Below are the main steps of FISIK and associated functions, as described in:

(1) de Oliveira and Jaqaman, Biophys. J. 2019. FISIK: Framework for the Inference of in Situ Interaction Kinetics from single-molecule imaging data.
https://pubmed.ncbi.nlm.nih.gov/31443908/.

(2) Guerrero et al., bioRXiv 2025. Inference of VEGFR2 dimerization kinetics on the cell surface by integrating single-molecule imaging and mathematical modeling.
https://doi.org/10.1101/2025.06.03.657760.

Software installation

The software has been tested on **Matlab R2018b** on **Linux 64-bit OS**. It includes a mex-file (maxWeightedMatching) that might need recompilation for other Operating Systems.

The software requires the following **Matlab toolboxes**:

- Statistics and Machine Learning Toolbox
- Control System Toolbox
- Robust Control Toolbox
- Optimization Toolbox
- Signal Processing Toolbox
- Symbolic Math Toolbox
- Image Processing Toolbox
- Datafeed Toolbox
- Bioinformatics Toolbox
- Computer Vision System Toolbox

The download package contains all functions (but not the Matlab toolboxes) necessary to run FISIK, except for those in the u-track package (for single particle tracking) and in the YALMIP package (for estimating the particle oligomerization state from its intensity and merging and splitting history). Please download those two packages separately:

- u-track: https://github.com/DanuserLab/u-track.
- YALMIP: https://yalmip.github.io/.

The descriptions below explain briefly the function(s) for each step of FISIK. For detailed information on the input and output arguments of these functions, please refer to the help section of each function.

FISIK steps and functions:

1. **Kinetic Monte Carlo simulations:**

    a. To generate the trajectories and interactions of molecules freely diffusing in 2D, where all molecules have the same diffusion coefficient (BJ 2019): **receptorAggregationSimpleOptimized.m**.

b. To generate the trajectories and interactions of molecules freely diffusing in 2D, where each molecule has a different diffusion coefficient drawn from a normal distribution (BJ 2019):
receptorAggregationVaryDiffCoef.m.

c. To generate the trajectories and interactions of molecules freely diffusing in 2D, where each molecule has a diffusion coefficient drawn from an arbitrary distribution, e.g. derived empirically from experimental data (as in the case of VEGFR2; bioRXiv 2025):
receptorAggregationVEGFR2ModeDistCorrFixedID.m

See supplied file "empirical distributions.mat" for example empirical distributions. These distributions were derived empirically from VEGFR2 (full length and truncation mutant ECTM) single-molecule imaging data, as used in the bioRXiv 2025 manuscript listed above.

d. To "label" a fraction of the simulated molecules and output their trajectories and interactions:

For trajectories generated from (a) or (b) above (BJ 2019):

i. Without photobleaching:
genReceptorInfoLabeled.m.

ii. With photobleaching:
genReceptorInfoLabeledPhotobleaching.m.

For trajectories generated from (c) above (bioRXiv 2025):
genReceptorInfoLabeledFixedID.m.

All functions take as input the output "receptorInfoAll" from the simulation functions listed in (a-c) above. Note that genReceptorInfoLabeled and genReceptorInfoLabeledFixedID are called from within their corresponding simulation functions as well.

2. Synthetic single-molecule movie generation (if desired):

Use the function:
getImageStackFromTracksDirect_New.m

This function takes as input the compound tracks output of the simulations, i.e., receptorInfoLabeled.compTracks (Step 1 above).

3. Refinement of u-track output (if desired):

To eliminate merges and splits that are most likely tracking artifacts (related to detection false positives and/or particles passing by each other), use the function:
refineTracksWithinMask.m

This function takes as input the tracks as output by u-track and then outputs tracks in the same format but after (1) removing simultaneous merges and splits, (2) removing merge-to-split events that are too short, and (3) removing split-to-merge events, split-to-end events, and start-to-merge events that are too short.

4. Oligomeric state estimation:

The procedure for estimating the labeled oligomeric state depends on the type of data (dynamic or static):

a. Dynamic trajectory data:

   i. Simulations where fluorophore intensity does not fluctuate (i.e. intensity ~ $N(1,0)$):
   aggregStateFromCompIntensity.m

   ii. Simulations where fluorophore intensity fluctuates over time:
   aggregStateFromCompTracksMIQP.m

   Both functions take as input the compound tracks of the labeled molecules. For tracks as directly output by the simulations (Step 1 above), the compound tracks are in the field "compTracks" inside the simulation output "receptorInfoLabeled". For tracks produced by u-track, the compound tracks are in the directory specified by the u-track package, potentially after track refinement (if performed).

b. Static snapshot data:

   Here the oligomer distribution data are derived from the last time point of the simulation using
   genAggregStateFromReceptorLabeledSuperRes.m.

   This function takes as input the output "receptorInfoAll" from the simulations (Step 1 above).

5. Oligomeric history analysis:

This applies only to dynamic data. It determines the start time and end time of each oligomeric state and the events (merge/association or split/dissociation) resulting in the start and end of the oligomeric state.

clusterHistoryFromCompTracks_aggregState_motion.m

This function takes as input the outputs "compTracksOut.defaultFormatTracks" and "compTracksOut.alternativeFormatTracks" from the oligomeric state estimation function (Step 4a above).

6. **Summary statistics calculation:**

   The calculated summary statistics depend on the type of data (dynamic or static):

   a. Dynamic trajectory data:

      To calculate the labeled oligomer densities, off rates and on rates (note that on rates are calculated for completeness; they are not used as summary statistics, as explained in manuscripts):
      clusterOnOffRatesDensityFromClustHistAndAggregState.m

      This function takes as input the output "clustHistoryMerged" from the oligomeric state history function (Step 5 above) and the variable "aggregStateMat," which is obtained by running the function **convStruct2MatIgnoreMS** on the output "compTracksOut.defaultFormatTracks" from the oligomeric state estimation function (Step 4a above).

   b. Static snapshot data:

      To calculate the labeled oligomer densities:
      clusterDensityStatic.m

      This function takes as input the output of the static data oligomeric state estimation function (Step 4b above).

7. **Mahalanobis distance calculation:**

   Given a set of summary statistics for the probe and another set for the target, the summary statistics are compared using the Mahalanobis distance as follows:

   a. If the summary statistics are to be processed and then compared (BJ 2019):
      calculationMahalonobisDistanceandPvalueDynamicStatic.m
      This can handle static and dynamic data.

   b. If the summary statistics are to be compared as is (bioRXiv 2025):
      calculationMahalonobisDistanceandPvalue.m
      This can handle dynamic data only.

   These functions take as input a matrix of probe summary statistics and a matrix of target summary statistics, where number of columns = number of repeats (whether simulations or experiments). See function help for full details. They then internally calculate the summary statistics' mean and covariance matrix for Mahalanobis distance calculation.

Additional analysis functions outside of the main FISIK pipeline:

A. calcLabeledParticlesVsTotal.m

Analytical mapping from labeled fraction of receptors and oligomeric state fractions in full population to labeled fraction of particles and apparent oligomeric state fractions in labeled subset of complexes.

B. getPhotobleachStepSize.m

Fitting of step functions into intensity traces and estimating fluorophore intensity properties from the fit (steps and intensity values in between).