A step-by-step guide to using Spot-On Matlab

Overview

Spot-On is an open-source package that allows kinetic modeling of single-particle tracking data and which comes in three equivalent versions: a web-interface that can be found at here, a command-line package written in Python that can be found here, and a Matlab package that can be found here. This guide covers the Matlab version, which is open-source and distributed under the GNU General Public License version 3+. Spot-On is free software, but distributed without any warranty.

Documentation

The purpose of this step-by-step guide is to provide a succinct overview of how to use the Matlab version of Spot-On. A full description of the underlying mathematical models, their numerical implementation and the full details of all user-defined parameters can be found at: https://spoton.berkeley.edu/SPTGUI/docs/

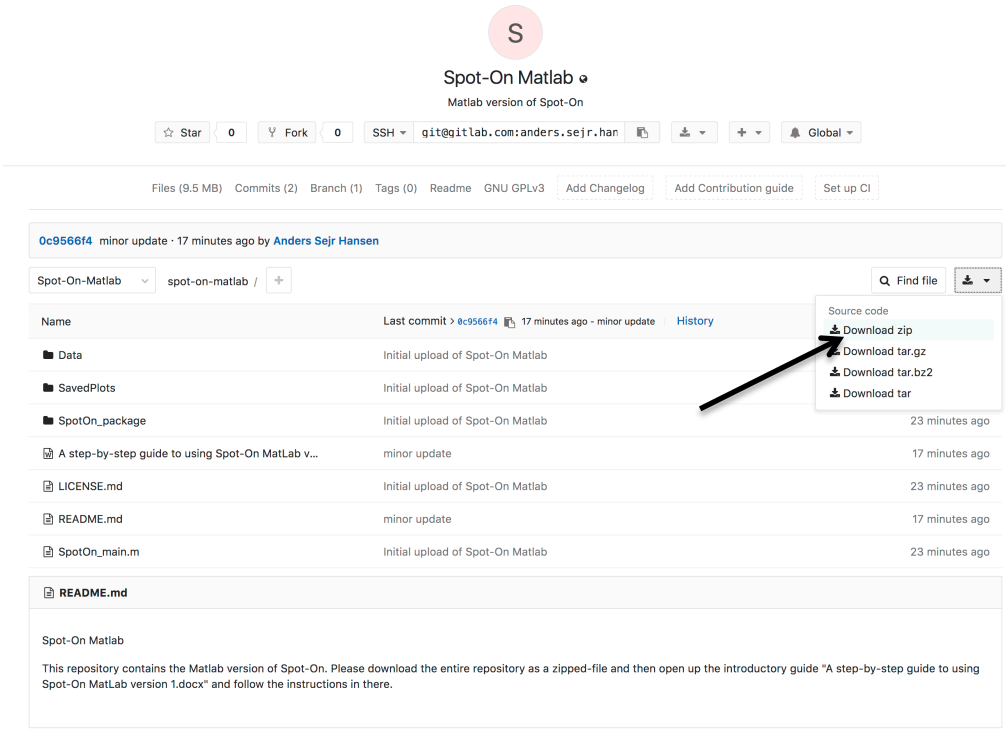
Spot-On Matlab version

Spot-On Matlab can be downloaded from GitLab: https://gitlab.com/tjian-darzacq-lab/spot-on-matlab

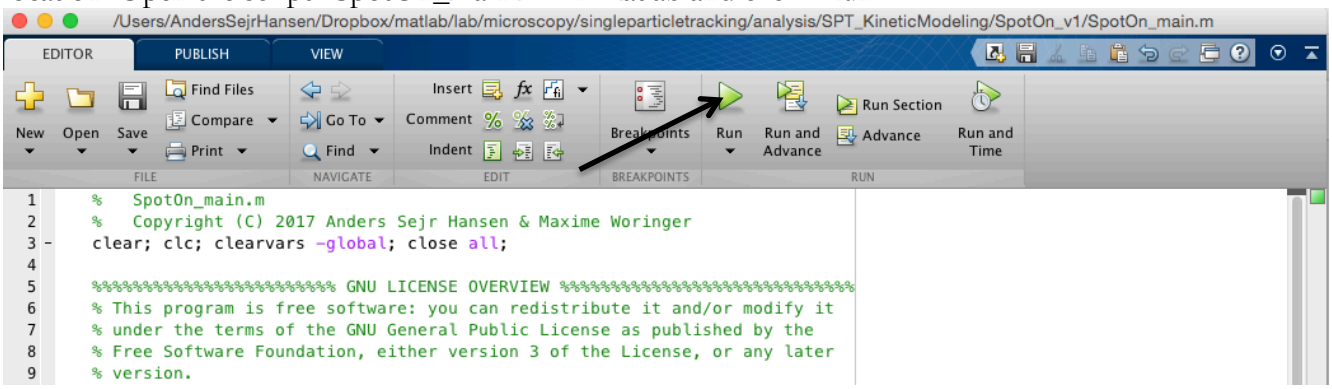
It consists of three directories and one main script, "SpotOn_main.m". The user can link to their data and adjust all important parameters in "SpotOn_main.m". All core code is found in the subdirectory "SpotOn_package", which the main script will call. Spot-On comes with a small example dataset, which is found in the subdirectory "Data". However, all data supporting this work is freely available at Dryad Digital Repository (URL pending). Moreover, once Spot-On runs, it will produce a series of figures, which will be saved to the subdirectory "SavedPlots". Please note that changing the directory organization or renaming anything will cause Spot-On to fail. Finally, please note that Spot-On does not perform localization and tracking of particles from raw images, but rather analyzes single particle tracking data, where the localization and tracking steps have already been performed, since a large number of publically available algorithms, ImageJ plugins and softwares that perform these steps are already available. For a partial list of these algorithms please click here.

Downloading and running Spot-On

Please download Spot-On as a zipped directory from GitLab. Once you click on the GitLab link, click "Download Zip" as illustrated below:



Once you have downloaded the zipped file, then unzip the directory and move it to an appropriate location. Open the script "SpotOn_main.m" in Matlab and click "Run"



Spot-On will then load it an example dataset (Halo-CTCF) and fit a 2-state kinetic model to each individual cell (5 cells total in DataSet 1) as well as to the merged data. Since fitting each single-cell takes a while, expect the code to take about 3 minutes to run (runtime on MacBook Pro laptop Mid-2015 running MATLAB R0214b). While the code runs, updates on the progress will be displayed in the Matlab "Command Window" and once the code finishes 5 figures will be displayed. Assuming this initial run is successful please proceed to the next section for a more detailed step-by-step guide to running Spot-On on your own data.

Detailed step-by-step guide to running Spot-On in Matlab

Please run Spot-On first on the provided example datasets to make sure that everything works. Here we provide a more detailed guide to loading in your own datasets, choosing the parameters appropriate for your dataset and interpreting the output.

1. Spot-On Matlab input data format: trackedPar

The Matlab version of Spot-On accepts only a single data format. Spot-On will take as input a series of Matlab workspace MAT-files such as "cell01.mat" (e.g. representing single particle tracking data from single cells). Please see the provided test data for an example. These MAT-files should contain a structure array named "trackedPar" (but MAT-files may contain any other number of variables as well). trackedPar should have the following fields:

- trackedPar.xy: xy is a matrix with 2 columns and a number of rows corresponding to the number of localizations in that trajectory. The first column is the x-coordinate and the second column is the y-coordinate. The units should be micrometers (μm).
- trackedPar.Frame: Frame is a column vector where each element is the frame where the particle was localized.
- trackedPar.TimeStamp: TimeStamp is a column vector where each element is the timepoint where the particle was localized.

If you use another format, please convert to the trackedPar format before using Spot-On Matlab.

2. How to make Spot-On read your MAT-files

For Spot-On to read your data, you need to provide a path for the directory where the MAT-files are stored as well as the names of all of the MAT-files. Additionally, Spot-On allows you to include or exclude individual cells to see how an outlier might bias the merged data or to remove a cell where the number of localizations per frame was too high to allow unambiguous tracking. Example DataSet 1 provides an illustration of how to tell Spot-On where the data is for data all in the same folder:

```
63 -    if DataSet == 1 % Example DataSet 1: a single replicate of Halo-hCTCF at 134 Hz
64 -        data_struct(1).path = [pwd, filesep, 'Data', filesep, 'CTCF_134Hz_rep1', filesep];
65 -        data_struct(1).workspaces = {'U2OS_C32_Halo-CTCF_PA-JF646_1ms-633nm_134Hz_rep1_cell01', 'U2OS_C32_Halo-CTCF_PA-JF646_
66 -        data_struct(1).Include = [1,2,3,4,5];
67 -        SampleName = 'U2OS C32 Halo-hCTCF; PA-JF646; ~134 Hz; rep1';
68
```

Line 64: give the path of the files.
Line 65: give the name of each MAT-file.
Line 66: Include: should have the same length as the number of workspaces.
Line 67: A sample name; this will be included in the plots and be the name of the saved files.

In case you would like Spot-On to read in data from multiple directories (e.g. if each replicate is in a separate directory and you would like to merge and analyze data from all replicates together), please use the format outlined for example DataSet 2:

```
69 -    elseif DataSet == 2 % Example DataSet 2: two replicates of Halo-hCTCF at 134 Hz
70          % 1st CTCF replicate:
71 -        data_struct(1).path = [pwd, filesep, 'Data', filesep, 'CTCF_134Hz_rep1', filesep];
72 -        data_struct(1).workspaces = {'U2OS_C32_Halo-CTCF_PA-JF646_1ms-633nm_134Hz_rep1_cell01', 'U2OS_C32_Halo-CTCF_PA-JF646_
73 -        data_struct(1).Include = [1,2,3,4,5];
74          % 3rd CTCF replicate:
75 -        data_struct(2).path = [pwd, filesep, 'Data', filesep, 'CTCF_134Hz_rep3', filesep];
76 -        data_struct(2).workspaces = {'U2OS_C32_Halo-CTCF_PA-JF646_1ms-633nm_134Hz_rep3_cell01', 'U2OS_C32_Halo-CTCF_PA-JF646_
77 -        data_struct(2).Include = [1,2,3,4,5];
78          % name of merged dataset
79 -        SampleName = 'U2OS C32 Halo-hCTCF; PA-JF646; ~134 Hz; two replicates';
80 -    end
```

As can be seen, simply increment "`data_struct(n)`", where `n` is the current directory to be added.

For adding your own datasets, simply add an additional line with "`elseif DataSet == 3`" and then include information about your dataset there and so forth. For example, assume you are on a Mac

and that you have a single file named "MyData.mat", that you would like to be read. Then define a DataSet 3 as shown below giving the full path to the MAT-file:

```
79 -    elseif DataSet == 3 % MyData.mat in some folder
80 -        data_struct(1).path = '/Users/anderssejrhansen/Dropbox/Data/';
81 -        data_struct(1).workspaces = {'MyData'};
82 -        data_struct(1).Include = [1];
83 -        SampleName = 'MyData: testing';
84 -    end
```

3. Adjust user-defined parameters

Spot-On requires the user to adjust a series of parameters to tell Spot-On some key experimental parameters (e.g. the frame-rate) and model-fitting parameters (e.g. whether fitting a 2-state or 3-state model is desired). We will describe each parameter in more detail below:

Choose DataSet (line 25-28)

```
25      %%%%% Choose DataSet
26 -    DataSet = 1;    % Use DataSet=1 for an example of how to process data for multiple cells from a single replicate
27                      % Use DataSet=2 for an example of how to process data for multiple cells from multiple replicates
28 -    data_struct = struct([]);
```

- DataSet: this parameter will determine which DataSet (line 61 and onwards) will be analyzed.

Acquisition Parameters (line 30-33)

```
30      %%%%% Acquisition Parameters:
31 -    TimeGap = 7.477; % delay between frames in milliseconds
32 -    dZ = 0.700; % The axial observation slice in micrometers; Rougly 0.7 um for the example data (HiLo)
33 -    GapsAllowed = 1; % The number of allowed gaps in the tracking
```

- **TimeGap**: time delay between frames in units of milliseconds. The example DataSet was recorded at ~134 Hz, so the TimeGap was ~7.477 ms.
- **dZ**: axial observation slice in units of micrometers. This parameter will depend somewhat on signal-to-noise conditions and imaging modality. But for a typical setup (HiLo or epi illumination, HaloTag or SNAP-Tag dyes), this is likely to be around 0.7 μm. The parameter tell Spot-On how far out-of-focus a molecule can be before it fails to be detected and it is important for accurately correcting for diffusing molecules gradually moving out-of-focus and thus being undersampled at longer time-intervals. In most cases 0.7 μm is reasonable, but for details on how to measure this please see (Hansen *et al.*, *eLife*, e25776, 2017) or the Spot-On FAQ.
- **GapsAllowed**: the number of gaps allowed during the tracking. Most tracking algorithms allow a user to choose how many gaps to allow (e.g. if a molecule is localization in frame 1, but not in frame 2, but again in frame 3, can it still be tracked?). If you set GapsAllowed to a smaller number than the number of gaps in your data Spot-On will fail. Currently, the Matlab version of Spot-On supports only 0, 1 or 2 gaps, but not more. Use GapsAllowed=1 for all the provided example data.

Data Processing Parameters (line 35-44)

```
35      %%%%% Data Processing Parameters:
36 -    TimePoints = 8; % How many delays to consider: N timepoints yield N-1 delays
37 -    BinWidth = 0.010; % Bin Width for computing histogram in micrometers (only for PDF; Spot-On uses 1 nm bins for CDF)
38 -    UseAllTraj = 0; % If UseAllTraj=1, all data from all trajectories will be used; If UseAllTraj=0, only the first X displa
39 -    JumpsToConsider = 4; % If UseAllTraj=0, the first JumpsToConsiders displacements for each dT where possible will be used.
40 -    MaxJumpPlotPDF = 1.05; % the cut-off for displaying the displacement histograms plots
41 -    MaxJumpPlotCDF = 3.05; % the cut-off for displaying the displacement CDF plots
42 -    MaxJump = 5.05; % the overall maximal displacements to consider in micrometers
43 -    SavePlot = 1; % if SavePlot=1, key output plots will be saved to the folder "SavedPlots"; Otherwise set SavePlot = 0;
44 -    DoPlots = 1; % if DoPlots=1, Spot-On will output plots, but not if it's zero. Avoiding plots speeds up Spot-On for batch
```

- **TimePoints**: how many time points to consider. If you allow N time points, this corresponds to considering displacements with a maximal time-delay of (N-1)Δt. Generally, we do not recommend going much above 50-60 ms unless you have an a very large

number of trajectories and/or very long trajectories, since otherwise, the displacement histograms at longer Δt tend to be undersampled.

- **BinWidth**: how finely to do binning for PDF fitting and plotting in units of micrometers. Generally 0.010 μm or 10 nm is reasonable, but if you have very small amounts of data you may want to increase it. Please note, Spot-On will always use a 1 nm bin width for CDF-fitting, since CDF fitting is not prone to binning artifacts.
- **UseAllTraj**: If UseAllTraj=1, all displacements will be used from each trajectory. If UseAllTraj=0, then the number of displacements used will depend on other parameters explained below. If UseAllTraj=1, a trajectory of N frames, will contribute N-1 displacements to the 1Δt histogram, N-2 displacements to the 2Δt histogram, …, N-k displacements to the kΔt histogram.
- **JumpsToConsider**: This parameter only comes into effect if UseAllTraj=0. If UseAllTraj=0, the first JumpsToConsider number of displacements will be used. This is empirically useful to correct for over-counting of slow-molecules not accounted for by the corrections implemented in the algorithm (for instance for undercounting due to motion-blur). Here if JumpsToConsider=4, for each trajectory, 4 jumps (if possible) will be used to build the jump length histogram. For example, if Number of timepoints=8 and JumpsToConsider=4, a trajectory of 9 frames will contribute 4 jumps to 1dT, 4 jumps to 2 dT, …, and 2 jumps to 7 dT. This is a semi-empirical way of correcting for additional biases towards bound molecules.
- **MaxJumpPlotPDF**: the cut-off on the x-axis for displaying the displacement histogram in units of micrometers. The parameter only affects the plots.
- **MaxJumpPlotCDF**: the cut-off on the x-axis for displaying the displacement cumulative distribution function in units of micrometers. The parameter only affects the plots.
- **MaxJump**: this parameter affects data-processing. For binning displacements, a maximum displacement has to be set, so this parameter should be set to a large value that should be at least as big as the largest displacement. Generally, 5.05 μm is reasonable for single-molecule tracking data in mammalian cells.
- **SavePlot**: If SavePlot=1, a PDF all plots will be saved to the folder "SavedPlots". If SavePlot=0, no plots will be saved.
- **DoPlots**: If DoPlots=1, plots will be displayed. If DoPlots=0, no plotting will be done. This can be useful if you are running Spot-On in batch on a lot of data and want to save time by avoiding the plotting step.

Model Fitting Parameters (line 46-59)

```
46      %%%%% Model Fitting Parameters:
47 -    ModelFit = 2; %Use 1 for PDF-fitting; Use 2 for CDF-fitting
48 -    DoSingleCellFit = 1; %Set to 1 if you want to analyse all single cells individually (slow).
49 -    NumberOfStates = 2; % If NumberOfStates=2, a 2-state model will be used; If NumberOfStates=3, a 3-state model will be use
50 -    FitIterations = 2; % Input the desired number of fitting iterations (random initial parameter guess for each)
51 -    FitLocError = 1; % If FitLocError=1, the localization error will fitted from the data
52 -    FitLocErrorRange = [0.010 0.075]; % min/max for model-fitted localization error in micrometers.
53 -    LocError = 0.035; % If FitLocError=0, LocError in units of micrometers will be used.
54 -    UseWeights = 0; % If UseWeights=0, all TimePoints are given equal weights. If UseWeights=1, TimePoints are weighted acco
55 -    D_Free_2State = [0.5 25]; % min/max Diffusion constant for Free state in 2-state model (units um^2/s)
56 -    D_Bound_2State = [0.0001 0.08]; % min/max Diffusion constant for Bound state in 2-state model (units um^2/s)
57 -    D_Free1_3State = [0.5 25]; % min/max Diffusion constant #1 for Free state in 3-state model (units um^2/s)
58 -    D_Free2_3State = [0.5 25]; % min/max Diffusion constant #2 for Free state in 3-state model (units um^2/s)
59 -    D_Bound_3State = [0.0001 0.08]; % min/max Diffusion constant for Bound state in 3-state model (units um^2/s)
```

- **ModelFit**: Determines whether fitting will be performed to the displacement histograms (PDF; ModelFit=1) or to the cumulative distribution function of displacements (CDF; ModelFit=2). CDF-fitting is always more precise, whereas PDF-fitting tends to slightly underestimate the fraction bound and the diffusion constant, likely because PDF-fitting is more prone to binning artifacts and undersampling. Thus, for all quantitative analysis,

CDF-fitting should be performed. However, displacement histograms often seem more intuitive, and for this reason Spot-On also allows PDF-fitting for making figures etc.

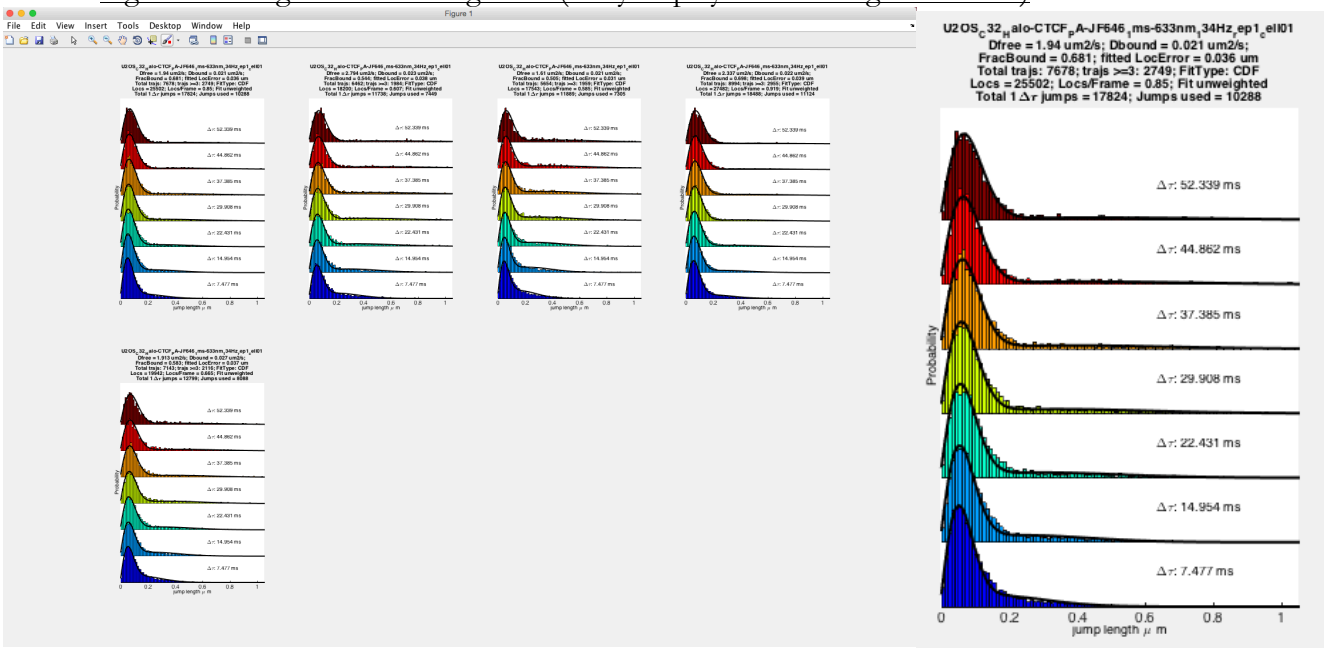
- **DoSingleCellFit**: If DoSingleCellFit=1, each single cell or each individual MAT-file will be analyzed and fitted separately. This is great for assessing how much cell-to-cell variability there is and for determining whether a single outlier is biasing the results, but also very slow, since fitting merged data takes about the same amount of time as fitting a single cell. If DoSingleCellFit=1, a figure will be displayed showing the fit to each single cell. Set DoSingleCellFit=0, to avoid single-cell fitting.
- **NumberOfStates**: the number of diffusive states in the model. Currently, Spot-On only supports 2 or 3 states. In most single particle tracking experiments of molecules that can engage scaffolds (e.g. transcription factors, which may bind chromatin), one of these states will correspond to a bound state. In the case of Halo-CTCF in human U2OS cells (the example data), this will manifest itself as the chromatin-bound state of CTCF exhibiting a very small *D* for the bound state, which likely corresponds to slow diffusion of chromatin. The other states will correspond to freely diffusive states. However, the user can change this to fit whatever is appropriate for their data by adjusting the lower and upper bounds listed in lines 53-57.
- **FitIterations**: Spot-On fits a mathematical model to the data using least-squares fitting implemented using the Levenberg-Marquardt algorithm using Matlab's function `lsqcurvefit`. Since this algorithm may occasionally get trapped in local minima in parameter space, for each iteration of the fitting Spot-On generates a random initial guess of the parameters, which differs between each iteration. Thus, increasing FitIterations, increases the probability that the globally optimal fit will be generated, but comes at the cost of slowing down the fitting. In practice, for all single-molecule tracking data we have tested so far, the globally best fit is always obtained in the first or second iteration of fitting, so we generally recommend keeping this parameter to 2-3.
- **FitLocError**: The user can either provide the localization error or precision with which single-molecules were localized or ask Spot-On to infer this from the model-fitting. If FitLocError=1, Spot-On will ignore the value provided in line 52 and infer the localization error from the model-fitting. In the example datasets provided with Spot-On, the localization error was around 35 nm
- **FitLocErrorRange**: the allowed range for fitting the localization error in units of micrometers. If you choose to have Spot-On fit the localization error, make sure this range is set reasonably based on your imaging conditions.
- **LocError**: if FitLocError=1, this parameter will be ignored since Spot-On will infer the localization error from fitting. However, if FitLocError=0, then Spot-On will use the user-supplied localization error, which has to be supplied in units of micrometers. If this parameter is set inaccurately, this will generally show up through poor fitting of the bound state and will cause the estimation of the bound diffusion constant to be inaccurate.
- **UseWeights**: if UseWeights=1, Spot-On will take into account the relative amount of data during the model-fitting. For example, there will almost always be a lot more 1dT displacements than e.g. 5dT displacements, such that there is more noise associated with the later dT histograms. If UseWeights=1, Spot-On will linearly weigh the histogram at each dT according to the relative amount of data during the least-squares fitting, which causes the later dT histograms to count less. This is also helpful in cases where only small amounts of data could be obtained and/or very few long trajectories could be obtained. In cases, where large amounts of data was obtained, weighing the model-fit has a minimal effect. If UseWeights=0, each dT histogram will carry equal weight.
- **D_free_2State**: allowed lower and upper bound for the faster diffusion constant for 2-state model-fitting in units of $\mu m^2/s$.

- **D_bound_2State**: allowed lower and upper bound for the slower diffusion constant for 2-state model-fitting in units of $\mu m^2/s$.
- **D_free1_3State**: allowed lower and upper bound for the first faster diffusion constant for 3-state model-fitting in units of $\mu m^2/s$.
- **D_free2_3State**: allowed lower and upper bound for the second faster diffusion constant for 3-state model-fitting in units of $\mu m^2/s$.
- **D_bound_3State**: allowed lower and upper bound for the slower diffusion constant for 3-state model-fitting in units of $\mu m^2/s$.

4. Figures plotted by Spot-On

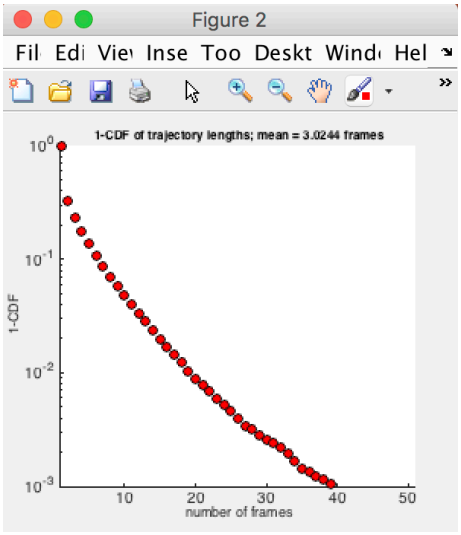
Spot-On will produce a series of figures for each run, the exact number of which will depend on the user input, but which fall into 5 types. We will discuss the 5 types of figures below.

Figure showing fit to each single cell (Only displayed if DoSingleCellFit=1)



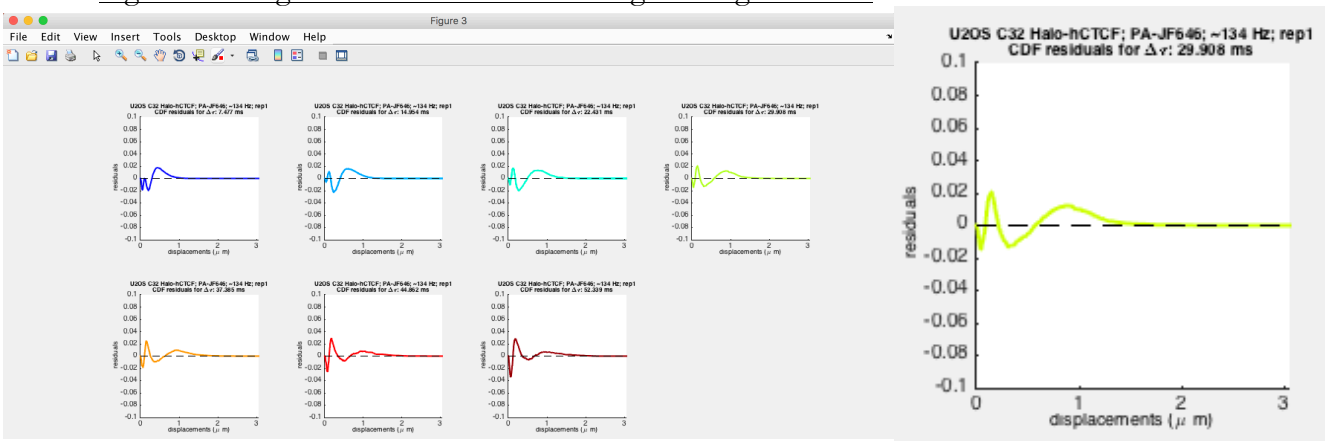
If DoSingleCellFit=1, then Spot-On will fit the model to each MAT-file individually and output this to a figure, showing a maximum of 8 plots. If more than 8 MAT-files are provided, additional figures will be outputted.

Figure showing survival probability of trajectory lengths



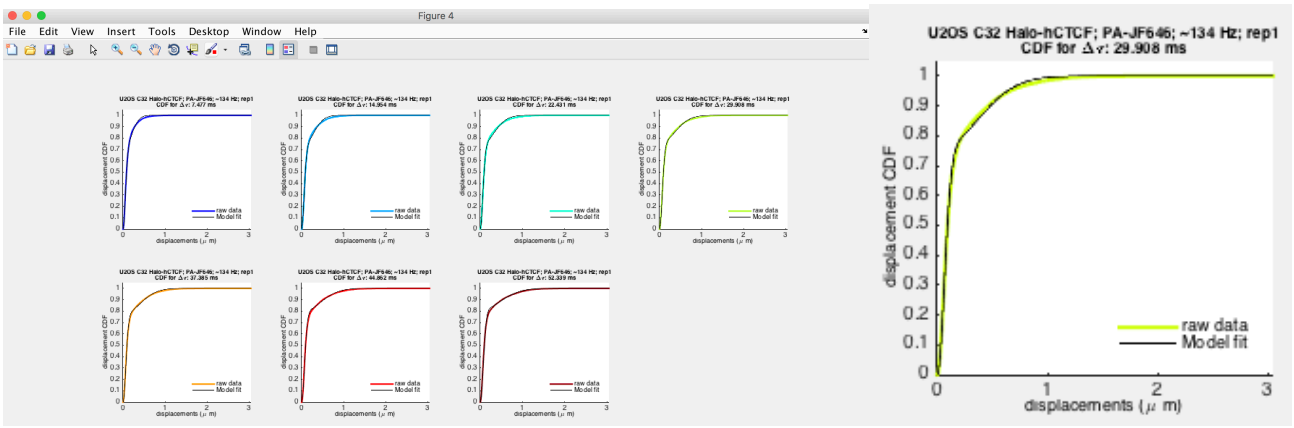
The plot shows the survival probability of all trajectories in units of frames. This can be useful for getting an idea of the distribution of trajectory lengths. Please note, the molecules can disappear both because of photobleaching and because of molecules moving out-of-focus. Therefore, cell lines imaged under the same conditions and with the same dye often exhibit difference. For example, Halo-3xNLS generally shows short trajectories since molecules rapidly move out-of-focus and since very few molecules appear bound.

Figure showing residuals from model-fitting to merged dataset



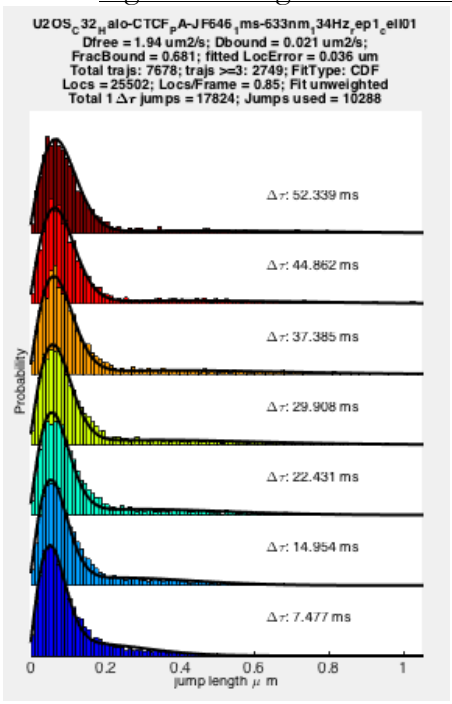
This figure shows the residuals (difference between data and fit) from fitting the model to the merged datset. If PDF-fitting was chosen, it will show the residuals for the PDF-fit, but if CDF-fitting was chosen, it will show the residuals for the CDF-fit. It can be helpful in accessing how well the model could fit the data, and where there were deviations, if any.

Figure showing the model-fit to the merged data displacements CDF



This figure shows the model-fit to the displacements CDF for all the user-defined time points. Please note that if the user asks Spot-On to perform PDF-fitting, the corresponding CDF-fit will also be computed using the best-fit PDF parameters and this figure will still display.

Figure showing the model-fit to the merged data displacements PDF



The last figure will show the model-fit to the displacement histograms (PDF), as well as summarize the values of key parameters and the underlying data as shown above. Please note, that even when the user asks Spot-On to perform CDF-fitting, Spot-On will still calculate the PDF-fit using the best-fit CDF parameters and display the plot above.

5. Variables outputted by Spot-On

Some users may prefer to plot the own figures from the Spot-On output or automatically run Spot-On on many datasets and automatically save the output. For these purposes, Spot-On returns an structure array called "output_struct":

```
Command Window
>> Output_struct

Output_struct =

    single_cell_model_params: [5x4 double]
       single_cell_model_PDF: [35x506 double]
       single_cell_model_CDF: [35x5051 double]
        single_cell_data_PDF: [35x506 double]
        single_cell_data_CDF: [35x5051 double]
         merged_model_params: [1x4 double]
            merged_model_PDF: [7x506 double]
            merged_model_CDF: [7x5051 double]
             merged_data_PDF: [7x506 double]
             merged_data_CDF: [7x5051 double]
   merged_model_residuals: [7x5051 double]
   list_of_model_parameters: {1x4 cell}
                r_bins_PDF: [1x506 double]
                r_bins_CDF: [1x5051 double]

fx >>
```

This structure array contains the variables shown above, namely the raw PDF+CDF and the model-fit PDF+CDF for both the single-cell and merged data. It also contains the best-fit model parameters for both the single-cell and merged data. Moreover, it contains the residuals from fitting the model to the merged data and "`list_of_model_parameters`", which contains the name of each parameter. Please note that each row is a different timepoint: for example, `merged_data_PDF`, is 7x506 matrix, where each column is the probability for a bin and each row is a different timepoint, in this case from 1Δt to 7Δt. Finally, `Output_struct` contains the bins used for the PDF and for the CDF: `r_bins_PDF` and `r_bins_CDF`.

As an example, say we want to plot the raw PDF, the model-fit as well as the raw CDF and the model-fit for time point 3. We can then do this using the following simple Matlab code:

```
figure('position',[200 200 600 250]); %[x y width height]
subplot(1,2,1); % plot PDF and fit
hold on;
plot(Output_struct(1).r_bins_PDF, Output_struct(1).merged_data_PDF(3,:), 'r-', 'LineWidth', 2);
plot(Output_struct(1).r_bins_PDF, Output_struct(1).merged_model_PDF(3,:), 'k-', 'LineWidth', 2);
legend('raw data', 'model fit PDF', 'Location', 'NorthEast');
xlabel('displacements (um)'); ylabel('probability'); title('raw data and PDF-fit');
xlim([0 1.05]);
hold off;

subplot(1,2,2); % plot CDF and fit
hold on;
plot(Output_struct(1).r_bins_CDF, Output_struct(1).merged_data_CDF(3,:), 'b-', 'LineWidth', 2);
plot(Output_struct(1).r_bins_CDF, Output_struct(1).merged_model_CDF(3,:), 'k-', 'LineWidth', 2);
legend('raw data', 'model fit CDF', 'Location', 'SouthEast');
xlabel('displacements (um)'); ylabel('probability'); title('raw data and CDF-fit');
xlim([0 2.05]);
hold off;
```

This code-snippet will then produce the figure below: