Tracking Routine Guide 14. August 2021

1 Getting started

This tracking routine uses *ThunderSTORM* to localize particles, a plugin for *fiji*.[1,2] The localizations can be found with *rapidSTORM* as well.[3] The particles are connected to trajectories with *swift*.[4] Swift is currently in development and new versions appear frequently. It is recommended to use the latest version. Parameter determination for *swift* and further data analysis is realized in *SPTAnalyser*. *SPTAnalyser* also handles *PALMTracer* data, a program that both localizes and connects the localizations.[A] The connections between the different softwares are visualized in fig. 1.
The *ThunderSTORM*.jar plugin has to be placed in the plugins folder of *fiji*.[B] Jupyterlab, the *SPTAnalyser* wheel and hide-code has to be installed. Navigate to the directory with the *SPTAnalyser* wheel and execute listing 1 in the anaconda promt as admin.[C] Install a HDF viewer.[D]

Listing 1: Commands to install jupyterlab *SPTAnalyser* and hide codes in the command line.

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

[A]https://www.iins.u-bordeaux.fr/team-sibarita-PALMTracer
[B]https://github.com/zitmen/thunderstorm/releases/tag/dev-2016-09-10-b1
[C]https://github.com/kirbs-/hide_code
[D]https://www.hdfgroup.org/downloads/hdfview/

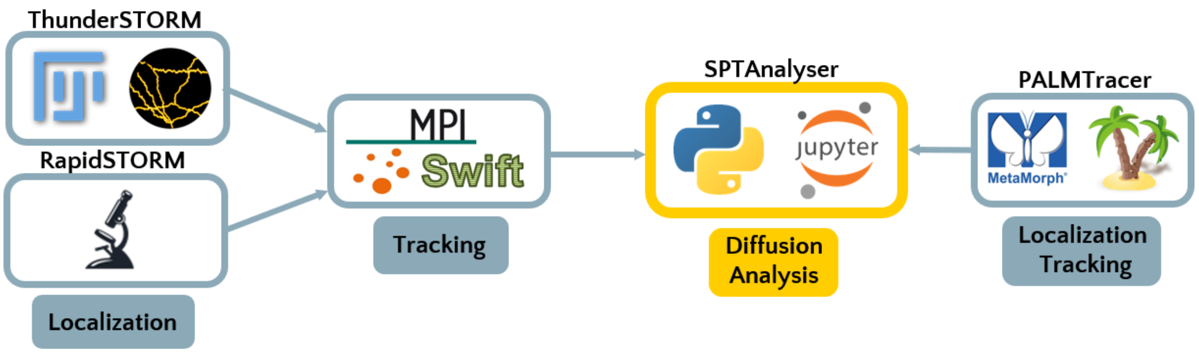


Fig. 1: Connections between the softwares. *Swift* is compatible with *ThunderSTORM* and *rapidSTORM* files. *SPTAnalyser* is compatible with *swift* and *PALMTracer* files.

2 Folder structure

Following folder structure is recommended for the workflow using ThunderSTORM, swift and SPTAnalyser (fig. 2). Individual folders contain data of each measured coverslip (CS) with background measurements (optinal), cell measurements and some swift preprocessing parameters that are determined per coverslip. Each cell folder (applies to background folder as well) contains rois, tifs and tracks. The roi folder consists of *.roi files, which mark the region of interest (ROI) in ThunderSTORM and *_size.csv, the measured size of the ROI. The tifs folder contains recorded *.tif movies and transmitted light images *_dl.tif. The tracks folder contains the localization file of ThunderSTORM which will adapt the *.tif file name of the measurement file, 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.

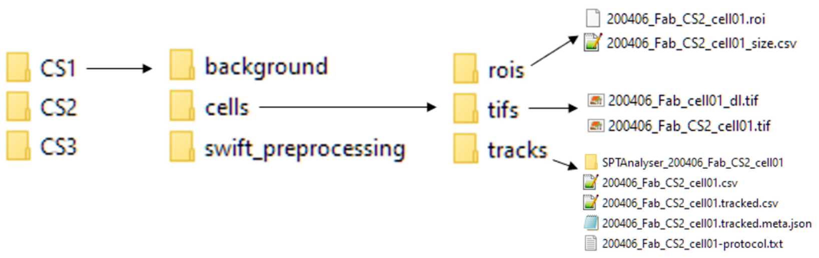
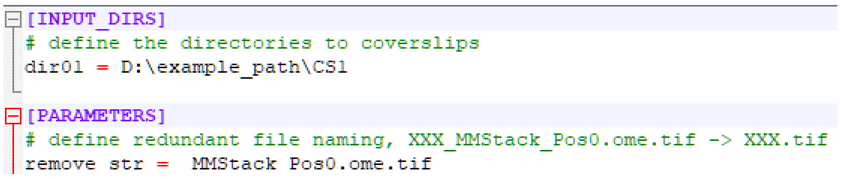

CS1
CS2
CS3
background
cells
swift_preprocessing
rois
tifs
tracks
200406_Fab_CS2_cell01.roi
200406_Fab_CS2_cell01_size.csv
200406_Fab_cell01_dl.tif
200406_Fab_CS2_cell01.tif
SPTAnalyser_200406_Fab_CS2_cell01
200406_Fab_CS2_cell01.csv
200406_Fab_CS2_cell01.tracked.csv
200406_Fab_CS2_cell01.tracked.meta.json
200406_Fab_CS2_cell01-protocol.txt

Fig. 2: Folderstructure created in the course of tracking routine.

2.1 Script to create structured folder

After measurement the *.tif files are distributed over multiple folders. This folder structure is cleaned up with a script and the resulting structure necessary for further batch processing. Use the adapt_folder_structure.py script with a config file (fig. 3) to automatically create the structure in fig. 2. Define the paths to coverslip directories and define possible redundant names in file (remove_str = ...). The string defined in remove_str will be removed. Example: cell01_MMStack_Pos0.ome.tif will be renamed to cell01.tif. If remove_str is empty, the original file names will be kept. Execute the script with listing 2. In order to use the script, the background measurements must contain „background", the cell measurements must contain „cell" and the transmitted light images must contain „tl" or „dl" in their file name.



```
[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. 3: Config file for adapt_folder_structure.py script. Multiple directories are possible.

Listing 2: Execution of script in anaconda shell to create folder structure in fig. 2.

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

3 *ThunderSTORM*

Load the SPT movie file and a transmitted light image of the cell into *fiji*. Circle the region of interest with a selection tool of your choice (fig. 4). Click on the transmitted light image window and open the ROI-manager (press *t* on the keyboard). Select the *.tif image and click on the area in the ROI-manager to transfer the selected area to the *.tif file. Click at *Measure* to calculate the area of the region in px and save this information (*_size.csv). Save the region as *.roi (*More* → *Save...*). Both will be needed at later points of the analysis.

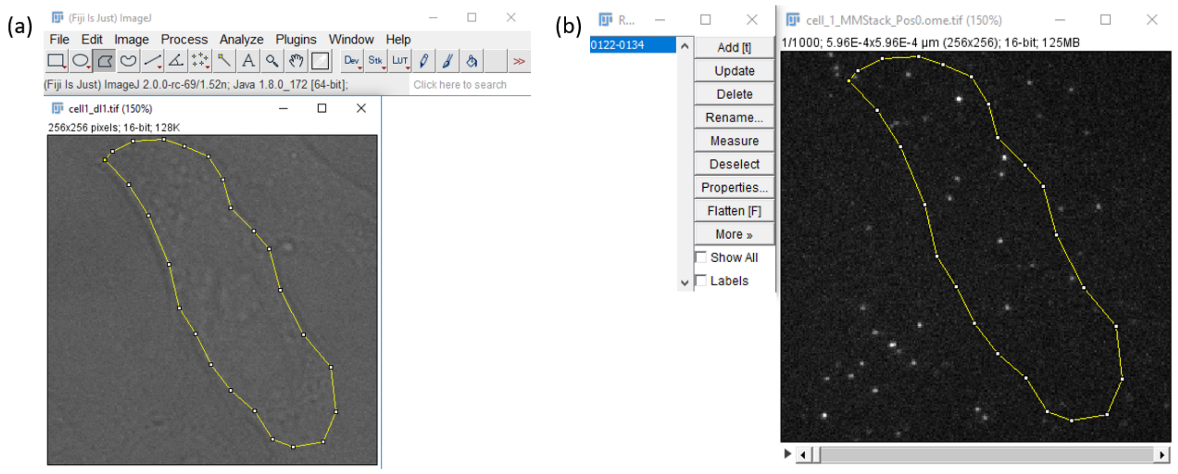

(a)
(Fiji Is Just) ImageJ
File Edit Image Process Analyze Plugins Window Help
(Fiji Is Just) ImageJ 2.0.0-rc-69/1.52n; Java 1.8.0_172 [64-bit];
Click here to search
cell1_dl1.tif (150%)
256x256 pixels; 16-bit; 128K
(b)
0122-0134
Add [t]
Update
Delete
Rename...
Measure
Deselect
Properties...
Flatten [F]
More »
Show All
Labels
cell_1_MMStack_Pos0.ome.tif (150%)
1/1000; 5.96E-4x5.96E-4 µm (256x256); 16-bit; 125MB

Fig. 4: (a) Selected region of a transmitted light cell image with the polygon selection tool. (b) Transferred region into a single particle tracking (SPT) movie.

Run the analysis with *Plugins* → *ThunderSTORM* → *Run analysis* (fig. 7). The following will briefly introduce the settings, for more information read the *ThunderSTORM* user guide.[E,F]
The camera setup has to be specified (fig. 6). 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. 5). 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*.

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

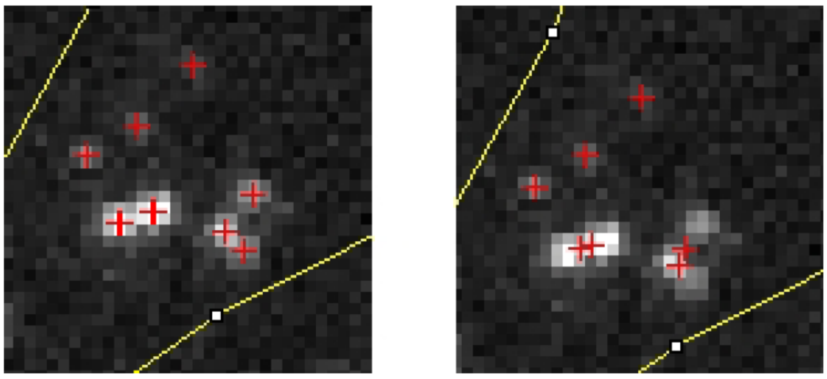
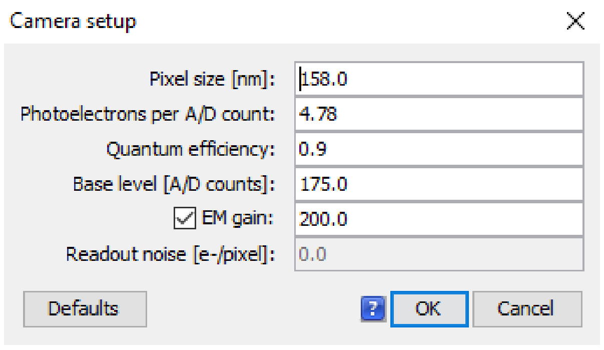


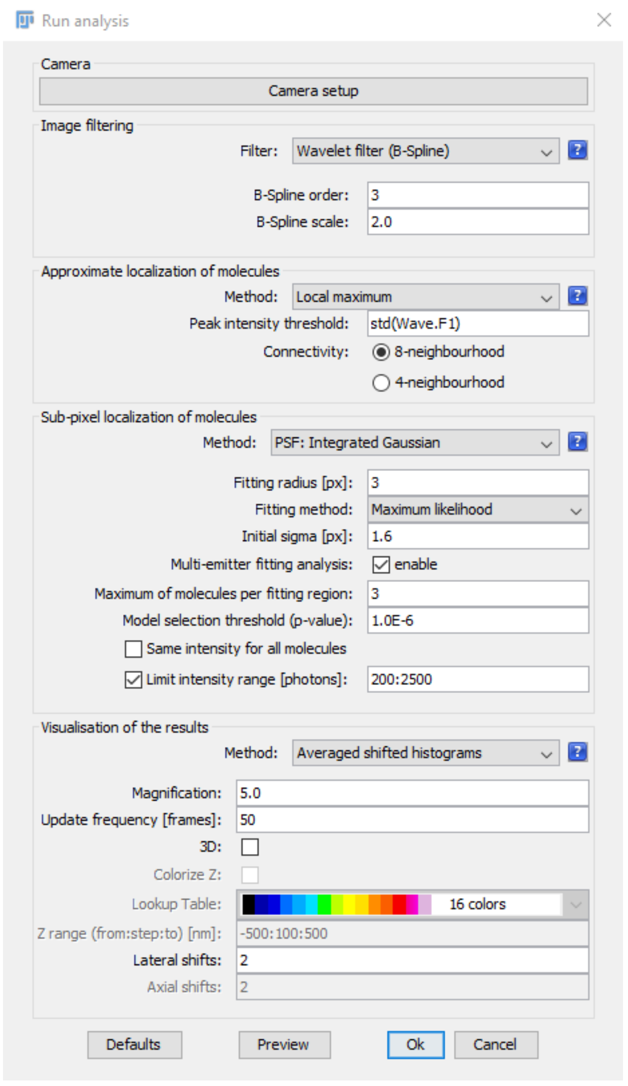
Fig. 5: (left) with multi-emitter fit (right) without multi-emitter fit



Camera setup

Pixel size [nm]:	158.0
Photoelectrons per A/D count:	4.78
Quantum efficiency:	0.9
Base level [A/D counts]:	175.0
☑ EM gain:	200.0
Readout noise [e-/pixel]:	0.0

Fig. 6: Example settings for the NSTORM-Setup (Camera X8884).



Run analysis

Camera

Camera setup

Image filtering

Filter: Wavelet filter (B-Spline)

B-Spline order: 3

B-Spline scale: 2.0

Approximate localization of molecules

Method: Local maximum

Peak intensity threshold: std(Wave.F1)

Connectivity: [✓ 8-neighbourhood] 4-neighbourhood

Sub-pixel localization of molecules

Method: PSF: Integrated Gaussian

Fitting radius [px]: 3

Fitting method: Maximum likelihood

Initial sigma [px]: 1.6

Multi-emitter fitting analysis: [✓ enable]

Maximum of molecules per fitting region: 3

Model selection threshold (p-value): 1.0E-6

☐ Same intensity for all molecules

[✓] Limit intensity range [photons]: 200:2500

Visualisation of the results

Method: Averaged shifted histograms

Magnification: 5.0

Update frequency [frames]: 50

3D: ☐

Colorize Z: ☐

Lookup Table: 16 colors

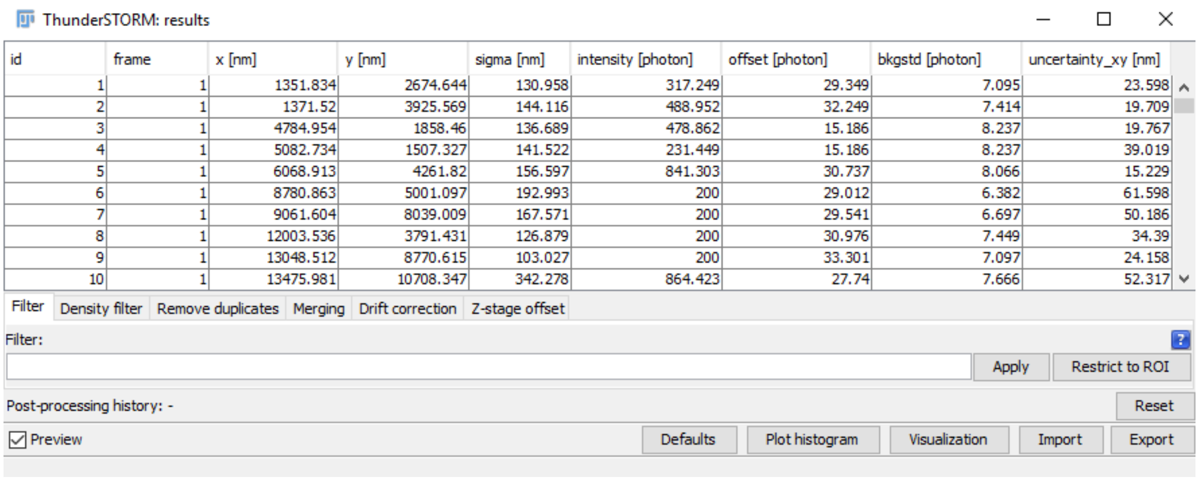
Z range (from:step:to) [nm]: -500:100:500

Lateral shifts: 2

Axial shifts: 2

Fig. 7: Recommended *ThunderSTORM* settings. The limit intensity range for the multi-emitter fit was set for ATTO 647N.

When the analysis terminates, the localizations are listed in a table (Fig. 8). Make sure that the units are [nm] and [photons], units 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 float precision 1 and all possible columns as well as the measurement protocol (fig. 9).



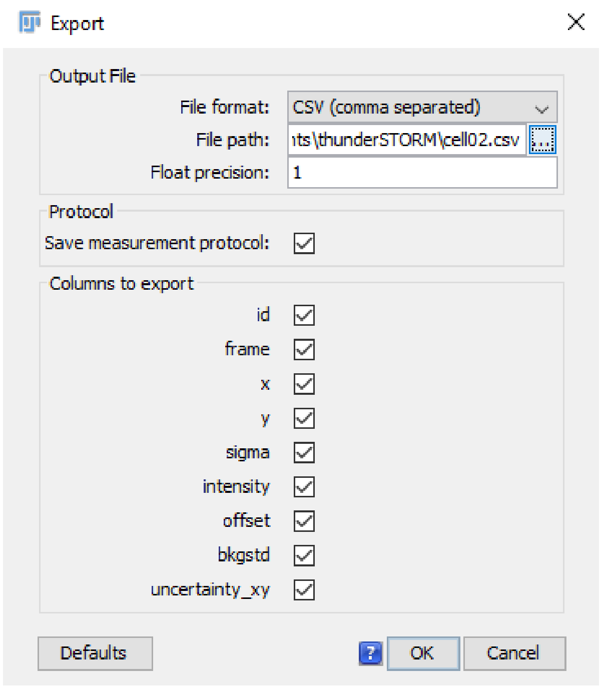
id	frame	x [nm]	y [nm]	sigma [nm]	intensity [photon]	offset [photon]	bkgstd [photon]	uncertainty_xy [nm]
1	1	1351.834	2674.644	130.958	317.249	29.349	7.095	23.598
2	1	1371.52	3925.569	144.116	488.952	32.249	7.414	19.709
3	1	4784.954	1858.46	136.689	478.862	15.186	8.237	19.767
4	1	5082.734	1507.327	141.522	231.449	15.186	8.237	39.019
5	1	6068.913	4261.82	156.597	841.303	30.737	8.066	15.229
6	1	8780.863	5001.097	192.993	200	29.012	6.382	61.598
7	1	9061.604	8039.009	167.571	200	29.541	6.697	50.186
8	1	12003.536	3791.431	126.879	200	30.976	7.449	34.39
9	1	13048.512	8770.615	103.027	200	33.301	7.097	24.158
10	1	13475.981	10708.347	342.278	864.423	27.74	7.666	52.317

Filter:

Post-processing history: -

☑ Preview

Fig. 8: ThunderSTORM results table with [nm] and [photon] as units.



Output File

File format: CSV (comma separated)

File path: ıts\thunderSTORM\cell02.csv

Float precision: 1

Protocol

Save measurement protocol: ☑

Columns to export

id ☑

frame ☑

x ☑

y ☑

sigma ☑

intensity ☑

offset ☑

bkgstd ☑

uncertainty_xy ☑

Fig. 9: Export the results as *.csv with float precision 1.

3.1 Macro for *ThunderSTORM*

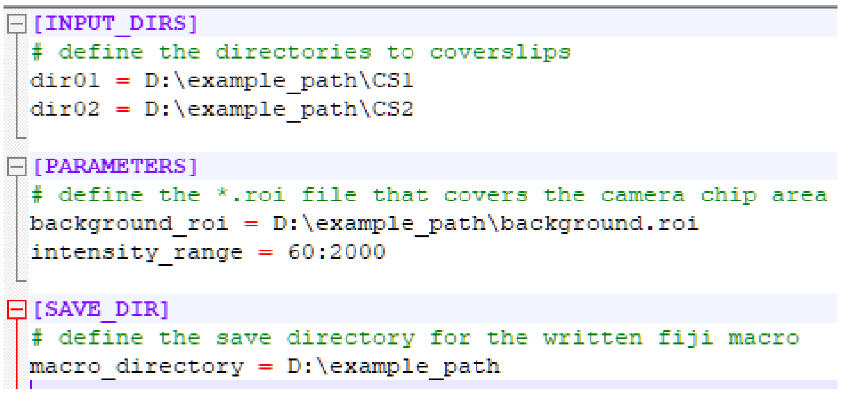
It is possible to run the localization analysis automatically for multiple files with a macro for *fiji* (listing 3). The file paths of the SPT movies have to be added to the target_cell array. Make sure to write all paths in quotes and use double backslashes. The names of the SPT movies are saved in 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. Check for continuous indexing starting from 0 for all arrays and for matching instances per array (target_cell[0] must be consistent with target_roi[0] ...). The macro runs the localization analysis, filters for duplicates and saves the results. **However, *ThunderSTORM* settings can't be changed with a macro and the analysis will always adapt the settings from the latest run. Therefore, start a test run with the settings of choice before running the macro.**

Listing 3: *Fiji* macro to run and save the localization analysis in *ThunderSTORM* automatically.

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

//Add the file path of the target objects
target_cells = newArray;
target_cells[0] = "C:\\Documents\\cell_01_MMStack_Pos0.ome.tif"
target_cells[1] = "C:\\Documents\\cell_02_MMStack_Pos0.ome.tif"

//Specify the tail of the path for each target object (example: "cell_name.tif")
cell_tif_names = newArray;
cell_tif_names[0] = "cell_01_MMStack_Pos0.ome.tif"
cell_tif_names[1] = "cell_02_MMStack_Pos0.ome.tif"

//Specify the paths to the *.roi files.
target_rois = newArray;
target_rois[0] = "C:\\Documents\\roi_cell01.roi"
target_rois[1] = "C:\\Documents\\roi_cell02.roi"

//Specify the paths where the analysis should be saved.
save_paths = newArray;
save_paths[0] = "C:\\Users\\pcoffice37\\Desktop\\cell19_macro.csv"
save_paths[1] = "C:\\Users\\pcoffice37\\Desktop\\cell20_macro.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();
}
```

3.2 Script to create macro for *ThunderSTORM*

Automatic creation of the macro is possible with a python script (write_TS_macro.py) and config file (fig. 10). Define the paths to coverslip directories, path to the background.roi file, intensity range for TS multi-emitter fit and save directory for the created macro. The folder structure in sec 2 must be followed. Execute the script with listing 4.



```
[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. 10: Config file for write_TS_macro.py script. Multiple directories are possible.

Listing 4: Execution of script in anaconda shell to write *fiji* macro with config file.

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

4 *swift*

***Swift* has multiple data and performance parameters that have to be considered for each dataset**. More information on the parameters are found in the user guide and in the swift_parameter_overview.pdf file. *Swift* version 0.2.0+ offers a graphical user interface that visualizes trajectories and should be explored. *SPTAnalyser* supports parameter estimations for the expected displacement, bleaching probability and localization precision. *Swift* and *SPTAnalyser* are also compatible with the localization software *rapidSTORM*. Cells with a low density based on the number of localizations should be chosen for the parameter estimation, because the tracking problem is less complicated to solve and relies less on the initial parameters given. The parameters should be determined and applied to a single measurement, a coverslip, a measurement day or a measurement condition depending on their variances.

4.1 Determination of the pre-analysis parameters with *SPTAnalyser*

Three parameters (precision, exp_displacement and p_bleach) are determined via Jupyter notebooks. Jupyter notebooks can be opened via the command line. Navitage to the folder with the notebooks and execute the command *jupyter notebook*.

4.1.1 Information on calculations

precision
Precision is the localization uncertainty in the x/y-plane in nm. It is determined in the Jupyter Notebook localizationUncertainty. The localization uncertainties of the trajectories are loaded from a localized file and a representative mean value is extracted by ln-transforming the localization uncertainties to generate normal distributed data, calculating the arithmetic mean and re-transforming the mean value. This procedure can be repeated with different localization files and eventually a representative localization uncertainty can be extracted. The parameter is important to classify immobile particles, as their displacement emerges from localization errors.

diffraction_limit
The diffraction limit specifies the minimal distance at which particles can still be told apart during localization, in [nm]. It is determined with framewise nearest neighbor based analysis. The minimal nearest neighbor per file is extracted. Multiple files should be analyzed to find the minimal distance.

exp_displacement
The expected displacement 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. *Swift* prefers longer connections if the value is large and vice versa.

p_bleach

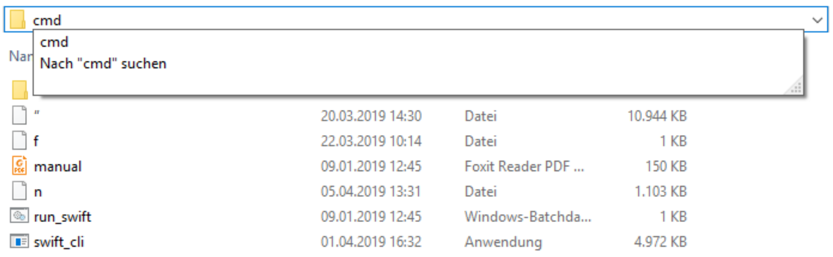
p_bleach is the probability per particle and frame to bleach, $\in [0;1]$. The higher the value, the more reluctantly *swift* makes connections, as shorter life times of trajectories are expected. Trajectories that exist for a certain time lag are subtracted from the normalized sum of all trajectories, represented as a histogram. The decay is fitted with a monoexponential function (eq. 1) based on an initial a value (= 1), an initial k value in s^{-1} (default = 0.5 s^{-1}) and the camera integration time Δt in s. It is possible to mask a number of values in the histogram, starting from time lag = 0, because some data deviate from a monoexponential decay especially in the first time lags. The remaining data is normalized and fitted as usual. The k value extracted from the monoexponential function and the time step between two frames Δt are inserted in the cumulative distribution function of an exponential distribution to determine p_bleach (eq. 2).

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

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

4.1.2 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 parameters exp_displacement and p_bleach are iteratively determined by running *swift* multiple times (with fixed precison and other fixed parameters determined for the dataset). Navigate to the swift_cli.exe in the explorer and start the command line from the folder (fig. 11). The first command „swift_cli.exe“ will start the tracking.exe, the second command is the file path to the localization file, tau is the camera integration time in [ms] and out_values specifies the output values. All possible output values are extracted with „all“. Other desired parameters should be added as well, e.g. p_switch is set to 0.01 (listing 5). Press enter to start the algorithm. A *.tracked.csv file is generated.



```
"            20.03.2019 14:30   Datei               10.944 KB
f            22.03.2019 10:14   Datei                    1 KB
manual       09.01.2019 12:45   Foxit Reader PDF ...   150 KB
n            05.04.2019 13:31   Datei                1.103 KB
run_swift    09.01.2019 12:45   Windows-Batchda...       1 KB
swift_cli    01.04.2019 16:32   Anwendung            4.972 KB
```

Fig. 11: Start the command line from the directory of the swift_cli.exe by entering ”cmd”.

Listing 5: *Swift* running in the command line to determine pre-analysis parameters.

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

The *.tracked.csv file is loaded in the expDisplacement and pBleach JupyterNotebook to determine the values of the parameters. The results are inserted in *swift* and rerun in the command line (listing 6). This process is repeated until the values converge.

Listing 6: *Swift* running in the command line with the additional parameters exp_displacement and p_bleach.

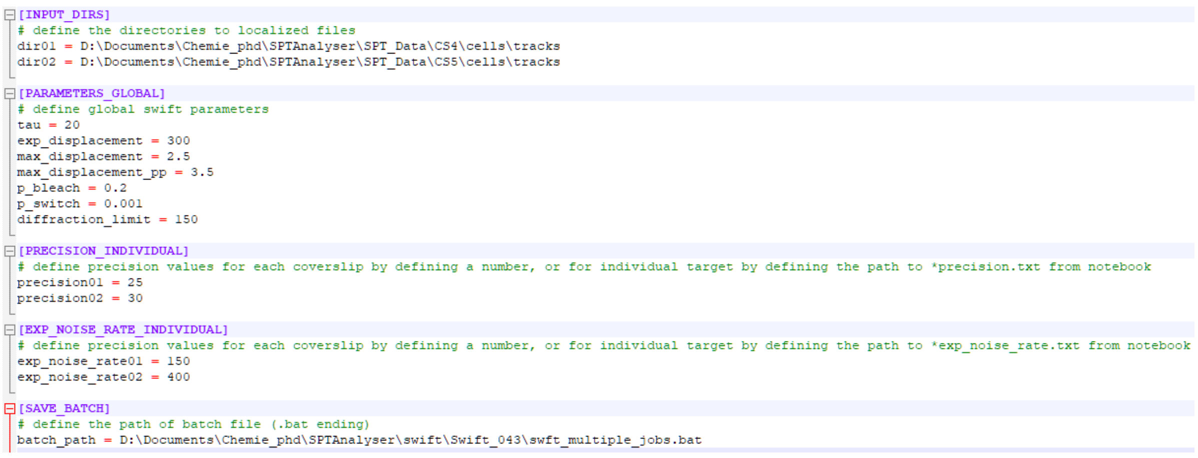
```
C:\Documents\swift> swift_cli.exe C:\Documents\Fab_CS2_cell01.csv
--precision "26.636" --p_switch "0.01" --tau "20" --out_values "all"
--exp_displacement "70.392" --p_bleach "0.006"
```

The connection is executed for multiple files of the same conditions with the same parameters with a batch script (listing 7). Insert the file paths of the localization files, declare the parameters and execute the batch script.

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 (
swift_cli.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.
```

4.1.3 Write batch file with script



```
[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. 12: Config file for write_swift_batch.py script

Listing 8: Execution of script in anaconda shell to write *fiji* macro with config file.

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

5 Diffusion analysis with *SPTAnalyser*

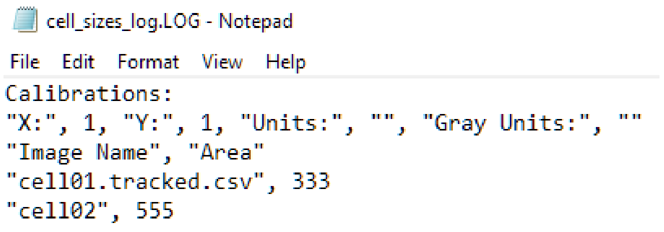
The diffusion analysis is executed in two Jupyter Notebooks: TrackAnalysis and Track-Statistics. The first notebook calculates diffusion coefficients, determines between the diffusion type immobile, confined, free and no type according to Michalet *et al.*,[5] Rossier *et al.*[6] and Harwardt *et al.*[7] The results are saved in a hdf5 file. The calculations might take several minutes depending on the magnitude of the dataset.
The second notebook reloads the information from the created hdf5 files and is therefore computationally quicker. Here, the data can be analyzed in detail. Trajectories can be filtered based on their diffusion type, diffusion coefficient and trajectory length. Global information for all cells is extracted and saved in a statistics file.

5.1 TrackAnalysis

TrackAnalysis handles tracked files from *swift* (localized with *ThunderSTORM* or *rapidSTORM*) 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.

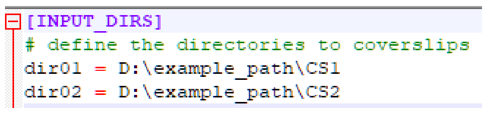
Define the cell areas [px^2] in a *.LOG file (like the one created in *PALMTracer*, fig. 13). You can copy a *.LOG file and use it multiple times. Add the name in quotes and the cell size in pixels^2. Either use the file name + file type or only the file name. If no fitting name is in the list, the amount of pixels on the camera is used as the cell size.



```
Calibrations:
"X:", 1, "Y:", 1, "Units:", "", "Gray Units:", ""
"Image Name", "Area"
"cell01.tracked.csv", 333
"cell02", 555
```

Fig. 13: Define cell sizes in pixels^2 in the *PALMTracer* .LOG file format.

The log file can also be created with the script write_cell_size_log.py and the config file write_cell_size_log_config.ini for multiple coverslips at once (fig. 14, listing 9).



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

Fig. 14: Config file for write_cell_size_log.py script. Multiple directories are possible.

Listing 9: Execution of script in anaconda shell to write log file with cell sizes for *SPTAnalyser*.

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

Several parameters have to be set. Therefore, files from one measurement condition should be analyzed at once. Parameters are the pixel size [nm], the amount of pixels on the camera and the camera integration time [s]. The units are transferred from px to μm and from frames to seconds.
Furthermore, the number of MSD values that are considered for the linear fit to determine the diffusion coefficient, the % of the MSD values that are considered for the fit that distinguishes between confined and free trajectories, the degree of freedom (2D tracking = 4), the minimal detectable diffusion coefficient (see chapter 5.3) 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. 15). 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.

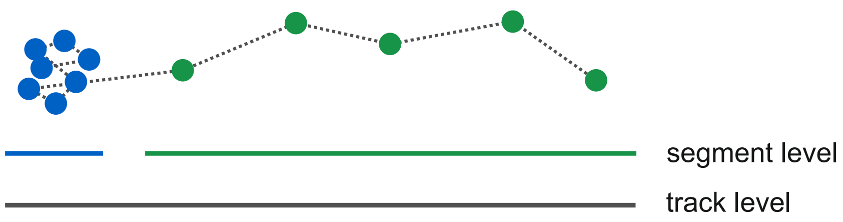


Fig. 15: Track and segment level of a trajectory. Segments represent different diffusion modes. The track level comprises the segments and represents the whole trajectory.

Click on "run the analysis" to start the analysis.

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 averaged over the cells according to section 5.7.4. The diffusion coefficients are represented in a histogram with the normalized frequency plotted against the diffusion coefficient with logarithmic bin size (fig. 16, 17). In order to represent diffusion coefficients from different cells, their frequency counts have to be normalized to the cell size.
The plots can be saved as vector graphics (as displayed), but all plot information (all MSD values, histogram with last applied bin size) is additionally saved in the statistics file in the TrackStatistics Jupyter Notebook.

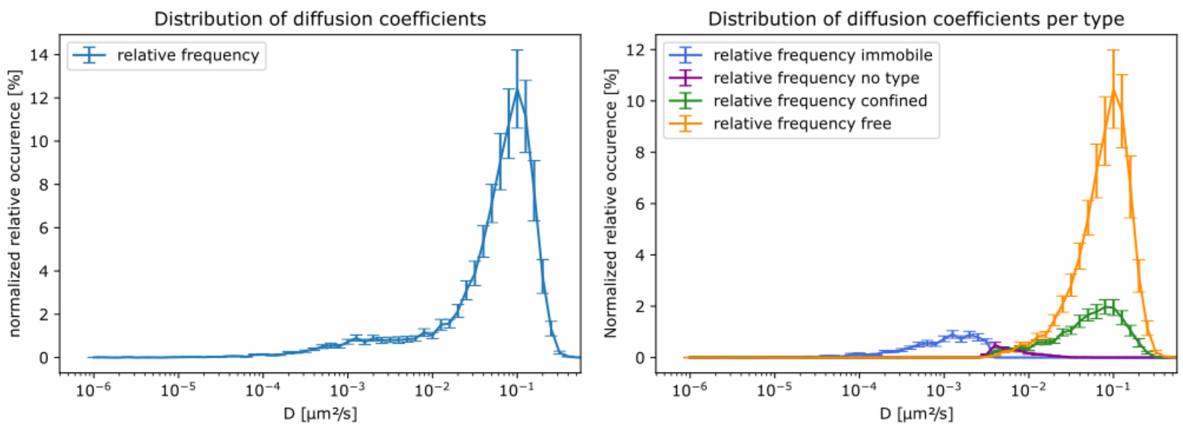


Fig. 16: Histograms of the diffusion coefficients from all analyzed files. The normalized occurrence in % is plotted against the diffusion coefficients in $\frac{\mu m^2}{s}$ with standard error of the means (SEM) and logarithmic bins.

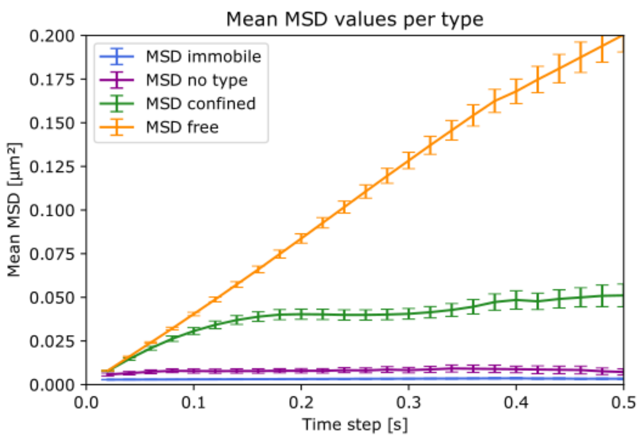
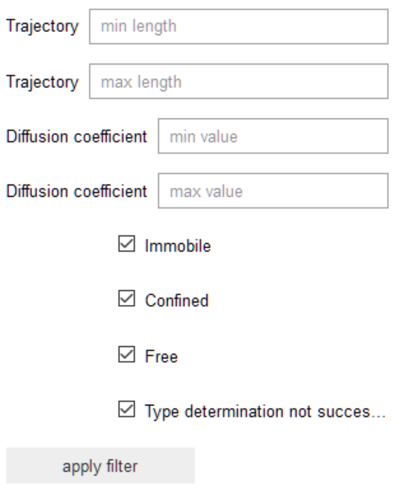


Fig. 17: Average MSD-values per type and cell.

5.2 TrackStatistics

The *.h5 files are loaded in the trackStatistics JupyterNotebook by choosing a directory. Filtering options are available (fig. 18). 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 5.5).


Trajectory: min length

Trajectory: max length

Diffusion coefficient: min value

Diffusion coefficient: max value

☑ Immobile

☑ Confined

☑ Free

☑ Type determination not succes...

Fig. 18: Filter options in TrackStatistics.

The diffusion types immobile and notype are treated separately and are additionally merged to one immobile fraction. It is mandatory to click on “Plot global diffusion histogram and MSD-plot” to calculate the global information before saving the files.

5.3 Determination of $\mathbf{D_{min}}$

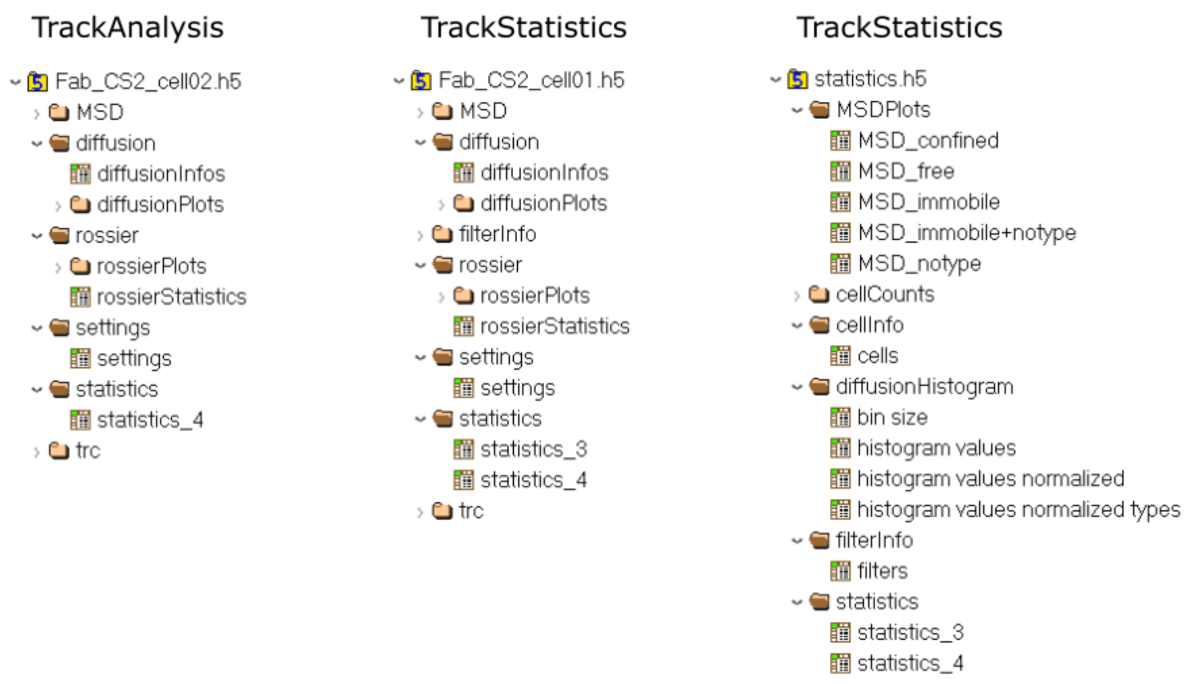
A D_{min} value based on the filtered data is needed to determine if a particle is immobile. All trajectories/segments with a diffusion coefficient smaller than D_{min} are labeled as immobile. D_{min} is derived from the dynamic localization error σ_{dyn} and easily computed by using the TrackAnalysis and TrackStatistics Jupyter Notebooks. The localized files are loaded in the TrackAnalysis Jupyter Notebook and analyzed without considering the default D_{min} value. For all trajectories/segments the diffusion coefficient and MSD(0) are calculated and used as variables to compute σ_{dyn}. 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} leads to a representative D_{min} value.

5.4 Background correction

The background correction is applied to the histogram of diffusion coefficients. Background measurements are analyzed like normal measurements and hdf5 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 measurements. Both, a background corrected diffusion histogram and a histogram without correction are saved.

5.5 File information

TrackAnalysis
“MSD” contains MSD Δt value pairs of all trajectories.
“diffusion” $\rightarrow$ “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” $\rightarrow$ “diffusionPlots” contains the first n MSD Δt value pairs, the linear fit and the residues.
“rossier” $\rightarrow$ “rossierStatistics” contains the trajectory id, the diffusion type (0=False, 1=True) and the parameters from eq. 6 to distinguish between confined and free diffusion, r = confinement radius, $D_{confined}$ = confined diffusion coefficient, chi^2. “rossier” $\rightarrow$ “rossierPlots” contains the the ath fraction of MSD Δt value pairs, the fit from eq. 6 and the residues.
“settings” contains the camera integration time dt, pixel size, cell size, $\tau_{\text{threshold}}$ from eq. 7, min trajectory length, fraction a of the MSD-values fitted with eq. 6, degrees of freedom dof, dynamic diffusion error based on eq. 4 and if analysis is based on track or segment (fig. 15).
“statistics” $\rightarrow$ “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.


TrackAnalysis

```
Fab_CS2_cell02.h5
  MSD
  diffusion
    diffusionInfos
    diffusionPlots
  rossier
    rossierPlots
    rossierStatistics
  settings
    settings
  statistics
    statistics_4
  trc
```

TrackStatistics

```
Fab_CS2_cell01.h5
  MSD
  diffusion
    diffusionInfos
    diffusionPlots
  filterInfo
  rossier
    rossierPlots
    rossierStatistics
  settings
    settings
  statistics
    statistics_3
    statistics_4
  trc
```

TrackStatistics

```
statistics.h5
  MSDPlots
    MSD_confined
    MSD_free
    MSD_immobile
    MSD_immobile+notype
    MSD_notype
  cellCounts
  cellInfo
    cells
  diffusionHistogram
    bin size
    histogram values
    histogram values normalized
    histogram values normalized types
  filterInfo
    filters
  statistics
    statistics_3
    statistics_4
```

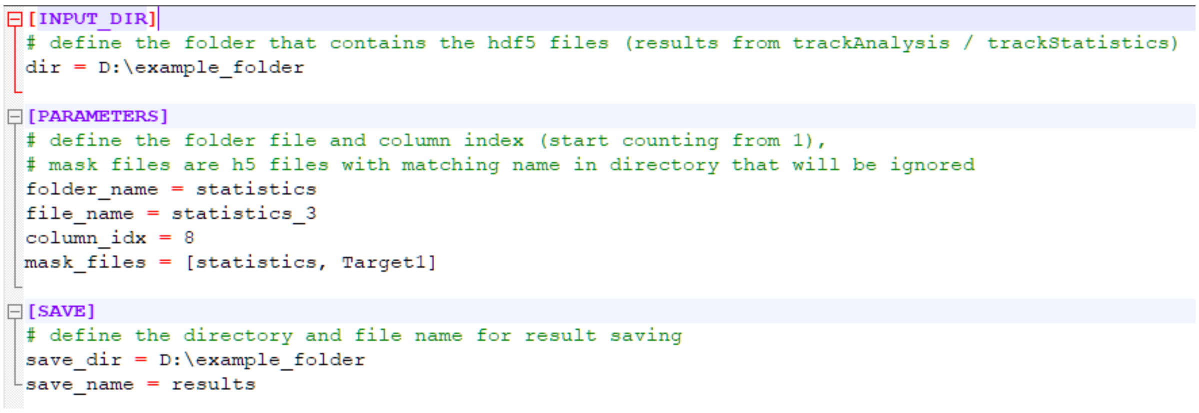
Fig. 19: TrackAnalysis saves a hdf5 file per cell. TrackStatistics saves a hdf5 file per cell containing the filtered trajectories and a statistics file that gives an overview of the filtered data of all loaded files in TrackStatistics.

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 notype. In “statistics_3” all immobile and notype trajectories are considered as immobile.

Additionally, a statistics file is created that summarizes information of all hdf5 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 hdf5 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.

5.6 Hdf5 file extraction

Values regarding defined columns of a hdf5 file can be extracted with the extract_hdf5_files.py script for multiple files. Define a folder that contains the hdf5 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 hdf5 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. 20: Config file for extract_hdf5_files.py script.

5.7 Calculations

For a more detailed explanation check out the literature.[5–7]

5.7.1 Diffusion coefficient

The diffusion coefficient for individual trajectories is obtained by fitting the first n points in the MSD plot with eq. 3, dof = degree of freedom, dof(2D diffusion) = 4.

$$MSD(\Delta t) = dof \cdot D \cdot \Delta t \tag{3}$$

5.7.2 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. 4 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 5 to calculate D_{min}.

$$\sigma_{dyn} = \sqrt{\frac{\langle MSD(0)\rangle + \frac{dof}{3}\langle D\rangle \cdot dt}{dof}} \tag{4}$$

$$D_{min} = \frac{\sigma_{dyn}^2}{dof \cdot n \cdot dt} \tag{5}$$

5.7.3 Diffusion types confined and free

Particles that are not immobile are categorized as confined or free based on the analysis by Rossier *et al.*[6] For each trajectory a fraction a of the MSD values are fitted with eq 6 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.

$$MSD(\Delta t) = \frac{4}{3}r_c^2 \cdot (1 - e^{-\Delta t/\tau}) \quad \rightarrow \quad \tau = \frac{r_c^2}{3D_c} \tag{6}$$

$$\tau_{threshold} = dt \cdot a \cdot l_{min} \cdot 0.5 \tag{7}$$

5.7.4 Statistics

Mean values and their SEM are calculated per cell (eq. 8 and 9). A global value is calculated by averaging the mean values of the cells (eq. 10). The error of the global value is calculated by averaging the SEM of the cells, according to Gaussian error propagation (eq. 11).

$$Y_i = \frac{1}{M}\sum_{j=1}^{M} y_j \tag{8}$$

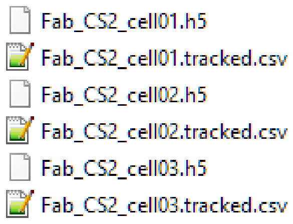
$$\Delta Y_i = \frac{\sqrt{\frac{(m_j - \bar{m})^2}{M-1}}}{\sqrt{M}} \tag{9}$$

$$Y_{global} = \frac{1}{N} \sum_{i=1}^{N} Y_i \tag{10}$$

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

6 Transition Counts with ***SPTAnalyser***

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. 21).



```
Fab_CS2_cell01.h5
Fab_CS2_cell01.tracked.csv
Fab_CS2_cell02.h5
Fab_CS2_cell02.tracked.csv
Fab_CS2_cell03.h5
Fab_CS2_cell03.tracked.csv
```

Fig. 21: 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 are 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. A mask value of 0 means no masking. All segments with a length <= " mask value are skipped, and the transition between the segment before and after are counted. 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. 22. In A the absolute counts per cell are shown. In B, transition counts were normalized per cell and summed to one to compare the occurrences of transition types. In C, transition counts were normalized per diffusion state so that the counts of transition types proceeding from the same diffusion state summed up to one to compare the occurrences of diffusion states in adjacent segments.

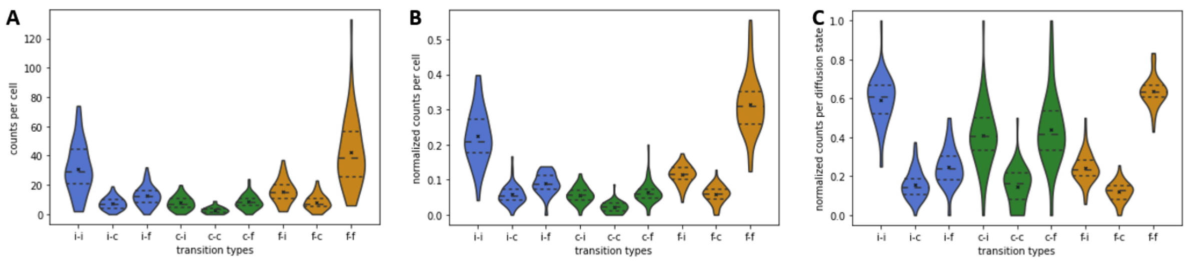


Fig. 22: Show code cells and adjust the axis ranges via xlim and ylim parameters.

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. 23). Plot axis ranges can be adjusted via the code cells (fig. 24). Significance tests of pairwise comparisons of the transition types are automatically saved and use Wilcoxon signed rank tests (SciPy). Levels of significance were 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 (***).

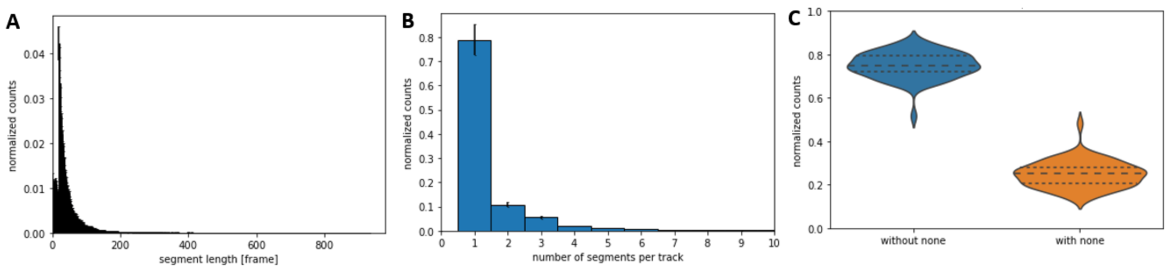
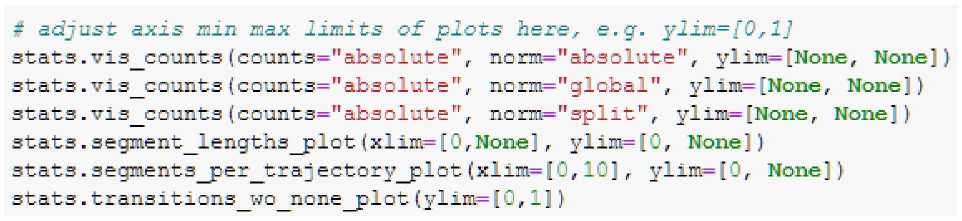


Fig. 23: Transiton counting results (A) absolute counts (B) counts normalized to one per cell (C) counts normalized to one per diffusion state.



```
# 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. 24: (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.

7 Literatur

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

[6] 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.

[7] 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.