Tracking Routine Manual

1 Getting started

1.1 Installation

This tracking routine uses ThunderSTORM to localize particles, a plugin for Fiji.[1,2] The localizations can also be found with rapidSTORM.[3] Particles are connected to trajectories with swift.[A][4] Parameter determination for swift and further data analysis is realized in SPTAnalyser. SPTAnalyser also processes PALMTracer[B] data, a program that both localizes and connects the localizations. The connections between the different softwares are shown in fig. 1.

Navigate to the directory with the SPTAnalyser *.whl file and execute listing 1 in Anaconda prompt. The ThunderSTORM.jar plugin[C] must be placed in the plugins folder of Fiji. Install a h5-viewer.[D]

Listing 1: Commands to install SPTAnalyser and dependencies in the Anaconda prompt.

```
conda create --name SPTAnalyser
conda activate SPTAnalyser
conda install pip pywin32
pip install jupyterlab
pip install SPTAnalyser-XXX-py3-none-any.whl
pip install jupyter_contrib_nbextensions
jupyter notebook
```

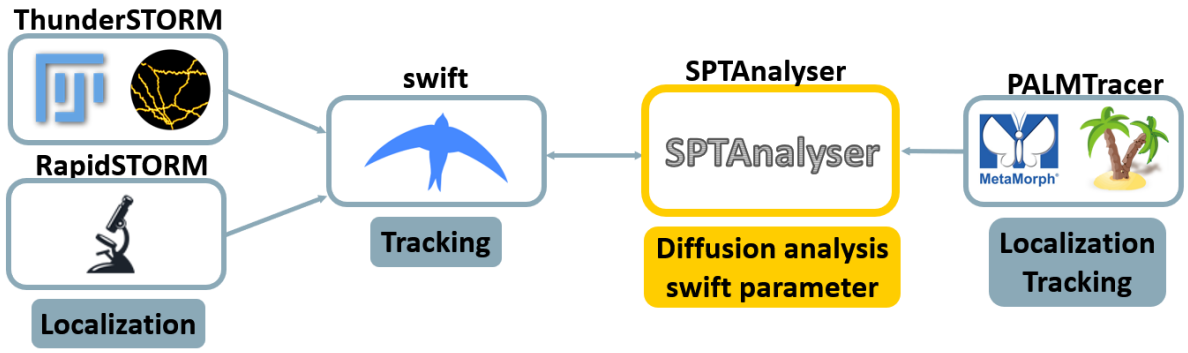

Fig. 1: Connections between the softwares. swift is compatible with ThunderSTORM and rapidSTORM. SPTAnalyser is compatible with swift and PALMTracer.

[A] http://bit.ly/swifttracking
[B] https://docs.google.com/forms/d/e/1FAIpQLSdxkJXmoc5uRqoL8kzCSV2Rv90kEfRMIoaPuk--Bpm0Lttb0g/viewform
[C] https://github.com/zitmen/thunderstorm/releases/tag/dev-2016-09-10-b1
[D] https://www.hdfgroup.org/downloads/hdfview/

1.2 How to run a notebook

SPTAnalyser is available in Jupyter Notebooks. Start them by following listing 2: Change to the directory of the notebooks (*.ipynb ending), activate the environment and start the notebooks, they will open in a browser. Configure the notebook settings on the first run by clicking on the Nbextensions tab, enabling configuration and selecting “Collapsible Headings” and “Hide input all” (fig. 2), to increase usability.

Listing 2: Commands to run SPTAnalyser after installation in the Anaconda prompt.

```
cd C:\Documents\SPTAnalyser
conda activate SPTAnalyser
jupyter notebook
```

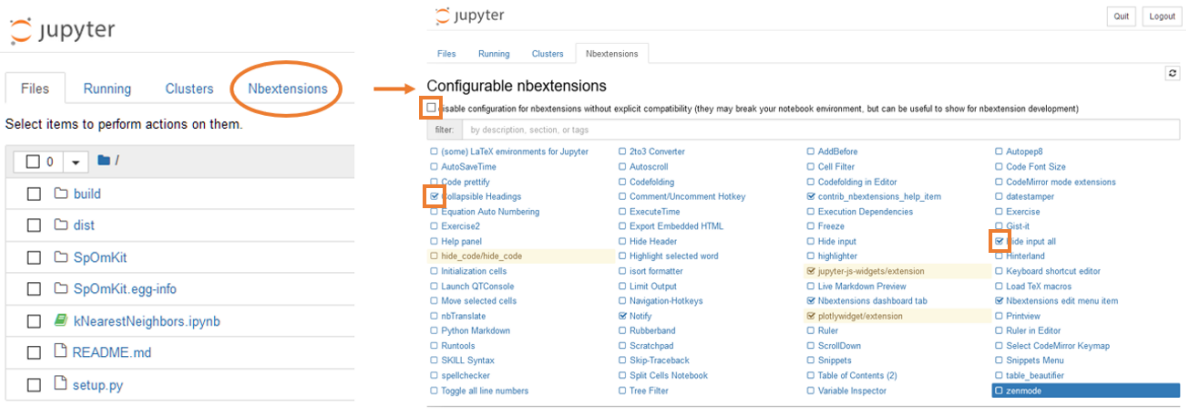

Fig. 2: Configure Jupyter Notebook settings. Enable “hide input all” to make code cells disappear. Enable “collapsible headings” to make sections disappear. Both settings will increase the overall usability by making the notebooks look cleaner.

In the files tab, select a notebook. A new tab opens where you have to click on restart the kernel to load the widgets (1), click on show codecell inputs to hide code (2), and you may collapse headings if needed (3) (Fig. 3).

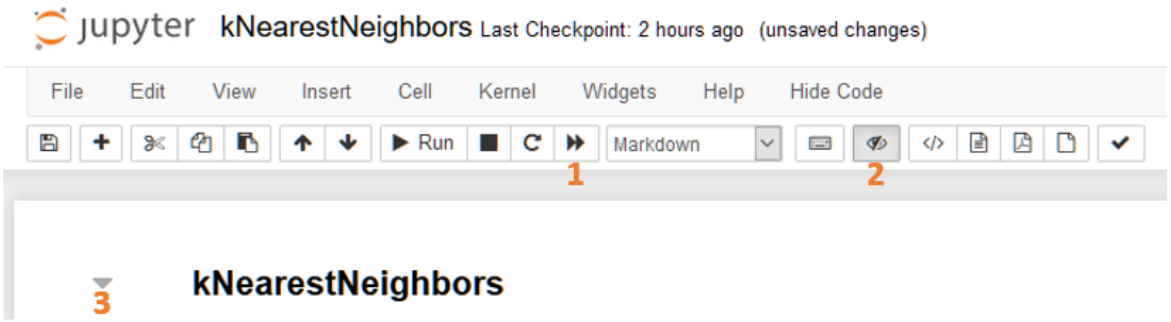

Fig. 3: Start and run a notebook.

It is possible to customize the default parameters of the widgets. “Hide input all” must be deactivated to see the code. All adjustable settings are highlighted with comments, just change the values according to your requirements (Fig. 4).

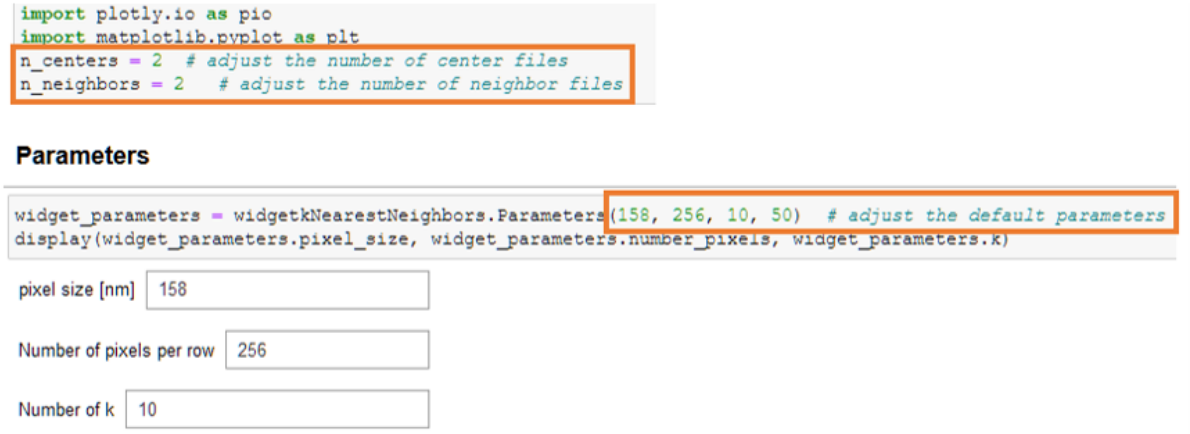

Fig. 4: Two examples of where to adjust the default parameters in the code. The places are marked with # comments.

1.3 Test dataset

The dataset[E] contains the following files of 60 HeLa cells marked with non-activating Fab ligand:

- *.csv file = localization file from ThunderSTORM
- *-protocol.txt = information file with ThunderSTORM parameters
- *_tracked.csv file = tracked file from swift
- *.tracked.meta.json = information file with swift parameters
- *.h5 = SPTAnalyser file from trackStatistics

The analysis steps in the notebooks require different input files:

- diffractionLimit - localization file
- expDisplacement - tracked file
- expNoiseRate - localization file
- pBleach - tracked file
- precision - localization file
- trackAnalysis - tracked file
- trackStatistics - file from trackAnalysis
- transitionCounts - tracked file and file from trackStatistics / trackAnalysis

[E]More data is available at `https://www.ebi.ac.uk/biostudies/studies/S-BSST712`

1.4 Recommended folder structure

The following folder structure is recommended for the workflow using ThunderSTORM, swift, and SPTAnalyser (fig. 5). Individual folders contain data of each measured coverslip (CS) with background measurements (optional), cell measurements, and some swift preprocessing parameters determined for each coverslip. Each cell folder (applies to background folder as well) contains rois, tifs, and tracks folder. The roi folder consists of *.roi files, which mark the region of interest (ROI) in ThunderSTORM, and *.csv files, which contain the measured size of the ROI. The tifs folder contains recorded *.tif movies and transmitted light images *_dl.tif or *_tl.tif. The tracks folder contains localization files from ThunderSTORM that will adapt the *.tif file names of the recorded movies, the tracked file in swift *.tracked.csv, meta information of the tracked file *.tracked.meta.json, and meta information of the ThunderSTORM file *-protocol.txt
For the naming it is recommended to name every cell individually, for example date, condition, coverslip number, and cell number to be able to distinguish all targets for later analysis.

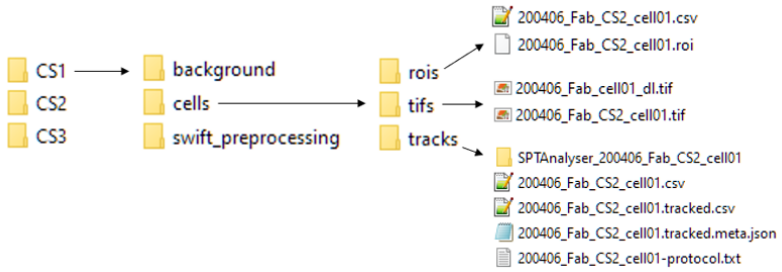

Fig. 5: Folder structure created in the course of the tracking routine.

1.4.1 Script to create folder structure

After measurement the *.tif files are distributed over multiple folders, by the measurement program Micromanager.[5] This folder structure is cleaned up with a script and the resulting structure (fig. 5) is necessary for further batch processing (and in generell recommended to keep an overview over the dataset). To use the script, the background measurement files must contain “background” in their names, the cell measurements “cell” and the transmitted light images “tl” or “dl”.
Execute the adapt_folder_structure.py script + config file in Anaconda prompt to automatically create the structure in (fig. 6, listing 3). Define the paths to coverslip directories, multiple directories can be handled at once. Additionally, redundant names in the

files can be removed, by defining remove_str = ... (empty = original file names are kept). Example: cell01_MMStack_Pos0.ome.tif will be renamed to cell01.tif.

```
[INPUT_DIRS]
# define the directories to coverslips
dir01 = D:\example_path\CS1

[PARAMETERS]
# define redundant file naming, XXX_MMStack_Pos0.ome.tif -> XXX.tif
remove str =  MMStack Pos0.ome.tif
```

Fig. 6: Config file for adapt_folder_structure.py script. Multiple directories are possible.

Listing 3: Execution of script in Anaconda prompt to create folder structure in fig. 5.

```
python adapt_folder_structure.py adapt_folder_structure_config.ini
```

2 ThunderSTORM

ThunderSTORM[1] is a localization software, available as a plugin for Fiji. The analysis requires transmitted light images and SPT movies of the target and information about the camera settings. Localized files are further processed in swift to create trajectories.

2.1 Mark regions of interest

Load the SPT movie and a transmitted light image of a cell into Fiji. Circle the region of interest with a selection tool (fig. 7). Select the transmitted light image window and open ROI manager, by pressing *t* on the keyboard. Select the SPT movie and click on the area in the ROI manager window to transfer the selected area to the movie. Click at *Measure* to calculate the area of the region in px and save this information as *.csv in the rois folder (choose same name as SPT movie, e.g. cell01.tif and cell01.csv). Save the region as *.roi (*More* → *Save...*). This will be needed at later points of the analysis.

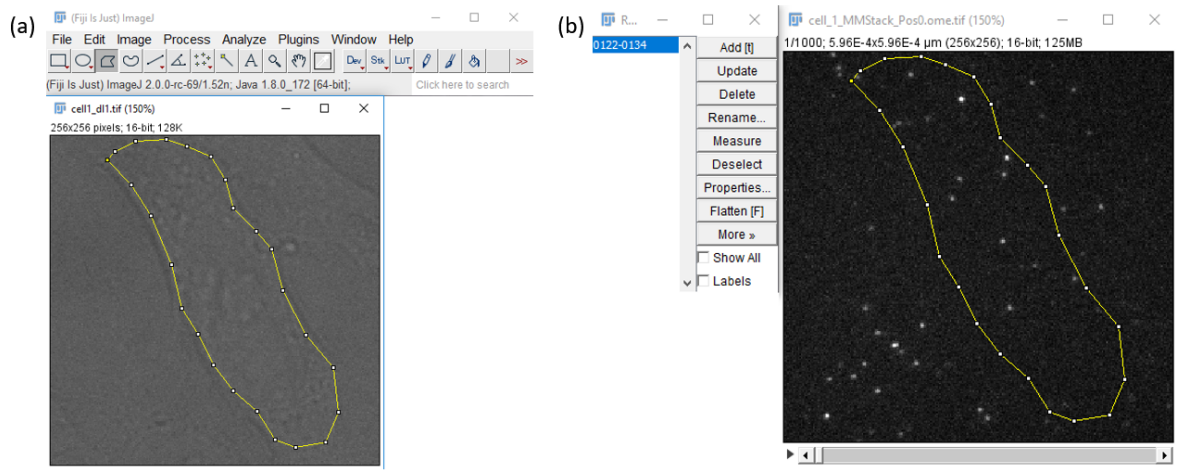

Fig. 7: (a) Selected region of a transmitted light cell image with the polygon selection tool. (b) Transferred region into a single particle tracking (SPT) movie.

2.2 Analysis of one file

Run the analysis in Fiji with *Plugins* → *ThunderSTORM* → *Run analysis* (fig. 10). The following will briefly introduce the settings, for more information read the Thun-

derSTORM user guide.[F,G]

The camera has to be specified (fig. 9). Correct pixel size is important for proper spatial calibration of the rendered images. Camera conversion gain and offset influence the estimates of localization precision. The recommended filter for the feature enhancement is the wavelet filter. The localization of the molecules can be found with the local maximum method in combination with 8-neighborhood-connectivity. This combination is reported to result in the highest F1 score. Integrated Gaussian fitting with maximum likelihood is recommended, but computationally costly. Check the multi-emitter fitting analysis to estimate the number of molecules detected as a single blob (fig. 8). Make sure to choose a realistic intensity range per molecule. The intensity range can be determined by localizing with single emitter fit and taking the mean ± standard deviation of the resulting log-transformed photon intensity distribution. Choose a visualization method and finally run the analysis by clicking *Ok*.

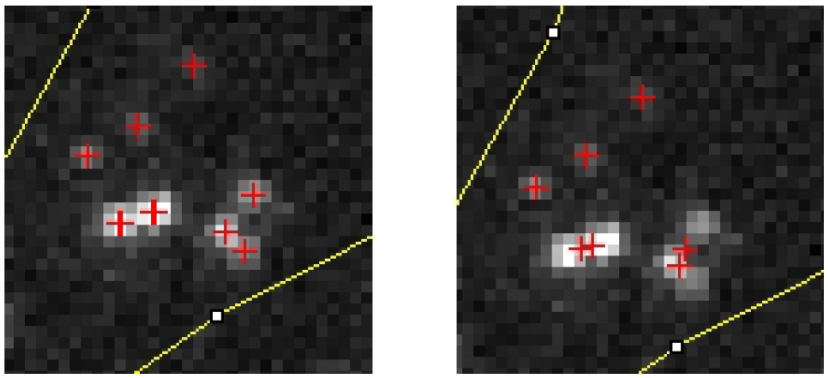

Fig. 8: (left) with multi-emitter fit (right) without multi-emitter fit

[F] https://www.researchgate.net/profile/Martin_Ovesny/publication/262638879_ThunderSTORM_Supplementary_Note_-_User%27s_Guide/links/0f3175384f21d24f56000000/ThunderSTORM-Supplementary-Note-Users-Guide.pdf

[G] http://www.neurocytolab.org/tscolumns/

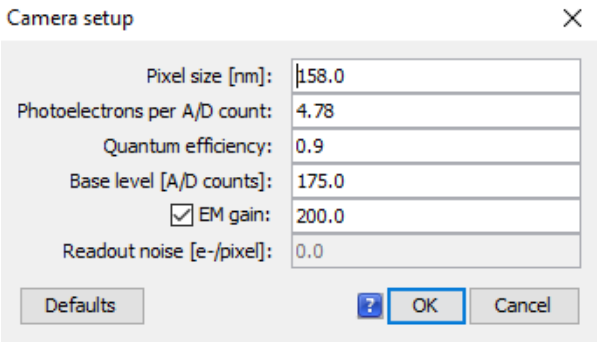

Fig. 9: Example settings for the N-STORM-Setup (Camera Andor iXon Ultra).

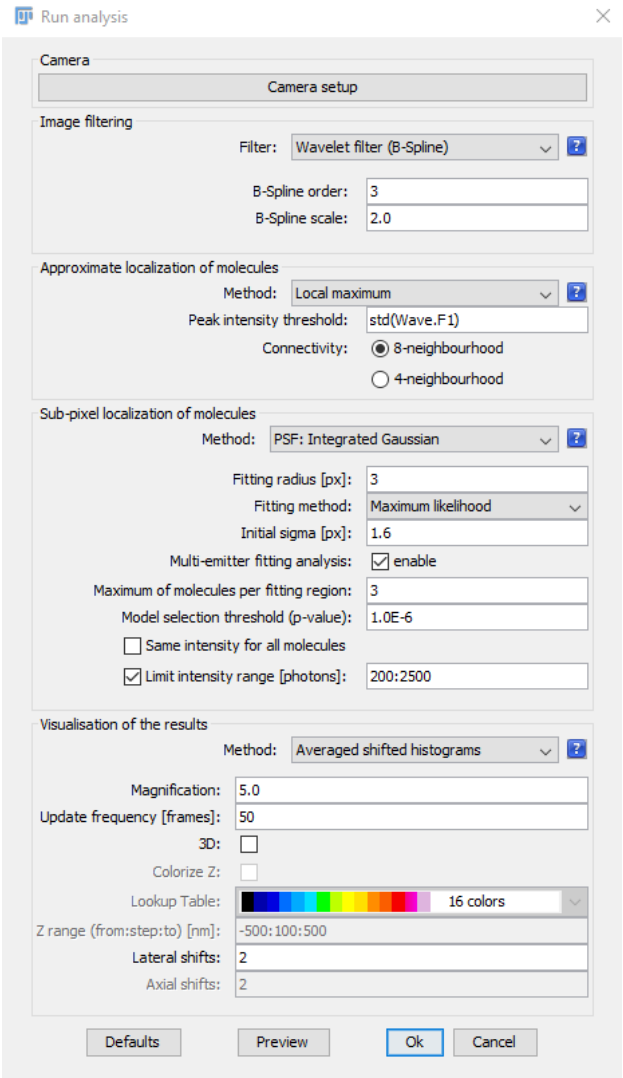

Fig. 10: Recommended ThunderSTORM settings.

The result is a list of localizations (fig. 11). Units must be in [nm] and [photons], and

can be changed by right clicking on the column header. The multi-emitter fit leads to doubled localizations. They can be removed by applying the remove duplicates filter. Export the information as *.csv with all possible columns. Check “Save measurement protocol” to extract a metafile containing information about the settings (fig. 12).

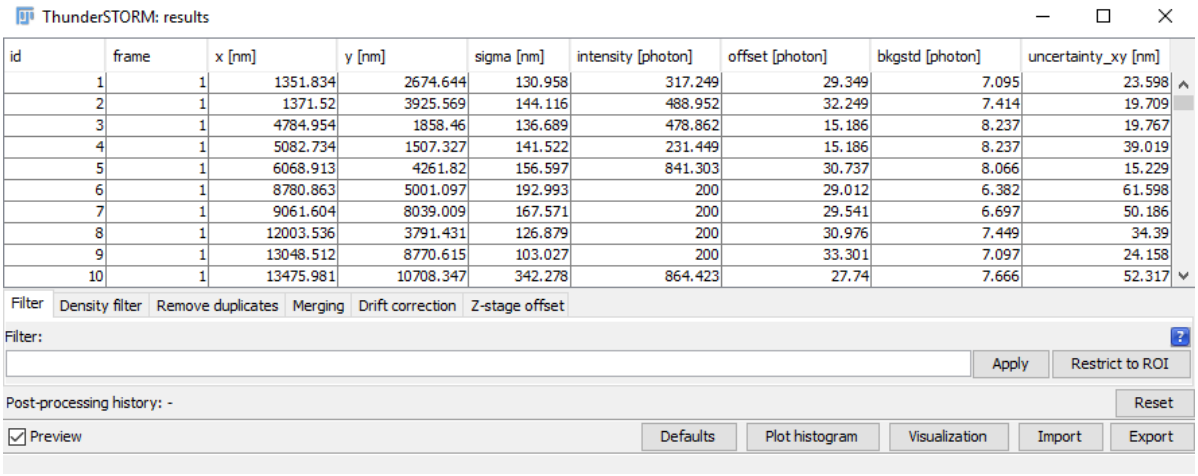

Fig. 11: ThunderSTORM results table with [nm] and [photon] as units.

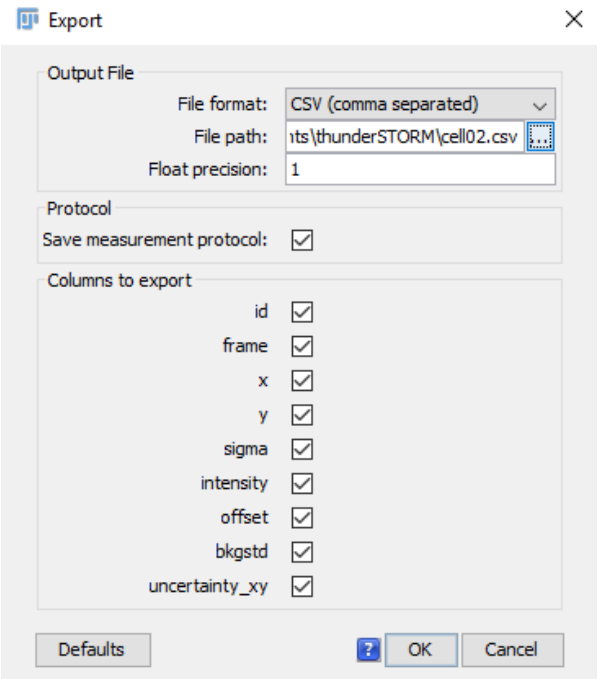

Fig. 12: Export the results as *.csv with all columns and also save the measurement protocol.

2.3 Batch processing of multiple files

Execute the localization analysis for multiple files automatically with a macro in Fiji (listing 4). The macro runs the localization analysis, filters for duplicates, and saves the results.

The file paths of the SPT movies are defined by the target_cells array, the names of the SPT movies by the cell_tif_names array. The *.roi files are added to the target_rois array. The paths to save the results have to be chosen and added to the save_paths array. Make sure to write all paths in quotes and use double backslashes. Check for continuous indexing starting from 0 for all arrays and for matching instances per index (target_cell[0] must be consistent with target_roi[0] ...).

Camera settings cannot be changed via the macro and have to be set correctly before running the analysis.

Listing 4: Fiji macro to run and save the localization analysis in ThunderSTORM for multiple SPT movies.

```
requires("1.45s");
setOption("ExpandableArrays", true);

//Add the file path of the target objects
target_cells = newArray;
target_cells[0] = "C:\\Documents\\200406_Fab_CS2_cell_01.tif"
target_cells[1] = "C:\\Documents\\200406_Fab_CS2_cell_02.tif"

//Specify the tail of the path for each target object (example: "cell_name.tif")
cell_tif_names = newArray;
cell_tif_names[0] = "200406_Fab_CS2_cell_01.tif"
cell_tif_names[1] = "200406_Fab_CS2_cell_02.tif"

//Specify the paths to the *.roi files.
target_rois = newArray;
target_rois[0] = "C:\\Documents\\200406_Fab_CS2_cell_01.roi"
target_rois[1] = "C:\\Documents\\200406_Fab_CS2_cell_02.roi"

//Specify the paths where the analysis should be saved.
save_paths = newArray;
save_paths[0] = "C:\\Users\\pcoffice37\\Desktop\\200406_Fab_CS2_cell_01.csv"
save_paths[1] = "C:\\Users\\pcoffice37\\Desktop\\200406_Fab_CS2_cell_02.csv"

//Run the analysis loop
for (i=0; i<target_cells.length ;i++){
open(target_cells[i]);
roiManager("Open", target_rois[i]);
roiManager("Select", i);
run("Run analysis", "filter=[Wavelet filter (B-Spline)] scale=2.0 order=3
detector=[Local maximum] connectivity=8-neighbourhood threshold=std(Wave.F1)
estimator=[PSF: Integrated Gaussian] sigma=1.6 fitradius=3 method=[Maximum likelihood]
full_image_fitting=false mfaenabled=true keep_same_intensity=false nmax=3
```

```
fixed_intensity=true expected_intensity=200:2500 pvalue=1.0E-6
renderer=[Averaged shifted histograms] magnification=5.0 colorize=false
threed=false shifts=2 repaint=50");
run("Show results table", "action=duplicates distformula=uncertainty_xy");
run("Export results", "floatprecision=1 filepath=" + save_paths[i] +
" fileformat=[CSV (comma separated)] sigma=true intensity=true
chi2=true offset=true saveprotocol=true x=true y=true bkgstd=true
id=true uncertainty_xy=true frame=true");
selectWindow(cell_tif_names[i]);
close();
}
```

2.3.1 Script to create macro

Automatic creation of the macro is possible by executing the python script write_TS_macro.py + config file in Anaconda prompt (fig. 13, listing 5). Define the paths to coverslip directories, the path to the background.roi file, define the intensity range for TS multi-emitter fit and a directory to save the created macro. For this process, the folder structure in sec 1.4 must be followed.

```
[INPUT_DIRS]
# define the directories to coverslips
dir01 = D:\example_path\CS1
dir02 = D:\example_path\CS2

[PARAMETERS]
# define the *.roi file that covers the camera chip area
background_roi = D:\example_path\background.roi
intensity_range = 60:2000

[SAVE_DIR]
# define the save directory for the written fiji macro
macro_directory = D:\example_path
```

Fig. 13: Config file for write_TS_macro.py script. Multiple directories are possible.

Listing 5: Execution of script in Anaconda prompt to write Fiji macro with config file.

```
python write_TS_macro.py write_TS_macro_config.ini
```

3 swift

swift[4] is a tracking software that is compatible with localization lists from ThunderSTORM and rapidSTORM. Tracked files can be further processed in SPTAnalyser with MSD-based diffusion state analysis and transition counting. swift has multiple data and performance parameters that have to be considered for each dataset. Data parameters can be determined with SPTAnalyser (sec. 4.1). More information on the parameters are found in the user manual and in the swift_parameter_overview.pdf file.

3.1 Analysis for one file

Swift can be run with a graphical user interface (swiftgui.exe) and via the command line (swft.exe). Swiftgui has the advantage of visualizing the trajectories and files should be frequently analyzed here to visually check the tracking results. We will focus on the command line approach as it allows batch processing.
Navigate to swft.exe and start the command line from the folder (fig. 14). To process a file, run listing 6. The first command “swft.exe” is starting the tacking programm, the second command is the file path to the localization file. It is mandatory to specify a camera integration time in ms and out_values as “all”. Further parameters can be set, here e.g. the bleaching probability of 0.01. Press enter to start the algorithm. A *.tracked.csv file will be created.

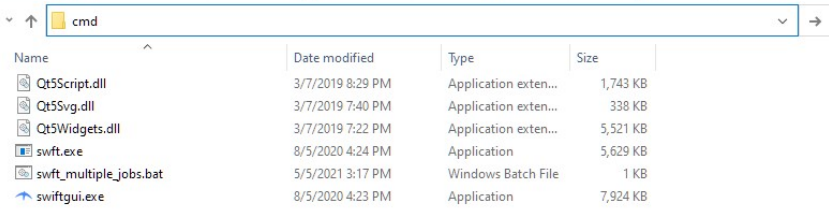

Fig. 14: Start the command line from the directory of the swft.exe by entering ”cmd”.

Listing 6: swift running in the command line to determine pre-analysis parameters.

```
C:\Documents\swift> swft.exe C:\Documents\Fab_CS2_cell01.csv
--precision "26.636" --p_bleach "0.01" --tau "20" --out_values "all"
```

3.2 Batch processing for multiple files

Multiple files are processed at once with a *.bat script. The file paths have to be defined followed by parameter specification (listing 7). The script is executed by double clicking it.

Listing 7: Batch script to run the tracking analysis in swift multiple times with the final parameters.

```
@echo off
set list=C:\\Documents\\data\\Fab_CS2_cell01.csv
set list=%list%;C:\\Documents\\data\\Fab_CS2_cell02.csv
set list=%list%;C:\\Documents\\data\\Fab_CS2_cell03.csv
set list=%list%;C:\\Documents\\data\\Fab_CS2_cell04.csv
set list=%list%;C:\\Documents\\data\\Fab_CS2_cell05.csv

FOR %%A IN (%list%) DO (
swft.exe %%A --tau "20" --p_bleach "0.011"
--exp_displacement "56.222" --precision "26.636"
--out_values "all" --p_switch "0.01"
)

ECHO All swift jobs are done.
PAUSE
```

3.2.1 Script to create batch file

The batch file can be automatically created for all localized files in multiple directory with write_swift_batch.py + write_swift_batch_config.ini (listing 8). In the config file, define the directories to the localized files and global parameters used for all files. For the parameters “precision” and “exp_noise_rate”, it is possible to set the parameters per directory (as in fig. 15) or individually per file by defining the path to the precision.txt and exp_noise_rate.txt files (SPTAnalyser outputs by running precision.ipynb and exp_noise_rate.ipynb, see chapter 4.1).

```
[INPUT_DIRS]
# define the directories to localized files
dir01 = D:\Documents\Chemie_phd\SPTAnalyser\SPT_Data\CS4\cells\tracks
dir02 = D:\Documents\Chemie_phd\SPTAnalyser\SPT_Data\CS5\cells\tracks

[PARAMETERS_GLOBAL]
# define global swift parameters
tau = 20
exp_displacement = 300
max_displacement = 2.5
max_displacement_pp = 3.5
p_bleach = 0.2
p_switch = 0.001
diffraction_limit = 150

[PRECISION_INDIVIDUAL]
# define precision values for each coverslip by defining a number, or for individual target by defining the path to *precision.txt from notebook
precision01 = 25
precision02 = 30

[EXP_NOISE_RATE_INDIVIDUAL]
# define precision values for each coverslip by defining a number, or for individual target by defining the path to *exp_noise_rate.txt from notebook
exp_noise_rate01 = 150
exp_noise_rate02 = 400

[SAVE_BATCH]
# define the path of batch file (.bat ending)
batch_path = D:\Documents\Chemie_phd\SPTAnalyser\swift\Swift_043\swft_multiple_jobs.bat
```

Fig. 15: Config file for write_swift_batch.py script

Listing 8: Execution of script in Anaconda prompt to write fiji macro with config file.

```
python write_swift_batch.py write_swift_batch_config.ini
```

3.2.2 How to execute

The parameter precision can be calculated directly from the localization files. Repeat the process with different files to eventually get a representative localization uncertainty for the measurements. Consider also the other parameters for swift that are not determined via Jupyter Notebooks.

The *.tracked.csv file is loaded in the expDisplacement and pBleach Jupyter Notebook to determine the values of the parameters. The results are inserted in swift and rerun in the command line (listing 9). This process is repeated until the values converge.

Listing 9: swift running in the command line with the additional parameters exp_displacement and p_bleach.

```
C:\Documents\swift> swft.exe C:\Documents\Fab_CS2_cell01.csv
--precision "26.636" --p_switch "0.01" --tau "20" --out_values "all"
--exp_displacement "70.392" --p_bleach "0.006"
```

4 SPTAnalyser

SPTAnalyser is compatible with localization list from ThunderSTORM and rapidSTORM, and tracking files from swift and PALMTracer. SPTAnalyser offers following functionalities: data parameter determination for swift (diffractionLimit.ipynb, expDisplacement.ipynb, expNoiseRate.ipynb, pBleach.ipynb, and precision.ipynb), MSD-based diffusion analysis (trackAnalysis.ipynb, trackStats.ipynb), and transition counting of diffusion states within trajectories (transitionCounts.ipynb). The general process of running a Jupyter Notebook is explained in chapter 1.2.

4.1 Determination of swift parameters

In swift different input parameters are possible. These parameters can be determined using SPTAnalyser. The parameters precision, diffraction_limit and exp_noise_rate should be determined first, as they can be extracted from the localization files before tracking. The exp_displacement and p_bleach parameters are extracted from tracked files and determined in an iterative process. Set all parameters including the previously determined parameters precision, diffraction_limit and exp_noise_rate and make initial estimates or use the default values for exp_displacement and p_bleach. Perform the tracking analysis and determine exp_displacement and p_bleach in their notebooks. Repeat the tracking step with the recalculated values for exp_displacement and p_bleach. After a few iterations (≈3-5x) the values converge (fig. 16). In the following chapters the parameters are explained in detail.

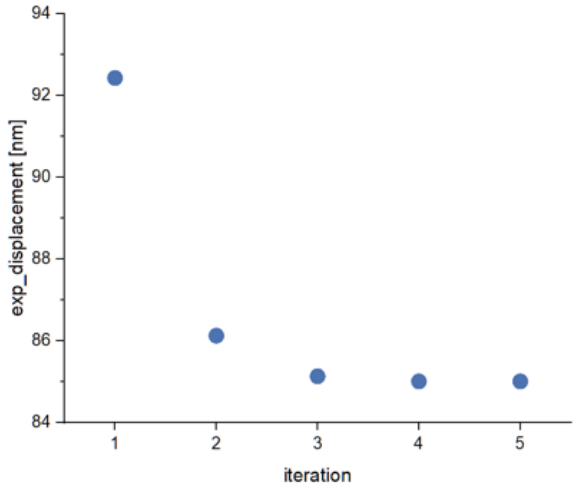

Fig. 16: The exp_displacement value converges after a few iterations of tracking and extracting the value with the expDisplacement.ipynb notebook.

4.1.1 Parameter explanations

precision

The parameter precision is the localization uncertainty in the x/y-plane in nm. The localization list contains a column with the localization uncertainties. To calculate a representative average, all values are ln-transformed to generate a normal distribution (fig. 17). From the ln-transformed values, the arithmetic mean is calculated, re-transformed and used as the parameter precision. It is recommended to calculate the precision for every target, as the localization uncertainty is highly influenced by the imaging plane. In swift, the parameter is relevant for detecting immobile particles as their apparent displacement is a result of the localization error.

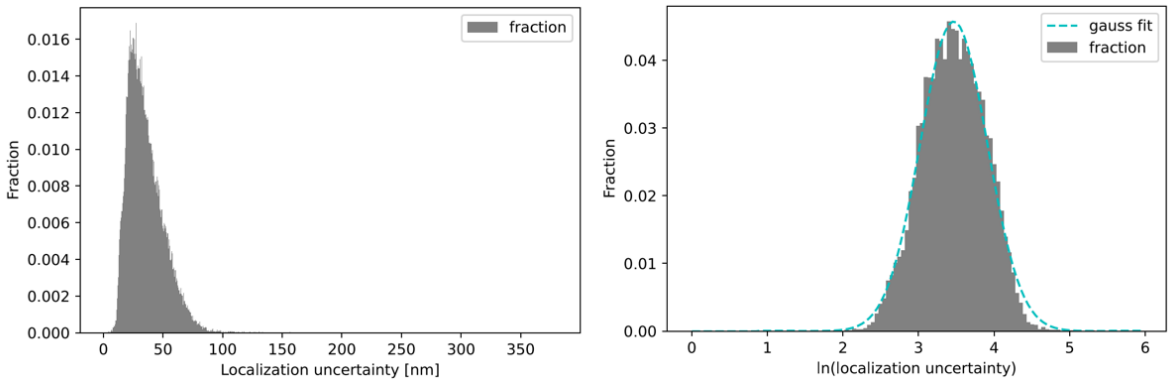

Fig. 17: (left) Histogram of localization uncertainties (right) histogram of logarithmized localization uncertainties. The average of logarithmized localization uncertainties is calculated, re-transformed, and used for the precision. The Gauss fit only assists in evaluating the histogram to be normally distributed in log-space.

exp_displacement

The expected displacement parameter is the expected undirected movement of a particle per frame in x\y direction in nm. The expected displacement is estimated by calculating the average of the mean jump distances (mjd) weighted by the number of data points used to calculate the mean jump distance (mjd_n) of all particles (fig. 18). The average is still a good estimate when multiple subpopulations are present. Immobile particles should not be taken into account since swift version 0.4.0. We recommend to calculate an average value for all targets in a condition. swift might prefer longer connections over shorter once for large expected displacement values.

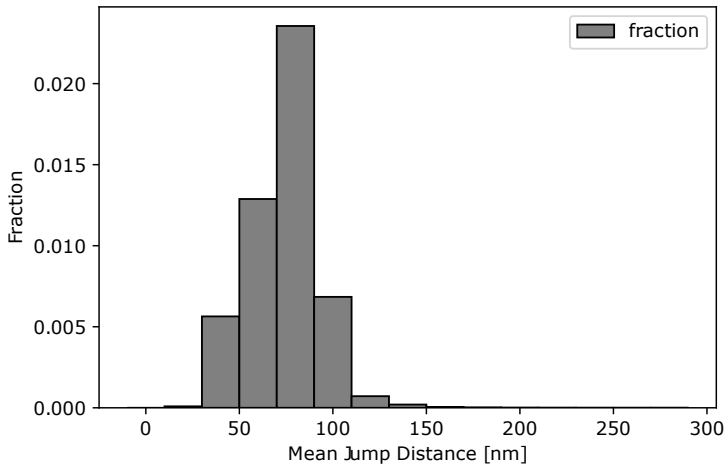

Fig. 18: Histogram of mean jump distances weighted by the lengths of trajectories.

diffraction_limit

The diffraction limit parameter is the minimum distance at which particles can still be distinguished during localization in nm, and is influenced by many variables such as the localization algorithm used. It is determined using a frame-wise nearest neighbor analysis. The minimum nearest neighbor distance of localizations per file is calculated. Multiple files should be analyzed and the minimum distances averaged to estimate this parameter. We recommend using the parameter for all targets in a condition or even for targets from multiple conditions. In swift, the parameter affects the probability of merging and splitting of trajectories, with high values promoting these occurrences.

p_bleach

p_bleach is the probability per particle and frame to bleach, $\in [0; 1]$. The number of trajectories that exist for a given time period are subtracted from the normalized sum of the total number of trajectories and plotted as a histogram (fig. 19). The decay rate is extracted with an exponential fit (eq. 1), with an initial a value (=1), an initial k value in s^{-1} (=0.5), and the camera integration time Δt in s. The cumulative distribution function of an exponential distribution is used to determine p_bleach (eq. 2). It is possible to mask a number of values in the histogram, starting from time step = 0, because some data deviate from a monoexponential decay especially in the first time steps. The remaining data is normalized and fitted as usual. In swift, the parameter influences the willingness of making connections. The higher the value, the more reluctantly connections are made, as shorter life times of trajectories are expected.

$$f(t; k; a) = a \cdot e^{-\Delta t \cdot k} \tag{1}$$

$$F(\Delta t; k) = 1 - e^{-\Delta t \cdot k} \tag{2}$$

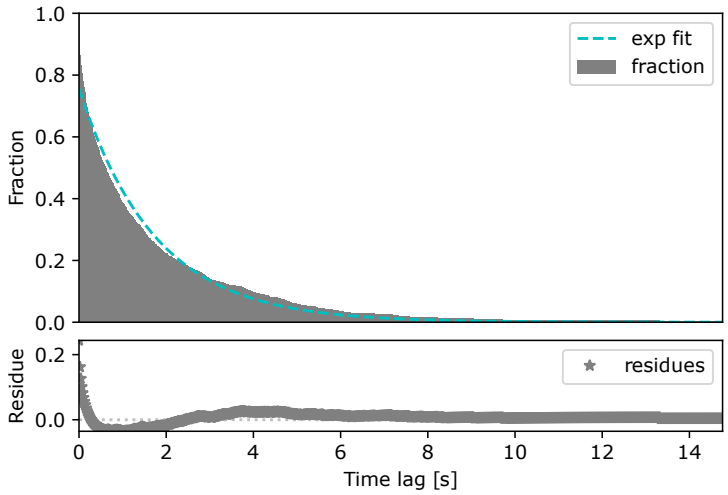

Fig. 19: The amount of trajectories existing after a time period is exponentially fitted and the decay rate used to calculate p_bleach.

exp_noise_rate

The expected noise rate estimates the percentage of false positives among all localizations in percent. The number of localizations per px^2 and frame is determined and averaged for the background measurements. The number of localizations per px^2 and frame is determined for the cell measurements. The expected noise rate is calculated by dividing the background average density through the individual cell density multiplyied by 100 (eq. 3). It is recommended to calculate the expected noise rate for each target as the parameter is influenced by localization density. In swift, the parameter is used in combination with the intensity of a localization to determine its individual likelihood to be noise. False positive points should not be connected into trajectories.

$$\frac{< density background >}{density cell_i} \cdot 100 \tag{3}$$

4.2 Diffusion analysis with SPTAnalyser

The diffusion analysis is executed in two Jupyter Notebooks: TrackAnalysis and TrackStatistics. The first notebook calculates diffusion coefficients and determines between the diffusion type immobile, confined, free, and notype according to Michalet *et al.*,[6] Rossier *et al.*[7] and Harwardt *et al.*[8] The results are saved in a h5 file. The calculations might take several minutes depending on the size of the dataset. The second notebook

reloads the information from the created h5 files. Here, the data can be analyzed in detail. Global information on the tracking results is available as well as filtering according to different criteria.

4.2.1 TrackAnalysis

TrackAnalysis handles tracked files from swift and PALMTracer. A directory from which all tracked files with fitting file ending are loaded has to be chosen (swift: *.tracked.csv, PALMTracer: *.trc).
PALMTracer creates multiple *.trc files and some of them do not contain tracking information. Either copy all *.trc files that contain the tracking information in one folder or use the option "mask words" that will ignore all files with the fitting file type but containing at least one of the mask words in the file name. Comma separate multiple mask words. Typically, mask words only have to be considered for PALMTracer data.

For the analysis, several parameters must be set. Therefore, files from one measurement condition should be analyzed at once. Parameters are the pixel size in nm, the amount of pixels on the camera, the camera integration time in s, the cell sizes, the number of MSD values that are considered for the linear fit to determine the diffusion coefficient, the % of the MSD values that are used for the fit that distinguishes between confined and free trajectories, the degree of freedom (2D tracking = 4), the minimal detectable diffusion coefficient, and the minimal trajectory length (all trajectories shorter are ignored) have to be set.

A specialty of swift is the distinction of different diffusion modes within a trajectory called segments (fig. 20). It is recommended to execute the diffusion analysis with segments instead of complete trajectories. However, PALMTracer data does not differentiate different diffusion modes and can only be executed on the track level.

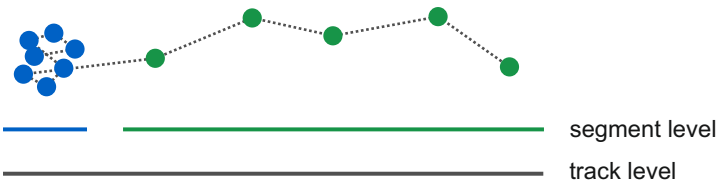

Fig. 20: Track and segment level of a trajectory. Segments represent different diffusion modes. The track level comprises the segments and represents the whole trajectory.

Get detailed information about a trajectory with "choose trajectory to plot". "Plot global diffusion histogram and MSD plot" yields a diffusion histogram and a MSD plot avera-

ged per diffusion state and cell. The diffusion coefficients are represented in a histogram with the normalized frequency plotted against the diffusion coefficient with logarithmic bin size (fig. 21, 22). Frequency counts of diffusion coefficients from different cells are normalized to their cell sizes.

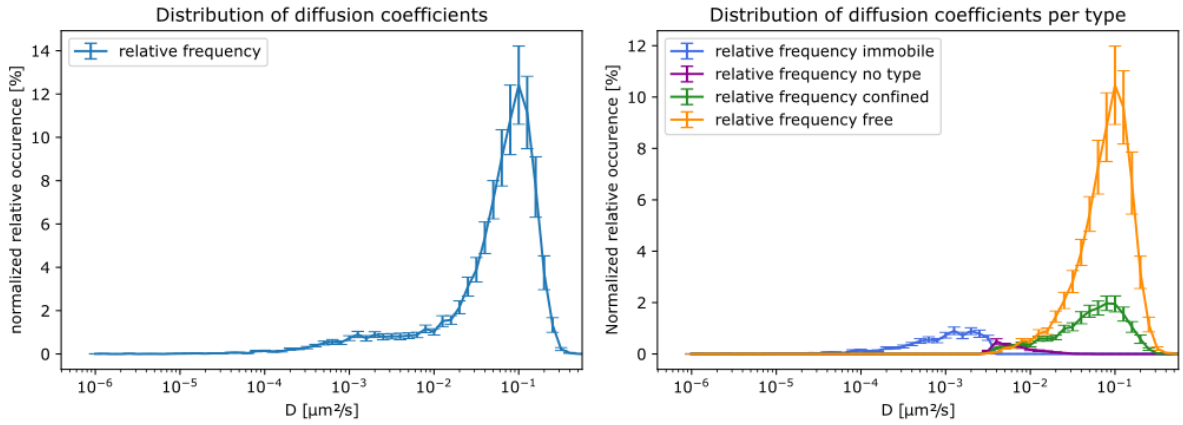

Fig. 21: Histograms of the diffusion coefficients from all analyzed files. The normalized occurrence in % is plotted against the diffusion coefficients in $\frac{\mu m^2}{s}$ with standard error of the means (SEM) and logarithmic bins.

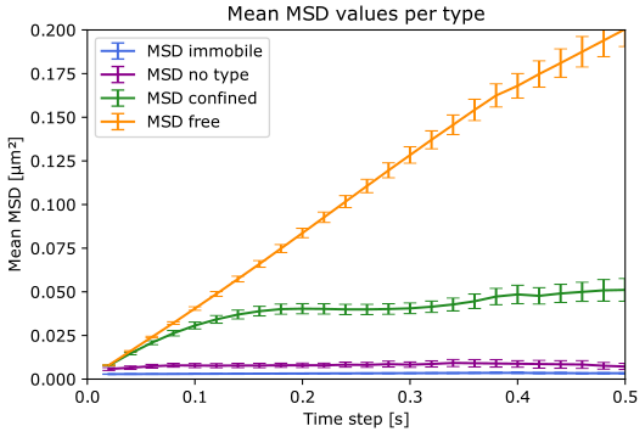

Fig. 22: Average MSD values per type and cell.

Define cell sizes

Define the cell areas with a *.LOG file (fig. 23), where the first column defines the file names that must match the file names of the tracked files (with or without file ending, both is possible) and the second column the cell area in px^2. If tracked files could not be matched with names in the list of the cell areas, the size of the field of view is used

as target size. PALMTracer automatically creates a file in the tracking routine. For the other programs, a *.LOG file can be created with the script write_cell_size_log.py + the config file write_cell_size_log_config.ini for multiple coverslips at once (fig. 24, listing 10).

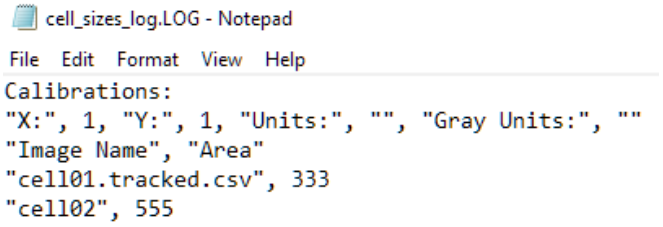

Fig. 23: Define cell sizes in pixels^2 in the PALMTracer .LOG file format.

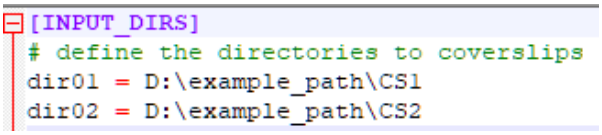

Fig. 24: Config file for write_cell_size_log.py script. Multiple directories are possible.

Listing 10: Execution of script in Anaconda prompt to write log file with cell sizes for SPTAnalyser.

```
python write_cell_size_log.py write_cell_size_log_config.ini
```

4.2.2 TrackStatistics

The *.h5 files from trackAnalysis are loaded in the trackStatistics Jupyter Notebook by choosing a directory (all *.h5 files within the folder and subfolders are loaded). Filtering options are available (fig. 25). Trajectories can be discarded by a minimal and maximal trajectory length, a minimal and maximal diffusion coefficient, and based on their diffusion type immobile, confined, free, and no successful type determination. The filtered dataset can be visualized similarly to TrackAnalysis and stored in *.h5 files along with some statistical informations (see section 4.2.5). The diffusion types immobile and notype are treated separately and statistic information is calculated for four groups (immobile, confined, free, and notype). Additionally, the immobile and notype fraction are merged to one group and statistic information is calculated for three groups (immobile+notype, confined, and free).

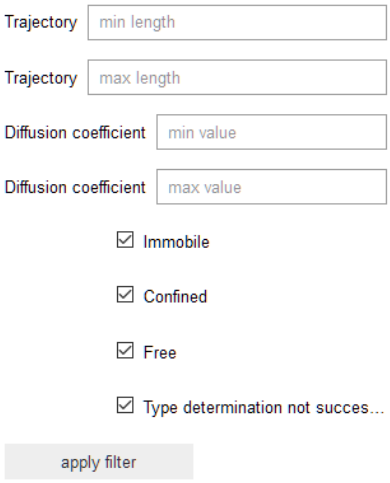

Fig. 25: Filter options in TrackStatistics.

4.2.3 Determination of D_{min}

A D_{min} value based on the filtered data is required to determine if a particle is immobile. All trajectories/segments with a diffusion coefficient less than D_{min} are labeled as immobile. D_{min} is derived from the dynamic localization error σ_{dyn} and can be easily computed by using the trackAnalysis and trackStatistics.
The localized files are loaded in trackAnalysis and analyzed without considering the default D_{min} value. The resulting *.h5 files are loaded into the TrackStatistics Jupyter Notebook and filtered. Based on the filtered dataset, σ_{dyn} is calculated per cell. The third quartile of σ_{dyn} results in a representative D_{min} value.

4.2.4 Background correction

Background correction is applied to the diffusion coefficient histogram (this is optional). Background measurements are analyzed like normal measurements and h5 files created with trackAnalysis. In the trackStatistics Jupyter Notebook, background measurements can be loaded under “load background .h5 files”. Their counts in the histogram are subtracted from the counts of the measurements. Both a background-corrected diffusion histogram and a histogram without correction are saved.

4.2.5 *.h5 file information

TrackAnalysis

"MSD" contains MSD Δt value pairs of all trajectories.

"diffusion" → "diffusionInfos" contains the trajectory id, diffusion coefficient and error (derived from linear fit uncertainty), y-intercept MSD(0), chi^2 (values vs linear fit) and trajectory length. "diffusion" → "diffusionPlots" contains the first n MSD Δt value pairs, the linear fit and the residues.

"rossier" → "rossierStatistics" contains the trajectory id, the diffusion type (0=False, 1=True) and the parameters from eq. 7 to distinguish between confined and free diffusion, r = confinement radius, $D_{confined}$ = confined diffusion coefficient, chi^2. "rossier" → "rossierPlots" contains the the ath fraction of MSD Δt value pairs, the fit from eq. 7 and the residues.

"settings" contains the camera integration time dt, pixel size, cell size, $\tau_{\text{threshold}}$ from eq. 8, min trajectory length, fraction a of the MSD values fitted with eq. 7, degrees of freedom dof, dynamic diffusion error based on eq. 5 and if analysis is based on track or segment (fig. 20).

"statistics" → "statistics_4" references to the four diffusion types immobile, confined, free, and notype. This section contains the type % within the cell, the mean diffusion coefficients, trajectory lengths and respective SEMs per diffusion type.

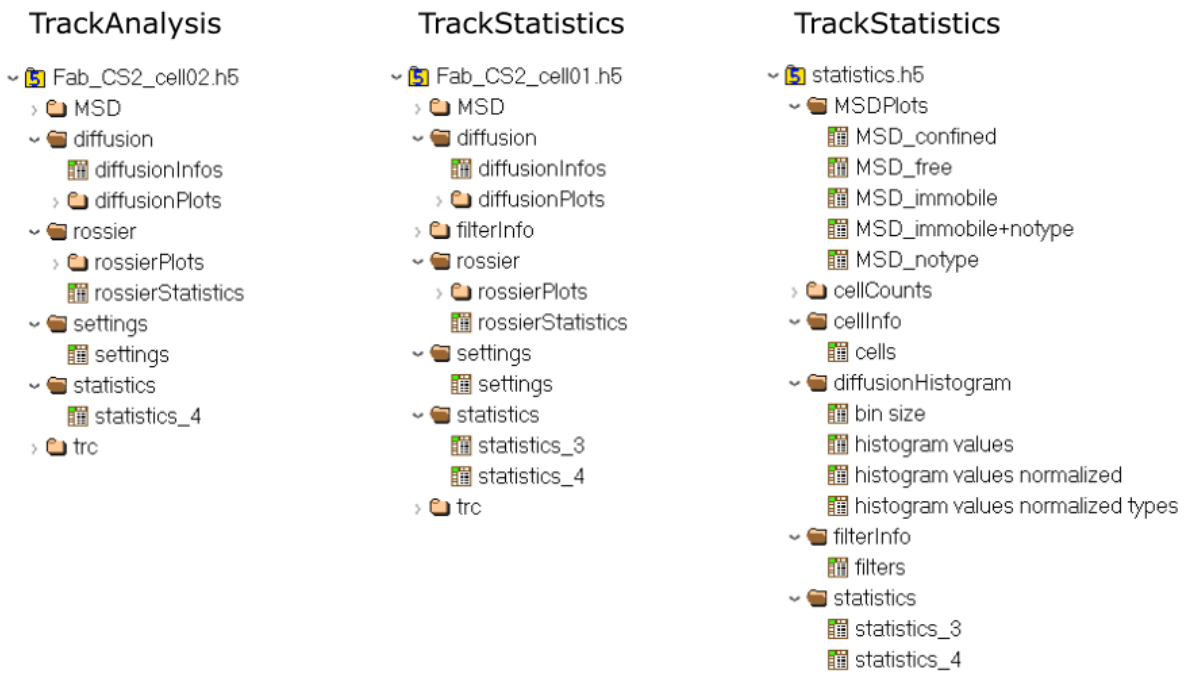

Fig. 26: TrackAnalysis saves a *.h5 file per cell. TrackStatistics saves a *.h5 file per cell containing the filtered trajectories and a statistics file that gives an overview of the filtered data of all loaded files in TrackStatistics.

TrackStatistics

In TrackStatistics the same file per cell like in TrackAnalysis is created. This file contains only the filtered trajectories and additional filter information in "filterInfo". "statistics" → "statistics_4" references to the four diffusion types immobile, confined, free and no-type. In "statistics_3" all immobile and notype trajectories are considered as immobile.

Additionally, a statistics file is created that summarizes information of all *:h5 files that were loaded into TrackStatistics.
"MSDPlots" contains the average of all MSD values per type and cell. "cellCounts" contains the frequency counts / area of diffusion coefficients with logarithmic bin size per cell. "cellInfo" contains the names of the loaded *.h5 files and the cell sizes. "diffusionHistogram" contains the bin size in log-space, the mean histogram values and SEM (not normalized, normalized, distinguished per type and normalized). "filterInfo" summarizes the applied filters. "statistics" contains the averages of diffusion types, diffusion coefficients and trajectory lengths. → "statistics_4" references to the four diffusion types immobile, confined, free and notype. In "statistics_3" all immobile and notype trajectories are considered as immobile.

4.2.6 *.h5 file extraction

Values regarding defined columns of a *.h5 file can be extracted with the extract_h5_files.py script for multiple files. Define a folder that contains the *.h5 files. Define the folder, file name, and column index of the target column, for example “rossier”, “rossierStatistics”, “8”, to extract the confined radii. The confined radii will be saved in a *_values.csv file, each column is a target. An additinal file *_mean.csv is created that contains mean, standard deviation and standard error per target column. If the target column is a single value (for example “statistics“, “statistics_3”, “1”, to target the percentage value of immobile and notype segments) the values are saved in a *_single_values.csv file with standard deviation and standard error. The parameter mask_files allows to define *.h5 file names that will be ignored. Define a folder and a file name to save results.

```
[INPUT_DIR]
# define the folder that contains the hdf5 files (results from trackAnalysis / trackStatistics)
dir = D:\example_folder

[PARAMETERS]
# define the folder file and column index (start counting from 1),
# mask files are h5 files with matching name in directory that will be ignored
folder_name = statistics
file_name = statistics_3
column_idx = 8
mask_files = [statistics, Target1]

[SAVE]
# define the directory and file name for result saving
save_dir = D:\example_folder
save_name = results
```

Fig. 27: Config file for extract_h5_files.py script.

4.2.7 Calculations of diffusion parameters

For a more detailed explanation check out the literature.[6–8]

Diffusion coefficient

The diffusion coefficient for individual trajectories is obtained by fitting the first n points in the MSD plot with eq. 4, dof = degree of freedom, dof(2D diffusion) = 4.

$$MSD(\Delta t) = dof \cdot D \cdot \Delta t \tag{4}$$

Diffusion type immobile

Based on the dynamic localization precision, a D_{min} threshold is determined. All trajectories with a diffusion coefficient smaller than D_{min} are classified as immobile.

The dynamic localization precision is determined per cell with eq. 5 by averaging the y-intercepts and the diffusion coefficients of all trajectories within the cell. The third quartile of dynamic localization precisions is plugged into eq 6 to calculate D_{min}.

$$\sigma_{dyn} = \sqrt{\frac{\langle MSD(0)\rangle + \frac{dof}{3}\langle D\rangle \cdot dt}{dof}} \tag{5}$$

$$D_{min} = \frac{\sigma_{dyn}^2}{dof \cdot n \cdot dt} \tag{6}$$

Diffusion types confined and free

Particles that are not immobile are categorized as confined or free based on the analysis by Rossier *et al.*[7] For each trajectory a fraction a of the MSD values are fitted with eq 7 and compared to a $\tau_{threshold}$ value. If $\tau < \tau_{threshold}$ the particle moves confined, if $\tau > \tau_{threshold}$ the particle moves freely. In some cases this fit does not converge and particles are labeled with no diffusion type. $\tau_{threshold}$ depends on the camera integration time dt, the fraction of MSD values a, the minimum length of trajectories l_{min}, and a factor 0.5.

$$MSD(\Delta t) = \frac{4}{3}r_c^2 \cdot (1 - e^{-\Delta t/\tau}) \quad \rightarrow \quad \tau = \frac{r_c^2}{3D_c} \tag{7}$$

$$\tau_{threshold} = dt \cdot a \cdot l_{min} \cdot 0.5 \tag{8}$$

4.2.8 Statistics

Mean values and their SEMs are calculated per cell (eq. 9 and 10). A global value is calculated by averaging the mean values of the cells (eq. 11). The error of the global value is calculated by averaging the SEMs of the cells, according to Gaussian error propagation (eq. 12).

$$Y_i = \frac{1}{M}\sum_{j=1}^{M} y_j \tag{9}$$

$$\Delta Y_i = \frac{\sqrt{\frac{(m_j - \bar{m})^2}{M-1}}}{\sqrt{M}} \tag{10}$$

$$Y_{global} = \frac{1}{N}\sum_{i=1}^{N} Y_i \tag{11}$$

$$\Delta Y_{global} = \frac{1}{N}\sum_{i=1}^{N}\frac{\delta Y_{global}}{\delta Y_i}\cdot\Delta Y_i = \frac{1}{N}\sum_{i=1}^{N}\Delta Y_i \tag{12}$$

4.3 Transition counts

In this notebook, the transitions of diffusion states within of segments within trajectories are counted. Required input files are matching (=same file name) *.h5 files form trackStatistics and *.tracked.csv files from swift (fig. 28).

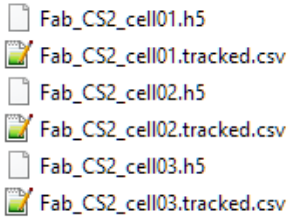

Fig. 28: Required input files for transition counting. Define the directory path containing matching files (=same filename) from trackStatistics (*.h5) and swift (*.tracked.csv).

For each single trajectory in which at least two segments were identified, the transition of the diffusion state between the segments is determined. For the three diffusion states of immobile (i), confined (c), and freely diffusing (f) particles, nine different transitions are distinguished: i-i, i-c, i-f, c-i, c-c, c-f, f-i, f-c, f-f. The number of diffusion state can be set to 4 to investigate no-type transitions. Unclassified segments that occurred between two segments can be neglected by defining a mask value. All segments with a length <= mask value in frames are skipped, and the transition between the segment before and after is counted. A mask value of 0 means no masking. A recommendation is to set the mask value to the classification threshold of diffusion states (e.g. if the diffusion type of segments with a minimal length of 20 frames is classified, mask value = 19). Results of transition counting are shown in fig. 29. Fig. 29 (A) shows the absolute counts per cell. In fig. 29 (B), transition counts were normalized per cell and summed to one to compare the occurrences of transition types. In fig. 29 (C), transition counts were normalized per diffusion state so that the counts of transition types proceeding from the same diffusion state sum up to one to compare the occurrences of diffusion states in adjacent segments. Further plots show the length of segments in frames, the number of segments per trajectory, and the number of counts without / with transitions that involve short, unclassified segments (fig. 30).

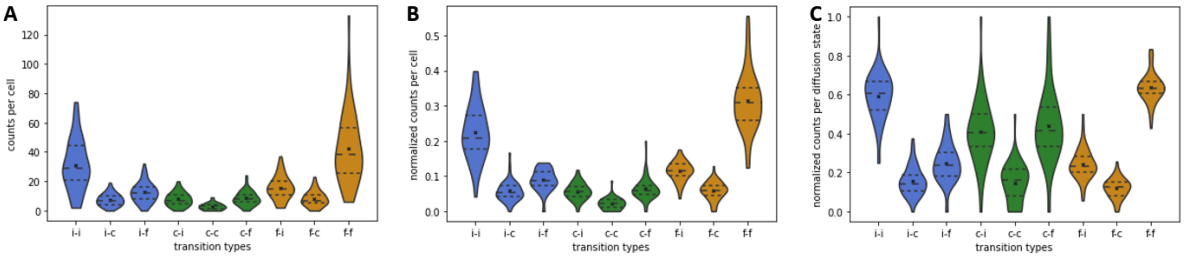

Fig. 29: Transiton counting results in (A) absolute counts, (B) counts normalized to one per cell, and (C) counts normalized to one per diffusion state.

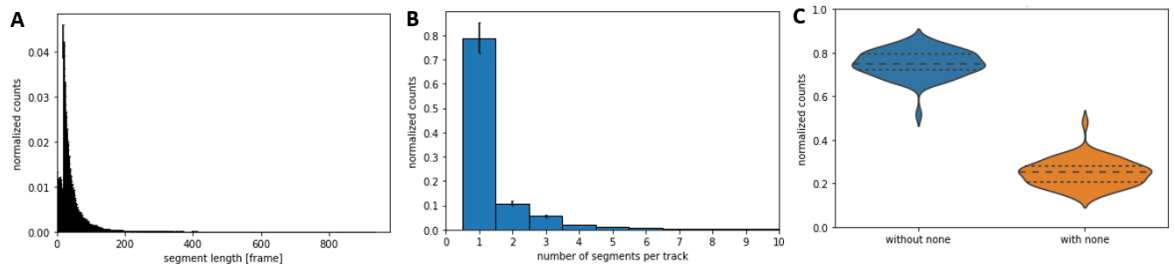

Fig. 30: (A) Lengths of segments within trajectories that contain at least one classified segment. (B) Number of segments per trajectory. (C) Relative occurence of transition counts without unclassified segments and with classified segments.

Plot axis ranges can be adjusted via the code cells (fig. 31). Significance tests of pairwise comparisons of the transition types are automatically saved and use Wilcoxon signed rank tests (SciPy). Levels of significance are classified as follows: $p > 0.05$ no significant difference (n.s.), $p < 0.05$ significant difference (*), $p < 0.01$ very significant difference (**), $p < 0.001$ highly significant difference (***).

```
# adjust axis min max limits of plots here, e.g. ylim=[0,1]
stats.vis_counts(counts="absolute", norm="absolute", ylim=[None, None])
stats.vis_counts(counts="absolute", norm="global", ylim=[None, None])
stats.vis_counts(counts="absolute", norm="split", ylim=[None, None])
stats.segment_lengths_plot(xlim=[0,None], ylim=[0, None])
stats.segments_per_trajectory_plot(xlim=[0,10], ylim=[0, None])
stats.transitions_wo_none_plot(ylim=[0,1])
```

Fig. 31: Show code cells and adjust the axis ranges via xlim and ylim parameters.

5 References

[1] M. Ovesny, P. Krizek, J. Borkovec, Z. Svindrych, G. M. Hagen, *Bioinformatics* **2014**, *30*, 2389–2390.

[2] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, A. Cardona, *Nat. Methods* **2012**, *9*, 676–682.

[3] S. Wolter, A. Löschberger, T. Holm, S. Aufmkolk, M.-C. Dabauvalle, S. van de Linde, M. Sauer, *Nat. Methods* **2012**, *9*, 1040–1041.

[4] M. Endesfelder, C. Schießl, B. Turkowyd, T. Lechner, U. Endesfelder, manuscript in prep.

[5] A. D. Edelstein, M. A. Tsuchida, N. Amodaj, H. Pinkard, R. D. Vale, N. Stuurman, *J. Biol. Methods* **2014**, *1*, e10.

[6] X. Michalet, *Phys. Rev. E* **2010**, *82*, 041914.

[7] O. Rossier, V. Octeau, J.-B. Sibarita, C. Leduc, B. Tessier, D. Nair, V. Gatterdam, O. Destaing, C. Albigès-Rizo, R. Tampé, L. Cognet, D. Choquet, B. Lounis, G. Giannone, *Nat. Cell Biol.* **2012**, *14*, 1057–1067.

[8] M.-L. I. E. Harwardt, P. Young, W. M. Bleymüller, T. Meyer, C. Karathanasis, H. H. Niemann, M. Heilemann, M. S. Dietz, *FEBS Open Bio* **2017**, *7*, 1422–1440.