

PyLattice

v0.2

August 2018

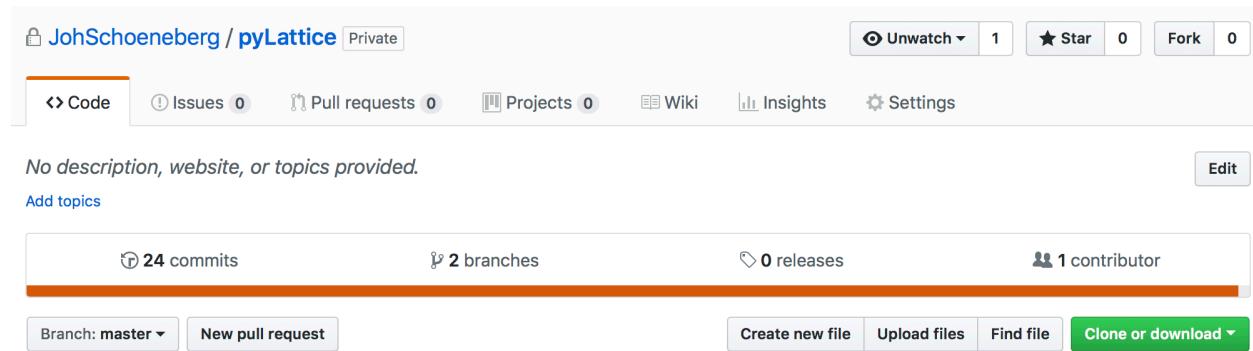
Joh Schöneberg

User Manual

Installation

Download the git repository to your computer from

<https://github.com/JohSchoeneberg/pyLattice> by using the green button as displayed below:



Requirements

- Python
- Jupyter notebook installation (e.g. Anaconda)
- A recent Matlab installation (Matlab 2017 was used for dev and testing)

Python

Some functionality of pyLattice is built in Python and resides in the pyLattice package on [PyPI](#).
Install the package by typing:

- pip install pyLattice

To install the requirements specific for pyLattice, the github repository contains a requirements.txt file. From within the pyLattice folder, type

- pip install -r requirements.txt

Jupyter

The best way to access and run the tools is through their Jupyter notebook interfaces:

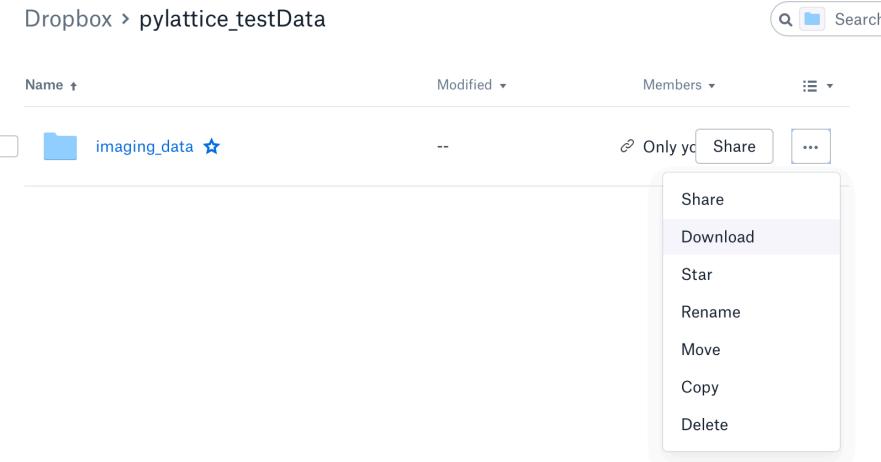
- Start a Jupyter notebook server (> Jupyter notebook) and navigate to the src/jupyter folder



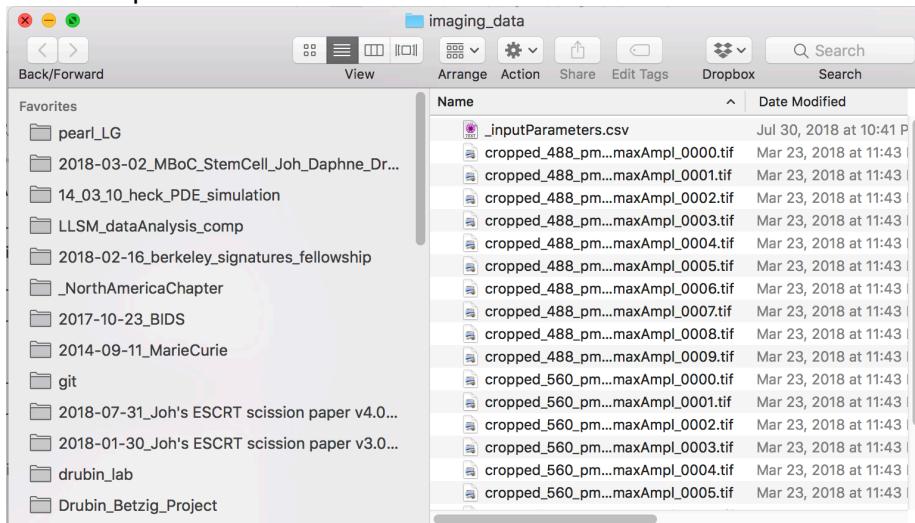
If the notebook requires additional python modules, install them via terminal: 'pip install [moduleName]'

Usage 0, Example Data

- Example input data can be downloaded here:
<https://www.dropbox.com/sh/2n86tvgr1a6i3h/AAAcJ84K0qdqq3USCtO1jUcra?dl=0>
- Download the example data (see screenshot)



- The example data should look like this:

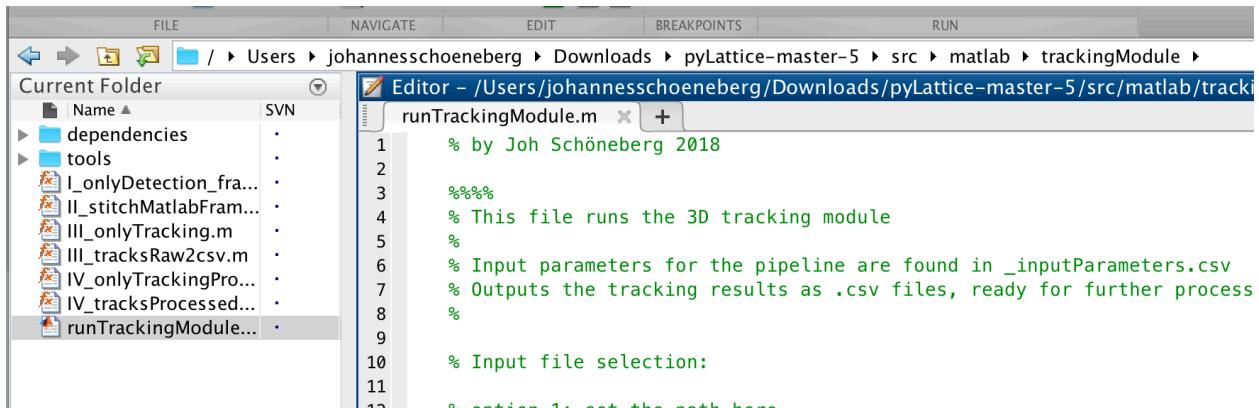


- In the '_inputParameters.csv' file, change the first two lines to match the location of the '_inputParameters.csv' in your filesystem. E.g:

```
inputDataFolder, ./ → inputDataFolder, /Users/johannesschoeneberg/Downloads/imaging_data/¬  
outputDataFolder, ./output → outputDataFolder, /Users/johannesschoeneberg/Downloads/imaging_data/output/¬
```

Usage 1, Run Particle Detection and Tracking Module

- Open Matlab.
- Navigate to [yourpath]/pyLattice/src/matlab/trackingModule



The screenshot shows the Matlab IDE interface. The current folder path is displayed at the top: /Users/johannesschoeneberg/Downloads/pyLattice-master-5/src/matlab/trackingModule. The left pane shows a file tree with several subfolders and files, including 'dependencies', 'tools', and scripts like 'I_onlyDetection_fra...', 'II_stitchMatlabFram...', 'III_onlyTracking.m', 'III_tracksRaw2csv.m', 'IV_onlyTrackingPro...', 'IV_tracksProcessed...', and 'runTrackingModule...'. The right pane is the 'Editor' window titled 'Editor - /Users/johannesschoeneberg/Downloads/pyLattice-master-5/src/matlab/trackingModule/runTrackingModule.m'. The code in the editor is:

```
% by Joh Schöneberg 2018
%%%%%
% This file runs the 3D tracking module
%
% Input parameters for the pipeline are found in _inputParameters.csv
% Outputs the tracking results as .csv files, ready for further process
%
% Input file selection:
% <--> Select the path here
```

- Run the file 'runTrackingModule' by opening it and clicking the 'Run' button (green arrow)
- A popup window appears in which you have to select the input data for the tracking module. Example input data is available (see above, step 0). Make sure that the path is correctly specified according to your system.
- Now, in the Matlab prompt, select the '_inputParameters.csv' file as the input for the tracking.
- The detection and tracking should now commence. Depending on your system, it will take a couple of minutes to finish the calculations.

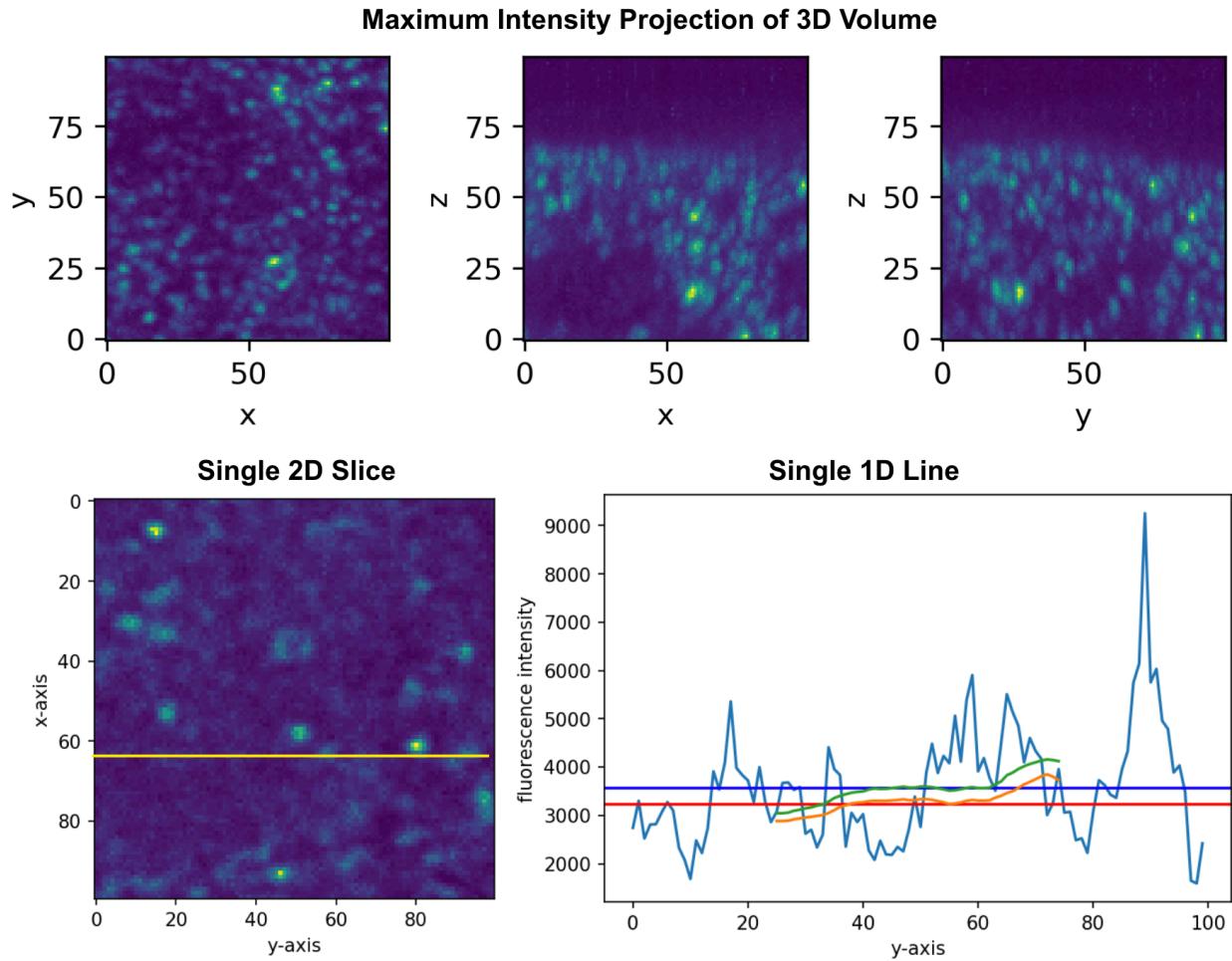
Usage 2, Lattice Light-Sheet Data Visualization and Preprocessing

latticeFrame_showFrame.ipynb

One of the first things when working with high resolution data is to visually look at the data. The notebook 'latticeFrame_showFrame.ipynb' provides the basic functionality to do that:

- Maximum intensity projection
- View individual 2D slices of the data
- View 1D lines of the data

For 3D volumetric renderings of the data, I recommend to use ChimeraX.

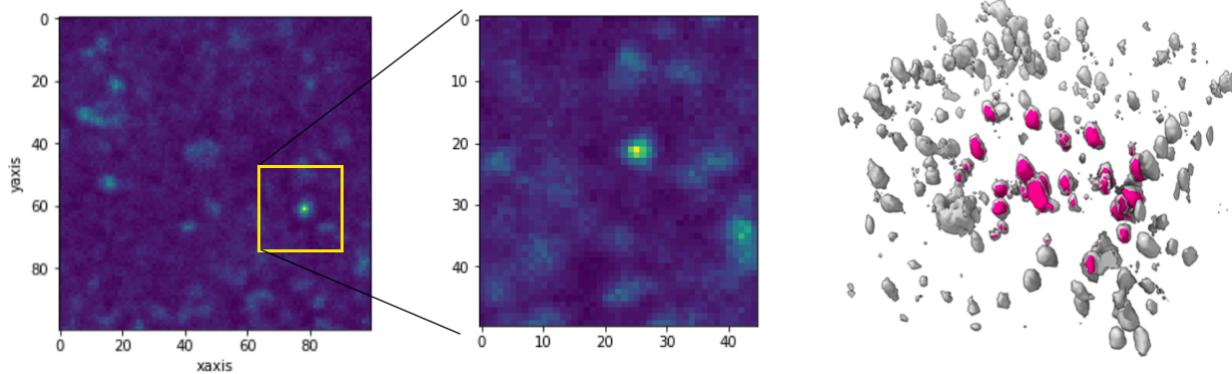


[latticeMovie_cropRegion.ipynb](#)

LLSM frames in respectively LLSM movies can reach substantial file sizes, cropping the movie can lead to faster processing times. Also, sometimes a certain region in the imaged 3D volume contains most of the relevant information. Or one simply wants to zoom in on a region.

This cropping tool allows to crop regions of interest out of large 3D LLSM movies:

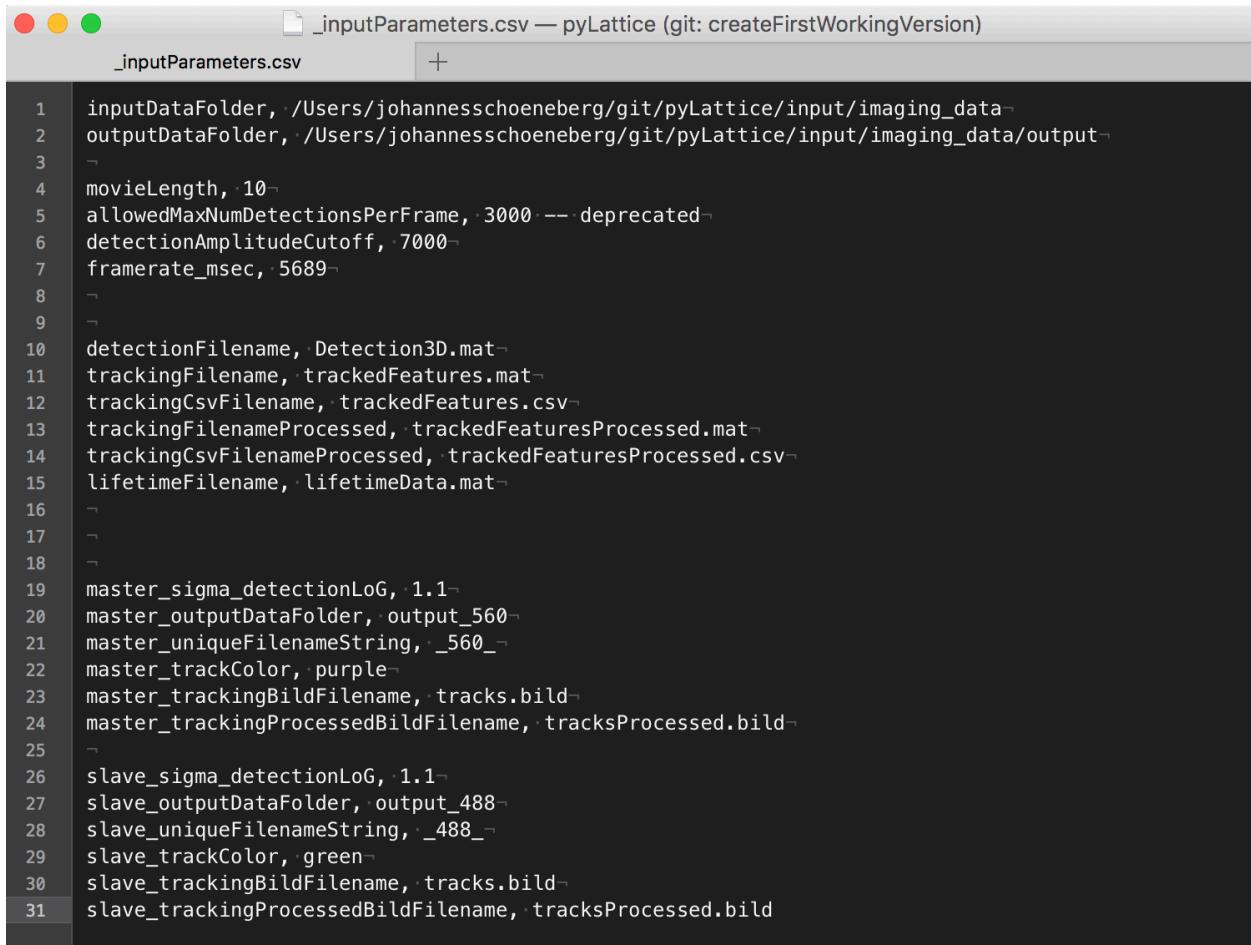
Cropping a smaller 3D volume out of a larger LLSTM frame



3D Particle Detection & Tracking

1. Adjust Input Parameters

- Navigate to pyLattice/input/imaging_data
- Open _inputParameters.csv
- Change the inputDataFolder and outputDataFolder variables such that they match your filesystem (top two lines)



The screenshot shows a terminal window with the title '_inputParameters.csv — pyLattice (git: createFirstWorkingVersion)'. The window displays the following CSV file content:

```
1 inputDataFolder, /Users/johannesschoeneberg/git/pyLattice/input/imaging_data
2 outputDataFolder, /Users/johannesschoeneberg/git/pyLattice/input/imaging_data/output
3 
4 movieLength, 10
5 allowedMaxNumDetectionsPerFrame, 3000 -- deprecated
6 detectionAmplitudeCutoff, 7000
7 framerate_msec, 5689
8 
9 
10 detectionFilename, Detection3D.mat
11 trackingFilename, trackedFeatures.mat
12 trackingCsvFilename, trackedFeatures.csv
13 trackingFilenameProcessed, trackedFeaturesProcessed.mat
14 trackingCsvFilenameProcessed, trackedFeaturesProcessed.csv
15 lifetimeFilename, lifetimeData.mat
16 
17 
18 
19 master_sigma_detectionLog, 1.1
20 master_outputDataFolder, output_560
21 master_uniqueFilenameString, _560_
22 master_trackColor, purple
23 master_trackingBildFilename, tracks.bild
24 master_trackingProcessedBildFilename, tracksProcessed.bild
25 
26 slave_sigma_detectionLoG, 1.1
27 slave_outputDataFolder, output_488
28 slave_uniqueFilenameString, _488_
29 slave_trackColor, green
30 slave_trackingBildFilename, tracks.bild
31 slave_trackingProcessedBildFilename, tracksProcessed.bild
```

2. Run the Matlab Tracking Module

- Open the /src/matlab/trackingModule folder and open 'runTrackingModule.m'
- Hit 'Run'.

The Command window should now display the progress doing the tracking:

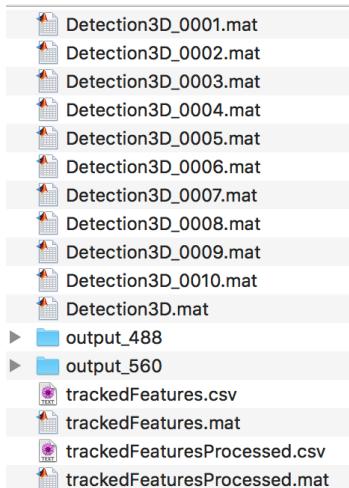
```
Command Window
>> runTrackingModule

paramFilePath =
    '/Users/johannesschoeneberg/git/pyLattice/src/matlab/trackingModule//...//.../input/_inputPar

-----
I_onlyDetection_framebyframe_nonParallel(): Start detection...
path =
    '/Users/johannesschoeneberg/git/pyLattice/src/matlab/trackingModule//...//.../input/_inputPar
fx
```

After a little while (few minutes) the tracking module should terminate.
The results of the tracking can be found in the output folder that you specified in step 1.

There should now be the following files in your output folder:



The trackedFeaturesProcessed.csv is the final output of the tracking module that we will use for further processing.

Detected Puncta

The first step in particle tracking detects the particles (puncta).

1. Visualize Detected Puncta

detectedPuncta_oneFrame_plot.ipynb

This Jupyter notebook allows you to visualize the detected puncta from an individual frame.

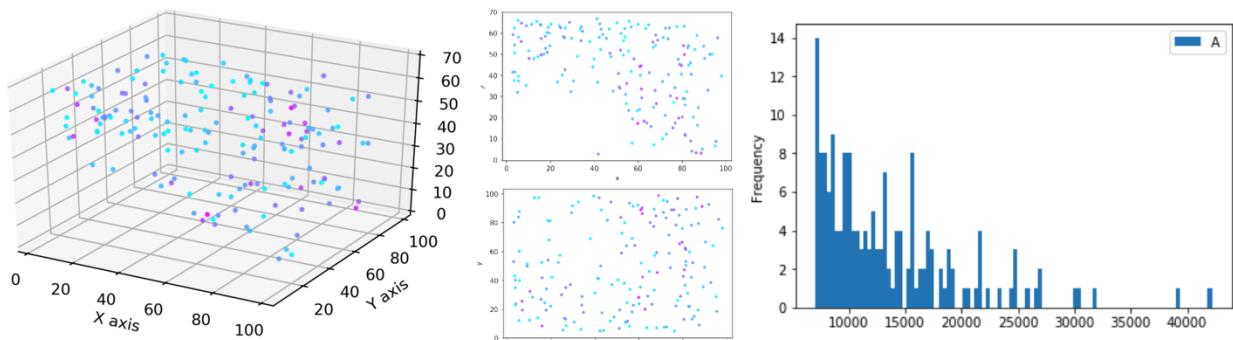


Fig 1. Plot the detected puncta with the ‘detectedPuncta_oneFrame_plot.ipynb’ notebook.

It provides code and functions to plot the detected puncta in 3D, 2D and has functionality for puncta selection based on intensity.

In addition, the notebook allows to convert the detected puncta into a [*.bild](#) file that can be visualized in the open source tool [ChimeraX](#) [1]:

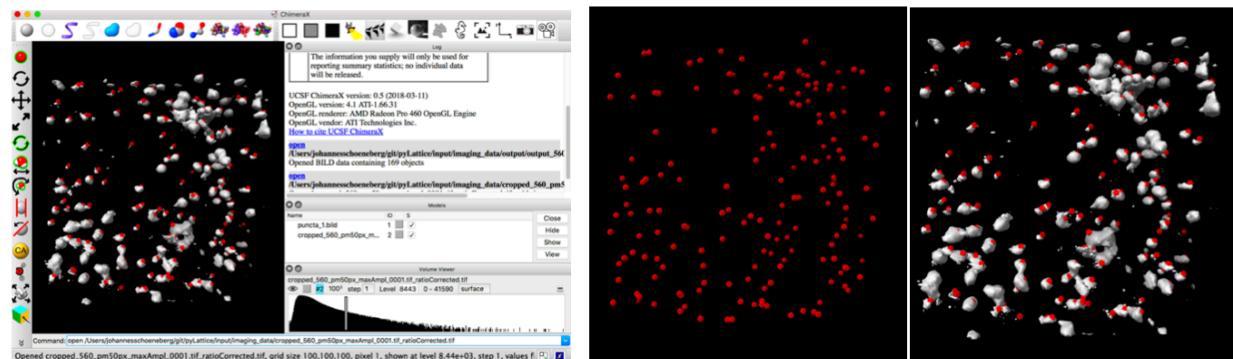
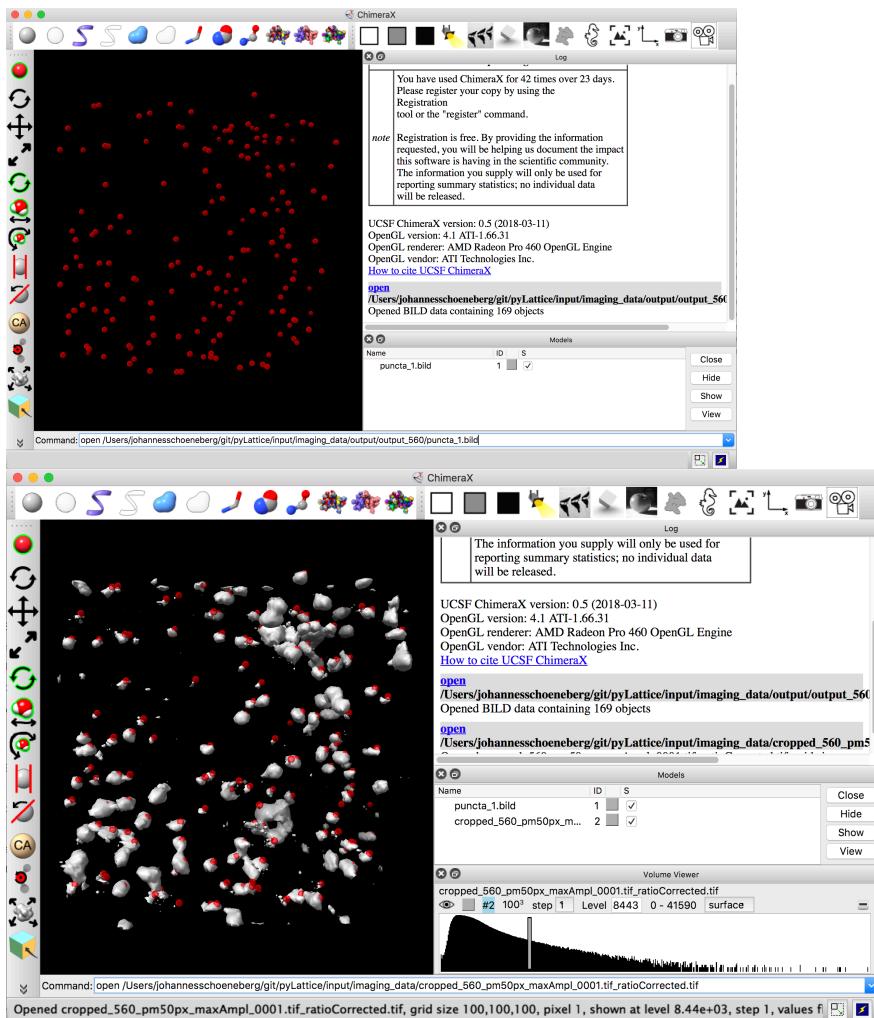
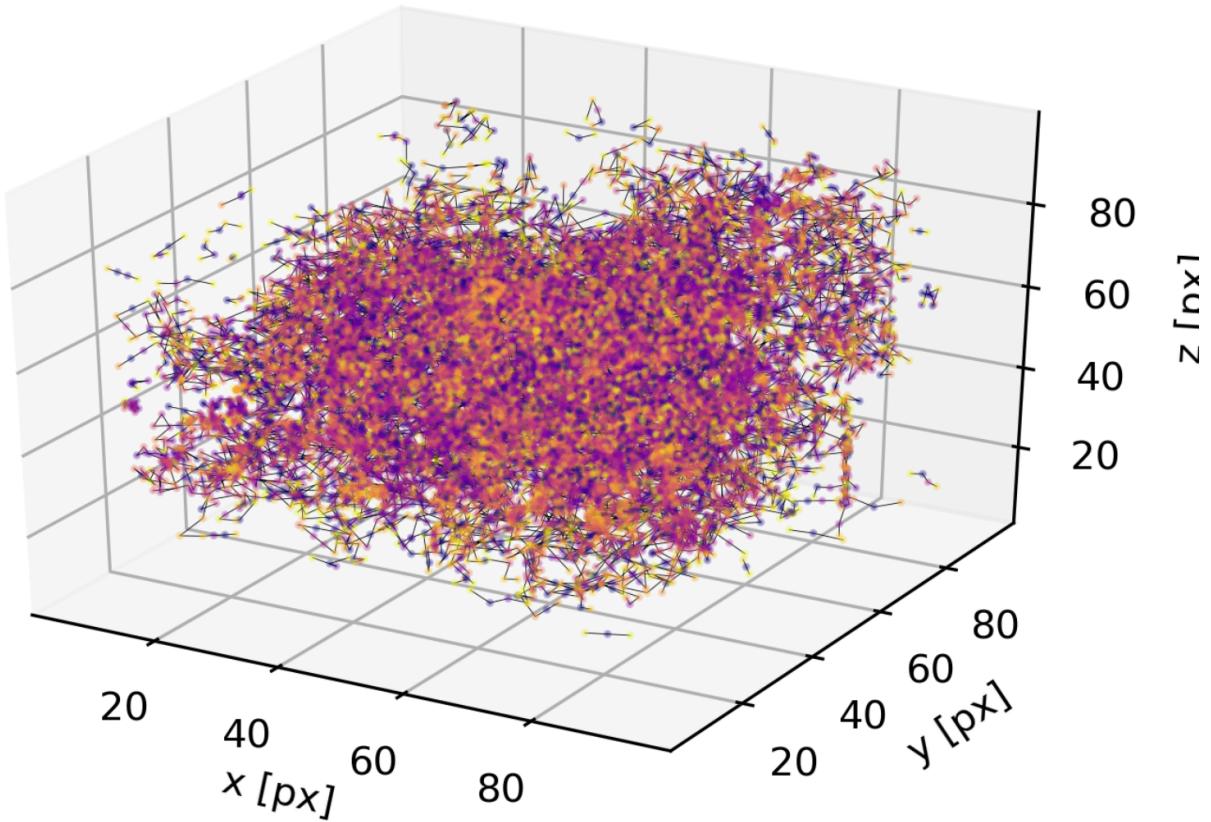


Fig 2. Display detected puncta using ChimeraX and overlay them with the raw data. Shown is ChimeraX’s UI (left), detected puncta (middle) and overlay (right).



Run: detectedTracksProcessed_plotAll_3D.ipynb, the result should look like this:



References:

- [1] [UCSF ChimeraX: Meeting Modern Challenges in Visualization and Analysis](#). Goddard TD, Huang CC, Meng EC, Pettersen EF, Couch GS, Morris JH, Ferrin TE. *Protein Sci.* 2018 Jan;27(1):14-25. doi: 10.1002/pro.3235.