**Summary of Code**

Last Updated: 24 February 2024

**‘BrightnessV27’**

BrightnessV27 is a FIJI code that can identify single-molecules through loop-thresholding and outputs information about single-molecule ROIs, trajectories, etc. It can also calculate the brightness of single-molecule images through an integration approach (with automatically generated histogram output). However, we note this approach is *not accurate* when the single-molecule dyes are turned on only a small percent of the total number of frames, which is very common in single-molecule imaging. Instead, trajectories should be post-processed by FluorTrajAnal 2pt6 and 3pt1 codes, which perform 2D Gaussian fitting to identify ‘on’ trajectories (and filter out ‘off’ trajectories which add to noise).

BrightnessV27 has a simple GUI and is designed to be used completely hands-off. It requires one package to be installed, ‘BAR’, if histogram plotting is enabled.

**‘ColorProcessing + LUTs’**

ColorProcessingV4 is a FIJI code that can generate color-thresholded single-molecule images. It requires one package to be installed, ‘BAR’. It should be run only after BrightnessV27 and the FluorTrajAnalysis workup, or with a code that generates comparable outputs.

ColorProcessing requires several inputs.

First, it requires a single frame image of the single-molecule image in which to color. This is generated by BrightnessV27 under Folder Label #5 (5\_frame\_average\_stacks\_threshold).

ColorProcessing also requires a ROIs to threshold. This is generated by Brightness V27 under Folder Label #7 (7\_ROIs). In the manuscript, Threshold\_SUM (the sum from the entire threshold sweep), 01\_ROIset\_SUM (not expanded) was always used.

ColorProcessing lastly requires a CSV table of photon values. This is generated by FluorTrajAlternate2pt6 or 3pt1 (parallel version of code). Note: the first column of the CSV MUST say ‘Budget’. With FluorTrajAlternate2pt6 or 3pt1 output, this column title will need to be entered in manually.

ColorProcessing 2-color LUTs are provided. The LUT used is specified in the FIJI file (//Run Color Coder).

**‘FluorTrajAnalV2pt6/V3pt1 (V3 is the parallel version)**

FluorTrajAnal is a MatLab code that post-processes BrightnessV27 output to produce detected photon values. Version 2 can process one single-molecule timelapse/ROI file and is useful for testing, as it contains a few extra test options (for example checking single frames in the single-molecule timelapse). Version 3 can process multiple single-molecule timelapses/ROI files without re-starting MatLab and is designed to be left running for long stretches of time.

For V2, the single-molecule timelapse location and ROI location must be specified (see code for instructions).

For V3, the root directors for single-molecule timelapses and ROIs must be specified (see code for instructions). This means all single-molecule timelapses and ROIs must be in respective directories together.

Additional Notes: FluorTrajAnalV1.7 (‘OLD Code’) uses a total integration approach similar to the FIJI code, but with extra cutoff/thresholding abilities. It is parallelized (can process multiple single-molecule timelapses/ROI files without re-running the code).

**‘FitFunctTest’**

FitFunctTest is a fitting function that is required for the use of FluorTrajAnal scripts.

**‘HistogramConvoluter’**

Used for double-sampling histograms. Can specify the number of repeats which determines how many times the double-sampling is performed (suggested value 100).

**‘PercentLoop’**

Loops over percent threshold values to find a value of the overlap between two histograms (getting a value of 2 or 200% means there is no overlap between samples, 1 or 100% means there is complete overlap between samples). Plots the overlap vs. the percent threshold for all values in the loop.

**‘Photon Budget Compiler’**

Photon budget compiler is a MatLab code that allows the merging of different detected photon trials into a single file. A cutoff for the maximum photons detected can be applied. The parallel version can merge detected photons not just within one folder (typically corresponding to a single date of experiments with *n* number of trials), but also merges multiple folders (corresponding to multiple dates of experiments). To make handling the compiled data easier, the parallel version creates folders for photons detected ‘budgets’ and median values ‘medians’ (used to determine standard deviation) with values from each folder.

**‘Recover Gauss’**

Recover Gauss is a MatLab code that can recover raw output from FluorTrajAnalV2/V3 very quickly (within seconds). Since the output of from these codes is exported in csv format, Recover Gauss re-imports these tables back into MatLab. ‘Recover Gauss’ can save the user a very large amount of time in comparison to re-running FluorTrajAnal codes again if small modifications to mathematical operations need to be performed.

**‘Bootstrapper’ (‘Other Code’)**

Bootstrapper (1 and 2) are MatLab codes that can be run to determine histogram spread (these could be run on compiled photon budget outputs). Bootstrapper 1 is the traditional bootstrapper algorithm (see *Chernick, M. R. Resampling Methods. WIREs Data. Mining. Knowl. Discov. 2012, 2, 255-262*), whereas Bootstrapper 2 is a variant. However, we found that standard deviation between trials is much larger than bootstrapper error within a trial, therefore standard deviation is reported in the manuscript and the bootstrapper code did not end up being needed.