Documentation

General Information

1.1 Overview

The software is used for analysis of 3D live images generated from lattice light sheet microscopy (LLSM). It is developed based on Aguet et al., 2016 (cmeAnalysis3D). The software consists of image deskew, point detection and point tracking, along with some post-analysis functions to quantify or visualize the detection or tracking results. Compared to the original method, we have these improvements:

a. add methods for color offset correction across different channels;
b. Improve efficiency of the detection method;
c. Developed detection methods under low SNR conditions;
d. Develop constrained optimization scheme for Gaussian curve fitting and mixture of Gaussians curve fitting;
e. Add filtering of noisy points;
f. Fix bugs in tracking;
g. Add distributed computing schemes for better scalability.

1.2 License

For additional information, send email to sup@berkeley.edu.

1.3 Requirements

This software requires an installation of Matlab (MathWorks, Natick, MA) with the following toolboxes:

- Curve Fitting Toolbox
- Image Processing Toolbox
- Optimization Toolbox
- Statistics Toolbox
- Parallel Computing Toolbox (optional)

The software has been tested using the following configuration(s):

Matlab version: 2019b and above
Operating systems (64-bit only): Mac OS X 10.15.2, Ubuntu 18.04, Windows 10

1.4 Instationation

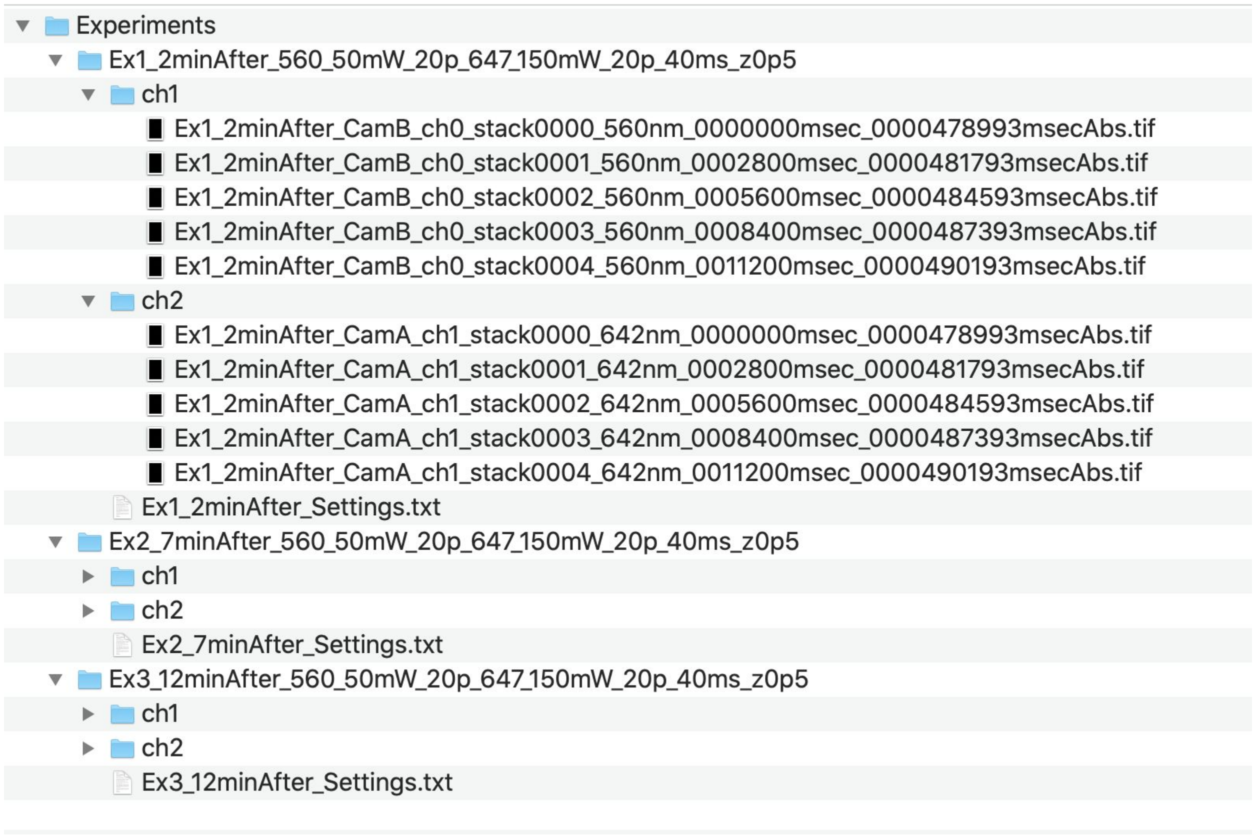
1. Download the latest version of the software from github (https://github.com/abcucberkeley/LLSM3DTools.git) via git clone or downloading as zip archive file. If downloading the zip file, extract it in your work directory.
2. Start Matlab
3. Add the LLSM3DTools directory containing the code to the Matlab search path. For more information, see http://www.mathworks.com/help/matlab/matlab_env/what-is-the-matlab-search-path.html

Data Requirement and Organization

The software is designed for the analysis of 5D lattice light sheet data (xyz + time + channels). The data organization is illustrated in Fig. 1 for automatic parsing:

- Each movie should be stored in a directory with a common identifier, i.e., “Exp” or “Ex”, followed by a number. The suffix should include the stage scan interval in the format of ‘_z0p5’, meaning 0.5 um.
- The frames of each movie must be stored as individual TIFF files (extension .tif or .tiff, case insensitive). If the frame rate is not provided and the frame file names also contain the relative time in the movie, i.e., “Ex1_XXX_0040msec_z0p5”, this is automatically parsed. The frame rate is calculated by the difference of two adjacency frames. If there are multiple channels, each channel must have its own subdirectory within the experimental directory. The master channel frames can optionally be stored directly in the experimental directory.
- Frames must have a frame number preceding the extension in the file name. There are no constraints on the remainder of the file name.

- Movie directories may be stored in a single parent directory, or further divided by experiment or acquisition session (see Fig. 1). The software can be run at the 'Condition' or 'Experiment' level; in both cases all movies are pooled for the analysis.
- Note: if there is no point spread function of the experiments, it may be necessary to obtain them by calibration of the microscope for the channels in the experiments. Then the calibration files may be put in a subdirectory under "Experiments".



```
▼ Experiments
    ▼ Ex1_2minAfter_560_50mW_20p_647_150mW_20p_40ms_z0p5
        ▼ ch1
              Ex1_2minAfter_CamB_ch0_stack0000_560nm_0000000msec_0000478993msecAbs.tif
              Ex1_2minAfter_CamB_ch0_stack0001_560nm_0002800msec_0000481793msecAbs.tif
              Ex1_2minAfter_CamB_ch0_stack0002_560nm_0005600msec_0000484593msecAbs.tif
              Ex1_2minAfter_CamB_ch0_stack0003_560nm_0008400msec_0000487393msecAbs.tif
              Ex1_2minAfter_CamB_ch0_stack0004_560nm_0011200msec_0000490193msecAbs.tif
        ▼ ch2
              Ex1_2minAfter_CamA_ch1_stack0000_642nm_0000000msec_0000478993msecAbs.tif
              Ex1_2minAfter_CamA_ch1_stack0001_642nm_0002800msec_0000481793msecAbs.tif
              Ex1_2minAfter_CamA_ch1_stack0002_642nm_0005600msec_0000484593msecAbs.tif
              Ex1_2minAfter_CamA_ch1_stack0003_642nm_0008400msec_0000487393msecAbs.tif
              Ex1_2minAfter_CamA_ch1_stack0004_642nm_0011200msec_0000490193msecAbs.tif
          Ex1_2minAfter_Settings.txt
    ▼ Ex2_7minAfter_560_50mW_20p_647_150mW_20p_40ms_z0p5
        ▶ ch1
        ▶ ch2
          Ex2_7minAfter_Settings.txt
    ▼ Ex3_12minAfter_560_50mW_20p_647_150mW_20p_40ms_z0p5
        ▶ ch1
        ▶ ch2
          Ex3_12minAfter_Settings.txt
```

Figure 1 Data structures recognized by the software

Data Analysis

Documentation for all functions provided with this software can be retrieved by entering help followed by the function name in the Matlab command prompt.

1.0 Load data

The data is loaded with the following command.

```
>> data = XR_locadConditionData3D();
```

Alternatively, the data can be loaded non-interactively (see also Section 2.3):

```
>> data = loadConditionData('PathToData', {'Ch1', 'Ch2'}, {'EGFP',...
'RFP'}, 'Parameters', [1.49 100 6.45e-6]);
```

Using the example of Fig. 1, PathToData would point to “Condition”.

Alternatively, the data can be loaded non-interactively (see also Section 2.3):

```
>> data = XR_loadConditionData('PathToData', {'Ch1', 'Ch2'},
{'EGFP',... 'RFP'}, 'Parameters', [1.49 100 6.45e-6]);
```

Using the example of Fig. 1, PathToData would point to “Condition”.

1.1 Deskew data

In image acquisition, the z-stacks of a frame are skewed and are not positioned properly. To solve the problem, we need first deskew data.

```
>>aname = 'Analysis';
>>data = deskewData(data, 'Overwrite', true, 'SkewAngle', 31.5, ...
'aname', aname);
>>XR_correctXZoffsetData3D(data, 'Overwrite', true);
```

The first line provides the result directory name (under the master channel directory). The second line is for the deskew for each channel, with directory name ‘DS’ under each channel directory. The parameter ‘overwrite’ means overwrite the results even the result files exist. ‘SkewAngle’ is the angle between objective and stage of the lattice light-sheet microscopy. The third line is an optional function for correct offset in X-axis and Z-axis across different channels.

1.2 Detection

1.2.0 Estimation of point spread function standard deviations

(Optional step) The software needs standard deviation of the point spread functions as parameters for point source detection. If they are unavailable by theory, and if there are calibration files, we can estimate psf standard deviations. If there are multiple channels, we can estimate psf standard deviations one by one.

```
>> ch1_psf_filename = ‘path_to_psf_file_for_ch1’;
>>[sigmaXY_ch1, sigmaZ_ch1] =
GU_estimateSigma3D(ch1_psf_filename,[]);
>> ch2_psf_filename = ‘path_to_psf_file_for_ch2’;
>>[sigmaXY_ch2, sigmaZ_ch2] =
GU_estimateSigma3D(ch2_psf_filename,[]);
...
```

Then, we can put all sigmas to a single matrix.

```
>> sigma_mat = [sigmaXY_ch1, sigmaZ_ch1 ./ data.zAniso;
                sigmaXY_ch2, sigmaZ_ch2 ./ data.zAniso];
```

Note that the standard deviation in Z direction should be divided by the Aniso factor if the calibration data is (approximately) isotropic.

1.2.1 Point source detection

Then, we can perform point detection with the following code:

```
>>apath = arrayfun(@(i) [i.source filesep aname
filesep],data,'unif',0);
>>XR_runDetection3D(data, 'Sigma', sigma_mat, 'Overwrite', true, ...
'ResultsPath', apath);
```

1.3 Tracking

After point source detection, we can link the points between adjacency time frames to create point trajectories. This step can be divided into three sub-steps: first we link points across frames; then we classify tracks into different groups based on whether they have gaps, merges or splits etc; finally, we may rotate tracks for visualization.

1.3.1 Tracking

```
data = data(~([data.movieLength]==1));
XR_runTracking3D(data, loadTrackSettings('Radius', [3, 6], ...
'MaxGapLength', 2), 'FileName', 'trackedFeatures.mat', 'Overwrite',
... true, 'ResultsPath', apath);
```

1.3.2 Tracking processing

```
>>XR_runTrackProcessing3D(data, 'Overwrite', true,...
                          'TrackerOutput', 'trackedFeatures.mat',...
                          'FileName', 'ProcessedTracks.mat', ...
                          'Buffer', [3, 3], 'BufferAll', false,...
                          'FitMixtures', false, 'WindowSize', [],...
                          'ResultsPath', apath);
```

1.3.3 Rotate tracking

```
>>rotateTracks3D(data, 'Overwrite', true, 'Crop', false, ...
'ResultsPath', apath);
```

1.4 Lifetime Analysis

```
>>lftRes = runLifetimeAnalysis3D(data);
>>cohorts = plotIntensityCohorts(data);
```

The first line is to run lifetime analysis and the second line is for visualization of the results.

2.1 Integrated function

Above, we separate the steps for better understanding of how the software works. To simplify the running, we also provide an integrated function that contains deskew, detection and tracking steps. After loading data and estimation of psf standard deviation, we can use the following code to perform all the analysis from step 1.1, 1.2.1, and 1.3.

```
>>XR_runDetTrack3d(data,'Sigma', sigma_mat, 'DetectionMethod', ...
           'lowSNR', 'aname', aname,  'zoffsetCorrection', true);
```

Demos

There are three demos: two for analysis of a single experiment (cell) with one is for integrated function, and another other is for step-by-step; one for three experiments and visualization and post-analysis of the detection and tracking results (one cell is not enough). Before running the demos, first download image data for the demos from the google drive link and then decompress the tarball in your desired location.

3.1 Integrated demo for a single experiment
To run the demo for the integrated demo for a single experiment, execute the following command

```
>> demo_point_detection_3d_integrated
```

or open the demo, click ‘Run’ or press ‘F5’.

It will promote to get the movie path, you can either go to the path in the level of experiments (‘Ex001XXXX’) or upper level. After getting the path, it requires the input of number of channels in the command line window, type ‘2’ for the demo. Then, it will promote for the paths of the channels. You can click ‘ch1’ for channel 1 with your mouse; and then you can click ‘ch1’ for channel 2 when it propped again. After loading data, it will run the remaining steps.

3.2 Step-by-step demo for a single experiment
The demo name is `demo_point_detection_3d_by_step.m`. The running methods are the same as in 3.1.

3.3 Integrated demo for multiple experiments
The demo name is `demo_point_detection_3d_multiple_experiments.m`. The running methods are the same as in 3.1.