1 Overview

This package is one of the supportive materials of paper "". It is designed for fluorescent emitter(molecule) tracking at super-high density. Other functions such as microscopy imaging movie simulation of motion molecules and mapping molecule motion barriers from trajectories are also included.

We program the core "sparse linking EM" algorithm using C language, and other codes are written in matlab. The C code has to be compiled before using, any compiler support C99 standard(gcc for linux, intel c compiler or mingw for windows) should be enough to do the work.

Our package doesn't require extra software or package to execute.

2 Installation and usage

Compile "testtracklspfull.c" under folder "Ccode" into executable file, name it "afulllsp.out" and place it into folder "bin". It is OK for linux users to enter folder "Ccode" and simply run "compile.sh", "compile.sh" calls gcc to compile the C codes:

gcc testtracklspfull.c TrackingFunctions.c EM_Tuning_Tracking.c EMfunctions.c -O3 -lm -o ../bin/afulllsp.out

Executable file "afulllsp.out" is called by matlab code "tracksolverlspfull.m" under folder "heterosolver". Make sure this matlab code correctly call "afulllsp" by either add the path of folder "bin" to system environment $PATH or modify the line in "tracksolverlspfull":

Add the path of folder "bin" to system environment $PATH, add this line to /.bashrc:

export PATH=$PATH:PathToYourSparseTracking/bin

Modify the line in "tracksolverlspfull":

system(['afulllsp.out ', f2, ' ', f3, ' <', f1, ' >', f4]);

Move 'afulllsp.out ' to your path of afulllsp.out (keep the space character after .out).

After the compile, paths to the functions should be added when users run our package in matlab, the script "addpathtx.m" will do the job. Enter the folder of our package and run addpathtx in matlab console.

If you want to avoid running addpathtx.m every time you start matlab, add the command to run "addpathtx.m" in matlab startup script, here's what the command should look like(on my computer, linux mint):

run('/home/username/SparseTracking/addpathtx.m');

3 Demos

Demos are under folder "demo", detailed comments are available in these scripts.

3.1 demo 1, Simulation of microscopy imaging movie of Brownian molecules

The simulation consists of 2 parts, conducted by function "SimuTracks". First, SimuTracks simulate the movement of emitters(molecules), second, Simu-

Tracks simulate the imaging of fluorescent emitters. The molecules in this demo follow Brownian motion. Parameters are commented in detail in this demo.

Other than "SimuTracks" we also provide other functions simulate molecules show different motion behaviour. These functions are located under folder "TrackingSimu"

3.2 demo 2, Solve trajectories from microscopy imagine movie(s)

This demo simulates microscopy imagine movie(s), and solve the trajectories from the movies. The simulation is same to the condition in figure 2, and the parameters of the simulation in figure 5 are also included in this demo(commented). Hyperparameter for the solver are tuned using a small "Training" set, movies with ground truth result of trajectories are provided in this "Training" set. For users "Training" set is always available from simulation, there may be other ways to obtain the "Training" set. As only 3 hyperparameters need to be tuned, the "Training" set doesn't need to be large as traditional machine learning algorithms required. Default combination of hyperparameters are defined in function "defaultparaset" under folder "paratunning".

Solvers located under folder "heterosolver", functions for hyperparameter tuning is under folder "paratunning". Integrated code is under folder "foruser".

3.3 demo 3, Reconstruct and visualize motion barriers from trajectories

This demo simulates molecules have confined motion, solve the trajectories, and reconstruct motion barriers from the solved trajectories. The result is visualized.

We realize that reconstruct motion barriers from trajectories is a meaningful question for biologists, and it is novel in biological imaging processing area. However, that is not the focus of our paper(package). This visualization only works when molecules are even distributed. The functions conducting the Monte Carlo algorithm are under folder "ink", as the algorithm works similar to adding a drop of ink into a bottle of water.

4 Data structure

4.1 microscopy imaging movie

3D matrix, each entry records the number of photons captured at the corresponding pixel. The coordinates of entry (i,j,k) is (i-1, j-1, frame k-1) as x,y,t all starts from 0.

4.2 trajectory set

Represented by $l \times 5$ matrix, 5 columns are [x,y,I,t,id]. x,y are coordinates, I are fluorescent intensities, t are frame numbers, id are non-negative numbers where 2 lines with same id belongs to the same fluorescent emitter (molecule).

4.3 trajectory set for emitter tracking solvers(general trajectory set)

In addition to trajectory set, emitter tracking solvers also use and update background fluorescent intensities. Since background fluorescent intensity is mathematically equivalent to a(or a mixture of some) special emitter, whose PSF (point spread function) cover the whole image area, we consider the trajectory set together with background intensity as general trajectory set.

A general trajectory set GT is represented as a matlab structure: $GT.trj$ is $l \times 5$ ordinary trajectory set, and $GT.no$ is a $T \times 1$ array, records background intensity of each frame.

In this version only even background intensity is supported, strategy to deal with uneven background fluorescent intensity is discussed in the appendix of our paper and will be supported in our next version.

5 Key functions

5.1 Molecule motion and microscopy images simulation

5.2 Emitter tracking solvers

5.3 Mapping molecule motion barriers from trajectories

5.4 Visualization

5.5 Operation of trajectory sets

6 Future work

As illustrated in the appendix of our paper, SparseTracking is able to handle non-even background fluorescent intensity, detailed algorithm has been presented. We will integrate this algorithm into our package in the subsequent version.

Also, we are considering increasing the package's support for emitters with non-Gaussian PSF(point spread function), this will allow users to use more accurate PSF models.

In our current implementation of Sparse Linking EM algorithm, all the positions and fluorescent intensities are updated in every EM iteration, this is unnecessary for emitter trajectories whose fluorescent intensity have already decay to zero. We are considering the removal of zero intensity emitters during the program running, this will increase the computational speed 5 times or even more.