singleParticleTracking Matlab Code

Main author: Dr Isabel Llorente Garcia. University of Oxford and University College London.

Matlab code for the detection and tracking in two dimensions (2D) of bright spots in microscope images (on a dark background). The code can be used to analyse fluorescence microscopy videos of living cells with fluorescently labelled proteins that appear on the images as bright spots.

Academic publication that uses this code:

Single-molecule in vivo imaging of bacterial respiratory complexes indicates delocalized oxidative phosphorylation, I. Llorente-Garcia, T. Lenn, H. Erhardt, O. L. Harriman, L.-N. Liu, A. Robson, S.-W. Chiu, S. Matthews, N. J. Willis, C. D. Bray, S.-H. Lee, J. Yen Shin, C. Bustamante, J. Liphardt, T. Friedrich, C. W. Mullineaux and M. C. Leake. **Biochimica et Biophysica Acta (BBA): Bioenergetics** 1837, 811-824 (2014). https://www.sciencedirect.com/science/article/pii/S0005272814000309?via%3Dihub

Guide to run analysis software

Steps to follow:

1. Preliminaries, setting up;
2. Find trajectories and output them to one excel file;
3. Generate lists of good tracks after visual inspection;
4. Analyse each good track.

1. Preliminaries:

Note that most of this code is at the moment made for analysing images which have two channels, top and bottom (e.g. green and red channels for two fluorescent labels that emit in two different wavelengths/colours).

You can look at the files under folder "scriptsThatHelpRunCode" (e.g., "runToAnalyse.m") for guidance on some types of operations.

Must have Microsoft excel installed.

Having installed the R2008b 7.7.0 version of Matlab (with the following toolboxes: curve fitting, statistics, signal processing), copy the code (the whole structure of codes and folders within singleParticleTracking) into one of your directories.

Add the directory where code was copied as a Matlab path by doing:

File | set path | add with sub folders

(just add mymatlabfiles to path, not all matlab folers in my documents (not the analysis space))

Keep an original copy of your data!

Then make a copy and paste into an analysis folder, the place where your image sequences (videos) are and where analysis result folders will be added. The program will output various files into that.

Note that running the code from a directory in a network drive is quite slow. It is much faster to copy data to your local C:\ drive and run the code from there.

Once Matlab is open, go to the directory described above (current directory).

- Move into correct data analysis directory, eg.:

```
cd('C:\Isabel\DataAnalysis\Ollie's\cyoA_mCherry_OlliesData\24October2012\in vivo cyoa-mCherry')
```

You can copy a “runToAnalyse.m” file, which contains the main functions used to get the tracks from the image sequences, from the “scriptsThatHelpRunCode” directory to the current directory with the new data you want to analyse.

All functions (.m files) are commented to explain what they do as well as inputs and outputs. Some functions have (at the end) examples of how to run the function and detailed explanations of the outputs.

- **Calculating frame averages** is a useful step to have a look at where cells are on the image and whether there are beads or other unusual elements on the image sequences (videos):

To do this, the general function is:

```
frame_avg = frameAverage(image_label, start_frame, end_frame, display_result, save_png)
```

This function calculates a frame average and saves a .png image of it in folder “frameAverages” within the current data directory.

Use doc frameAverage for more help about the function inputs.

Example of how to run this function:

```
frameAverage('1030_4_TIRF', 1, 'end', 1, 1);
```

2. Find trajectories and output them to one excel file

The functions used are FindTrajects.m and linkTrajSegments.m. The first one finds bright spots, finds their centres and joins spots in subsequent frames into trajectory segments. The second one joins segments into longer trajectories and outputs the resulting tracks to an excel file.

Important: you always need to run linkTrajSegments.m, so that an output excel file is generated for subsequent analysis.

Use “doc name of function” for further info/help.
The above functions require the correct parameter settings (size of spots, etc) to function.
Set the parameters for each function in external functions called:

- paramsForFindTrajects.m, for function FindTrajects.m.
- paramsForLinkTrajSegments.m, for function linkTrajSegments.m.

There are two folders located in the same folder (myMatlabFiles) where all the Matlab functions for the code are. These are:
...\Isabel\myMatlabFiles\Eg_ParameterScripts and
...\Isabel\myMatlabFiles\scriptsThatHelpRunCode.
The first folder contains examples of the **Parameter Files** that you need to copy and place into the analysis folder where your image sequences are saved ("current directory" for Matlab). Then you need to modify the values of the parameters in these files to find the optimum ones for the image sequences you are looking at, and save the changes before running other functions.
The second folder contains examples of scripts used to guide you in running the analysis (all the "runToAnalyse"-type files for different kinds of analysis are there).

To set the parameter values in paramsForFindTrajects.m, look at the bright spots in the single frames to estimate their size in pixels, make sure the size `inner_circle_radius` in pixels fits the whole bright spot, and make `subarray_halfwidth` a couple of pixels larger than that on each side. Bear in mind that the total spot intensity will be calculated within the inner_circle_mask, and that the background intensity will be calculated from the area inside the square subarray of the given halfwidth, outside the inner circle.
Make sure you do optimise this parameter values for each data set, do not just run with any values!

For paramsForFindTrajects.m, the key parameters are:

```
subarray_halfwidth = 15; % (Default: 8 pixels). Halfwidth of image
square subarray
inner_circle_radius = 10; % (Default: 5 pixels). Radius of inner
circular mask that moves inside the fixed square subarray.
gauss_mask_sigma = 5; (this one is quite robust to change)
```
(eg for Charlotte’s data)

```
% PARAMETERS for deciding if we accept a spot centre found by the
function findSpotCentre1frame or not:
sigmaFit_min = -inner_circle_radius; % minimum acceptable sigma of
gaussian fit to spot, in pixels (2) (-3).
sigmaFit_max = inner_circle_radius; % maximum acceptable sigma of
gaussian fit to spot, in pixels (4) (3).
SNR_min = 2; % minimum acceptable signal-to-noise ratio (at least 2)
(SNR as defined in findSpotCentre1frame.m).
```

```
rsq_min = 0.2; % minimum acceptable r-square value (0.2) (goodness of
gaussian fit to spot).
```

Parameters SNR_min and rsq_min will determine how many detected spots you accept for further analysis.

```
% PARAMETERS for building trajectories:
% For linking spots in current and previous frames:
d_01_max = 5; % max distance in pixels between spot centres in current
and previous frames, for linking them into a trajectory (5).
Iratio_01_min = 0.5; % min ratio of total spot intensities (after bgnd
subtraction) (0.5).
Iratio_01_max = 3; % max ratio of total spot intensities (after bgnd
subtraction) (frame k-1/frame k) (large enough value (3) to account for
blinking).
SigmaRatio_01_min = 0.5; % min ratio of spot widths (sigma of Gaussian
fit) (0.5).
SigmaRatio_01_max = 2; % max ratio of spot width (sigma of Gaussian fit)
(2).
```

These are important! Take into account the mobility of your spots and your frame rate, and how far they might move between two frames… Also the allowed ratio of intensities and ratio of widths to connect two bright spots into a track.

```
% For linking loose spots in current frame and 2 frames ago (jump of 1
frame in trajectory):
d_02_max = 5; % max distance in pixels between spot centres in current
frame and 2 frames ago. (default: 5)
Iratio_02_min = 0.2; % min ratio of total spot intensities (after bgnd
subtraction).
Iratio_02_max = 3; % max ratio of total spot intensities (after bgnd
subtraction).
SigmaRatio_02_min = 0.5; % min ratio of spot widths (sigma of Gaussian
fit).
SigmaRatio_02_max = 2; % max ratio of spot width (sigma of Gaussian
fit).
```

Same as before but for connecting one spot with another one two frames ago.
If you don't want to do this, make `d_02_max` = 0.

```
% Parameter to exclude a region from accepting spots:
exclude_region = 0;
```

If this parameter is set to 1, you will be excluding a framing border of width (`subarray_halfwidth`) around the top half of the image, and the same for the bottom half of the image, so spots in the central region of the image will not be detected.
If the parameter is set to 0, no framing border is excluded from the analysis, use 0 for images which are not split into two colour channels.

For paramsForLinkTrajSegments.m, the parameters to tweak are:

```
% PARAMETERS for linking trajectory segments:
% For linking end spot in one trajectory with start spot in another
trajectory:
```

```
d_01_max = 5; % max distance in pixels between spot centres.(5)
Iratio_01_min = 0.5; % min ratio of total spot intensities (after bgnd
subtraction).(0.2)
Iratio_01_max = 3; % max ratio of total spot intensities (after bgnd
subtraction). (Large enough to account for blinking, note that Iratio is
for frame k-1/frame k.)(3)
SigmaRatio_01_min = 0.5; % min ratio of spot widths (sigma of Gaussian
fit).(0.5)
SigmaRatio_01_max = 2; % max ratio of spot width (sigma of Gaussian
fit). (2)
Frames_away_max = 2; % max separation in frames (i.e., prop to time) for
trajectory segments to be linked.(default = 2). Write 1 if you want no
jumps (no frame jumps) in the segment linking.
% At most we skip one frame when Frames_away_max = 2.
```

In general, you should keep these the same as the parameters named the same within function findTrajects.m.

```
- Find trajectories and output them to one excel file:
Generic functions and inputs:

spot_results = FindTrajects(image_label,start_frame,end_frame)

linkTrajSegments(image_label,start_frame,end_frame,spot_results,data_set
_label)
```

You need to find the first frame where there is fluorescence emission in the image sequence, as well as the last frame you want to analyse. Do this by using the Andor SOLIS software to open the image. Choose the second brightest frame to make sure (eg, frame 18 to 300).

```
s4 = FindTrajects('1030_4_TIRF',18,300);
linkTrajSegments('1030_4_TIRF',18,300,s4,'EGFR_eGFP');

save 'resultStructures' 's*' % save all result structures in a .mat
file.
```

Make sure the name of the output of function “FindTrajects”, i.e., “s4” in the above example, is the same as the name of the fourth input of function “linkTrajSegments”.
Last input of function “linkTrajSegments” is just a label for your output excel files with the trajectory data.

You can stop the analysis at any time by pressing Ctrl+C at any time on the command window.

The first function takes a while to run. Usually the best option is to leave it running overnight for a few image sequences, writing a “runToAnalyse.m” script of instructions.

3. Generate lists of good tracks after visual inspection

goThroughTracksVideo(image_label,n_traj_start,n_traj_end,minPointsTraj)

This function will show you a video of the found and accepted tracks so that you can label them as good (1) or not (0). This generates the file "good_track_nums_3ob.mat" inside the current directory. It is useful to do a visual inspection in case there are beads on the image or tracks found outside the cell region, bad tracking or other anomalies...
The minimum value that minPointsTraj can take is 3.An

goThroughTracksVideo('3ob',1,'end',3)

As you see a video of the trajectory overlayed on the image sequence, enter 1 to accept a track as “good” and press enter, or enter 0 and press Enter to reject that track.s

This generates the file good_track_nums_1030_4_TIRF.mat in the current directory.

4. Analyse each good track

```
Then use showManyTrajAnalysis2.m (which calls showTrajAnalysis2.m) to
produce one analysis excel file and graph per analysed track:
```

Similarly to before, you need to use a file with parameters:
Hereon the notes are not so thorough, think and choose the right analysis choices...

You need a copy of paramsForShowTrajAnalysis2.m in the current Matlab directory and to adjust the values of the parameters within this script to the desired values. Get a copy of that file from folder: ...\
myMatlabFiles**Eg_ParameterScripts.**

```
% See paramsForShowTrajAnalysis2.m
%
showManyTrajAnalysis2(image_label,n_traj_start,n_traj_end,start_frame,ts
amp,pixelsize_nm,showVideo,minPointsTraj)

showManyTrajAnalysis2('1030_4_TIRF',1,'end',18,0.03,40,1,3)
```

A folder is generated which contains the excel and graph files for the analysis of each track:
`EGFR_eGFP_1030_4_TIRF`

Getting the initial intensities from all tracks

Need to obtain a reliable photobleaching time constants

This was done with the Oxphos dataset by fitting the whole cell intensity (not the best way), see 'fullCellIntensity.m' guide with runToanalysise_WholeCell_I

Once you have tau, use `analyseSetOfTracks2` to back-fit the intensity of all tracks with a fixed exponential time constant to get a fixed intensity.

Or could be done by making a histogram with the initial intensities(I0s) of all tracked spots (would be better) or tau

Colocalisation analysis

Find the script file `runToAnalyseColocalis.m` to use as a guide
(look for it within folder ...myMatlabFiles**scriptsThatHelpRunCode**)

Might need to modify the functions or the videos (merge them) since the code's functions are written to use the split array for two colour channels (as with the oxphos data).

The cross correlation method is very sensitive to edges therefore all the 30s in the argument of eg
```
% [ybf497 xbf497] =
overlapBF('497',1,'end',30,30,30,30,1,1);
```

Colocalisation on frame averages

Better to use regions of interest around the cells than the whole frame,

Eg the line
```
% quickGraphColocalisation('498','497',7,'end',roi498,'_roi')
```
In `runToAnalyseColocalis`

When the function is run, it will create a folder called collocalisResults and put the results in it. (the same will be done in the frame by frame case.
Isabel makes folders for specific runs eg colocoalasitionFRameByFrame (see in coloaclisResults)

Colocalisation frame-by-frame

Eg
```
% colocFrameByFrame498 =
colocalisationFrameByFrame('498','497',7,7+24,7,'end',roi498,'_roi',0.144,1.5);
```

Colocalisation track by track

A folder called colocResultsTRacCKBYTrack is made and takes the ruslts

```
      %
analyseSetOfTracksColocalis(minNumPointsInTrack,minNumFramesOverlap)
```

In vitro **analysis – (recommencing on 18/12/12, notes)**

Analysis of single-molecule bright spots on slides, trying to quantify number of molecules per spot and single molecule brightness.

Using ‘analyseFixedBrightSpots.m ‘ in in vitro folder in myMatlabFiles

Similar format to the parameter files,

We have to change params within this file in the invitro analysis

The following two parameters are the most important to get right.

It’s better to have many sparse images than one densly spotted image (and could have a large sub array half width, giving a better background measure). Ideally you’d have a slightly larger innercircle radius (a bit larger than the spot, gives a better measure of intensity of spot))

Open video and find frame and count diameter of a spot in pixels

~10 frames across

Then add a couple and divide by two to get the half width for ‘innercircleradius’ parameter

6

Then make the square around it at least 2 pixels more on each side then put the square radius, ‘subarrayhalf width’.

8

```
            subarray_halfwidth = 8; % (Default: 8 pixels). Halfwidth of
image square subarray
% ROI, typically a square of size 17x17 pixels.
inner_circle_radius = 6
```

Check other spots to see if the measured spot is representative.

Then set

Think of roughly fwhm wrt 6 so either 3 or 4

```
      gauss_mask_sigma = 2; % (Default: 2 pixels). Size in pixels of the
applied Gaussian mask.
```

Think of roughly fwhm wrt 6 so either 2 or 3 (get the order of magnitude right)

```
guess_sigma_Fit = 3; %
```

Change the min snr

```
      params.SNR_min = 1.3; % minimum acceptable signal-to-noise ratio
(at least 2) (SNR as defined in findSpotCentre1frame.m).
```

play with this to make is only accept spots that are gaussiany enough shape (eg top hat would have a low rsquare value)

```
      params.rsq_min = 0.2; % minimum acceptable r-square value (0.2)
(goodness of gaussian fit to spot).
```

Save

Go to run to analyse.m

Now making the image_lable be the identifying umbers in the tiff file name.

A couple of things needed changing

Made the region of interest the whole array

And allows you to set the min and max of the second plot on intensity against time

See these in runtoanalyse, these were then found in analysefixedbrightspots to change

```
% limitsROI = [250 500 250 500];
% limitsYzoom = [-2000 6000];
```