\_\_\_\_\_Insallation

The code referenced herein was created in order to quickly and easily analyze tracking data obtained by using imageJ’s manual tracking, although it can be used for any X and Y values.

Because some of the code relies on other bits of code it is important to add the entire “track analysis” folder to MATLAB’s path. This tells matlab to search for functions in this folder.

Press “set path”, on matlab’s home page.

\_

Press “add with subfolders”.

Add the location of the “track analysis”.

Press “Save”.

Good. Now all functions should be available on your machine.

As many of the functions are based on, or are mere implementation of Jean-Yves Tinevez’s MSD analyzer, I highly recommend going over his documentation.

Full description of MSD analyzer can be found here: https://tinevez.github.io/msdanalyzer/.

Instructions for using analyzeTracks.m

first create and initialize a documentation variable like so:

my\_documentation = documentation

you will get:

documentation with properties:

name: []

date: '05-Apr-2014'

description: []

pixelSize: []

frameInterval: []

timeUnits: []

spaceUnits: []

Define its properties according to your imaging parameters like so:

my\_documentation.frameInterval = 2;

my\_documentation.pixelSize = 0.239;

my\_documentation.timeUnits = 'sec';

my\_documentation.spaceUnits = 'µm';

I find it comfortable to save the data in a structure format, with one element being the “ma” (which stands for “MSD analyzer”), and the other one being the results.

myExperiment.ma = ma;

myExperiment.results = results;

First, to create the ma variable, you need to create a cell array with each cell containing a nX3 track in this format –Frame/X/Y.

You can create the ma like so –

myExperiment.ma = msdanalyzer(2,'µm', 'sec')

And add the tracks like so-

myExperiment.ma = myExperiment.ma.addAll(tracks)

If your tracks are stored as excel files, with each track contained in one sheet, or the track number indicated by a marker (look at “sampleFile.XLS” for reference), you can use the “createMaFromXls.m” function, automatically import all tracks and create the ma, like so:

myExperiment.ma = combineMa(createMaFromXls(fullPath, my\_documentation))

If your tracks are stored in text files in the format of “sampleFile.txt”, you can use createMaFromText:

myExperiment.ma = createMaFromTxt(files, my\_documentation);

After adding the ma file, you should run the next command, to obtain alpha values (if you plan on using them):

myExperiment.ma = myExperiment.ma.fitLogLogMSD

How to find a problematic cell –

Click the cursor data icon C:\Users\user\Desktop\Untitled-2.jpg and select a unique point(one which the track does not share with another track). A bubble appears with x and y values. Right click the bubble and click "Export cursor data to workspace". Give the variable whatever name you choose, for example "cursor\_info". Then, call this function:

RTfindTrack(control,cursor\_info)

The program will output the track number.

you can now use the built in function “plot tracks” in the ma object like so:

myExperiment.ma.plotTracks

I highly recommend using the plot tracks method at this point as it can visualy reveal any tracks that clearly contain an error, like too big of a variation in y axis. Such tracks should be omitted from the analysis.

You can now call anlayzeTracks function to create the results table (maNew is the filtered data):

[myExperiments.results, maNew] = analyzeTracks (myExperiments.ma, my\_documentation);

Put the filtered data into ma:

myExperiments.ma = maNew

Re-analyze with the filtered data:

[myExperiments.results, maNew] = analyzeTracks (myExperiments.ma, my\_documentation);

The results table contains all information for each track. For a more detailed explanation about each parameter, view comments in "analyzeTracks.m":

edit analyzeTracks

With the data analyzed, you can now use several visualization funtions:

To plot distance over time:

RTplotMSD2(control, treated, my\_documentation,{'one','two'})

To plot MSD:

RTplotMSD(control.ma,treated.ma,my\_documentation)

To plot histograms of instantaneous velocities:

RTplotVelocityHistograms(control,treated, my\_documentation,{'one','two'})

To plot bars:

RTcompare(control,treated, my\_documentation)

To plot pie chart:

RTpie(control,treated)

Plot parameters over time:

RTtrackMap(control,my\_documentation,1)

Where '1' is the track number you want to look at. If, based on the data from the result table, you find that track #15 is interesting, then write 15 instead of 1.

Count unique values in excel

=SUM(IF(FREQUENCY(B1:B244,B1:B244)>0,1))

using Rtexperiment.m (named historically after Rabies Team)

This is recommended when you have a folder containing subfolders with one or more txt files.

The code will generate a matlab structure with sub-catagories named after the subfolders.

For example, suppose you have a folder “my experiment”, whithin it there are three subfolders:

1. control (containing any number of txt files).

2. treatment\_A (etc..)

3. treatment\_B (etc..)

to create the structure write:

myCoolExperiment = RTexperiment ('/Users/michael/imaging/my experiment',documentation, 3).

then, you can call any subexperiment simply by typing:

myCoolExperiment.treatment\_A

prerequisites:

recommended - dir2.m

last updated 14.12.16