rsnaped

Release 0.1.0

unknown

Apr 21, 2023

CONTENTS

rSNAPed: A software for single-molecule image tracking, simulation and parameter estimation. Created on Fri Jun 26 22:10:24 2020 Authors: Luis U. Aguilera, William Raymond, Brooke Silagy, Brian Munsky.

class rsnaped.rsnaped.**AugmentationVideo**(*video: ndarray*, *predefined_angle=None*)

This class is intended to be used for data augmentation. It takes a video and perform random rotations in the X and Y axis.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

predefined_angle
[int, with values 0, 90, 180, 270, 360, or None. Optional, None is the default.] Indicates the specific angle of rotation.

random_rotation()

Method that performs random rotations of a video in the Y and X axis.

Returns

video_random_rotation
[np.uint16] Array with dimensions [T, Y, X, C].

class rsnaped.rsnaped.**BandpassFilter**(*video: ndarray*, *low_pass: float = 0.5*, *high_pass: float = 10*)

This class is intended to apply high and low bandpass filters to the video. The format of the video must be [T, Y, X, C]. This class uses **difference_of_gaussians** from skimage.filters.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

low_pass
[float, optional] Lower pass filter intensity. The default is 0.5.

high_pass
[float, optional] Higher pass filter intensity. The default is 10.

apply_filter()

This method applies high and low bandpass filters to the video.

Returns

video_filtered
[np.uint16] Filtered video resulting from the bandpass process. Array with format [T, Y, X, C].

class rsnaped.rsnaped.**BeadsAlignment**(*first_image_beads: ndarray*, *second_image_beads: ndarray*, *spot_size: int = 5*, *min_intensity: float = 100*, *show_plot=True*)

This class is intended to detected and align spots detected in the various channels of an image with dimensions [C, Y, X]. The class returns a homography matrix that can be used to align the images captured from two different cameras during the experiment. Notice that this class should only be used for images taken from a microscope setup that uses two cameras to image various channels.

Parameters

first_image_beads
[NumPy array] Array with a simple image with dimensions [Y, X].

second_image_beads
[NumPy array] Array with a simple image with dimensions [Y, X].

spot_size
[int, optional] Average size of the beads, The default is 5.

min_intensity
[float, optional] Minimal expected intensity for the beads. The default is 400.

show_plot
[Bool, optional] Shows a plot with the detected beads in the image. The default is True.

`make_beads_alignment()`

This method aligns a list of spots detected in an image with dimensions [C, Y, X] and returns a homography matrix.

Returns

homography_matrix
[object] The homography matrix is a 3x3 matrix. This transformation matrix maps the points between two planes (images).

class rsnaped.rsnaped.**CamerasAlignment**(*video: ndarray*, *homography_matrix*, *target_channels: list = [1]*)

This class is intended to align the images from multiple channels by applying a matrix transformation using the homography

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

homography_matrix
[object] The homography matrix object generated with the class BeadsAlignment.

target_channels
[List of int, optional] Lower bound to normalize intensity. The default is [1].

`make_video_alignment()`

This method transforms the video by multiplying the target channels by the homography matrix.

Returns

transformed_video
[np.uint16] Transformed video. Array with dimensions [T, Y, X, C].

class rsnaped.rsnaped.**Cellpose**(*video: ndarray*, *num_iterations: int = 5*, *channels: list = [0, 0]*, *diameter: float = 120*, *model_type: str = 'cyto'*, *selection_method: str = 'max_cells_and_area'*)

This class is intended to detect cells by image masking using **Cellpose** . The class uses optimization to maximize the number of cells or maximize the size of the detected cells.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

num_iterations
[int, optional] Number of iterations for the optimization process. The default is 5.

channels
[List, optional] List with the channels in the image. For gray images use [0, 0], for RGB images with intensity for cytosol and nuclei use [0, 1] . The default is [0, 0].

diameter
[float, optional] Average cell size. The default is 120.

model_type
[str, optional] To detect between the two options: 'cyto' or 'nuclei'. The default is 'cyto'.

selection_method
[str, optional] Option to use the optimization algorithm to maximize the number of cells or maximize the size options are 'max_area' or 'max_cells' or 'max_cells_and_area'. The default is 'max_cells_and_area'.

`calculate_masks()`

This method performs the process of image masking using **Cellpose**.

Returns

selected_masks
[List of NumPy arrays] List of NumPy arrays with values between 0 and the number of detected cells in the image, where a number larger than zero represents the masked area for each cell, and 0 represents the area where no cells are detected.

`class rsnaped.rsnaped.CellposeSelection(`*mask: ndarray*, *video: ndarray*, *selection_method: str = 'max_area'*, *particle_size: int = 5*, *selected_channel: int = 0*`)`

This class is intended to select the cell with the maximum area (max_area) or with the maximum number of spots (max_spots) from a masked image previously detect cells by **Cellpose**

Parameters

mask
[NumPy array] Arrays with values between 0 and the number of detected cells in the image, where a number larger than zero represents the masked area for each cell, and 0 represents the area where no cells are detected. An array of images with dimensions [Y, X].

video
[NumPy array] An array of images with dimensions [T, Y, X, C].

selection_method
[str, optional] Options used by the optimization algorithm to select a cell based on the number of cells or the number of spots. The options are: 'all_cells_in_image','max_area' or 'max_spots'. The default is 'maximum_area'.

particle_size
[int, optional] Average particle size. The default is 5.

selected_channel
[int, optional] Channel where the particles are detected and tracked. The default is 0.

`select_mask()`

This method selects the cell with the maximum area (max_area) or with the maximum number of spots (max_spots) from a masked image.

Returns

selected_mask
[NumPy array, with Boolean values, where 1 represents the masked area, and 0 represents the area outside the mask.] An array of images with dimensions [Y, X].

class rsnaped.rsnaped.**ConvertToStandardFormat**(*video: ndarray*, *time_position: int = 0*, *height_position: int = 1*, *width_position: int = 2*, *channel_position: int = 3*)

This class contains two methods to:

1. Transform any numpy array of images into the format [T, Y, X, C].
2. Convert an image into an array video with a single time point (this last is necessary for compatibility).

Parameters

video
[NumPy array] Array of images. This class accepts arrays with formats: [Y, X], [T, Y, X], [T, Y, X, C], or any other permutation of channels, the user must specify the position of each dimension in the original video by defining the parameters: time_position, height_position, width_position, channel_position.

time_position
[int, optional] Position for the dimension for the time in the original video array. The default is 0.

height_position
[int, optional] Position for the dimension for the y-axis (height) in the original video array. The default is 1.

width_position
[int, optional] Position for the dimension for the x-axis (width) in the original video array. The default is 2.

channel_position
[int, optional] Position for the channel's dimension in the original video array. The default is 3.

image_to_video()

Method that converts an image into a video with one frame. This process is done for compatibility with the rest of the classes.

Returns

video_correct_order
[np.uint16] Array with dimensions [T, Y, X, C].

transpose_video()

Method that transposes an unsorted video to the standard [T, Y, X, C]

Returns

video_correct_order
[np.uint16] Array with dimensions [T, Y, X, C].

class rsnaped.rsnaped.**Covariance**(*intensity_array=None*, *dataframe_particles=None*, *selected_field='ch1_int_mean'*, *max_lagtime=100*, *show_plot=True*, *figure_size=(6, 4)*, *step_size=1*)

This class calculates the auto-covariance from an intensity time series.

Parameters

dataframe_particles
[pandas dataframe] Dataframe with fields [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR, ch1_SNR, ch2_SNR].

selected_field
[str, optinal] Field in the datafre to be used to calculate autocovariance. The deafault is 'ch1_int_mean'

max_lagtime
[int, optional] Is the time interval used to compute the mean square displacement. Trackpy uses a default of 100.

show_plot
[bool, optional] If True, it displays a plot of Covariance vs time. The default is True.

step_size
[int, optional.] Frame rate in seconds. The default is 1 frame per second.

`calculate_autocovariance()`

Method that runs the simulations for the multiplexing experiment.

Returns

mean_acf_data
[NumPy array] 2D Array with dimensions [particle, Time]

err_acf_data
[NumPy array] 2D Array with dimensions [particle, Time]

lags
[NumPy array] 1D Array with dimensions [Time]

decorrelation_time
[float] Float indicating the decorrelation time.

auto_correlation_matrix
[Numpy array] 2D Array with dimenssions [particle, lags].

`class rsnaped.rsnaped.GaussianFilter(`*video: ndarray*, *sigma: float = 1*`)`

This class is intended to apply high and low bandpass filters to the video. The format of the video must be [T, Y, X, C]. This class uses **difference_of_gaussians** from skimage.filters.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

low_pass
[float, optional] Lower pass filter intensity. The default is 0.5.

high_pass
[float, optional] Higher pass filter intensity. The default is 10.

`apply_filter()`

This method applies high and low bandpass filters to the video.

Returns

video_filtered
[np.uint16] Filtered video resulting from the bandpass process. Array with format [T, Y, X, C].

`class rsnaped.rsnaped.GaussianLaplaceFilter(`*video: ndarray*, *sigma: float = 1*`)`

This class is intended to apply high and low bandpass filters to the video. The format of the video must be [T, Y, X, C]. This class uses **difference_of_gaussians** from skimage.filters.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

low_pass
[float, optional] Lower pass filter intensity. The default is 0.5.

high_pass
[float, optional] Higher pass filter intensity. The default is 10.

`apply_filter()`

This method applies high and low bandpass filters to the video.

Returns

video_filtered
[np.uint16] Filtered video resulting from the bandpass process. Array with format [T, Y, X, C].

`class rsnaped.rsnaped.Intensity(`*video: ndarray*, *particle_size: int = 5*, *trackpy_dataframe: Optional[object] = None*, *spot_positions_movement: Optional[ndarray] = None*, *dataframe_format: str = 'short'*, *method: str = 'disk_donut'*, *step_size: float = 1*, *show_plot: bool = True*`)`

This class is intended to calculate the intensity in the detected spots.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

particle_size
[int, optional] Average particle size. The default is 5.

trackpy_dataframe
[pandas data frame or None (if not given).] Pandas data frame from trackpy with fields [x, y, mass, size, ecc, signal, raw_mass, ep, frame, particle]. The default is None

spot_positions_movement
[NumPy array or None (if not given).] Array of images with dimensions [T, S, y_x_positions]. The default is None

dataframe_format
[str, optional] Format for the dataframe the options are : 'short' , and 'long'. The default is 'short'. "long" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR]. "short" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, x, y].

method
[str, optional] Method to calculate intensity the options are : 'total_intensity' , 'disk_donut' and 'gaussian_fit'. The default is 'disk_donut'.

step_size
[float, optional] Frame rate in seconds. The default is 1 frame per second.

show_plot
[bool, optional] Allows the user to show a plot for the optimization process. The default is True.

`calculate_intensity()`

This method calculates the spot intensity.

Returns

dataframe_particles
[pandas dataframe] Dataframe with fields [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR].

array_intensities_mean
[Numpy array] Array with dimensions [S, T, C].

time_vector
[Numpy array] 1D array.

mean_intensities: Numpy array
Array with dimensions [S, T, C].

std_intensities
[Numpy array] Array with dimensions [S, T, C].

mean_intensities_normalized
[Numpy array] Array with dimensions [S, T, C].

std_intensities_normalized
[Numpy array] Array with dimensions [S, T, C].

`class` rsnaped.rsnaped.**MaskingImage**(*video: ndarray*, *mask: ndarray*)

This class is intended to apply a mask to the video. The video format must be [T, Y, X, C], and the format of the mask must be [Y, X].

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

mask
[NumPy array, with Boolean values, where 1 represents the masked area, and 0 represents the area outside the mask.] An array of images with dimensions [Y, X].

`apply_mask()`

This method applies high and low bandpass filters to the video. The method uses **difference_of_gaussians** from skimage.filters.

Returns

video_removed_mask
[np.uint16] Video with zero values outside the area delimited by the mask. Array with format [T, Y, X, C].

`class` rsnaped.rsnaped.**MergeChannels**(*directory: str*, *substring_to_detect_in_file_name: str = '.*_C0.tif'*, *save_figure: bool = False*)

This class takes images as arrays with format [Z,Y,X] and merge then in a numpy array with format [Z, Y, X, C]. It recursively merges the channels in a new dimension in the array. Minimal number of Channels 2 maximum is 4

Parameters

directory: str or PosixPath
Directory containing the images to merge.

substring_to_detect_in_file_name: str
String with the prefix to detect in the files names.

save_figure: bool, optional
Flag to save the merged images as .tif. The default is False.

`merge()`

Method takes all the videos in the folder and merge those with similar names.

Returns

list_file_names
[List of strings] List with strings of names.

list_merged_images
[List of NumPy arrays.] List of NumPy arrays with format np.uint16 and dimensions [Z, Y, X, C].

number_files
[int.] Number of merged images in the folder.

class rsnaped.rsnaped.**MetadataImageProcessing**(*metadata_filename_ip*, *list_video_paths*, *files_dir_path_processing*, *particle_size*, *selected_channel_tracking*, *selected_channel_segmentation*, *intensity_calculation_method*, *mask_selection_method*, *use_optimization_for_tracking*, *average_cell_diameter*, *min_percentage_time_tracking*, *dataframe_format*, *list_segmentation_succesful*, *list_frames_videos*)

This class is intended to generate a metadata file containing used dependencies, user information, and parameters used to process single molecule gene expression experiments.

Parameters

The parameters for this class are defined in the SimultedCell class

write_metadata()

This method writes the metadata file.

class rsnaped.rsnaped.**MetadataSimulatedCell**(*metadata_filename*, *video_dir*, *masks_dir*, *list_gene_sequences*, *list_number_spots*, *list_target_channels_proteins*, *list_target_channels_mRNA*, *list_diffusion_coefficients*, *list_label_names*, *list_elongation_rates*, *list_initiation_rates*, *number_cells=1*, *simulation_time_in_sec=100*, *step_size_in_sec=1*, *frame_selection_empty_video='gaussian'*, *spot_size=7*, *spot_sigma=1*, *intensity_scale_ch0=None*, *intensity_scale_ch1=None*, *intensity_scale_ch2=None*, *simulated_RNA_intensities_method='constant'*, *basal_intensity_in_background_video=20000*, *list_files_names_outputs=[]*)

This class is intended to generate a metadata file containing used dependencies, user information, and parameters used to generate the simulated cell.

Parameters

The parameters for this class are defined in the SimultedCell class

write_metadata()

This method writes the metadata file.

class rsnaped.rsnaped.**ParticleMotion**(*trackpy_dataframe*, *microns_per_pixel*, *step_size_in_sec*, *max_lagtime=100*, *show_plot=True*, *remove_drift=False*)

This class is intended to calculate the mean square displacement in the detected spots. This class uses trackpy.motion.emsd.

Parameters

trackpy_dataframe
[pandas data frame or None (if not given).] Pandas data frame from trackpy with fields [x, y, mass, size, ecc, signal, raw_mass, ep, frame, particle]. The default is None

microns_per_pixel
[float] Factors to converts microns per pixel.

step_size_in_sec
[float] Factor that sets the frames per second in the video.

max_lagtime
[int, optional] Is the time interval used to compute the mean square displacement. Trackpy uses a default of 100.

show_plot
[bool, optional] If True, it displays a plot of MSD vs time. The default is True.

remove_drift
[bool, optional] This flags removes the drift in the dataframe.

`calculate_msd()`

This method calculates the MSD for all the spots in the dataframe.

Returns calculated_diffusion_coefficient : float

Calculated mean square displacement for the particle ensemble.

MSD_series
[pandas dataframe] Dataframe with fields [msd, time].

`class rsnaped.rsnaped.PhotobleachingCalculation(`*video: ndarray*, *mask: ndarray*, *step_size: int = 1*, *selected_channel: int = 1*, *show_plot: bool = 1*`)`

This class is intended to calculate the intensity in the detected spots.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

particle_size
[int, optional] Average particle size. The default is 5.

trackpy_dataframe
[pandas data frame or None (if not given).] Pandas data frame from trackpy with fields [x, y, mass, size, ecc, signal, raw_mass, ep, frame, particle]. The default is None

spot_positions_movement
[NumPy array or None (if not given).] Array of images with dimensions [T, S, x_y_positions]. The default is None

method
[str, optional] Method to calculate intensity the options are : 'total_intensity' , 'disk_donut' and 'gaussian_fit'. The default is 'disk_donut'.

step_size
[float, optional] Frame rate in seconds. The default is 1 frame per second.

show_plot
[bool, optional] Allows the user to show a plot for the optimization process. The default is 1.

class rsnaped.rsnaped.**PipelineTracking**(*video: ndarray*, *mask: Optional[ndarray] = None*, *particle_size: int = 5*, *file_name: str = 'Cell.tif'*, *selected_channel_tracking: int = 0*, *selected_channel_segmentation: int = 0*, *intensity_calculation_method: str = 'disk_donut'*, *mask_selection_method: str = 'max_spots'*, *show_plot: bool = 1*, *use_optimization_for_tracking: bool = 1*, *real_positions_dataframe=None*, *average_cell_diameter: float = 120*, *print_process_times: bool = 0*, *min_percentage_time_tracking=0.4*, *intensity_threshold_tracking=None*, *dataframe_format='short'*, *create_pdf=True*, *path_temporal_results=None*, *image_index: int = 0*)

A pipeline that allows cell segmentation, spot detection, and tracking of spots.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

mask
[NumPy array, optional] Array of images with dimensions [Y, X]. The default is None, and it will perform segmentation using Cellpose.

particle_size
[int, optional] Average particle size. The default is 5.

file_name
[str, optional] The file name for the simulated cell. The default is 'Cell.tif'.

selected_channel_tracking
[int, optional] Integer indicating the channel used for particle tracking. The default is 0.

selected_channel_segmentation: int, optional
Integer indicating the channel used for segmenting the cytosol. The default is 0.

intensity_calculation_method
[str, optional] Method to calculate intensity the options are : 'total_intensity' , 'disk_donut' and 'gaussian_fit'. The default is 'disk_donut'..

mask_selection_method
[str, optional] Option to use the optimization algorithm to maximize the number of cells or maximize the size. The options are 'max_area' or 'max_cells'. The default is 'max_area'.

show_plot
[bool, optional] Allows the user to show a plot for the optimization process. The default is 1.

use_optimization_for_tracking
[bool, optional] Option to run an optimization process to select the best filter for the tracking process. The default is 0.

real_positions_dataframe
[Pandas dataframe.] A pandas data frame containing the position of each spot in the image. This dataframe is generated with class SimulatedCell, and it contains the true position for each spot. This option is only intended to be used to train algorithms for tracking and visualize real vs detected spots. The default is None.

average_cell_diameter
[float, optional] Average cell size. The default is 120.

print_process_times
[bool, optional] Allows the user the times taken during each process. The default is 0.

intensity_selection_threshold_int_std
[float, optional. The default is None, and it uses a default value or an optimization method if use_optimization_for_tracking is set to TRUE.] Threshold intensity for tracking

min_percentage_time_tracking = float, optional
Value that indicates the minimal (normalized) percentage of time to consider a particle as a detected trajectory during the tracking process. The number must be between 0 and 1. The default is 0.3.

intensity_threshold_tracking
[float or None, optional.] Intensity threshold value to be used during tracking. If no value is passed, the code attempts to calculate this value. The default is None.

dataframe_format
[str, optional] Format for the dataframe the options are : 'short' , and 'long'. The default is 'short'. "long" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR]. "short" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, x, y].

create_pdf
[bool, optional] Flag to indicate if a pdf report is needed. The default is True.

path_temporal_results
[path or str, optional.] Path used to store results to create the PDF. The default is None.

image_index
[int, optional] Index indicating a counter for the image. The default is 0.

`run()`

Runs the pipeline.

Returns

dataframe_particles
[pandas dataframe] Dataframe with fields [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y].

selected_mask
[Numpy array] Array with the selected mask. Where zeros represents the background and one represent the area in the cell.

array_intensities
[Numpy array] Array with dimensions [S, T, C].

time_vector
[Numpy array] 1D array.

mean_intensities: Numpy array
Array with dimensions [S, T, C].

std_intensities
[Numpy array] Array with dimensions [S, T, C].

mean_intensities_normalized
[Numpy array] Array with dimensions [S, T, C].

std_intensities_normalized
[Numpy array] Array with dimensions [S, T, C].

`class rsnaped.rsnaped.Plots`

This class contains miscellaneous plots that are constantly used during the generation of the simulated data.

`plot_image_channels`(*selected_time_point=0*)

This function plots all the channels for the original image.

`class` rsnaped.rsnaped.`ReadImages`(*directory: str*)

This class reads all .tif images in a given folder and returns the names of these files, path, and number of files.

Parameters

directory: str or PosixPath
Directory containing the images to merge.

`read`()

Method takes all the videos in the folder and merge those with similar names.

Returns

list_images
[List of NumPy arrays.] List of NumPy arrays with format np.uint16 and dimensions [Z, Y, X, C] or [T, Y, X, C] .

list_file_names
[List of strings] List with strings of names.

number_files
[int.] Number of images in the folder.

`class` rsnaped.rsnaped.`RemoveExtrema`(*video: ndarray*, *min_percentile: float = 1*, *max_percentile: float = 99*, *selected_channels: Optional[list] = None*)

This class is intended to remove extreme values from a video. The format of the video must be [Y, X] , [Y, X, C] , [Z, Y, X, C] or [T, Y, X, C].

Parameters

video
[NumPy array] Array of images with dimensions [Y, X] , [Y, X, C] , [Z, Y, X, C] or [T, Y, X, C].

min_percentile
[float, optional] Lower bound to normalize intensity. The default is 1.

max_percentile
[float, optional] Higher bound to normalize intensity. The default is 99.

selected_channels
[List or None, optional] Use this option to select a list channels to remove extrema. The default is None and applies the removal of extrema to all the channels.

`remove_outliers`()

This method normalizes the values of a video by removing extreme values.

Returns

normalized_video
[np.uint16] Normalized video. Array with dimensions [T, Y, X, C] or image with format [Y, X].

`class` rsnaped.rsnaped.`ReportPDF`(*directory*, *pdf_report_name*, *list_file_names*, *list_segmentation_succesful*)

This class intended to create a PDF report including the images generated during the pipeline for segmentation and tracking.

Parameters

directory_results: str or PosixPath
Directory containing the images to include in the report.

pdf_report_name
[str] Name of the pdf repot.

list_segmentation_succesful
[list] List indicating if the segmentation was sucessful for the image.

list_file_names
[list,] List with the file names.

This PDF file is generated, and it contains the processing steps for each image in the folder.

`create_report()`

This method creates a PDF with the original images, images for cell segmentation and images for the spot detection.

`class rsnaped.rsnaped.SSA_rsnapsim(`*gene_file*, *ke=10*, *ki=0.03*, *frames=300*, *frame_rate=1*, *n_traj=20*, *t_burnin=1000*, *use_Harringtonin=False*, *use_FRAP=False*, *perturbation_time_start=0*, *perturbation_time_stop=None*`)`

This class uses rsnapsim to simulate the single-molecule translation dynamics of any gene.

Parameters

gene_file
[str,] Path to the location of a FASTA file.

ke
[float, optional.] Elongation rate. The default is 10.0.

ki: float, optional.
Initiation rate. The default is 0.03.

frames: int, optional.
Total number of simulation frames in seconds. The default is 300.

n_traj: int, optional.
Number of trajectories to simulate. The default is 20.

frame_rate
[int, optional.] Frame rate per second. The default is 1.

t_burnin
[int , optional] time of burnin. The default is 1000

use_Harringtonin: bool, optional
Flag to specify if Harringtonin is used in the experiment. The default is False.

use_FRAP: bool
Flag to specify if FRAP is used in the experiment. The default is False.

perturbation_time_start: int, optional.
Time to start the inhibition. The default is 0.

perturbation_time_stop
[int, opt.] Time to start the inhibition. The default is None.

Outputs:

`simulate()`

Method runs rSNAPsim and simulates the single molecule translation dynamics.

Returns

ssa_int
[NumPy array.] Contains the SSA trajectories with dimensions [Time_points, simulated_trajectories].

ssa_ump
[NumPy array.] SSA trajectories in UMP(units of mature protein). SSA trajectories normalized by the number of probes in the sequence. Array with dimensions [Time_points, simulated_trajectories].

time_vector: NumPy array with dimensions [1, Time_points].
Time vector used in the simulation.

class rsnaped.rsnaped.**ScaleIntensity**(*video: ndarray*, *scale_maximum_value: int = 1000*, *ignore_channel: Optional[bool] = None*)

This class is intended to scale the intensity values in a video. The format of the video must be [T, Y, X, C].

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C].

scale_maximum_value
[int, optional] Maximum intensity value to scaled the video. The default is 1000.

ignore_channel
[int or None, optional] Use this option to ignore the normalization of a given channel. The default is None.

apply_scale()

This method is intended to scale the intensity values of a video.

Returns

scaled_video
[np.uint16] Scaled video. Array with dimensions [T, Y, X, C].

class rsnaped.rsnaped.**SimulateRNA**(*shape_output_array*, *rna_intensity_method='constant'*, *mean_int=10*)

This class simulates RNA intensity.

Parameters

shape_output_array: tuple
Desired shape of the array, a tuple with two elements where the first element is the number of trajectories and the second element represents the number of time points.

rna_intensity_method
[str, optional.] Method to generate intensity. The options are 'constant' and 'random'. The default is 'constant'.

min_int
[float, optional.] Value representing the minimal intensity in the output array. the default is zero.

max_int
[float, optional.] Value representing the maximum intensity in the output array. the default is 10.

mean_int
[float, optional.] Value representing the mean intensity in the output array. the default is 5.

simulate()

Method that simulates the RNA intensity

Returns

rna_intensities
[NumPy.] NumPy arrays with format np.int32 and dimensions equal to shape_output_array.

class rsnaped.rsnaped.**SimulatedCell**(*base_video: ndarray*, *video_for_mask: Optional[ndarray] = None*, *mask_image: Optional[ndarray] = None*, *number_spots: int = 10*, *number_frames: int = 20*, *step_size: float = 1*, *diffusion_coefficient_pixel_per_second: float = 0.01*, *simulated_trajectories_ch0: Optional[ndarray] = None*, *size_spot_ch0: int = 5*, *spot_sigma_ch0: int = 1*, *simulated_trajectories_ch1: Optional[ndarray] = None*, *size_spot_ch1: int = 5*, *spot_sigma_ch1: int = 1*, *simulated_trajectories_ch2: Optional[ndarray] = None*, *size_spot_ch2: int = 5*, *spot_sigma_ch2: int = 1*, *ignore_ch0: bool = False*, *ignore_ch1: bool = False*, *ignore_ch2: bool = False*, *intensity_calculation_method: str = 'disk_donut'*, *perform_video_augmentation: bool = 0*, *frame_selection_empty_video: str = 'shuffle'*, *ignore_trajectories_ch0: bool = False*, *ignore_trajectories_ch1: bool = False*, *ignore_trajectories_ch2: bool = False*, *intensity_scale_ch0: float = 1*, *intensity_scale_ch1: float = 1*, *intensity_scale_ch2: float = 1*, *dataframe_format: str = 'short'*)

This class takes a base video, and it draws simulated spots on top of the image. The intensity for each simulated spot is proportional to the stochastic simulation given by the user.

Parameters

base_video
[NumPy array] Array of images with dimensions [T, Y, X, C].

video_for_mask
[NumPy array, optional] Array of images with dimensions [T, Y, X, C]. Use only if the base video has been edited, and an empty video is needed to calculate the mask. The default is None.

mask_image
[NumPy array, optional] Numpy Array with dimensions [Y, X]. This image is used as a mask for the simulated video. The mask_image has to represent the same image as the base_video and video_for_mask

number_spots
[int] Number of simulated spots in the cell. The default is 10.

number_frames
[int] The number of frames or time points to simulate. In seconds. The default is 20.

step_size
[float, optional] Step size for the simulation. In seconds. The default is 1.

diffusion_coefficient_pixel_per_second
[float, optional] The diffusion coefficient for the particles' Brownian motion. The default is 0.01.

simulated_trajectories_ch0
[NumPy array, optional] An array of simulated trajectories with dimensions [S, T] for channel 0. The default is None and indicates that the intensity will be generated, drawing random numbers in a range.

size_spot_ch0
[int, optional] Spot size in pixels for channel 0. The default is 5.

spot_sigma_ch0
[int, optional] Sigma value for the simulated spots in channel 0, the units are pixels. The default is 2.

simulated_trajectories_ch1
[NumPy array, optional] An array of simulated trajectories with dimensions [S, T] for channel 1. The default is None and indicates that the intensity will be generated, drawing random numbers in a range.

size_spot_ch1
[int, optional] Spot size in pixels for channel 1. The default is 5.

spot_sigma_ch1
[int, optional] Sigma value for the simulated spots in channel 1, the units are pixels. The default is 2.

simulated_trajectories_ch2
[NumPy array, optional] An array of simulated trajectories with dimensions [S, T] for channel 2. The default is None and indicates that the intensity will be generated, drawing random numbers in a range.

size_spot_ch2
[int, optional] Spot size in pixels for channel 2. The default is 5.

spot_sigma_ch2
[int, optional] Sigma value for the simulated spots in channel 2, the units are pixels. The default is 2.

ignore_ch0
[bool, optional] A flag that ignores channel 0 returning a NumPy array filled with zeros. The default is 0.

ignore_ch1
[bool, optional] A flag that ignores channel 1 returning a NumPy array filled with zeros. The default is 0.

ignore_ch2
[bool, optional] A flag that ignores channel 2 returning a NumPy array filled with zeros. The default is 0.

intensity_calculation_method
[str, optional] Method to calculate intensity the options are : 'total_intensity' , 'disk_donut' and 'gaussian_fit'. The default is 'disk_donut'.

using_for_multiplexing
[bool, optional] Flag that indicates that multiple genes are simulated per cell.

perform_video_augmentation
[bool, optional] If true, it performs random rotations the initial video. The default is 1.

frame_selection_empty_video
[str, optional] Method to select the frames from the empty video, the options are : 'constant' , 'shuffle', 'loop', and 'linear_interpolation'. The default is 'shuffle'.

ignore_trajectories_ch0
[bool, optional] A flag that ignores plotting trajectories in channel 0. The default is False.

ignore_trajectories_ch1
[bool, optional] A flag that ignores plotting trajectories in channel 1. The default is False.

ignore_trajectories_ch2
[bool, optional] A flag that ignores plotting trajectories in channel 2. The default is False.

intensity_scale_ch0
[float , optional] Scaling factor for channel 0 that converts the intensity in the stochastic simulations to the intensity in the image.

intensity_scale_ch1
[float , optional] Scaling factor for channel 1 that converts the intensity in the stochastic simulations to the intensity in the image.

intensity_scale_ch2
[float , optional] Scaling factor for channel 2 that converts the intensity in the stochastic simulations to the intensity in the image.

dataframe_format
[str, optional] Format for the dataframe the options are : 'short' , and 'long'. The default is 'short'. "long" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean,

ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR]. "short" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, x, y].

`make_simulation()`

This method generates the simulated cell.

Returns

tensor_video
[NumPy array uint16] Array with dimensions [T, Y, X, C]

spot_positions_movement
[NumPy array] Array with dimensions [T, Spots, position(y, x)]

tensor_mean_intensity_in_figure
[NumPy array, np.float] Array with dimensions [T, Spots]

tensor_std_intensity_in_figure
[NumPy array, np.float] Array with dimensions [T, Spots]

dataframe_particles
[pandas dataframe] Dataframe with fields [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR].

`class rsnaped.rsnaped.SimulatedCellDispatcher(`*initial_video: ndarray*, *list_gene_sequences: list*, *list_number_spots: list*, *list_target_channels_proteins: list*, *list_target_channels_mRNA: list*, *list_diffusion_coefficients: list*, *list_label_names: list*, *list_elongation_rates: list*, *list_initiation_rates: list*, *simulation_time_in_sec: float*, *step_size_in_sec: float*, *mask_image: Optional[ndarray] = None*, *image_number: int = 0*, *perform_video_augmentation: bool = True*, *frame_selection_empty_video: str = 'shuffle'*, *spot_size: int = 5*, *intensity_scale_ch0=1*, *intensity_scale_ch1=1*, *intensity_scale_ch2=1*, *dataframe_format='short'*, *simulated_RNA_intensities_method='constant'*, *spot_sigma=1*, *ignore_ch0: bool = False*, *ignore_ch1: bool = False*, *ignore_ch2: bool = False*, *scale_intensity_in_base_video: bool = False*, *basal_intensity_in_background_video: int = 20000*, *microns_per_pixel: float = 1.0*, *use_Harringtonin=False*, *perturbation_time_start=0*, *perturbation_time_stop=None*, *use_FRAP=False*)

This class takes a base video and simulates a multiplexing experiment, and it draws simulated spots on top of the image. The intensity for each simulated spot is proportional to the stochastic simulation given by the user.

Parameters

initial_video
[NumPy array] Array of images with dimensions [T, Y, X, C].

list_gene_sequences
[List of strings] List where every element is a gene sequence file.

list_number_spots
[List of int] List where every element represents the number of spots to simulate for each gene.

list_target_channels_proteins
[List of int with a range from 0 to 2] List where every element represents the specific channel where the spots for the nascent proteins will be located.

list_target_channels_mRNA
[List of int with a range from 0 to 2] List where every element represents the specific channel where the spots for the mRNA signals will be located.

list_diffusion_coefficients
[List of floats] List where every element represents the diffusion coefficient for every gene.

list_label_names
[List of str] List where every element contains the label for each gene.

list_elongation_rates
[List of floats] List where every element represents the elongation rate for every gene.

list_initiation_rates
[List of floats] List where every element represents the initiation rate for every gene.

simulation_time_in_sec
[int] The simulation time in seconds. The default is 20.

step_size_in_sec
[float, optional] Step size for the simulation. In seconds. The default is 1.

mask_image
[NumPy array, optional] Numpy Array with dimensions [Y, X]. This image is used as a mask for the simulated video. The mask_image has to represent the same image as the base_video and video_for_mask.

image_number
[int, optional] Cell number used as an index for the data frame. The default is 0.

perform_video_augmentation
[bool, optional] If true, it performs random rotations the initial video. The default is True.

frame_selection_empty_video
[str, optional] Method to select the frames from the empty video, the options are : 'constant' , 'shuffle' and 'loop'. The default is 'shuffle'.

spot_size
[int, optional] Spot size in pixels. The default is 5.

spot_sigma
[int, optional.] Sigma value used to generate a gaussian Point Spread Fucntion. The default is 1.

intensity_scale_ch0
[float , optional] Scaling factor for channel 0 that converts the intensity in the stochastic simulations to the intensity in the image.

intensity_scale_ch1
[float , optional] Scaling factor for channel 1 that converts the intensity in the stochastic simulations to the intensity in the image.

intensity_scale_ch2
[float , optional] Scaling factor for channel 2 that converts the intensity in the stochastic simulations to the intensity in the image.

simulated_RNA_intensities_method
[str, optinal] Method used to simulate RNA intensities in the image. The optiions are 'constant' or 'random'. The default is 'constant'

dataframe_format
[str, optional] Format for the dataframe the options are : 'short' , and 'long'. The default is 'short'. "long" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR]. "short" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, x, y].

ignore_ch0
[bool, optional] A flag that ignores channel 0 returning a NumPy array filled with zeros. The default is 0.

ignore_ch1
[bool, optional] A flag that ignores channel 1 returning a NumPy array filled with zeros. The default is 0.

ignore_ch2
[bool, optional] A flag that ignores channel 2 returning a NumPy array filled with zeros. The default is 0.

scale_intensity_in_base_video
[bool, optional] Flag to scale intensity to a maximum value of 10000. This arbritary value is selected based on the maximum intensities obtained from the original images. The default is False.

basal_intensity_in_background_video
[int, optional] if the base video is rescaled, this value indicates the maximum value to rescale the original video. The default is 20000

use_Harringtonin: bool, optional
Flag to specify if Harringtonin is used in the experiment. The default is False.

use_FRAP: bool
Flag to specify if FRAP is used in the experiment. The default is False.

perturbation_time_start: int, optional.
Time to start the inhibition. The default is 0.

perturbation_time_stop
[int, opt.] Time to start the inhibition. The default is None.

`make_simulation()`

Method that runs the simulations for the multiplexing experiment.

Returns

tensor_video
[NumPy array uint16] Array with dimensions [T, Y, X, C]

dataframe_particles
[pandas dataframe] Dataframe with fields [cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR].

list_ssa
[List of NumPy arrays] List of numpy arrays with the stochastic simulations for each gene. The format is [S, T], where the dimensions are S = spots and T = time.

`class rsnaped.rsnaped.Trackpy(`*video: ndarray*, *mask: ndarray*, *particle_size: int = 5*, *selected_channel: int = 0*, *minimal_frames: int = 5*, *optimization_iterations: int = 1000*, *use_default_filter: bool = True*, *FISH_image: bool = False*, *show_plot: bool = True*, *intensity_threshold_tracking=None*, *image_name: str = 'temp_tracking.png'*`)`

This class is intended to detect spots in the video by using **Trackpy**.

Parameters

video
[NumPy array] Array of images with dimensions [T, Y, X, C]. In case a FISH image is used, the format must be [Z, Y, X, C], and the user must specify the parameter FISH_image = 1.

mask
[NumPy array, with Boolean values, where 1 represents the masked area, and 0 represents the area outside the mask.] An array of images with dimensions [Y, X].

particle_size
[int, optional] Average particle size. The default is 5.

selected_channel
[int, optional] Channel where the particles are detected and tracked. The default is 0.

minimal_frames
[int, optional] This parameter defines the minimal number of frames that a particle should appear on the video to be considered on the final count. The default is 5.

optimization_iterations
[int, optional] Number of iterations for the optimization process to select the best filter. The default is 30.

use_default_filter
[bool, optional] This option allows the user to use a default filter if TRUE. Else, it uses an optimization process to select the best filter for the image. The default is = True.

show_plot
[bool, optional] Allows the user to show a plot for the optimization process. The default is True.

FISH_image
[bool, optional.] This parameter allows the user to use FISH images and connect spots detected along multiple z-slices. The default is 0.

intensity_selection_threshold_int_std
[float, optional. The default is None, and it uses a default value or an optimization method if use_optimization_for_tracking is set to TRUE.] Threshold intensity for tracking

intensity_threshold_tracking
[float or None, optional.] Intensity threshold value to be used during tracking. If no value is passed, the code attempts to calculate this value. The default is None.

image_name
[str , optional] Name for the image for tracking. The default is 'temp_tracking.png'.

`perform_tracking()`

This method performs the tracking of the particles using trackpy.

Returns

trackpy_dataframe
[pandas data frame.] Pandas data frame from trackpy with fields [x, y, mass, size, ecc, signal, raw_mass, ep, frame, particle].

number_particles
[int.] The total number of detected particles in the data frame.

video_filtered
[np.uint16.] Filtered video resulting from the bandpass process. Array with format [T, Y, X, C].

`class rsnaped.rsnaped.Utilities`

This class contains miscellaneous methods to perform tasks needed in multiple classes. No parameters are necessary for this class.

`convert_to_int8`(*rescale=True*)

This method converts images from int16 to uint8. Optionally, the image can be rescaled and stretched.

Parameters

image
[NumPy array] NumPy array with dimensions [T,Y, X, C]. The code expects 3 channels (RGB). If less than 3 values are passed, the array is padded with zeros.

rescale
[bool, optional] If True it rescales the image to the min and max intensity to 0 and 255. The default is True.

`extract_field_from_dataframe`(*dataframe=None*, *selected_time=None*, *selected_field='ch1_int_mean'*, *use_nan_for_padding=False*)

This function extracts the selected_field as a vector for a given frame. If selected_time is None, the code will return the extracted data as a NumPy array with dimensions [number_particles, max_time_points]. The maximum time points are defined by the longest trajectory.

Input

dataframe_path: Patlibpath or str, optional.
Path to the selected dataframe.

dataframe
[pandas dataframe] Dataframe with fields [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR].

selected_field
[str,] selected field to extract data.

selected_time
[int, optional] Selected time point to extract information. The default is None, and indicates to select all time points.

selected_field
[str, optional.] selected dataframe to extract data.

use_nan_for_padding: bool, optional
Option to fill the array with Nans instead of zeros. The default is false.

Returns

extracted_data
[Selected Field for each particle. NumPy array with dimensions [number_particles, max_time_points] if selected_time is None. The maximum time points are defined by the longest trajectory. Short trajectories are populated with zeros or Nans.] If selected_time is giive the code will return all existing vlues that meet that condtion.

`remove_extrema_in_vector`(*max_percentile=99*)

This function is intended to remove extrema data given by the max percentiles specified by the user

`class` rsnaped.rsnaped.`VisualizerCrops`(*list_videos: list*, *list_selected_particles_dataframe: list*, *particle_size: int = 5*)

This class is intended to visualize the spots detected b trackPy as crops in an interactive widget.

Parameters

list_videos
[List of NumPy arrays or a single NumPy array.] Images or videos to visualize. The format is a list of Numpy arrays where each array follows the convention [T, Y, X, C].

list_selected_particles_dataframe
[pandas data frame.] A pandas data frame containing the position of each spot in the image.

particle_size
[int, optional] Average particle size. The default is 5.

`make_video_app()`

This method returns two objects (controls and output) to display a widget.

Returns

controls
[object] Controls from interactive to use with ipywidgets **display**.

output
[object] Output values from from interactive to use with ipywidgets **display**.

`class rsnaped.rsnaped.VisualizerImage(`*list_videos: list*, *list_videos_filtered: Optional[list] = None*, *list_selected_particles_dataframe: Optional[list] = None*, *list_files_names: Optional[list] = None*, *list_mask_array: Optional[list] = None*, *list_real_particle_positions: Optional[list] = None*, *selected_channel: int = 0*, *selected_time_point: int = 0*, *normalize: bool = False*, *individual_figure_size: float = 5*, *image_name: str = 'temp_image.png'*, *show_plot: bool = True*, *save_image_as_file: bool = False*, *colormap='plasma'*`)`

This class is intended to visualize videos as 2D images. This class has the option to mark the particles that previously were selected by trackPy.

Parameters

list_videos
[List of NumPy arrays or a single NumPy array] Images or videos to visualize. The format is a list of Numpy arrays where each array follows the convention [T, Y, X, C] or an image array with format [Y, X].

list_videos_filtered
[List of NumPy arrays or a single NumPy array or None] Images or videos to visualize. The format is a list of Numpy arrays where each array follows the convention [T, Y, X, C]. The default is None.

list_selected_particles_dataframe
[pandas data frame, optional] A pandas data frame containing the position of each spot in the image. The default is None.

list_files_names
[List of str or str, optional] List of file names to display as the title on the image. The default is None.

list_mask_array
[List of NumPy arrays or a single NumPy array, with Boolean values, where 1 represents the masked area, and 0 represents the area outside the mask.] An array of images with dimensions [Y, X].

selected_channel
[int, optional] Allows the user to define the channel to visualize in the plotted images. The default is 0.

selected_time_point
[int, optional] Allows the user to define the time point or frame to display on the image. The default is 0.

normalize
[bool, optional] Option to normalize the data by removing outliers. The code removes the 1 and 99 percentiles from the image. The default is False.

individual_figure_size
[float, optional] Allows the user to change the size of each image. The default is 5.

list_real_particle_positions
[List of Pandas dataframes or a single dataframe, optional.] A pandas data frame containing the position of each spot in the image. This dataframe is generated with class SimulatedCell, and it contains the true position for each spot. This option is only intended to be used to train algorithms for tracking and visualize real vs detected spots. The default is None.

image_name
[str, optional.] Name for the image. The default is 'temp_image.png'.

show_plot: bool, optional
Flag to display the image to screen. The default is True.

save_image_as_file
[bool, optional,] Flag to save image as png. The default is False.

`plot()`

This method plots a list of images as a grid.

Returns

None.

`class rsnaped.rsnaped.VisualizerVideo(`*list_videos: list*, *dataframe_particles=None*, *list_mask_array: Optional[list] = None*, *show_time_projection_spots: bool = False*, *normalize: bool = False*, *step_size_in_sec: float = 1*`)`

This class is intended to visualize videos as interactive widgets. This class has the option to mark the particles that previously were selected by trackPy.

Parameters

list_videos
[List of NumPy arrays or a single NumPy array] Images or videos to visualize. The format is a list of Numpy arrays where each array follows the convention [T, Y, X, C] or an image array with format [Y, X].

dataframe_particles
[pandas data frame, optional] A pandas data frame containing the position of each spot in the image. The default is None.

list_mask_array
[List of NumPy arrays or a single NumPy array, with Boolean values, where 1 represents the masked area, and 0 represents the area outside the mask.] An array of images with dimensions [Y, X].

show_time_projection_spots
[int, optional] Allows the user to display the projection of all detected spots for all time points on the current image. The default is False.

normalize
[bool, optional] Option to normalize the data by removing outliers. The code removes the 1 and 99 percentiles from the image. The default is False.

step_size_in_sec
[float, optional] Step size in seconds. The default is 1.

`make_video_app()`

This method returns two objects (controls and output) that can be used to display a widget.

Returns

controls
[object] Controls from interactive to use with ipywidgets **display**.

output
[object] Output values from from interactive to use with ipywidgets **display**.

class rsnaped.rsnaped.**VisualizerVideo3D**(*list_videos: list*)

This class is intended to visualize 3d videos as interactive widgets.

Parameters

list_videos
[List of NumPy arrays or a single NumPy array] Images or videos to visualize. The format is a list of Numpy arrays where each array follows the convention [T, Y, X, C] or an image array with format [Y, X].

make_video_app()

This method returns two objects (controls and output) that can be used to display a widget.

Returns

controls
[object] Controls from interactive to use with ipywidgets **display**.

output
[object] Output values from from interactive to use with ipywidgets **display**.

rsnaped.rsnaped.**simulate_cell**(*video_dir*, *list_gene_sequences*, *list_number_spots*, *list_target_channels_proteins*, *list_target_channels_mRNA*, *list_diffusion_coefficients*, *list_elongation_rates*, *list_initiation_rates*, *list_label_names=None*, *masks_dir=None*, *number_cells=1*, *simulation_time_in_sec=100*, *step_size_in_sec=1*, *save_as_tif=True*, *save_dataframe=True*, *save_as_gif=False*, *frame_selection_empty_video='gaussian'*, *spot_size=7*, *spot_sigma=1*, *intensity_scale_ch0=None*, *intensity_scale_ch1=None*, *intensity_scale_ch2=None*, *dataframe_format='long'*, *simulated_RNA_intensities_method='constant'*, *store_videos_in_memory=False*, *scale_intensity_in_base_video=False*, *basal_intensity_in_background_video=20000*, *microns_per_pixel=1*, *select_background_cell_index=None*, *perform_video_augmentation=True*, *use_Harringtonin=False*, *use_FRAP=False*, *perturbation_time_start=0*, *perturbation_time_stop=None*, *save_metadata=False*, *name_folder=None*)

This function is intended to simulate single-molecule translation dynamics in a cell. The result of this simultion is a .tif video and a dataframe containing the transation spot characteristics.

Parameters

video_dir
[NumPy array] Path to an initial video file with the following format Array of images with dimensions [T, Y, X, C].

masks_dir
[NumPy array] Path to an image with containing the mask used to segment the original video. The image has the following dimensions [Y, X]. Optional, the default is None.

list_gene_sequences
[List of strings] List where every element is a gene sequence file.

list_number_spots
[List of int] List where every element represents the number of spots to simulate for each gene.

list_target_channels_proteins
[List of int with a range from 0 to 2] List where every element represents the specific channel where the spots for the nascent proteins will be located.

list_target_channels_mRNA
[List of int with a range from 0 to 2] List where every element represents the specific channel where the spots for the mRNA signals will be located.

list_diffusion_coefficients
[List of floats] List where every element represents the diffusion coefficient for every gene. The units are microns^2 per second. The code automatically convert it to pixels^2 per second during the simulation.

list_label_names
[List of str or int] List where every element contains the label for each gene. Optional the default is None. None will assign an integer label to each gene from 0 to n, ehere n is the number of genes-1.

list_elongation_rates
[List of floats] List where every element represents the elongation rate for every gene.

list_initiation_rates
[List of floats] List where every element represents the initiation rate for every gene.

simulation_time_in_sec
[int] The simulation time in seconds. The default is 20.

step_size_in_sec
[float, optional] Step size for the simulation. In seconds. The default is 1.

save_as_tif
[bool, optional] If true, it generates and saves a uint16 (High) quality image tif file for the simulation. The default is True.

save_dataframe
[bool, optional] If true, it generates and saves a pandas dataframe with the simulation. Dataframe with fields [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y]. The default is 0.

save_as_gif
[bool, optional] If true, it generates and saves a .gif with the simulation. The default is 0.

frame_selection_empty_video
[str, optional] Method to select the frames from the empty video, the options are : 'constant' , 'shuffle' and 'loop'. The default is 'shuffle'.

spot_size
[int, optional] Spot size in pixels. The default is 5.

spot_sigma
[int, optional.] Sigma value used to generate a gaussian Point Spread Fucntion. The default is 1.

intensity_scale_ch0
[float , optional] Scaling factor for channel 0 that converts the intensity in the stochastic simulations to the intensity in the image.

intensity_scale_ch1
[float , optional] Scaling factor for channel 1 that converts the intensity in the stochastic simulations to the intensity in the image.

intensity_scale_ch2
[float , optional] Scaling factor for channel 2 that converts the intensity in the stochastic simulations to the intensity in the image.

simulated_RNA_intensities_method
[str, optinal] Method used to simulate RNA intensities in the image. The optiions are 'constant' or 'random'. The default is 'constant'

dataframe_format
[str, optional] Format for the dataframe the options are : 'short' , and 'long'. The default is 'short'. "long" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y, ch0_SNR,ch1_SNR,ch2_SNR]. "short" format generates this dataframe: [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, x, y].

scale_intensity_in_base_video
[bool, optional] Flag to scale intensity to a maximum value of 10000. This arbritary value is selected based on the maximum intensities obtained from the original images. The default is False.

basal_intensity_in_background_video
[int, optional] if the base video is rescaled, this value indicates the maximum value to rescale the original video. The default is 20000

select_background_cell_index
[int in range 0 to 8, optional] Index to select an specific cell for the background. Integer in range 0 to 8, or use None to select a random value.

perform_video_augmentation
[bool, optional] If true, it performs random rotations the initial video. The default is True.

use_Harringtonin: bool, optional
Flag to specify if Harringtonin is used in the experiment. The default is False.

use_FRAP: bool
Flag to specify if FRAP is used in the experiment. The default is False.

perturbation_time_start: int, optional.
Time to start the inhibition. The default is 0.

perturbation_time_stop
[int, opt.] Time to start the inhibition. The default is None.

name_folder
[None or str, optional] This string indicates the name of the solve where the results are stored. The default is None and it generates a name based on the parameters used for the simulation.

Returns

dataframe_particles
[pandas dataframe] Dataframe with fields [image_number, cell_number, particle, frame, ch0_int_mean, ch1_int_mean, ch2_int_mean, ch0_int_std, ch1_int_std, ch2_int_std, x, y].

selected_mask
[Numpy array] Array with the selected mask. Where zeros represents the background and one represent the area in the cell.

array_intensities
[Numpy array] Array with dimensions [S, T, C].

time_vector
[Numpy array] 1D array.

mean_intensities: Numpy array
Array with dimensions [S, T, C].

std_intensities
[Numpy array] Array with dimensions [S, T, C].

mean_intensities_normalized
[Numpy array] Array with dimensions [S, T, C].

std_intensities_normalized
[Numpy array] Array with dimensions [S, T, C].

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX

R

S

T

U

V

W