This document describes the steps to run the computational analysis pipeline for Single Molecule Imaging – Fluorescent Speckle Microscopy (SMI-FSM) experiments, as described in "Multiscale imaging and quantitative analysis of plasma membrane protein-cortical actin interplay" (1).

CONTENTS

1 GLOSSARY

FSM	Fluorescent Speckle Microscopy
QFSM	Quantitative Fluorescent Speckle Microscopy
SM	Single molecule
SMI	Single Molecule Imaging
ROI	Region Of Interest
MVRG	Multivariate regression

2 SOFTWARE INSTALLATION

The software is written in MATLAB. It has been tested on **MATLAB R2020a** on **Windows 64-bit** and **Linux 64-bit OS**.

Required MATLAB toolboxes:

- Optimization Toolbox
- Signal Processing Toolbox
- Image Processing Toolbox
- Statistics and Machine Learning Toolbox
- Financial Toolbox
- Curve Fitting Toolbox
- Computer Vision Toolbox

The SMI-FSM package contains all functions (but not the Matlab toolboxes) necessary to run SMI-FSM analysis, except for:

- U-track: Needed for single-particle tracking (*Jaqaman et al., Nature Methods 5, pp. 695-702 (2008) (2)*). Please download from https://github.com/DanuserLab/u-track.
- QFSM: Needed for quantitative speckle analysis (see *Mendoza et al., Current Protocols in Cytometry 62:2.18.1–2.18.26 (2012) (3)*). Please download from https://cloud.biohpc.swmed.edu/index.php/s/JaQD4u3SDkxXjZO/download.
- YALMIP: Needed for estimating particle apparent assembly state from its intensity and merging and splitting history (see *de Oliveira and Jaqaman, Biophysical Journal 117(6):1012-1028 (2019) (4)*). Please download from https://yalmip.github.io/.

Please download all aforementioned software packages to complement the provided software in the SMI-FSM package.

3 EXAMPLE DATA

Please download the zip file from https://cloud.biohpc.swmed.edu/index.php/s/wLAGX5qqFGWjgPj After unzipping, the folder "exampleData" in the download contains example datasets to test the provided software and ensure that it is installed correctly. "exampleData" contains the output of u-track and QFSM, as described above. It contains the following folders, subfolders, and files:

A. "**Speckle**": This folder is where the actin FSM movies and subfolders storing QFSM analysis results are saved. There are three subfolders:
    a. Folder "**m-01-01**" illustrates the case when cell ROI masks were created through the QFSM package. It has the "**QFSMPackage**" subfolder, which contains the following subfolders as automatically generated from the QFSM software package:
        i. refined_masks
        ii. speckleTracks
        iii. speckles
        iv. kineticAnalysis

        These are the minimally required QFSM results to run the SMI-FSM analysis software. Note that user can create cell ROI masks through either the QFSM package (as done for this example movie) or manually (as done for next example movie).
    b. Folder "**m-01-02**" illustrates the case when cell ROI masks were created manually outside of the QFSM package (See section 7 "Instructions for setting up manual ("external") ROI"). It has the "**External_ROI**" subfolder (which contains the cell masks) and the "**QFSMPackage**" subfolder, which contains the minimally required QFSM results to run the SMI-FSM analysis software.
    c. Folder "**m-01-03**" corresponds to the actin movie "m-01-03.tif" and contains the full results output by QFSM. To view these results, run in the MATLAB Command Window:

        `movieSelectorGUI`

        Click "Open" and select "**m-01-03.mat**" in this folder. After the results are loaded, click "View".

B. "**SingleMolecule**": This folder is where single molecule movies and subfolders of u-track particle tracking results are saved. There are three subfolders, which are the SM counterparts of the QFSM subfolders described above:
    a. Folder "**m-01-01**" contains the "**TrackingPackage**" subfolder, which contains the "**tracks**" subfolder as automatically generated from the u-track software package.

b. Folder "**m-01-02**" contains the "**TrackingPackage**" subfolder, which contains the "**tracks**" subfolder as automatically generated from the u-track software package.

As above, "m-01-01" and "m-01-02" contain the minimally required u-track results to run the SMI-FSM analysis software.

c. Folder "**m-01-03**" contains the full u-track results corresponding to the single-molecule movie "m-01-03.tif" for visualization. To view these results, in the MATLAB Command Window, run:

```
movieSelectorGUI
```

Click "Open" and select "**m-01-03.mat**" in this folder. After the results are loaded, click "View".

C. "**exampleScript.m**": This MATLAB script contains instructions and commands to run SMI-FSM analysis pipeline on the provided example data.

D. "**exampleInputs_II.mat**": This MAT-file contains example inputs for running section II of exampleScript.m.

Even to run the analysis on the example dataset, users MUST create paths locating u-track and QFSM results corresponding to their local file folders, as instructed in the next section of this manual. The supplied "fsmPaths" and "utrackPaths" variables are examples only and will not run as-is.

E. "**exampleInputs_III.mat**": This MAT-file contains example inputs for running section III of exampleScript.m.

F. "**exampleInputs_IV.mat**": This MAT-file contains example inputs for running section IV of exampleScript.m.

G. "**exampleOutputs_II.mat**": This MAT-file contains example outputs expected from running section II of exampleScript.m. Because only section II requires users to create inputs different from those provided in the download, the variables in this file allow for quick run of the subsequent sections of exampleScript.m with only example variables.

4 IMAGING DATA FORMATTING, PARTICLE TRACKING AND QFSM ANALYSIS TO RUN SMI-FSM ANALYSIS

For analysis, the SMI and FSM counterparts of an SMI-FSM movie must be separated and saved as two separate movies (both as multi-tiff files). In the case of different sampling rates (similar to our work, where an SMI frame is acquired every 0.1 s while an FSM frame is acquired every 5 s), any empty actin frames must be removed. In the manuscript (1), this was done with the MetaMorph software (Molecular Devices, San Jose, CA).

Then, the SMI movie must be analyzed using u-track for particle tracking, while the FSM movie must be analyzed using the QFSM package for speckle detection, tracking and analysis. Please see the manuals of those software packages for detailed instructions on how to run them. Here, we briefly describe their output as relevant for SMI-FSM analysis.

U-track: The results of u-track are stored in a folder named "TrackingPackage" with the following necessary subfolder and file.

Subfolders	MATLAB file inside the subfolder	MATLAB variable inside the MATLAB file
tracks	Channel_1_tracking_result.mat	tracksFinal

QFSM: The results of QFSM are stored in a folder named "QFSMPackage" with the following necessary subfolders and files.

Subfolders	MATLAB or TIF files inside the subfolder	MATLAB variable inside the MATLAB file
speckles	cands_01.mat cands_02.mat …	cands
speckleTracks	_tracks.mat	MPM
kineticAnalysis	kineticMaps_01.mat kineticMaps_02.mat …	kinScore
refined_masks/ refined_masks_for_channel_1	refined_mask_m-01-01_c1_t01.tif refined_mask_m-01-01_c1_t02.tif …	

5 INSTRUCTIONS

This manual describes the basic options and parameters as performed in the manuscript referenced above (1). For all the functions of the pipeline, see each function's documentation for further instructions on different options and formats for inputs and outputs. There are 4 instruction sections.

5.1 PREPARING INPUTS FOR SMI-FSM ANALYSIS

This section contains 5 sub-sections (defining 6 input variables in total).

i. Locating QFSM results. The paths to QFSM analysis results must be saved in a cell array, called, e.g., "fsmPaths", with 1 entry per FSM movie. To set up "fsmPaths", navigate in MATLAB to the first FSM movie folder (e.g., m-01-01) and then run in the Command Window:

```
fsmPaths{1,1} = pwd;
```

Then navigate to the second movie folder, and run:

```
fsmPaths{2,1} = pwd;
```

And so on and so forth until all movies to be analyzed are added.

ii. Locating u-track results. The paths to u-track analysis results must be saved in a cell array, called, e.g., "utrackPaths", with 1 entry per SMI movie. The number of SMI movies in a dataset must equal the number of FSM movies in a dataset. There are two methods to create the utrackPaths input:

Method 1 (Manual)	Method 2 (Automatic, but requires a specific folder structure as described in "Recommended folder structure" section below (section 6))

In MATLAB, navigate to the first SMI movie folder (e.g., m-01-01). This folder should contain a subfolder called "TrackingPackage". Navigate to inside "TrackingPackage" folder. In the Command Window, run:
`utrackPaths{1,1} = pwd;`
Then navigate to the second movie "TrackingPackage" subfolder, and run:
`utrackPaths{2,1} = pwd;`
And so on and so forth until all movies to be analyzed are added
	In the Command Window in MATLAB, run:
	`utrackPaths = getUtrackPackPaths(fsmPaths);`

Examples of the `fsmPaths` and `utrackPaths` inputs are provided in "exampleData". However, to run the provided example, users will need to modify the paths so that they correspond to their local filesystem structure.

iii. Parameters for SM analysis. Parameters for extracting u-track results, processing them and calculating SM tracklet properties must be saved as a structure, called, e.g., "smConditions", defined as follows:

Suppose there are N movies to be processed, i.e., N = length(utrackPaths) (= length(fsmPaths)). In the Command Window in MATLAB, run:

```
smConditions = struct('minLft',10*ones(N,1), 'rad_density',7*ones(N,1),
'windAvg',50*ones(N,1), 'divFlag',(-1)*ones(N,1), 'maskLoc',2*ones(N,1));
```

Table of fields in smConditions. Fields in blue must be used with "recommended value" (software does not support any other value).

Field names	Recom-mended Value	Description	Notes
minLft	10	Minimum lifetime (in SMI frames) of an SM track to be employed for analysis.	SMI-FSM uses divide-and-conquer moment scaling spectrum motion analysis from *Vega et al., Biophys. J. 114, 1018-1025 (2018) (5)*, which uses a rolling window of 11 for initial track segmentation. Hence, removing single-molecule tracks of 10 frame or less reduces the number of unnecessary tracks having to go through the analysis.
rad_density	7	Radius (in pixels) around an SM localization to calculate the local SM density per SMI frame.	
windAvg	50	Number of SMI frames corresponding to each FSM frame.	Determined by user's image acquisition scheme. In our work, an SMI frame is acquired every 0.1 s, while an FSM frame is acquired every 5 s, thus windAvg = 5/0.1 = 50.
divFlag	-1	Scheme for temporal assignment of SMI frames to FSM frames. -1 indicates that SMI frames 1-51 are assigned to FSM	Currently, SMI-FSM software only supports divFlag = -1.

		frame 1, SMI frames 51-101 are assigned to FSM frame 2, etc.	
maskLoc	1 or 2	Flag indicating the type of mask to use for outlining the area of interest for analysis. 1: External mask (i.e., manually drawn) 2: Refined mask from QFSM package	It is possible that, within one dataset, some movies have automated QFSM package masks (preferred) and others have manual external masks (when needed if the automated masking failed). In this case, please interactively open maskLoc in MATLAB and manually update the maskLoc value for individual movies.

iv. Parameters for speckle analysis. Parameters for extracting QFSM results, processing them and calculating individual speckle properties must be saved as a structure, called, e.g., "actinConditions", defined as follows:

Suppose there are N movies to be processed, i.e., N = length(fsmPaths) (= length(utrackPaths)). In the Command Window in MATLAB, run:

```
actinConditions = struct('minLft',2*ones(N,1), 'rad_density',7*ones(N,1),
'pos_std',zeros(N,1), 'maskLoc',2*ones(N,1));
```

Table of fields in actinConditions. Fields in blue must be used with "recommended value" (software does not support any other value).

Field name	Recom-mended Value	Description	Notes
minLft	2	Minimum lifetime (in FSM frames) of a speckle to be employed for analysis.	Determined by user's image acquisition scheme. In our analysis, minLft = 2, in order to remove ghost speckles (6).
rad_density	7	Radius (in pixels) around a speckle to calculate its neighborhood density.	The speckle's neighborhood density is distinct from the local speckle density calculated around

			SM tracklets (described below). It is calculated in the code, but it is in fact not used for any further analysis in the manuscript.
pos_std	0	Logical flag to determine whether Gaussian noise is to be added to speckle positions.	Currently, SMI-FSM software only supports Pos_std = 0, i.e. Gaussian noise is not added to speckle positions.
maskLoc	1 or 2	Same as in smConditions structure above	Same as in smConditions structure above. Note that `actinConditions.maskLoc = smConditions.maskLoc`.

v. Parameters for assigning SMI and FSM data into subcellular regions.

If of interest, the SMI and FSM data can be separately assigned into subcellular regions, namely "edge" and "center" regions. This is achieved by first subdividing the cell mask into these regions, and then applying the mask subregion classification to the SMI and FSM data.

In the MATLAB Command Window, run:

```
maskListClass = classifyMaskGrid(pathsToMasks, 'numErosion', 3);
```

The input `pathsToMasks` is the list of folders with .tif files of cell masks. Number of erosions is recommended as 3, which gives an edge region of roughly 4 μm from the cell edge. See function for details.

Applying this classification to the SMI or FSM data requires the variable `boxClassCondition`, which is defined as follows:

```
boxClassCondition.extractClass = [1 2];
boxClassCondition.firstFrame = combineConditions.firstFsmFrame;
```

vi. Parameters for SMI and FSM data integration. These parameters must be saved in two structures, called, e.g., "combineConditions" and "smactinFlag", defined as follows:

Suppose there are N SMI-FSM movie pairs to be processed. In the Command Window in MATLAB, run:

```
combineConditions = struct('matchRadius',3.5*ones(N,1),
'fsmInterval',50*ones(N,1), 'firstFsmFrame',1*ones(N,1));
```

```
smactinFlag = struct('combFlag',10*ones(N,1), 'synthData',zeros(N,1),
'randFlag',100*ones(N,1));
```

Table of fields in combineConditions.

Field names	Value	Description	Note
matchRadius	3.5	Search radius (in pixels) for matching SM tracklets to their neighboring speckles.	Determined by user's camera registration shift and spatial range of interest for SMI-FSM data integration. In our work, we employed a radius of 3.5 pixels to account for camera registration shifts in our microscope setup.
fsmInterval	50	Number of SMI frames corresponding to each FSM frame.	Determined by user's image acquisition scheme. In our work, an SMI frame is acquired every 0.1 s, while an FSM frame is acquired every 5 s, thus fsmInterval = 5/0.1 = 50. Same as smConditions.windAvg.
firstFsmFrame	1	The FSM frame that corresponds to the first SMI frame.	For some experiments, it might be desirable to acquire multiple FSM frames before the first SMI frame. In this case, if x FSM frames were acquired before the start of SMI imaging, then firstFsmFrame = x + 1.

Table of fields in smactinFlag. Fields in blue must be used with "recommended value" (software does not support any other value).

Field names	Value	Description	Note
combFlag	10	Flag indicating how SM tracklets and speckles are combined.	Currently, SMI-FSM software only supports input 10, which indicates speckle matching based on the frame-by-frame coordinates of the SM tracklets.
synthData	0	Flag indicating whether integrated SMI-FSM data	Currently, SMI-FSM software only supports synthData = 0.

		are synthetic (i.e., simulated) or not (i.e., experimental).	
randFlag	100	Number of times data would be randomized when multivariate regression is performed.	In our work, randFlag = 100 in section III (Instructions for running core SMI-FSM data integration functions). Because the functions for analyzing the biphasic relationship perform data randomization separately from regression analysis, randFlag = 0 in section IV (Instructions for running analysis of biphasic relationship in SMI-FSM integrated data).

5.2 INSTRUCTIONS FOR RUNNING SMI-FSM DATA INTEGRATION FUNCTIONS

The descriptions below explain the core function(s) of SMI-FSM data integration (see manuscript (1) for details):

`smPropPerTimeInterval.m:` processes SM tracklet data.

`actinPropPerTimeInterval.m:` processes speckle data.

`cleanActinPropPerTimeInt.m:` cleans up speckle data.

`smactinPropPerTimeInt.m:` integrates SM tracklet and speckle data.

`smactinMat.m:` reformats integrated SM tracklet and speckle data for downstream analysis.

`retainTassmInliers.m:` removes outliers from integrated SMI-FSM data.

`smactinClean.m:` reformats clean, integrated SMI-FSM data for downstream analysis

a. Processing of SM tracklet data

In the Command Window in MATLAB, run:

```
[smPropPerTimeInt, threshMet] = smPropPerTimeInterval(utrackPaths,
smConditions, fsmPaths, [], smSpanRadius);
```

The input `smSpanRadius` is defined as follows:

$$smSpanRadius = \frac{speckle\ domain\ size}{your\ pixel\ size} = \frac{3000\ (nm)}{your\ pixel\ size\ (nm)}(pixels)$$

Recommended value for speckle domain size is ~3 μm or 3000 nm, following Lee G et al. (7) and Lee K et al. (8). In our analyses, `smSpanRadius = 33.7 pixels.`

If of interest, to add subregion (edge/center) information to SM tracklets, run:

```
smPropPerTimeIntInMask = retainSmPropInMask(smPropPerTimtInt, threshMet);
```

```
[smPropPerTimeBoxed, smTrackletIdPerBox] =
gridSmPerTimeInt(smPropPerTimeIntInMask);

smPropPerTimeIntByBoxClass = stratifySmTrackletByBoxClass(smPropPerTimeBoxed,
maskListClass, boxClassCondition, smTrackletIdPerBox);
```

The inputs `maskListClass` and `boxClassCondition` are defined in section ***5.1. Preparing inputs for SMI-FSM analysis***.

The following three variables contain equivalent analysis results, just for the different subsets of data (as classified above):

- `smPropPerTimeInt` contains SMI information throughout the whole cell;
- `smPropPerTimeIntByBoxClass(:,1)` contains SMI information in the cell center;
- `smPropPerTimeIntByBoxClass(:,2)` contains SMI information in the cell edge.

b. Processing and cleaning up of speckle data

In the Command Window in MATLAB, run:

```
[actinPropPerTimeInt] = actinPropPerTimeInterval(fsmPaths, actinConditions);
```

To remove low actin speckle displacements that are dominated by noise, displacements with magnitude below an empirically determined value are set to zero (`speedThresh`; see manuscript (1) for details on reasoning and experiments needed to determine `speedThresh`). For this, run:

```
[actinPropPerTimeIntClnSpeed] = cleanActinPropPerTimeInt(actinPropPerTimeInt,
speedThresh);
```

Note: This function can also remove speckles with incomplete lifetimes by setting them to NaN (See manuscript and function documentation for details.) This option is available but not used in manuscript (1).

If of interest, to add subregion (edge/center) information to speckles, run:

```
[~, ~, actinPropPerTimeBoxed, ~, ~, speckleIdPerBox] =
gridActinPropPerTimeInt(actinPropPerTimeIntClnSpeed, 'enumerate');
```

```
actinPropPerTimeIntByBoxClass =
stratifySpeckleByBoxClass(actinPropPerTimeBoxed, maskListClass,
boxClassCondition, speckleIdPerBox);
```

The inputs `maskListClass` and `boxClassCondition` are defined in section ***5.1. Preparing inputs for SMI-FSM analysis***.

The following three variables contain equivalent analysis results, just for the different subsets of data (as classified above):

- `actinPropPerTimeIntClnSpeed` contains actin speckle information throughout the whole cell;
- `actinPropPerTimeIntByBoxClass(:,1)` contains actin speckle information in the cell center;
- `actinPropPerTimeIntByBoxClass(:,2)` contains actin speckle information in the cell edge.

c. Integrating and cleaning up of SM tracklet and associated local speckle data

The following inputs are needed to proceed:

The inputs "`smPropPerTimeInt`" and "`threshMet`" are outputs of the function "`smPropPerTimeInterval`" from section ***Processing of SM tracklet data***. The input "`threshMet`" indicates SM tracklets located within the ROI cell mask. The input "`actinPropPerTimeIntClnSpeed`" is the output of the function "`cleanActinPropPerTimeInt`" from section ***Processing and cleaning up of speckle data***. The inputs "`combineConditions`" and "`smactinFlag`" were described in ***Preparing inputs for SMI-FSM analysis***.

In the Command Window in MATLAB, run the following set of commands:

```
[combinePropPerTimeInt] = smactinPropPerTimeInt(smPropPerTimeInt,
actinPropPerTimeInt, combineConditions, threshMet, smactinFlag);
```

```
[combinePropPerTimeIntClnSpeed] = smactinPropPerTimeInt(smPropPerTimeInt,
actinPropPerTimeIntClnSpeed, combineConditions, threshMet, smactinFlag);
```

If subregion analysis is of interest, these functions can be called with `smPropPerTimeIntByBoxClass(:,1)` or `smPropPerTimeIntByBoxClass(:,2)`, and `actinPropPerTimeIntByBoxClass(:,1)` or `actinPropPerTimeIntByBoxClass(:,2)` instead.

```
[tassm] = smactinMat(combinePropPerTimeInt, combineConditions, smactinFlag);

[tassmClnSpeed] = smactinMat(combinePropPerTimeIntClnSpeed, combineConditions,
smactinFlag);
```

The outputs of running these commands, i.e., tassm and tassmClnSpeed, are cell arrays containing integrated SMI-FSM data for each movie in the analysis. Each cell in the array contains three cells.

- The first cell contains local speckle properties around the SM tracklets, with columns as follows:
    - 1: Magnitude of average speckle displacement (i.e., individual displacements are first averaged as vectors, and then the magnitude of this vector is calculated)
    - 2: Extent of co-movement between speckles
    - 3: Speckle density
    - 4: Average speckle lifetime
    - 5: Average speckle intensity
    - 6: Average speckle background intensity
    - 7: Density-weighted average speckle intensity
    - 8: Density weighted average speckle background intensity
    - 9: Average of speckle displacement magnitudes (i.e., individual displacement magnitudes are directly averaged)
- The second cell contains SM tracklet properties, with columns as follow:
    - 1: SM net speed over tracklet
    - 2: SM intensity, averaged over tracklet
    - 3: SM diffusion coefficient
    - 4: SM density
    - 5: SM spatial span
    - 6: SM apparent assembly state
    - 7: SM diffusion type
- The third cell contains the x- and y-coordinates of each SM tracklet.

For details on the formulations and formats of the properties listed, see function documentation and manuscript. The list above contains more properties than those used in the manuscript. For details on the biological interpretation of the subset of properties used in the manuscript (mainly speckle properties 3, 5 and 9 and SM properties 2, 3 and 7), see manuscript (1).

To remove outliers, by replacing them with NaN, for each speckle or SM property, in the Command Window in MATLAB, run:

```
[tassmInl, flagInl, tassmInlClnSpeed] = retainTassmInliers(tassm, thresholds, tassmClnSpeed);
```

The input "`thresholds`" contains empirically determined thresholds for outlier detection of each local speckle property around the SM tracklets and each SM tracklet property, as listed above in the first and second cells of the output "tassm". Using the interquartile range method (also called Tukey's fences) (MATLAB function isoutlier(data, 'quartiles')), an outlier is defined as a value that is below Q1-k(Q3-Q1) or above Q3+k(Q3-Q1), where Q1 and Q3 are the lower and upper quartiles, respectively, and k is the threshold input by user. We used the following threshold values for outlier detection (see manuscript (1) for details):

```
thresholds = {[3 NaN 2.5 NaN 4 2.5 2 1.5 2.5] [6 7 2 NaN NaN NaN NaN]};
```

To clean up the speckle properties "magnitude of average speckle displacement" and "average of speckle displacement magnitudes" by replacing any 0 values with NaN (to exclude them from further analysis), in the Command Window in MATLAB, run:

```
[tassmDblClnSpeed] = smactinClean(tassmInlClnSpeed, 'speckle', colIndxArr, thresholdArr);
```

We set the inputs as follows, where 1 indicates "magnitude of average speckle displacement" and 9 indicates "average of speckle displacement magnitudes":

```
colIndxArr = [1 9]
thresholdArr = [0 0]
```

The outputs `tassmInl`, `tassmInlClnSpeed`, and `tassmDblClnSpeed` are structured similarly to `tassm` and `tassmClnSpeed`.

5.3 Instructions for running functions for multivariate regression (MVRG) analysis of integrated SMI-FSM data

The core functions for SMI-FSM MVRG analysis are:

`runSmactinAggMovExplain.m`: performs MVRG analysis of an SMI-FSM dataset.

`multiCompareIndCellsMVRG.m`: performs statistical tests comparing the MVRG analysis results of multiple SMI-FSM datasets.

It is recommended that user navigates to a new, empty folder before running these functions.

See each function's documentation for detailed instructions on the different options and formats for inputs and outputs. In brief, these functions use the following core inputs:

- "**tassmDblClnSpeed**" is the output of function "smactinClean" from section ***5.2.c. Integrating and cleaning up of SM tracklet and associated local speckle data.***
- "**smactinFlag**" is the variable described in section ***Preparing inputs for SMI-FSM analysis.***
- "**spekLabelBase**" and "**smLabelBase**" are cell arrays of the names, respectively, of speckle and SM properties of interest. The list of properties is described in section ***5.2.c.***
- "**propSpk**" and "**propSm**" are integer vectors corresponding, respectively, to the number of properties of interest in "spekLabelBase" and "smLabelBase". The numbers are listed in section ***5.2.c***.
- "**maxNumPropVect**" corresponds to the total number of speckle properties and total number of SM properties in "tassmDblClnSpeed". Per section ***5.2.c***, this input will be [9 6] for 9 speckle properties and 6 SM properties (SM diffusion type in column 7 is not counted here).
- "**mvrgAggMatFlag**" indicates whether normalization should be performed before MVRG.
- "**cutOffInfo**" is the threshold at which data from "tassmDblClnSpeed" is split into 2 subsets before any analysis.
- "**outlierParam**" contains empirically-determined parameters for a by now defunct outlier detection process (not used in manuscript (1)). Please set this input to [ ].
- "**sigThreshInput**" p-value thresholds for statistical tests.

a. ***To perform MVRG analysis***, in the Command Window in MATLAB, run:

```
smifsmResults = runSmactinAggMovExplain(tassmDblClnSpeed, [], smactinFlag,
propSpk, propSm, maxNumPropVect, mvrgAggMatFlag, cutOffInfo, outlierParam,
spekLabelBase, smLabelBase, sigThreshInput);
```

Results of MVRG analysis are output into the following fields in the smifsmResults structure:

- "testResultSplitAbv" contains results of MVRG analysis on data above the input threshold.

- "`testResultSplitBel`" contains results of MVRG analysis on data below the input threshold.
- "`testResultAndSplit{1,1}`" contains results of MVRG analysis on the total input data.

These three fields contain equivalent analysis results, just for the different subsets of data, as explained above.

Visual results of the MVRG coefficients are output in two folders:

- "above": this folder contains results of MVRG analysis on data above the input threshold.
- "below": this folder contains results of MVRG analysis on data below the input threshold.

b. ***To visually compare between the different MVRG analysis results***, in the Command Window in MATLAB, run:

- If user is comparing 2 different MVRG analysis results:

```
[~, pValueIndCell] = multiCompareIndCellsMVRG(spekLabelBase, smLabelBase,
true, smifsmResults1.testResultSplitAbv, smifsmResults2.testResultSplitAbv)
```

- If user is comparing more than 2 different MVRG analysis results:

```
[~, pValueIndCell] = multiCompareIndCellsMVRG(spekLabelBase, smLabelBase,
smifsmResults1.testResultSplitAbv , smifsmResults2.testResultSplitAbv,
smifsmResults3.testResultSplitAbv)
```

The example above shows 3 inputs of MVRG analysis result; however, function "multiCompareIndCellsMVRG" can handle more than 3. Optimal figure plotting was tested for up to 5 input results.

These example commands employ the MVRG results of the data above the input threshold. To compare the results of MVRG analysis of data below the input threshold, input `smifsmResults1/2/3.testResultSplitBel` instead of `smifsmResult1/2/3.testResultSplitAbv` in the commands above. To compare the results of MVRG analysis of total data, input `smifsmResult1/2/3.testResultAndSplit{1,1}` instead.

Visualization of x MVRG analysis results and their comparisons is output in (x+1) folders:

- Folders "**indCellsMVRG1_**", "**indCellsMVRG2_**", etc. contain visualizations of individual analysis results.

- Folder "**compareSigTtest**" contains comparisons between 2 analysis results, or folder "**compareSigAnova**" contains comparisons of more than 2 analysis results.

5.4 INSTRUCTIONS FOR RUNNING ANALYSIS OF BIPHASIC RELATIONSHIP IN INTEGRATED SMI-FSM DATA

See each function's documentation for detailed instructions on different options and formats for inputs and outputs. In brief, the functions in this section use the following core inputs:

- "**`tassmDblClnSpeed`**" is the output of function "`smactinClean`" from section ***5.2.c. Integrating and cleaning up of SM tracklet and associated local speckle data***.
- "**`smactinFlag`**" is the variable described in section ***5.1. Preparing inputs for SMI-FSM analysis***.
- "**`scanInfo`**" describes the values of the thresholds that will split the data from "`tassmDblClnSpeed`" into 2 subsets (above and below each threshold).
- "**`motionType`**" indicates the SM diffusion type (i.e., free diffusion, etc.) of interest.
- "**`figureInfo`**" and "**`figTitleString`**" describes how the output figures will be formatted and saved.
- "**`propStruct`**" and "**`propVect`**" indicates the properties of interest on which to perform analysis and visualization.

In the Command Window in MATLAB, run:

```
thresholdsTable = scan4smSpeckleThreshold(tassmDblClnSpeed, [], smactinFlag,
scanInfo, motionType, figureInfo, propStruct)
```

If the user chooses to generate the three output figures demonstrating the model-fitting assessment at each scanning threshold, it is recommended that the function `scan2smSpeckleThreshold` is run in a new, empty folder, because it will automatically save the figures in the current folder.

To visualize the biphasic relationship of the properties of interest per individual cell (e.g., if `cellNo = 1`, function will visualize the first cell of the dataset), user can run the following in the MATLAB Command Window to generate plots of the fit lines on top of the data scatterplot:

```
plotBiphasicLines(smifsmResults.testResultAndSplit{1, 1}, motionType, propVect,
thresholdsTable, cellNo, figTitleString);
```

The input "propVect" can be derived from the input "propStruct" as follow:

```
propVect = [propStruct{1}, propStruct{2} + propStruct{3}(1)];
```

6 RECOMMENDED FOLDER STRUCTURE

The following folder structured is required if input `utrackPaths` is extracted with Method 2 (through function `getUtrackPackPaths.m`) (see section 5.1). Folders that contain the results of u-track and QFSM software must be labelled starting with "m" (e.g., m-01-01, m-02-01, etc.) and must be identical to each other in any corresponding SMI and FSM movie pair. In the following example, the "SingleMolecule" and "Speckle" folders are in the user's local drive named "Z".

```
Z:.
├───SingleMolecule
│   ├───m-01-01
│   │   └───TrackingPackage
│   │       └───tracks
│   ├───m-01-02
│   │   └───TrackingPackage
│   │       └───tracks
│   └───m-01-03
│       ├───backups
│       ├───ImportedCellMask
│       ├───TrackingPackage
│       │   ├───GaussianMixtureModels
│       │   │   └───detections_for_channel_1
│       │   ├───MotionAnalysis
│       │   └───tracks
│       └───TransientDiffusionAnalysis
└───Speckle
    ├───m-01-01
    │   └───QFSMPackage
    │       ├───kineticAnalysis
    │       ├───refined_masks
    │       │   └───refined_masks_for_channel_1
    │       ├───speckles
    │       └───speckleTracks
    ├───m-01-02
    │   ├───External_ROI
    │   └───QFSMPackage
    │       ├───kineticAnalysis
    │       ├───speckles
    │       └───speckleTracks
    └───m-01-03
        ├───backups
        ├───External_ROI
        └───QFSMPackage
            ├───kineticAnalysis
            ├───masks
            │   └───masks_for_channel_1
            ├───noise
            │   └───_cropped_images
            ├───refined_masks
            │   └───refined_masks_for_channel_1
            ├───speckles
            └───speckleTracks
```

7 INSTRUCTIONS FOR SETTING UP MANUAL ("EXTERNAL") ROI

This section describes how to set up manual "external" ROI through the Windowing Protrusion software (8) if needed. Follow the 3 steps below.

Step 1: Download the Windowing Protrusion software package (8) from the following link:

- https://github.com/DanuserLab/Windowing-Protrusion

Step 2: Draw cell mask ROI using the actin movies from within the Movie Selection window:

In MATLAB Command Window, run:

```
movieSelectorGUI
```

This will bring up a window called "Movie selection". Select "Open" to choose your movie of interest. In the example movie, user can try opening "**m-01-03**". Click "Yes to all" when asked about relocating the channels.

After loading your movie list, select "Tools" > "Add region of interest" in the "Movie selection" window. This will pop up the "Region of interest" window. Select "Draw new region of interest". This will pop up your movie in a new window called "Select the region of interest". Draw your ROI and click "Save". This ROI will be used for all frames.

In your Speckle folder, you will see the binary mask named "roiMask.tif". For an actin FSM movie with N frames, create a new "External_ROI" folder and copy this mask inside it N times. Change the name of these masks to the following format: "roiMask_nn.tif" where nn is frame 01 to N (single digit numbers must be preceded by 0. For example, for first frame nn=01, etc.).

For example, the provided "m-01-03.tif" movie has 11 frames. In the provided "External_ROI" folder, you will see 11 .tif files named as followed:
"roiMask_01.tif"
"roiMask_02.tif"
…
"roiMask_11.tif"

Step 3: Use Windowing Protrusion package (8) to convert the manual masks to a format understandable by QFSM package.
In brief, in the Movie selection window, select “Segmentation” analysis package and continue through the two steps for making and refining binary masks.

After these 3 steps, proceed with the QFSM package as normal (as if the masks were created automatically in the QFSM package).

8 REFERENCED WORKS

1. Dasgupta, A., H.-T. Ngo, D. Tschoerner, N. Touret, B. d. Rocha-Azevedo, and K. Jaqaman. 2023. Multiscale imaging and quantitative analysis of plasma membrane protein-cortical actin interplay. *bioRxiv*.2023.2001.2022.525112, doi: 10.1101/2023.01.22.525112, https://www.biorxiv.org/content/biorxiv/early/2023/01/23/2023.01.22.525112.full.pdf.
2. Jaqaman, K., D. Loerke, M. Mettlen, H. Kuwata, S. Grinstein, S. L. Schmid, and G. Danuser. 2008. Robust single-particle tracking in live-cell time-lapse sequences. *Nat Methods*. 5(8):695-702, doi: 10.1038/nmeth.1237, https://www.ncbi.nlm.nih.gov/pubmed/18641657.
3. Mendoza, M. C., S. Besson, and G. Danuser. 2012. Quantitative fluorescent speckle microscopy (QFSM) to measure actin dynamics. *Curr Protoc Cytom*. Chapter 2:Unit2 18, doi: 10.1002/0471142956.cy0218s62, https://www.ncbi.nlm.nih.gov/pubmed/23042526.
4. de Oliveira, L. R., and K. Jaqaman. 2019. FISIK: Framework for the Inference of In Situ Interaction Kinetics from Single-Molecule Imaging Data. *Biophys J*. 117(6):1012-1028, doi: 10.1016/j.bpj.2019.07.050, https://www.ncbi.nlm.nih.gov/pubmed/31443908.
5. Vega, A. R., S. A. Freeman, S. Grinstein, and K. Jaqaman. 2018. Multistep Track Segmentation and Motion Classification for Transient Mobility Analysis. *Biophys J*. 114(5):1018-1025, doi: 10.1016/j.bpj.2018.01.012, https://www.ncbi.nlm.nih.gov/pubmed/29539390.
6. Ponti, A., P. Vallotton, W. C. Salmon, C. M. Waterman-Storer, and G. Danuser. 2003. Computational analysis of F-actin turnover in cortical actin meshworks using fluorescent speckle microscopy. *Biophys J*. 84(5):3336-3352, doi: 10.1016/S0006-3495(03)70058-7, https://www.ncbi.nlm.nih.gov/pubmed/12719263.
7. Lee, G., G. Leech, P. Lwin, J. Michel, C. Currie, M. J. Rust, J. L. Ross, R. J. McGorty, M. Das, and R. M. Robertson-Anderson. 2021. Active cytoskeletal composites display emergent tunable contractility and restructuring. *Soft Matter*. 17(47):10765-10776, doi: 10.1039/d1sm01083b, https://www.ncbi.nlm.nih.gov/pubmed/34792082.
8. Lee, K., H. L. Elliott, Y. Oak, C. T. Zee, A. Groisman, J. D. Tytell, and G. Danuser. 2015. Functional hierarchy of redundant actin assembly factors revealed by fine-grained registration of intrinsic image fluctuations. *Cell Syst*. 1(1):37-50, doi: 10.1016/j.cels.2015.07.001, https://www.ncbi.nlm.nih.gov/pubmed/26273703.