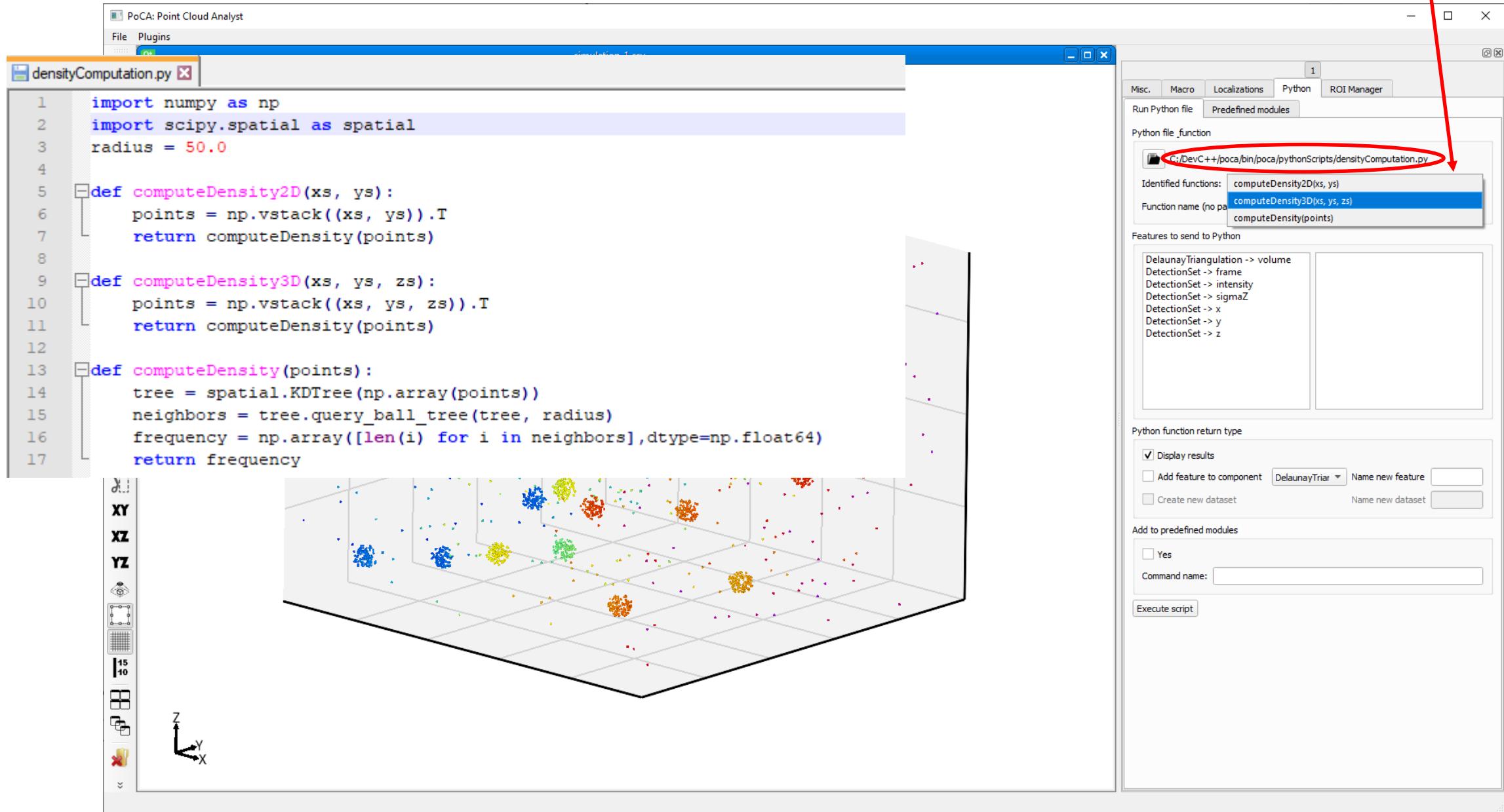
Python

Open the « densityComputation.py » Python file



Python

PoCA automatically identifies function name if the structure is « def funcName: »



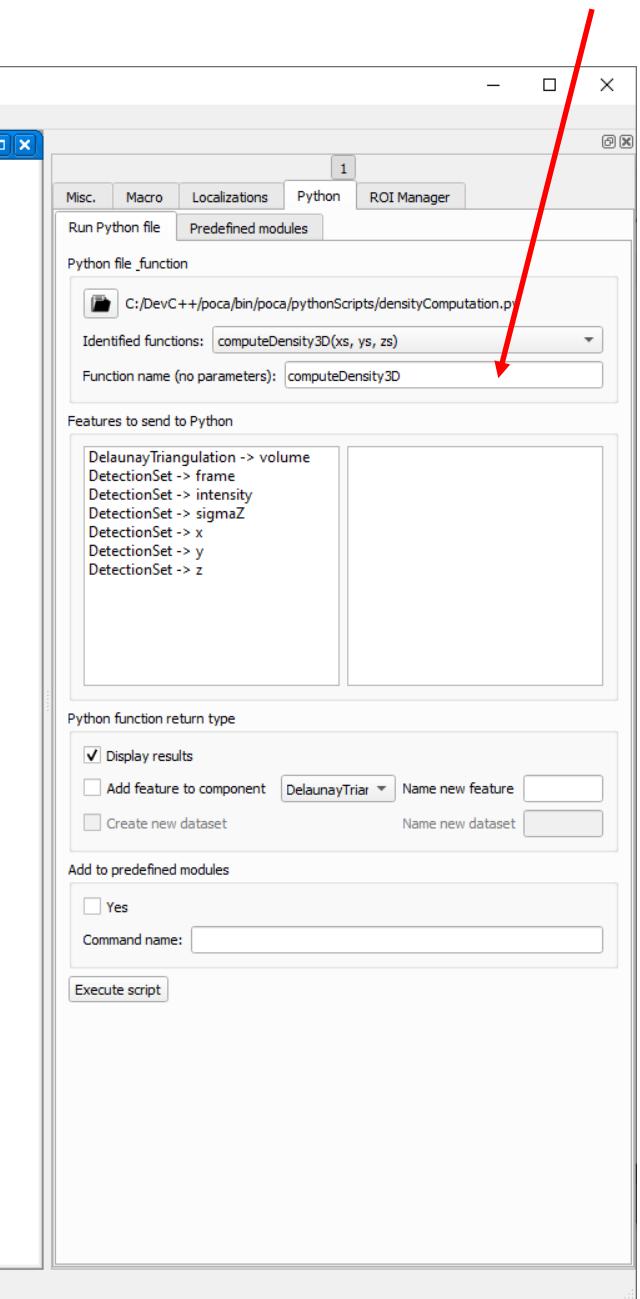
Currently, PoCA can only transfer 1 dimension arrays to Python
So only computeDensity2D & computeDensity3D can be used
computeDensity is an internal function that have a multidimensional array as input

Python

The screenshot shows the PoCA software interface. On the left, there is a 3D point cloud visualization window showing a grid-based spatial distribution of points in a 3D space. A vertical toolbar on the far left provides various viewing options like XY, XZ, YZ, and camera controls. In the center, a code editor window titled "densityComputation.py" displays the following Python script:

```
1 import numpy as np
2 import scipy.spatial as spatial
3 radius = 50.0
4
5 def computeDensity2D(xs, ys):
6     points = np.vstack((xs, ys)).T
7     return computeDensity(points)
8
9 def computeDensity3D(xs, ys, zs):
10    points = np.vstack((xs, ys, zs)).T
11    return computeDensity(points)
12
13 def computeDensity(points):
14     tree = spatial.KDTree(np.array(points))
15     neighbors = tree.query_ball_tree(tree, radius)
16     frequency = np.array([len(i) for i in neighbors], dtype=np.float64)
17     return frequency
```

Choosing a function automatically fills the function name
If the function was not in the list, it can be filled by the user in the text edit



Select « add feature to component »

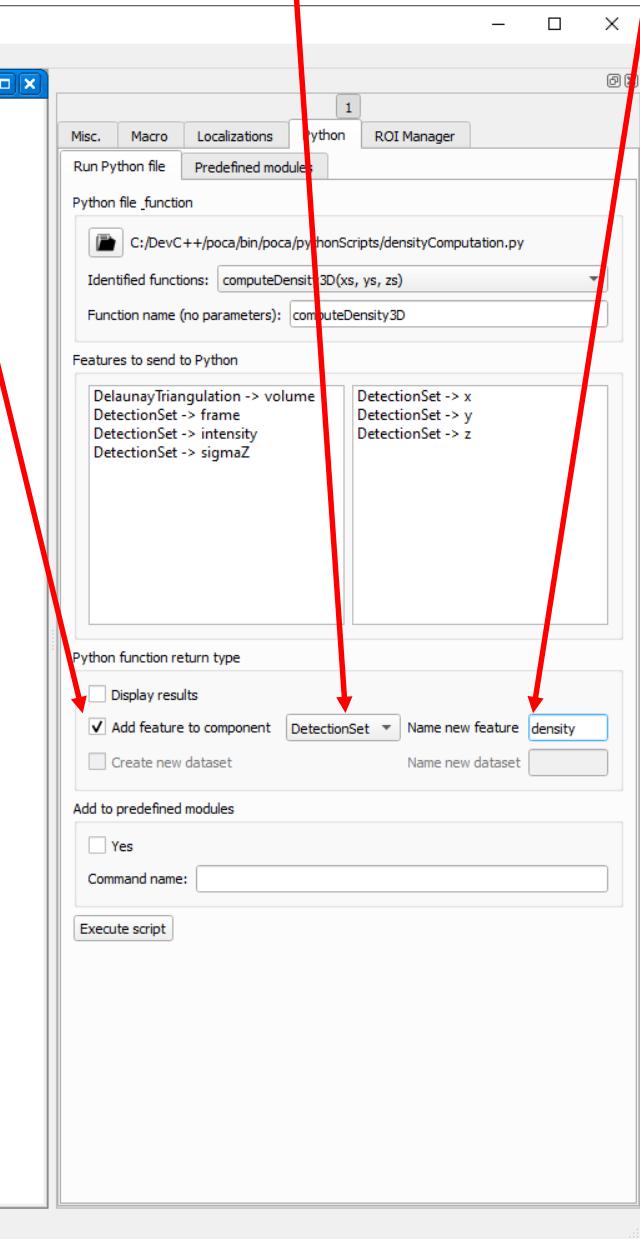
Name the added feature

Python

computeDensity2D & computeDensity3D both return 1 dimensional array (the computed density)

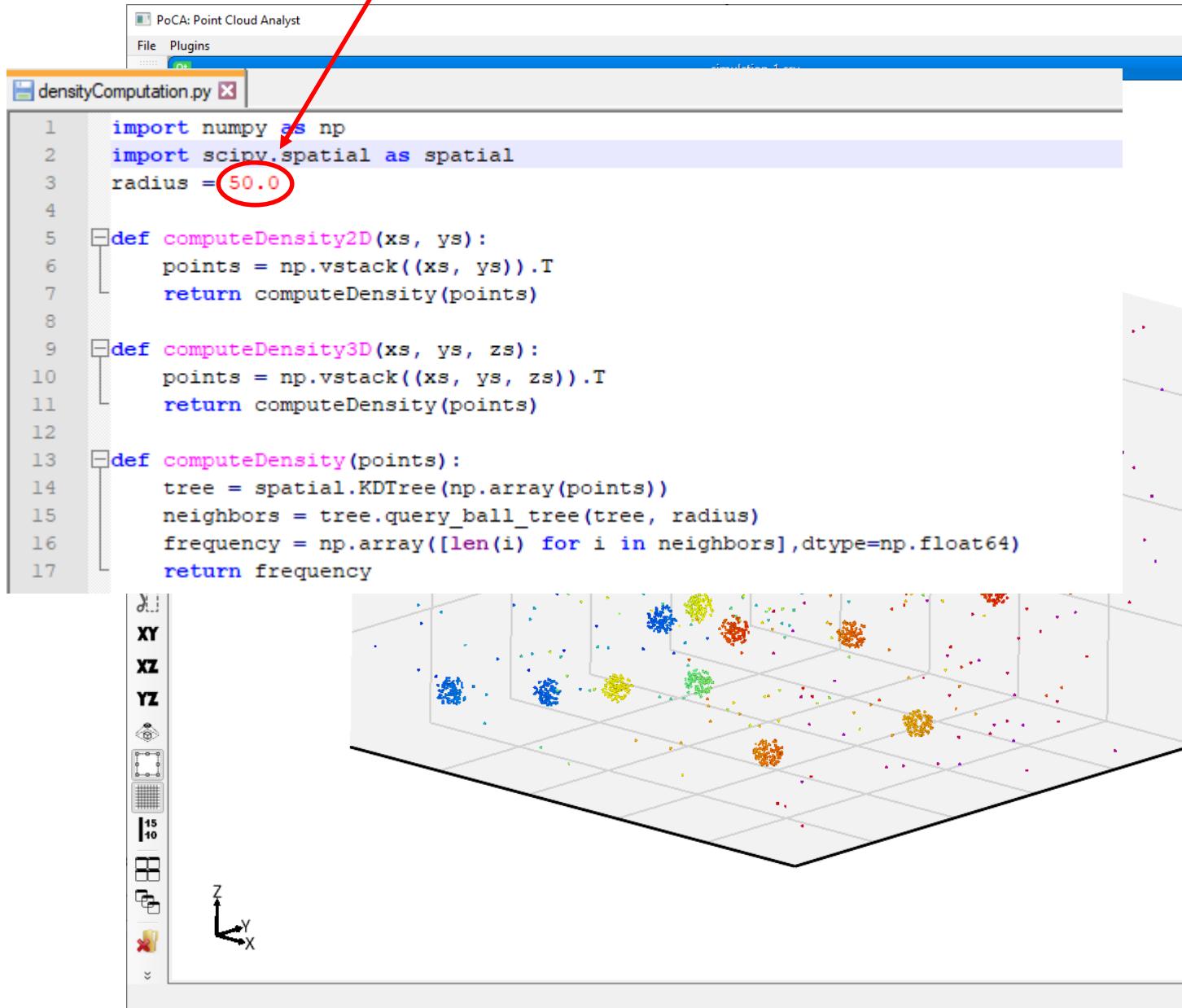
```
1 import numpy as np
2 import scipy.spatial as spatial
3 radius = 50.0
4
5 def computeDensity2D(xs, ys):
6     points = np.vstack((xs, ys)).T
7     return computeDensity(points)
8
9 def computeDensity3D(xs, ys, zs):
10    points = np.vstack((xs, ys, zs)).T
11    return computeDensity(points)
12
13 def computeDensity(points):
14     tree = spatial.KDTree(np.array(points))
15     neighbors = tree.query_ball_tree(tree, radius)
16     frequency = np.array([len(i) for i in neighbors], dtype=np.float64)
17     return frequency
```

Select the correct component



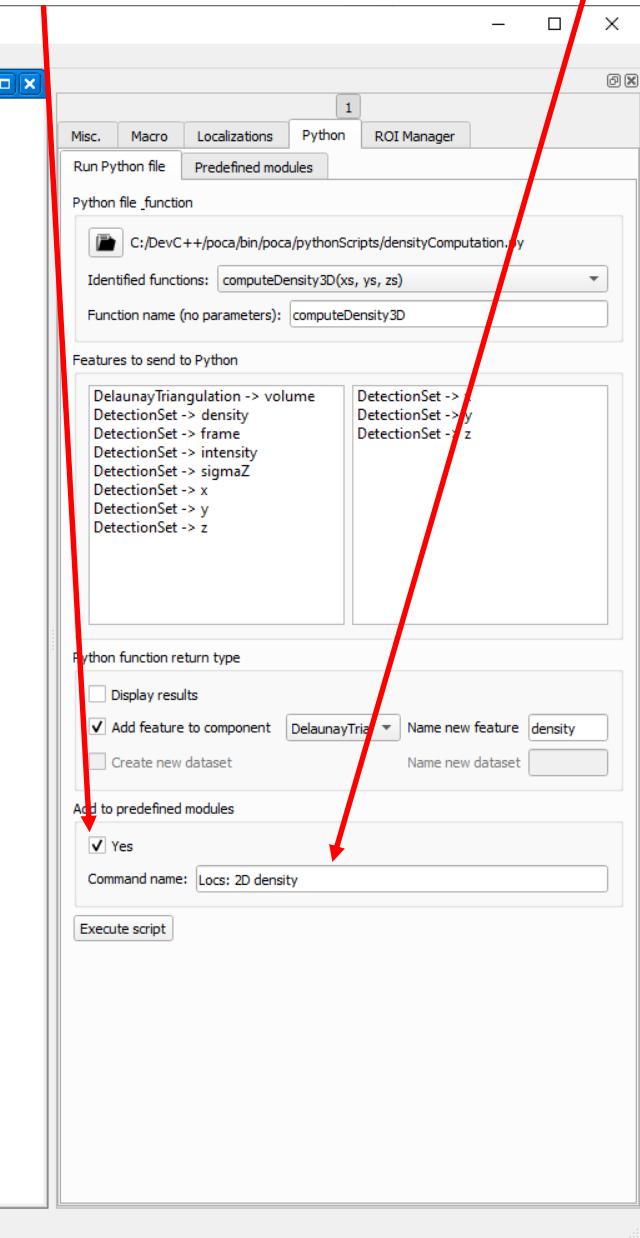
ComputeDensity computes, for each loc, the number of points in its vicinity defined by radius

Radius is hard-coded in the script



Any Python script can be added as predefined module, check « Yes »

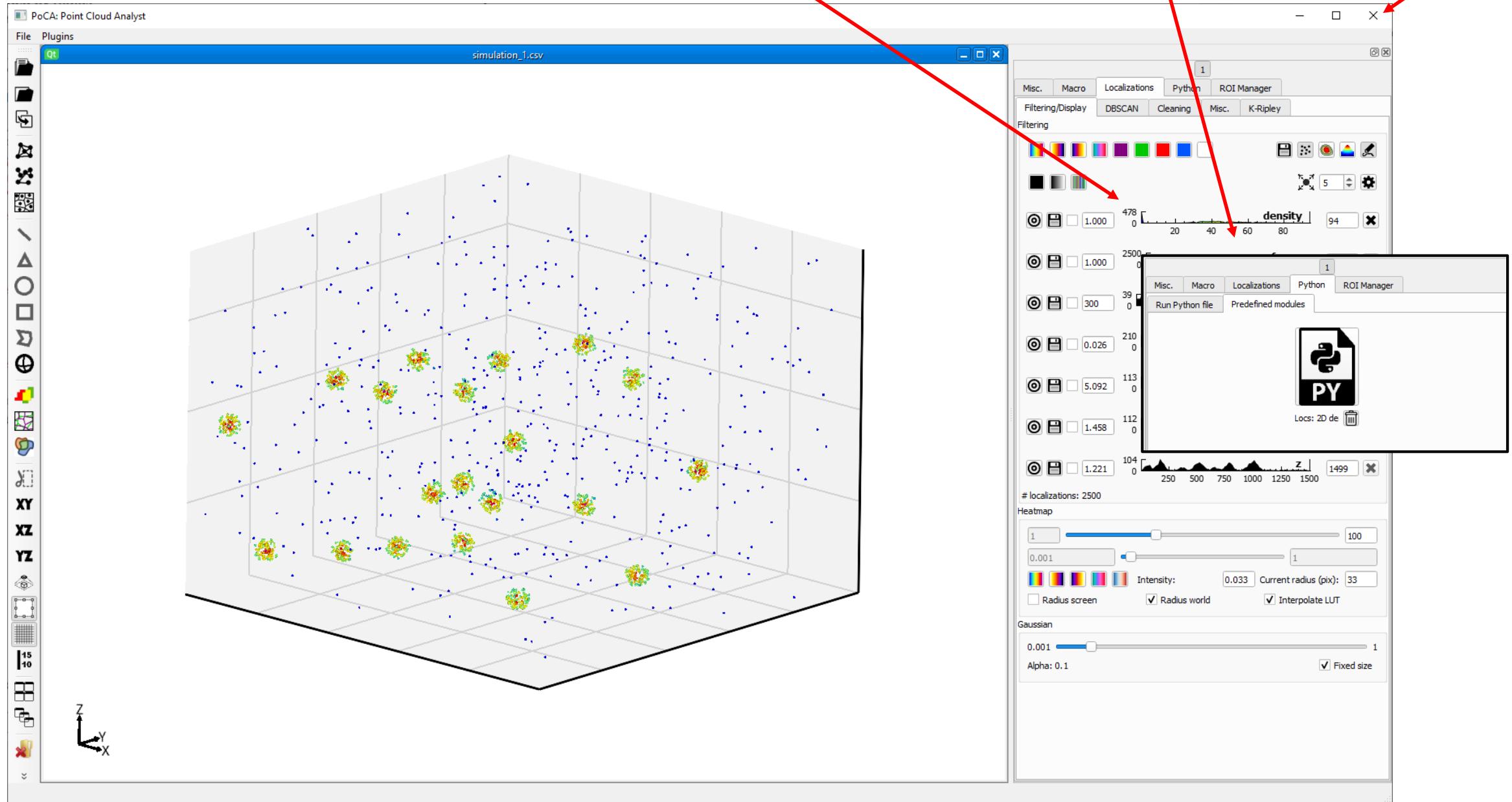
Give a name to the Python command



Python

Computed density have been added
to the localizations

The script has been added as a predefined
module. It should be loaded on the next PoCA
lauch (if PoCA was properly closed ; with the
cross)



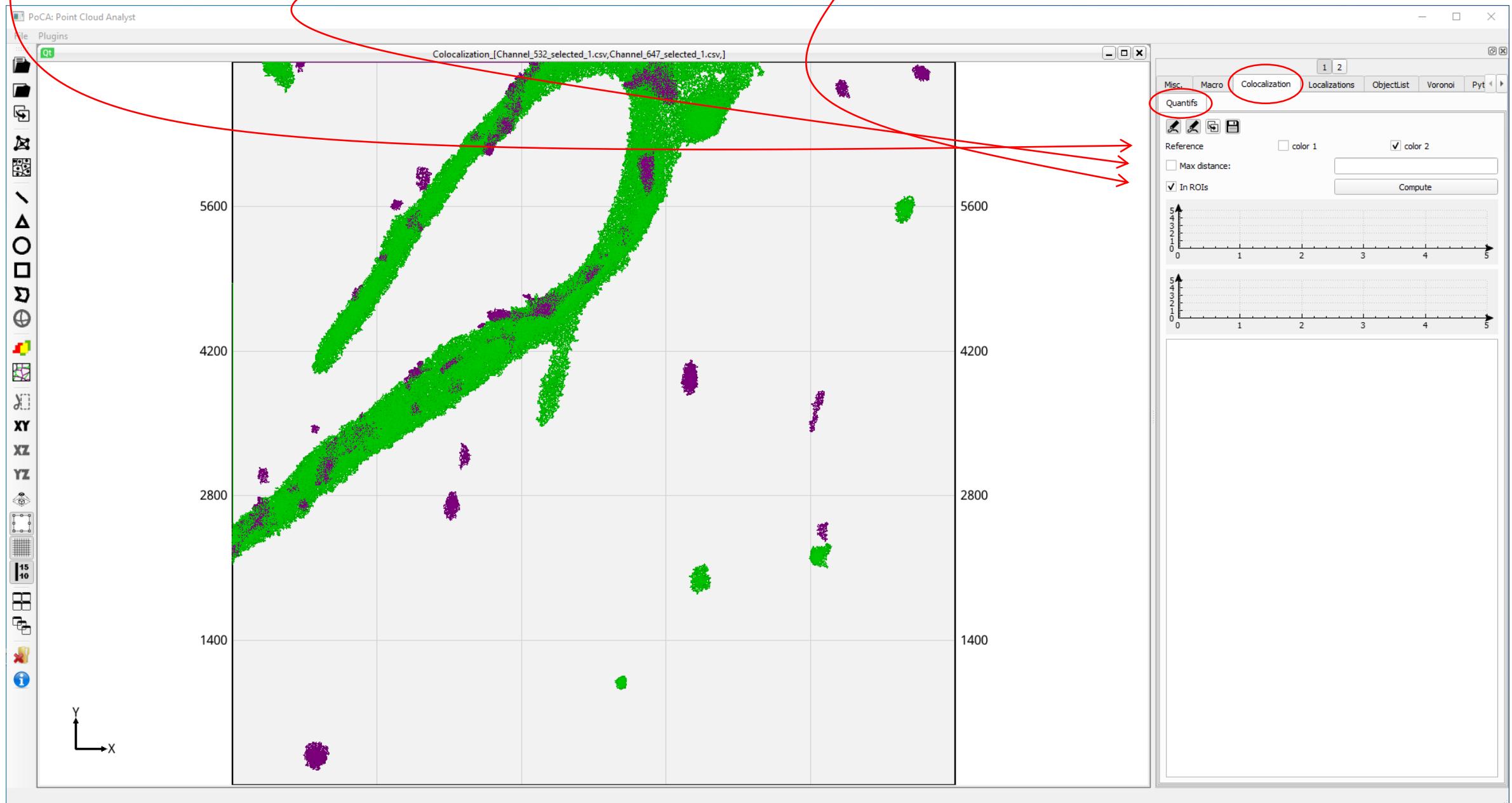
Nearest localizations multicolor plugin

You can use it after segmenting the 2 colors and creation a colocalization object

Reference defines the color acting as reference: the distance computation will be done between the objects (centroids & outline) of the other color and the outline of the reference

Max distance checked: only keeps objects of the other color that have distances (both centroid & outline) to the reference below the limit

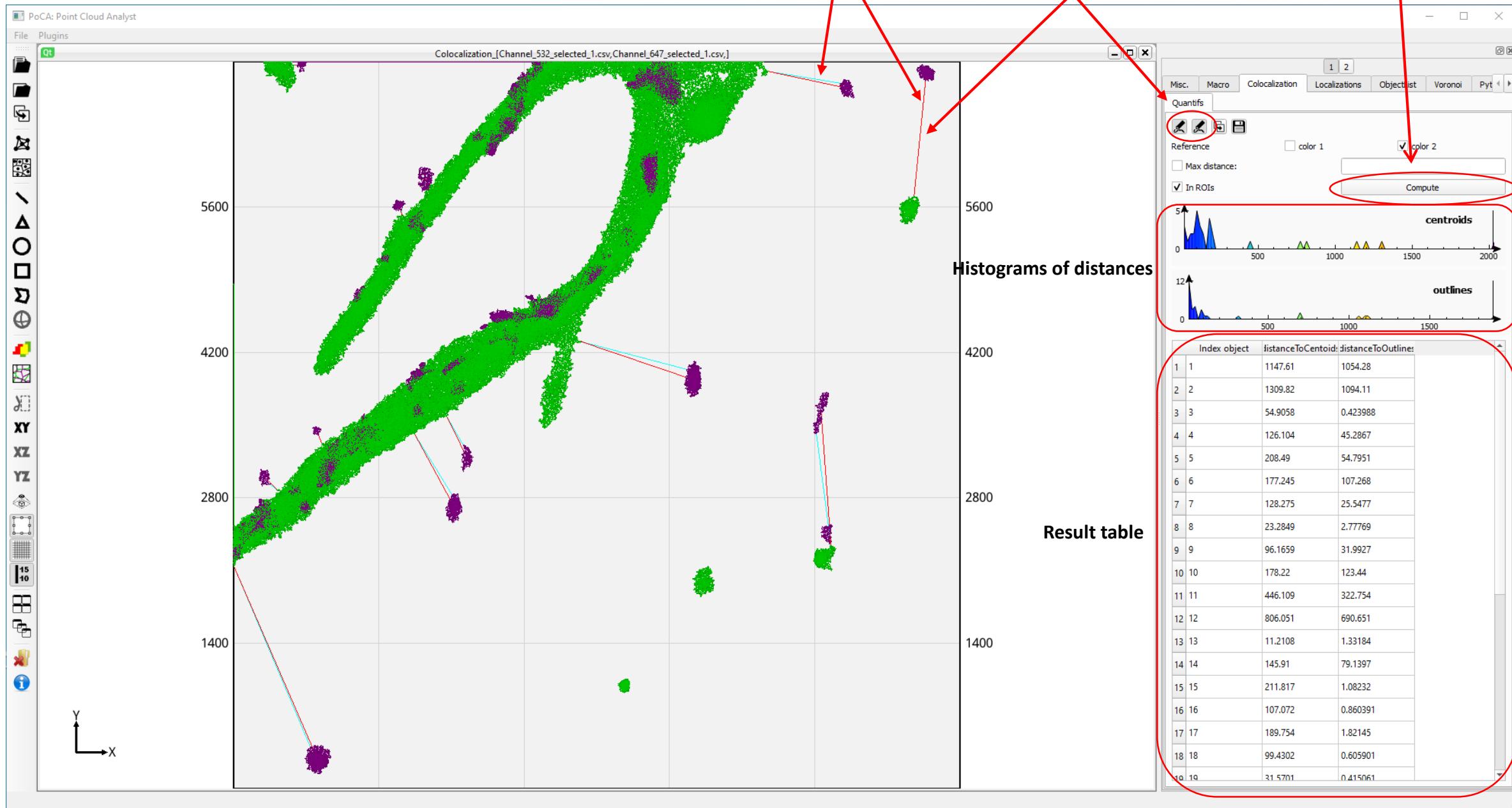
In ROIs checked: keep only objects inside the ROIs if some ROIs are defined



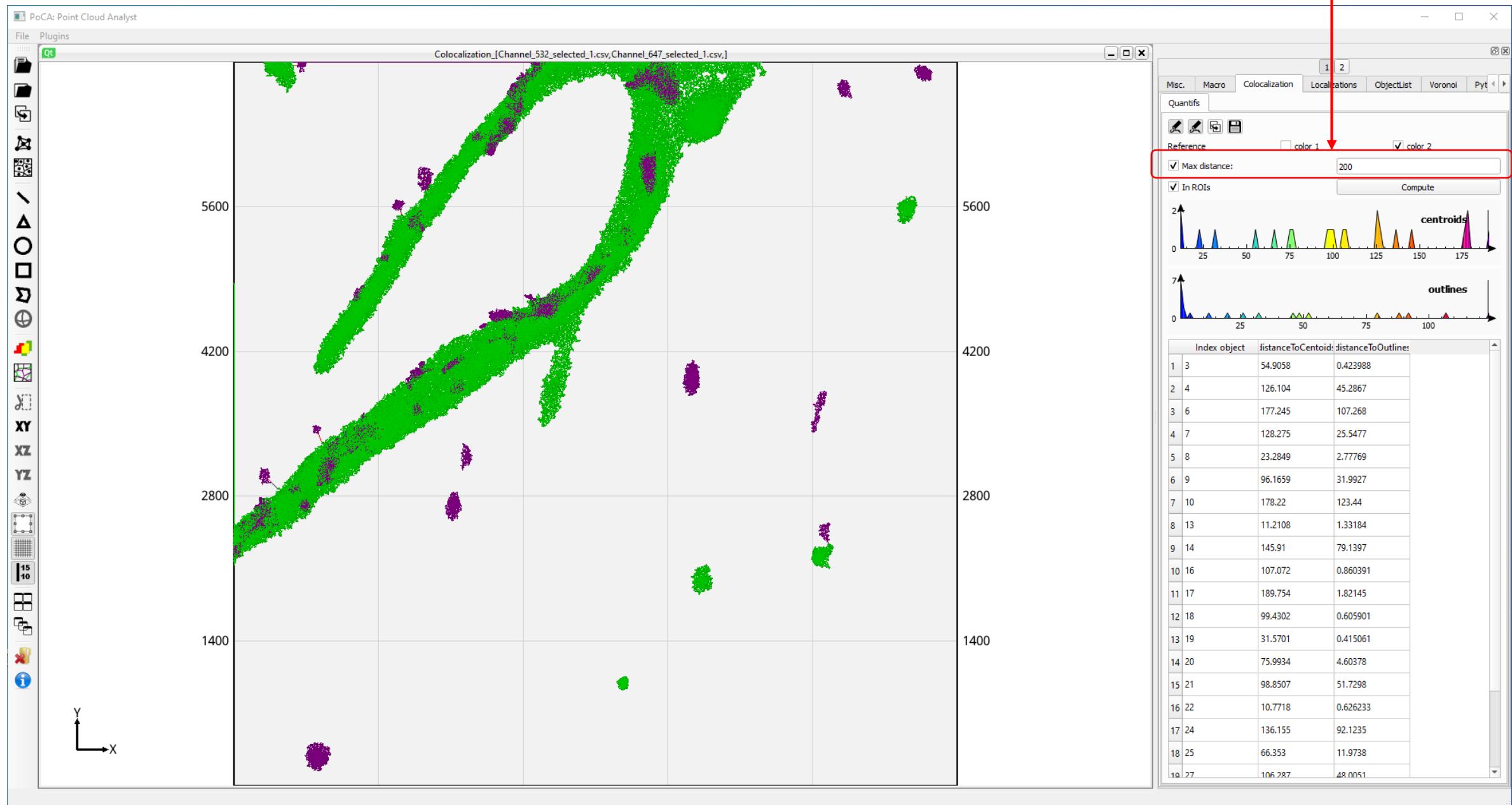
Red lines show the distance to centroids
Cyan lines show the distance to outlines

Toggle line displays

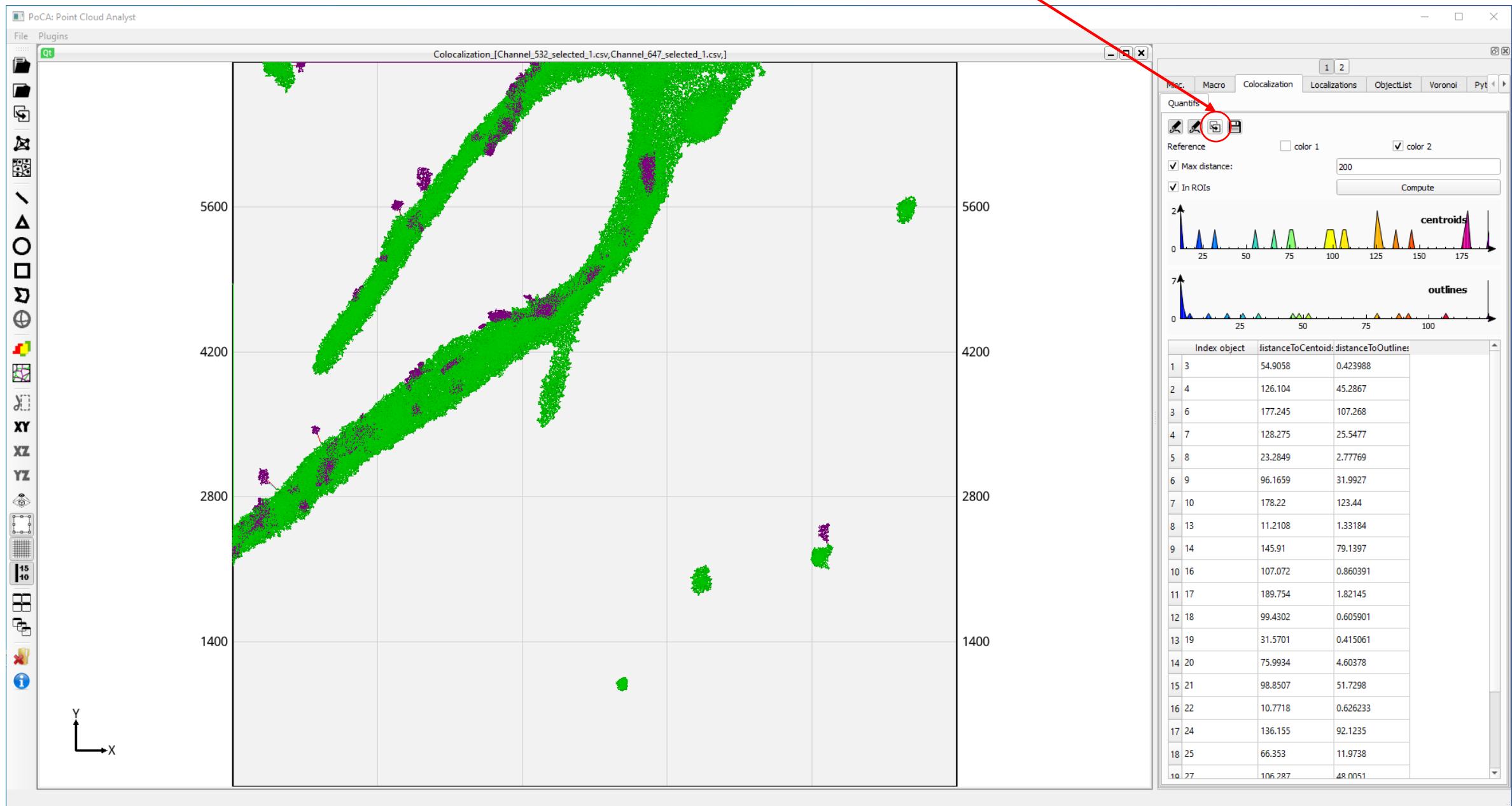
Run computation of the distances



Applying a max distance of 200 nm



Pressing this button remove the display of any object that is above the max distance (only for the non-referent color)



Pressing this button opens a dialog for saving the distances

