# Trackrefiner

***Release 1.3***

**Atiyeh Ahmadi**

**Jun 08, 2025**

# CONTENTS:

**Version**: 1.3 **Release Year**: 2025

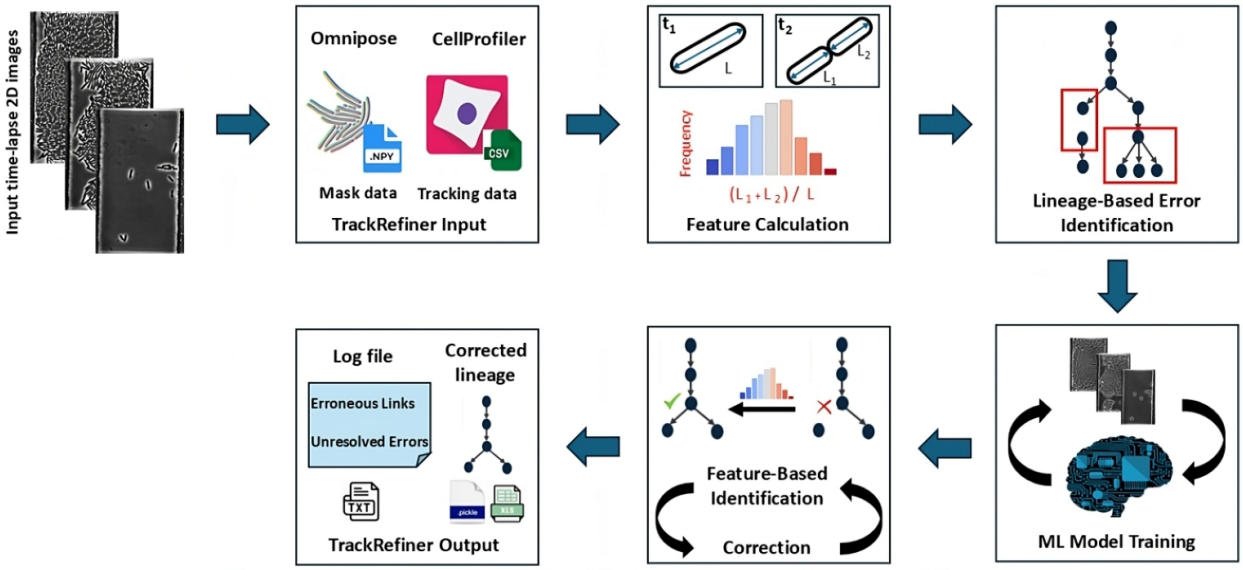
CHAPTER

ONE

# TRACKREFINER: A PRECISION TRACKING TOOL FOR BACILLUS CELL TRACKING

**TrackRefiner** is a post-processing tool specifically designed to refine the tracking outputs of CellProfiler, a widely used image processing tool. It improves tracking accuracy by refining segmentation and tracking outputs, pruning incorrect links, and adding missing ones. TrackRefiner combines traditional tracking methods with machine learning classifiers, requiring minimal user input while leveraging a data-driven approach to optimize results.

The general pipeline of TrackRefiner is illustrated in the following figure:




**Explore more:**

- Installation Guide

## 1.1 Modules

### 1.1.1 Module: processCellProfilerData

processCellProfilerData.**process_objects_data**(*is_gui_mode*, *cp_output_csv*, *segmentation_res_dir*, *neighbor_csv*, *interval_time*, *doubling_time*, *elongation_rate_method*, *pixel_per_micron*, *assigning_cell_type*, *intensity_threshold*, *disable_tracking_correction*, *clf*, *n_cpu*, *image_boundaries*, *dynamic_boundaries*, *out_dir*, *save_pickle*, *verbose*, *command*)

Processes CellProfiler output data, performs tracking correction, assigns cell types, and calculates additional features related to the life history of bacteria.

**Parameters**

- **is_gui_mode** (*bool*) – Flag indicating whether the function is called from the GUI (True) or the command line (False). Used for GUI-friendly error handling and messages.
- **cp_output_csv** (*str*) – Path to the CellProfiler output file in CSV format, which contains measured bacterial features.
- **segmentation_res_dir** (*str*) – Path to the directory containing npy files (segmentation results exported from the CellProfiler pipeline).
- **neighbor_csv** (*str*) – Path to a CSV file containing information about the neighbors of each object.
- **out_dir** (*str*) – Path to the directory where processed output files will be saved.
- **interval_time** (*float*) – Time interval between consecutive images (in minutes).
- **doubling_time** (*float*) – Minimum life history duration of bacteria (in minutes) for analysis.
- **elongation_rate_method** (*str*) – Method for calculating the elongation rate. Options: - 'Average': Computes average growth rate. - 'Linear Regression': Estimates growth rate using linear regression.
- **pixel_per_micron** (*float*) – Conversion factor for pixels to microns (e.g., default value is 0.144).
- **assigning_cell_type** (*bool*) – If True, assigns cell types to objects based on intensity thresholds.
- **intensity_threshold** (*float*) – Threshold to classify objects into specific cell types based on intensity values.

- **disable_tracking_correction** (*bool*) – If True, disables tracking correction and performs calculations directly on the CellProfiler output data.
- **clf** (*str*) – Name of the classifier to be used for model training during tracking correction. Options: 'LogisticRegression', 'GaussianProcessClassifier', 'C-Support Vector Classifier'.
- **n_cpu** (*int*) – Number of CPU cores to be used for parallel computing. *- -1* indicates that all available CPUs will be utilized.
- **image_boundaries** (*str*) – Boundary limits defined for all time steps to filter out objects outside the image area.
- **dynamic_boundaries** (*str*) – Path to a CSV file specifying boundary limits for each time step. The file should contain columns: *Time Step*, *Lower X Limit*, *Upper X Limit*, *Lower Y Limit*, *Upper Y Limit*.
- **save_pickle** (*bool*) – If True, results are saved in *.pickle* format.
- **verbose** (*bool*) – If True, displays warnings and additional details during processing.
- **command** (*str*) – User's Command Statement

**Returns**
None. Processed data and outputs are saved to the specified `out_dir`.

**Return type**
None

### 1.1.2 Module: findFixTrackingErrors

findFixTrackingErrors.**calculate_bacteria_features_and_assign_flags**(*dataframe*, *intensity_threshold*, *assigning_cell_type*, *neighbor_df*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*)

Calculate bacterial features and assign initial flags for further analysis.

This function computes features related to bacterial behavior, such as movement, division, life history, and spatial relationships. Additionally, it assigns flags and classifications (e.g., division flags, unexpected events, and daughters assignment) for downstream analysis.

**Parameters**

- **dataframe** (*pd.DataFrame*) – Input dataframe containing bacterial measured bacterial features.
- **intensity_threshold** (*float*) – Threshold value used for intensity-based cell type assignment.
- **assigning_cell_type** (*bool*) – Whether to assign cell types based on intensity values.
- **neighbor_df** (*pd.DataFrame*) – Dataframe containing information about neighboring bacteria.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centers. Example: {'x': 'Center_X', 'y': 'Center_Y'}
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.

- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.

**Returns**

tuple:

- **dataframe** (*pd.DataFrame*): Updated dataframe with calculated features and assigned flags.
- **neighbor_list_array** (*lil_matrix*): Sparse matrix indicating neighboring relationships between bacteria.
- **cell_type_array** (*np.array*): Array containing assigned cell types if *assigning_cell_type* is True; otherwise, an empty array.

findFixTrackingErrors.**find_fix_tracking_errors**(*cp_output_df*, *sorted_seg_npy_files_list*, *neighbors_df*, *center_coord_cols*, *all_rel_center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *label_col*, *interval_time*, *doubling_time*, *pixel_per_micron=0.144*, *intensity_threshold=0.1*, *assigning_cell_type=True*, *disable_tracking_correction=False*, *clf=None*, *n_cpu=-1*, *image_boundaries=None*, *dynamic_boundaries=None*, *out_dir=None*, *verbose=True*)

This function detects and corrects tracking errors in bacterial tracking data. It applies a comprehensive pipeline that includes filter out objects outside the image area, noise removal, feature calculation, training machine learning models for behavioral analysis, resolving tracking errors, and generating logs.

**Key Steps:**

1. **Unit Conversion:**
   Converts tracking data units (e.g., pixel to micron) for consistency.
2. **Boundary Checking:**
   Ensures bacteria remain within defined image or dynamic boundaries.
3. **Noise Removal:**
   Removes small or low-intensity objects that may represent noise.
4. **Feature Calculation:**
   Computes features like length, direction, and motion alignment for all bacteria.
5. **Model Training:**
   Trains machine learning models for bacterial behavior: - **Division vs. Non-Division**: Identifies dividing bacteria. - **Continuity**: Tracks the life history of bacteria across frames.
6. **Error Resolution:**
   - Resolves over assigned daughters, redundant parent links and missing connectivity.
   - Handles unexpected beginnings and ends in tracking.
   - Restores tracking links removed due to errors in previous steps.
7. **Label Propagation:**
   Reassigns sequential IDs and propagates labels for corrected data.
8. **Log Generation:**
   Creates a detailed log of all corrections, errors, and progress.

**Parameters**

- **cp_output_df** (*pandas.DataFrame*) – Input DataFrame from the CellProfiler pipeline containing tracking information and measured bacterial features.
- **sorted_seg_npy_files_list** (*list*) – List of sorted *.npy* files containing segmentation masks for each time step.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame representing neighbor relationships between bacteria.
- **center_coord_cols** (*dict*) – Dictionary specifying the x and y coordinate column names, e.g., *{'x': 'Center_X', 'y': 'Center_Y'}*.
- **all_rel_center_coord_cols** (*dict*) – Relative center coordinate columns for mapping bacteria in segmentation data.
- **parent_image_number_col** (*str*) – Column name for the parent bacterium's image number.
- **parent_object_number_col** (*str*) – Column name for the parent bacterium's object number.
- **label_col** (*str*) – Column name for bacteria labels.
- **interval_time** (*float*) – Time interval (in minutes) between consecutive frames.
- **doubling_time** (*float*) – Expected doubling time (in minutes) for bacterial division.
- **pixel_per_micron** (*float*) – Conversion factor from pixels to microns (default: *0.144*).
- **intensity_threshold** (*float*) – Threshold to classify objects into specific cell types based on intensity values. (default: *0.1*).
- **assigning_cell_type** (*bool*) – If True, assigns cell types to objects based on intensity thresholds.
- **disable_tracking_correction** (*bool*) – Flag to disable tracking error correction (default: *False*).
- **clf** (*str*) – Classifier to use for machine learning models (default: *None*).
- **n_cpu** (*int*) – Number of CPUs to use for parallel processing (default: *-1*, use all available CPUs).
- **image_boundaries** (*str*) – Boundary limits defined for all time steps to filter out objects outside the image area.
- **dynamic_boundaries** (*str*) – Path to a CSV file specifying boundary limits for each time step. The file should contain columns: *Time Step*, *Lower X Limit*, *Upper X Limit*, *Lower Y Limit*, *Upper Y Limit*.
- **out_dir** (*str*) – Directory to save output files and logs.
- **verbose** (*bool*) – If True, displays warnings and additional details during processing.

**Returns:**

tuple:

- **pandas.DataFrame df**: Corrected tracking data.
- **list logs_list**: List of progress and log messages.
- **pandas.DataFrame logs_df**: Detailed log of tracking corrections.
- **pandas.DataFrame identified_tracking_errors_df**: Identified tracking errors before correction.
- **pandas.DataFrame fixed_errors**: Corrected tracking errors.

- **pandas.DataFrame remaining_errors_df**: Remaining errors after corrections.
- **pandas.DataFrame neighbors_df**: Updated neighbors DataFrame.
- **numpy.ndarray cell_type_array**: Array of assigned cell types.

### 1.1.3 Module: multiRegionsDetection

multiRegionsDetection.**map_and_detect_multi_regions**(*df*, *sorted_seg_npy_files_list*, *pixel_per_micron*, *center_coord_cols*, *all_rel_center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *verbose*)

Map objects to segmentation files, detect multi-region areas, and prepare pixel data for tacking correction.

This function maps objects from the tracking dataframe (*df*) to their corresponding regions in segmentation files (*npy* format). It detects multi-region areas where disconnected regions share the same color and flags them for correction in a separate function. For all objects, their pixels are mapped into a sparse coordinate matrix and color array for further analysis.

**Parameters**

- **df** (*pd.DataFrame*) – Dataframe containing bacterial measured bacterial features.
- **sorted_seg_npy_files_list** (*list*) – sorted List of file paths to the segmentation files in *.npy* format.
- **pixel_per_micron** (*float*) – Conversion factor from pixels to microns for spatial measurements.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of object centers. Example: {'x': 'Center_X', 'y': 'Center_Y'}
- **all_rel_center_coord_cols** (*dict*) – Dictionary containing x and y coordinate column names.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.
- **verbose** (*bool*) – Whether to print detailed information about the processing steps for debugging.

**Returns**

A tuple containing:

- **df** (*pd.DataFrame*): Updated dataframe with mapped object data. Objects with incorrect regions in the segmentation file are flagged as noise.
- **coordinate_array** (*csr_matrix*): Sparse matrix indicating the pixel locations of objects.
- **color_array** (*np.ndarray*): Array of RGB color values for each object.

### 1.1.4 Module: multiRegionsCorrection

multiRegionsCorrection.**modify_existing_object**(*df*, *regions_color*, *faulty_row*, *img_array*, *regions_coordinates*, *coordinate_array*, *color_array*)

Modifies attributes of an existing object in the image and updates the relevant data structures.

**Parameters**

- **df** (*pandas.DataFrame*) – Input DataFrame containing object data, including tracking and feature information.
- **regions_color** (*list*) – list representing the colors assigned to different regions.
- **faulty_row** (*dict*) – A dictionary containing details about the faulty object, including indices used for updates.
- **img_array** (*numpy.ndarray*) – Image array where object regions are represented by their respective colors.
- **regions_coordinates** (*list of numpy.ndarray*) – A list of arrays where element indices are region indices, and elements are arrays of coordinates for each region.
- **coordinate_array** (*csr_matrix*) – Boolean array tracking encoded spatial coordinates for each object.
- **color_array** (*numpy.ndarray*) – Array storing the color of each object.

**Returns**

tuple:

Updated DataFrame (*df*), image array (*img_array*), coordinate array (*coordinate_array*), and color array (*color_array*).

multiRegionsCorrection.**multi_region_correction**(*df*, *img_array*, *distance_df_cp_connected_regions*, *img_number*, *regions_center_particles_stat*, *regions_center_particles*, *regions_color*, *regions_coordinates*, *regions_features*, *all_rel_center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *min_distance_prev_objects_df*, *coordinate_array*, *color_array*)

Corrects multi-region objects by identifying mismatched or faulty associations and updating or removing objects as necessary. Ensures consistency between regions and objects while maintaining accurate data records.

**Parameters**

- **df** (*pandas.DataFrame*) – Input DataFrame containing object data.
- **img_array** (*numpy.ndarray*) – Image array representing the current state of the image.
- **distance_df_cp_connected_regions** (*pandas.DataFrame*) – DataFrame containing pairwise distances between objects (based on CellProfiler records) and regions.
- **img_number** (*int*) – Current image number
- **regions_center_particles_stat** (*list*) – List of status for regions' center points.
- **regions_center_particles** (*list*) – List of coordinates for regions' center points.
- **regions_color** (*list*) – list of region colors in current time step.
- **regions_coordinates** (*list of numpy.ndarray*) – A list of arrays where each region index maps to an array of its coordinates.
- **regions_features** (*list of dict*) – A list of dictionaries where each region index maps to its features (e.g., orientation, axis lengths).
- **all_rel_center_coord_cols** (*dict*) – Dictionary containing x and y coordinate column names.
- **parent_image_number_col** (*str*) – Name of the parent image number column in the DataFrame.

- **parent_object_number_col** (*str*) – Name of the parent object number column in the DataFrame.
- **min_distance_prev_objects_df** (*pandas.DataFrame*) – DataFrame containing previous object distance information.
- **coordinate_array** (*csr_matrix*) – Boolean array tracking encoded spatial coordinates for each object.
- **color_array** (*numpy.ndarray*) – Array storing the colors of objects.

**Returns**

tuple: Updated DataFrame (*df*), coordinate array (*coordinate_array*), and color array (*color_array*).

multiRegionsCorrection.**remove_object_from_image**(*df*, *faulty_row*, *img_array*, *regions_coordinates*)

Removes an object from the image by removing its region in the image array and marking it as noise in the DataFrame.

**Parameters**

- **df** (*pandas.DataFrame*) – Input DataFrame containing object data, including tracking and feature information.
- **faulty_row** (*dict*) – A dictionary containing details about the faulty object, including indices used for updates.
- **img_array** (*numpy.ndarray*) – Image array where object regions are represented by their respective colors.
- **regions_coordinates** (*list of numpy.ndarray*) – A list of arrays where element indices are region indices, and elements are arrays of coordinates for each region.

**Returns**

tuple: Updated DataFrame (*df*) and image array (*img_array*).

multiRegionsCorrection.**update_object_records**(*df*, *faulty_row*, *img_array*, *regions_coordinates*, *regions_color*, *regions_features*, *all_center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *coordinate_array*, *color_array*)

Updates the records and attributes of a multi-region object after determining the best-fitting region.

**Parameters**

- **df** (*pandas.DataFrame*) – Input DataFrame containing object data, including tracking and feature information.
- **faulty_row** (*dict*) – A dictionary containing details about the faulty object, including indices used for updates.
- **img_array** (*numpy.ndarray*) – Image array where object regions are represented by their respective colors.
- **regions_coordinates** (*list of numpy.ndarray*) – A list of Arrays where element indices are region indices, and elements are arrays of coordinates for each region.
- **regions_color** (*list*) – list representing the colors assigned to different regions.
- **regions_features** (*list of dict*) – A list of dictionaries where keys are region indices, and values contain feature data for each region (e.g., orientation, major/minor axis lengths).
- **all_center_coord_cols** (*dict*) – A dictionary mapping coordinate types (*x*, *y*) to their respective column names in the DataFrame.

- **parent_image_number_col** (*str*) – Column name in the DataFrame representing the parent image number.
- **parent_object_number_col** (*str*) – Column name in the DataFrame representing the parent object number.
- **coordinate_array** (*csr_matrix*) – Boolean array tracking encoded spatial coordinates for each object.
- **color_array** (*numpy.ndarray*) – Array storing the color of each object.

**Returns**

tuple: Updated DataFrame (*df*), image array (*img_array*), coordinate array (*coordinate_array*), and color array (*color_array*).

Updates include:

- **Center Coordinates**: Updates the x and y center positions (*center_x*, *center_y*) in the DataFrame.
- **Shape Attributes**: Updates the major axis length (*AreaShape_MajorAxisLength*), minor axis length (*AreaShape_MinorAxisLength*), and orientation (*AreaShape_Orientation*) in radians.
- **Parent Information**: Resets the parent image number and parent object number to *0*.

### 1.1.5 Module: regionOfInterestFilter

regionOfInterestFilter.**filter_objects_outside_boundary**(*x_min*, *x_max*, *y_min*, *y_max*, *df*, *pixel_per_micron*, *center_coord_cols*)

Identifies and marks objects outside a user-defined boundary in the image, applying the same boundary limits across all time steps.

**Parameters**

- **x_min** (*float*) – The minimum x-coordinate (in micrometers) for the boundary.
- **x_max** (*float*) – The maximum x-coordinate (in micrometers) for the boundary.
- **y_min** (*float*) – The minimum y-coordinate (in micrometers) for the boundary.
- **y_max** (*float*) – The maximum y-coordinate (in micrometers) for the boundary.
- **df** (*pandas.DataFrame*) – The DataFrame containing tracking data and measured bacterial features for objects. This DataFrame is updated in place.
- **pixel_per_micron** (*float*) – The conversion factor from pixels to micrometers.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of object centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

**Returns**

pandas.DataFrame: The updated DataFrame with objects outside the boundary marked as noise (*noise_bac=True*).

regionOfInterestFilter.**filter_objects_outside_dynamic_boundary**(*dynamic_boundaries*, *df*, *pixel_per_micron*, *center_coord_cols*)

Identifies and marks objects outside user-defined, time-dependent boundaries in the image.

**Parameters**

- **dynamic_boundaries** (*pandas.DataFrame*) – A DataFrame containing time-dependent boundary limits for each time step. It includes columns specifying lower and upper limits for x and y coordinates (*Lower X Limit*, *Upper X Limit*, *Lower Y Limit*, *Upper Y Limit*) and the corresponding time step (*Time Step*).
- **df** (*pandas.DataFrame*) – The main DataFrame containing tracking data and measured bacterial features for all objects. This DataFrame is updated in place.
- **pixel_per_micron** (*float*) – The conversion factor from micrometers to pixels.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of object centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

**Returns**

pandas.DataFrame: The updated DataFrame with objects outside the boundaries marked as noise (*noise_bac=True*).

regionOfInterestFilter.**find_inside_boundary_objects**(*boundary_limits*, *dynamic_boundaries*, *df*, *center_coord_cols*, *pixel_per_micron*)

Identifies and retains objects within specified boundary limits, either static or time-dependent, by filtering out objects outside these boundaries.

**Parameters**

- **boundary_limits** (*str*) – A comma-separated string defining static boundary limits in micrometers as *x_min, x_max, y_min, y_max*. If provided, these limits are applied consistently across all time steps.
- **dynamic_boundaries** (*str*) – Path to a CSV file containing time-dependent boundary limits. The CSV must include columns specifying *Lower X Limit*, *Upper X Limit*, *Lower Y Limit*, *Upper Y Limit*, and *Time Step*. This parameter is used if *boundary_limits* is *None*.
- **df** (*pandas.DataFrame*) – The main DataFrame containing tracking data and measured bacterial features for all objects. This DataFrame is updated in place.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of object centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **pixel_per_micron** (*float*) – The conversion factor from micrometers to pixels.

**Returns**

pandas.DataFrame: The updated DataFrame with objects outside the boundaries marked as noise (*noise_bac=True*)

### 1.1.6 Module: noiseObjects

noiseObjects.**detect_and_remove_noise_bacteria**(*df*, *neighbors_df*, *parent_image_number_col*, *parent_object_number_col*)

Detect and remove noise bacteria from tracking data.

This function identifies noise objects (bacteria) based on major length and removes them from the dataframe. It iteratively detects noise objects, removes them, and updates neighbor relationships.

**Parameters**

- **df** (*pd.DataFrame*) – Dataframe containing bacterial tracking data, including features such as object major length
- **neighbors_df** (*pd.DataFrame*) – Dataframe containing information about neighboring bacteria relationships.

- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.

**Returns**

tuple:

- **df** (*pd.DataFrame*): Updated dataframe with noise bacteria removed.
- **neighbors_df** (*pd.DataFrame*): Updated neighbor relationships after removing noise bacteria.

noiseObjects.**update_tracking_after_removing_noise**(*df*, *noise_objects_df*, *neighbors_df*, *parent_image_number_col*, *parent_object_number_col*)

Update tracking and neighbor relationships after removing noise bacteria.

This function updates the tracking data and neighbor relationships when noise bacteria are identified and flagged. It: - Removes relationships where noise bacteria are parents. - Cleans up incorrect neighbor relationships involving noise bacteria.

**Parameters**

- **df** (*pd.DataFrame*) – Dataframe containing bacterial tracking data, including IDs, features, and noise flags.
- **noise_objects_df** (*pd.DataFrame*) – Subset of the dataframe containing the detected noise bacteria to be removed.
- **neighbors_df** (*pd.DataFrame*) – Dataframe containing neighbor relationship information for bacteria.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.

**Returns**

tuple:

- **df** (*pd.DataFrame*): Updated dataframe with noise bacteria flagged and their parent relationships reset.
- **neighbors_df** (*pd.DataFrame*): Updated neighbor relationships with noise bacteria removed.

### 1.1.7 Module: bacterialBehaviorModelTraining

bacterialBehaviorModelTraining.**train_bacterial_behavior_models**(*df*, *neighbors_df*, *neighbor_list_array*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *output_directory*, *clf*, *n_cpu*, *coordinate_array*)

Trains machine learning models to predict bacterial behaviors, including division, non-division, and distinguishing between divided and non-divided bacteria.

**Parameters**

- **df** (*pandas.DataFrame*) – DataFrame containing bacterial tracking data with spatial, temporal, and neighbor information.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing bacterial neighbor relationships for all time steps.
- **neighbor_list_array** (*csr_matrix*) – Sparse matrix representing neighbor relationships for efficient lookup.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **parent_image_number_col** (*str*) – Column name representing the image number of parent bacteria.
- **parent_object_number_col** (*str*) – Column name representing the object number of parent bacteria.
- **output_directory** (*str*) – Directory to save the trained models and performance logs.
- **clf** (*str*) – Classifier type (e.g., *'LogisticRegression'*, *'GaussianProcessClassifier'*, *'SVC'*).
- **n_cpu** (*int*) – Number of CPUs to use for parallel processing during model training.
- **coordinate_array** (*np.ndarray*) – Array of spatial coordinates for evaluating bacterial connections.

**Returns:**

**tuple: Three trained machine learning models:**

- **division_vs_continuity_model**: Model to distinguish between division and continuity links.
- **continuity_links_model**: Model to predict the behavior of continuity links.
- **division_links_model**: Model to analyze the characteristics of division links.

### 1.1.8 Module: mlModelDivisionVsContinuity

mlModelDivisionVsContinuity.**train_division_vs_continuity_model**(*connected_bac*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *output_directory*, *clf*, *n_cpu*, *coordinate_array*)

Constructs a machine learning model to compare divided and non-divided bacteria. The model uses features such as movement, spatial relationships, neighbor changes, and bacterial morphology to classify each instance.

**Workflow:**

- Splits the dataset into "divided" and "non-divided" bacteria based on duplication in parent relationships.
- Extracts and calculates features for each group, including: - Intersection over Union (IoU). - Spatial distances. - Neighbor differences and ratios. - Direction of motion and motion alignment angles. - Length change ratios.
- Labels the "divided" group as negative and the "non-divided" group as positive.
- Combines the features and trains the specified classifier.

**Parameters**

- **connected_bac** (*pandas.DataFrame*) – DataFrame containing connected bacteria data, including spatial, temporal, and neighbor information.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **parent_image_number_col** (*str*) – Column name representing the image number of parent bacteria.
- **parent_object_number_col** (*str*) – Column name representing the object number of parent bacteria.
- **output_directory** (*str*) – Directory where model outputs (e.g., performance logs, trained models) will be saved.
- **clf** (*str*) – Classifier type (e.g., *'LogisticRegression'*, *'GaussianProcessClassifier'*, *'SVC'*).
- **n_cpu** (*int*) – Number of CPUs to use for parallel processing during training.
- **coordinate_array** (*np.ndarray*) – Array of spatial coordinates for evaluating bacterial links.

**Returns:**

sklearn.Model:

Trained machine learning model to classify divided and non-divided bacteria.

### 1.1.9 Module: mlModelContinuity

mlModelContinuity.**train_continuity_links_model**(*df*, *connected_bac_high_chance_to_be_correct_with_neighbors_info*, *neighbors_df*, *neighbor_list_array*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *output_directory*, *clf*, *n_cpu*, *coordinate_array*)

Constructs a machine learning model to predict whether a bacterium will continue its life history.

**Workflow:**

- Extracts features for bacteria and their neighbors, including: - Size relationships (e.g., *LengthChangeRatio*). - Spatial relationships (e.g., *IoU*, distances, neighbor differences). - Motion alignment.
- Trains the specified classifier on the engineered features.
- Outputs the trained model.

**Parameters**

- **df** (*pandas.DataFrame*) – The complete dataset containing bacterial tracking information across time steps.
- **connected_bac_high_chance_to_be_correct_with_neighbors_info** (*pandas.DataFrame*) – Subset of bacteria with high probabilities of being correctly linked, including neighbor-related information.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing details about bacterial neighbors.
- **neighbor_list_array** (*csr_matrix*) – Sparse matrix representation of bacterial neighbor relationships.

- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **parent_image_number_col** (*str*) – Column name representing the image number of parent bacteria.
- **parent_object_number_col** (*str*) – Column name representing the object number of parent bacteria.
- **output_directory** (*str*) – Directory where model outputs (e.g., performance logs, trained models) will be saved.
- **clf** (*str*) – Classifier type (e.g., *'LogisticRegression'*, *'GaussianProcessClassifier'*, *'SVC'*).
- **n_cpu** (*int*) – Number of CPUs to use for parallel processing during training.
- **coordinate_array** (*scipy.sparse.csr_matrix*) – Sparse matrix of spatial coordinates for evaluating bacterial links.

**Returns:**

sklearn.Model:

Trained machine learning model to predict non-divided bacteria.

### 1.1.10 Module: mlModelDivision

mlModelDivision.**train_division_links_model**(*df*, *connected_bac_high_chance_to_be_correct_with_neighbors_info*, *neighbor_list_array*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *output_directory*, *clf*, *n_cpu*, *coordinate_array*)

Constructs a machine learning model to predict whether a bacterium will divide and be born daughters.

**Workflow:**

- Extracts features for bacteria pairs with division signals (*mother-daughter* relationships).
- Filters outliers based on size and neighbor characteristics.
- Computes key features such as: - Intersection over Union (IoU). - Spatial distances. - Neighbor changes (difference and common neighbors). - Angle between mother and daughter bacteria.
- Assigns positive labels for actual divisions and negative labels for unrelated bacteria pairs (link between daughter and source's neighbor).
- Trains the specified machine learning classifier using the derived features.

**Parameters**

- **df** (*pandas.DataFrame*) – The complete dataset containing bacterial tracking information across time steps.
- **connected_bac_high_chance_to_be_correct_with_neighbors_info** (*pandas.DataFrame*) – Subset of bacteria with high probabilities of being correctly linked, including neighbor-related information.
- **neighbor_list_array** (*csr_matrix*) – Sparse matrix representation of bacterial neighbor relationships.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

- **parent_image_number_col** (*str*) – Column name representing the image number of parent bacteria.
- **parent_object_number_col** (*str*) – Column name representing the object number of parent bacteria.
- **output_directory** (*str*) – Directory where model outputs (e.g., performance logs, trained models) will be saved.
- **clf** (*str*) – Classifier type (e.g., *'LogisticRegression'*, *'GaussianProcessClassifier'*, *'SVC'*).
- **n_cpu** (*int*) – Number of CPUs to use for parallel processing during training.
- **coordinate_array** (*scipy.sparse.csr_matrix*) – sparse matrix of spatial coordinates for evaluating bacterial links.

**Returns:**
sklearn.Model:

Trained machine learning model to classify divided bacteria relationships.

### 1.1.11 Module: machineLearningPipeline

machineLearningPipeline.**draw_kde_plot**(*pos_class_prob_values_dict*, *model_type*, *clf*, *output_directory*, *dsc*)

Draws and saves a KDE plot for the probability distributions of positive class probabilities across different trainings (Training on all data, Training on train split, Evaluation on test split).

**Parameters**

- **pos_class_prob_values_dict** (*dict*) – A dictionary where keys represent training type and values are arrays of positive class probabilities for the respective class.
- **model_type** (*str*) – Descriptive name of the model type (e.g., division_vs_continuity).
- **clf** (*str*) – Name of the classifier used.
- **output_directory** (*str*) – Directory where the plot will be saved.
- **dsc** (*str*) – Description for the data being plotted.

**Returns:**
None

machineLearningPipeline.**extract_feature_importance**(*pipeline*)

Extracts feature importance or weights from a trained machine learning pipeline.

**Parameters**
**pipeline** (*sklearn.pipeline.Pipeline*) – Trained machine learning pipeline.

**Returns**
pandas.DataFrame with columns 'Feature' and 'Weight', representing feature names and their corresponding weights/importance. Returns an empty DataFrame if the classifier does not support feature importance.

machineLearningPipeline.**extract_positive_class_probability_values**(*pipeline*, *x_dat*, *dat*)

Extracts the positive class probability values from a trained model pipeline.

**Parameters**

- **pipeline** (*sklearn.pipeline.Pipeline*) – Trained machine learning pipeline.

- **x_dat** (*pandas.DataFrame*) – DataFrame containing feature data.
- **dat** (*pandas.DataFrame*) – DataFrame containing additional data, including real labels.

**Returns**
numpy.ndarray containing the positive class probability values for the positive class.

machineLearningPipeline.**log_model_performance**(*output_directory*, *model_type*, *pos_class_prob_values_dict*, *pref_measures_dict*, *feature_importance_df*, *clf*, *num_full_pos_neg_class_samples*, *dsc*, *clas_names*)

Logs the performance metrics of a machine learning model and saves the results to files, including a probability distribution plot and a text log file.

**Parameters**

- **output_directory** (*str*) – Directory where output files, including performance logs and plots, will be saved.
- **model_type** (*str*) – Descriptive name of model (division_vs_continuity, continuity, division)
- **pos_class_prob_values_dict** (*dict*) – Dictionary containing positive class probabilities for different sets (Training on all data, Training on train split, Evaluation on test split).
- **pref_measures_dict** (*dict*) – Dictionary of performance measures for each dataset.
- **feature_importance_df** (*pandas.DataFrame*) – DataFrame containing feature names and their importances or weights.
- **clf** (*str*) – Name of the classifier used.
- **num_full_pos_neg_class_samples** (*dict*) – Dictionary with sample counts for positive and negative classes across trainings ( Training on all data, Training on train split, Evaluation on test split).
- **dsc** (*str*) – Description for the data being evaluated.
- **clas_names** (*list*) – List of class names used in performance metrics.

**Returns:**
None

machineLearningPipeline.**make_full_data**(*x_dat*, *y_dat*)

Combines feature data and target labels into a single DataFrame.

**Parameters**

- **x_dat** (*pandas.DataFrame*) – DataFrame containing feature data.
- **y_dat** (*pandas.Series*) – Series containing target labels.

**Returns**
pandas.DataFrame with an added 'real_label' column containing target labels.

machineLearningPipeline.**numb_positive_negative_class_samples**(*df*)

Counts the number of positive and negative samples in a DataFrame.

**Parameters**
**df** (*pandas.DataFrame*) – DataFrame containing a 'real_label' column with class labels.

**Returns**
Tuple (n_negative, n_positive) with counts of negative and positive samples.

machineLearningPipeline.**performance_calculation**(*y_dat*, *y_predicted*)

Calculates performance metrics including accuracy, specificity, and sensitivity.

**Parameters**

- **y_dat** (*numpy.ndarray*) – True labels.
- **y_predicted** (*numpy.ndarray*) – Predicted labels.

**Returns**

List of performance metrics [tn, fp, fn, tp, accuracy, specificity, sensitivity].

machineLearningPipeline.**train_model**(*merged_df*, *feature_list*, *columns_to_scale*, *model_type*, *output_directory*, *clf*, *n_cpu*)

Trains a machine learning model using the specified classifier and evaluates its performance. The function applies preprocessing, trains the model, and calculates various performance metrics such as accuracy, specificity, and sensitivity.

**Parameters**

- **merged_df** (*pandas.DataFrame*) – The input DataFrame containing the features and target labels for training the model. The target labels are expected to be in a column named *label*.
- **feature_list** (*list*) – A list of column names in merged_df representing the features used for training.
- **columns_to_scale** (*list*) – A list of feature columns in merged_df that should be scaled using StandardScaler.
- **model_type** (*str*) – Descriptive name of model (division_vs_continuity, continuity, division)
- **output_directory** (*str*) – The directory where output files, including performance logs and plots, will be saved.
- **clf** (*str*) –

  **The name of the classifier to use. Supported classifiers include:**

  - 'LogisticRegression'
  - 'GaussianProcessClassifier'
  - 'C-Support Vector Classifier'
- **n_cpu** (*int*) – Number of CPU cores to use for parallel processing in the classifier.

**Returns:**

sklearn.pipeline.Pipeline

**returns**

A trained machine learning pipeline containing the preprocessing steps and the classifier.

### 1.1.12 Module: findOutlier

findOutlier.**calculate_bacterial_length_boundary**(*df*)

Calculates the statistical boundary (average and standard deviation) for bacterial length values.

**Parameters**

**df** (*pandas.DataFrame*) – DataFrame containing the column "AreaShape_MajorAxisLength".

**Returns**

dict: A dictionary with 'avg' (average) and 'std' (standard deviation) for bacterial lengths.

findOutlier.**calculate_length_change_ratio_boundary**(*df*)

Calculates the statistical boundary for bacterial length change ratios below 1.

**Parameters**
**df** (*pandas.DataFrame*) – DataFrame containing the column "Length_Change_Ratio".

**Returns**
dict: A dictionary with 'avg' (average) and 'std' (standard deviation) for the ratios below 1.

findOutlier.**calculate_lower_statistical_bound**(*feature_boundary_dict*)

Calculates the lower statistical boundary using the formula: avg - 1.96 * std.

**Parameters**
**feature_boundary_dict** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation).

**Returns**
float: The lower statistical boundary.

findOutlier.**calculate_max_daughter_to_mother_boundary**(*df*)

Calculates the statistical boundary (average and standard deviation) for the maximum daughter-to-mother length ratio.

**Parameters**
**df** (*pandas.DataFrame*) – DataFrame containing the column "Max_Daughter_Mother_Length_Ratio".

**Returns**
dict: A dictionary with 'avg' (average) and 'std' (standard deviation) for the ratio.

findOutlier.**calculate_sum_daughter_to_mother_len_boundary**(*df*)

Calculates the statistical boundary (average and standard deviation) for the sum_daughter-to-mother length ratio.

**Parameters**
**df** (*pandas.DataFrame*) – DataFrame containing the column "Total_Daughter_Mother_Length_Ratio".

**Returns**
dict: A dictionary with 'avg' (average) and 'std' (standard deviation) for the ratio.

findOutlier.**calculate_upper_statistical_bound**(*feature_boundary_dict*)

Calculates the upper statistical boundary using the formula: avg + 1.96 * std.

**Parameters**
**feature_boundary_dict** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation).

**Returns**
float: The upper statistical boundary.

findOutlier.**detect_daughter_to_mother_length_outliers**(*df*)

Detects outliers in the daughter-to-mother length ratios.

**Parameters**
**df** (*pandas.DataFrame*) – DataFrame containing "Total_Daughter_Mother_Length_Ratio" and "Max_Daughter_Mother_Length_Ratio" columns.

**Returns**
pandas.DataFrame: A subset of the input DataFrame containing rows that exceed the calculated upper bounds for daughter-to-mother ratios or have ratios > 1.

findOutlier.**detect_length_change_ratio_outliers**(*df*)

Detects bacterial length change ratio outliers

**Parameters**

**df** (*pandas.DataFrame*) – DataFrame containing the column "Length_Change_Ratio" for analysis.

**Returns**

float: The maximum outlier value for bacterial length change ratios, or NaN if none exist.

findOutlier.**find_bacterial_length_outliers**(*values*)

Identifies bacterial length outliers

**Parameters**

**values** (*numpy.ndarray*) – Array of bacterial length values to analyze.

**Returns**

float: The maximum outlier value below the calculated lower bound, if any exist. Otherwise, returns NaN.

### 1.1.13 Module: overAssignedDaughters

overAssignedDaughters.**resolve_over_assigned_daughters_link**(*df*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *division_links_model*, *coordinate_array*)

This function identifies parent bacteria with over-assigned daughter links (OAD), evaluates the validity of these links using a machine learning model, and resolves the issue by: - Removing invalid daughter links. - Updating features for parent bacteria with corrected daughters. - Propagating corrected features to the daughter bacteria.

**Parameters**

- **df** (*pandas.DataFrame*) – DataFrame containing tracking data for bacteria, including division-related features.
- **parent_image_number_col** (*str*) – Column name for the parent image numbers.
- **parent_object_number_col** (*str*) – Column name for the parent object numbers.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **division_links_model** (*sklearn.Model*) – Machine learning model used to evaluate the probability of valid division links.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating spatial relationships.

**Returns**: :returns pandas.DataFrame: Updated DataFrame with corrected division links and propagated features.

### 1.1.14 Module: redundantParentLink

redundantParentLink.**detect_and_resolve_redundant_parent_link**(*df*, *neighbor_df*, *neighbor_list_array*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *continuity_links_model*, *coordinate_array*)

Detects and resolves redundant parent links (RPL) in bacterial lineage tracking by evaluating mother-daughter relationships based on geometric, motion, and neighbor-based features. This function performs iterative checks to detect RPLs.

**Parameters**

- **df** (*pandas.DataFrame*) – DataFrame containing tracking data for bacteria.
- **neighbor_df** (*pandas.DataFrame*) – DataFrame containing neighbor relationships between bacteria.
- **neighbor_list_array** (*lil_matrix*) – Matrix representing neighbor connections between bacteria.
- **parent_image_number_col** (*str*) – Column name in *df* for the parent bacterium's image number.
- **parent_object_number_col** (*str*) – Column name in *df* for the parent bacterium's object number.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **continuity_links_model** (*sklearn.Model*) –

  **Machine learning model used to evaluate which target bacterium the source bacterium should link to in order to**
  continue its life history.
- **coordinate_array** (*csr_matrix*) – Matrix of spatial coordinates used for geometric and spatial calculations.

**Returns:**

pandas.DataFrame:

Updated DataFrame with resolved RPLs and recalculated features.

### 1.1.15 Module: missingConnectivityLink

missingConnectivityLink.**detect_and_resolve_missing_connectivity_link**(*df*, *neighbors_df*, *neighbor_matrix*, *doubling_time*, *interval_time*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *division_vs_continuity_model*, *continuity_links_model*, *division_links_model*, *coordinate_array*)

Detect and resolve missing connectivity links in bacterial tracking data.

This function identifies missing connectivity links in bacterial tracking, finds candidate sources for linking with target bacteria, and evaluates candidates based on statistical conditions, biological plausibility, and machine learning models. If valid candidates are found, the function modifies the tracking data to establish new links.

**Parameters**

- **df** (*pd.DataFrame*) – Dataframe containing bacterial tracking data, including IDs, life histories, and coordinates.
- **neighbors_df** (*pd.DataFrame*) – Dataframe containing information about neighboring bacteria.
- **neighbor_matrix** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **doubling_time** (*float*) – Estimated doubling time for bacteria, used to set thresholds for division and growth.
- **interval_time** (*float*) – Time interval between frames in the tracking data.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centers. Example: {'x': 'Center_X', 'y': 'Center_Y'}
- **division_vs_continuity_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria.
- **continuity_links_model** (*sklearn.Model*) – Machine learning model used to evaluate candidate links for non-divided bacteria.
- **division_links_model** (*sklearn.Model*) – Machine learning model used to validate division links.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating candidate links (calculation of IOU).

**Returns**

pd.DataFrame:

Updated dataframe with resolved connectivity links and adjusted bacterial life histories.

missingConnectivityLink.**resolve_missing_connectivity_links**(*df*, *neighbors_df*, *neighbor_matrix*, *source_bac_idx*, *source_bac*, *link_probability*, *new_target_bac_idx*, *link_type*, *target_frame_bacteria*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*)

Resolve missing connectivity links by removing incorrect links and establishing new ones.

This function adjusts bacterial tracking data to resolve missing connectivity by removing redundant or incorrect links and creating new ones. Based on the *link_probability* and the *link_type*, it establishes or removes connections for division links (*'division'*) or continuity links (*'continuity'*).

**Parameters**

- **df** (*pd.DataFrame*) – Dataframe containing bacterial tracking data, including IDs, relationships, and coordinates.
- **neighbors_df** (*pd.DataFrame*) – Dataframe containing neighbor relationship information for bacteria.
- **neighbor_matrix** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **source_bac_idx** (*int*) – Index of the source bacteria whose tracking link is being resolved.

- **source_bac** (*pd.Series*) – Data for the source bacteria whose link needs to be adjusted.
- **link_probability** (*float*) – Probability value from a machine learning model. The value *1 - link_probability* represents the likelihood that the proposed new link is valid.
- **new_target_bac_idx** (*int*) – Index of the target bacteria to establish a new link with.
- **link_type** (*str*) – Type of link to handle: - *'division'*: Handles division events. - *'continuity'*: Handles continuity of the same bacteria across frames.
- **target_frame_bacteria** (*pd.DataFrame*) – Subset of the dataframe containing bacteria in the target time frame.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centers. Example: {'x': 'Center_X', 'y': 'Center_Y'}

**Returns**

pd.DataFrame:

Updated dataframe with resolved missing connectivity link and adjusted bacterial tracking features.

### 1.1.16 Module: unexpectedBeginning

unexpectedBeginning.**handle_unexpected_beginning_bacteria**(*df*, *neighbors_df*, *neighbor_matrix*, *interval_time*, *doubling_time*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *division_vs_continuity_model*, *continuity_links_model*, *division_links_model*, *color_array*, *coordinate_array*)

Handle unexpected beginning bacteria by finding and validating candidate sources, and creating new links.

This function processes bacteria flagged with *unexpected_beginning* by identifying candidate target bacteria in the next frame. It validates potential links using statistical conditions, biological plausibility, and machine learning models. If validated, it creates new links to resolve tracking gaps.

**The function handles two scenarios:**

- **continuity**: The source bacterium's life history continues to the next frame.
- **division**: The source bacterium divides, linking to two daughter bacteria.

**Parameters**

- **df** (*pd.DataFrame*) – Dataframe containing bacterial measured bacterial features.
- **neighbors_df** (*pd.DataFrame*) – Dataframe containing neighbor relationship information for bacteria.
- **neighbor_matrix** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **interval_time** (*float*) – Time interval between frames in the tracking data.

- **doubling_time** (*float*) – Estimated doubling time for bacteria, used to determine plausible division events.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centers. Example: {'x': 'Center_X', 'y': 'Center_Y'}
- **division_vs_continuity_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria.
- **continuity_links_model** (*sklearn.Model*) – Machine learning model used to evaluate candidate links for continuity events.
- **division_links_model** (*sklearn.Model*) – Machine learning model used to validate division links.
- **color_array** (*np.ndarray*) – Array representing the colors of objects in the tracking data. This is used for mapping bacteria from the dataframe to the coordinate array for spatial analysis.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating candidate links.

**Returns**

pd.DataFrame

Updated dataframe with resolved links for unexpected beginning bacteria.

unexpectedBeginning.**resolve_unexpected_beginning_bacteria**(*df*, *neighbors_df*, *neighbor_list_array*, *unexpected_beginning_bac_idx*, *unexpected_beginning_bac*, *link_probability*, *source_bac_idx*, *link_type*, *unexpected_begging_bac_time_step_bacteria*, *maintenance_cost*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_col*, *redundant_link_division_df*)

Resolves unexpected beginning bacteria by establishing new links to them and, if needed, modifying or removing existing links to source bacteria, based on the specified link type and probabilities.

**Parameters**

- **df** (*pandas.DataFrame*) – DataFrame containing bacterial life history and tracking information.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing neighboring relationships between bacteria, including their image and object numbers.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix representing neighbor connections between bacteria.
- **unexpected_beginning_bac_idx** (*int*) – Index of the unexpected beginning bacterium being resolved.

- **unexpected_beginning_bac** (*pandas.Series*) – Series containing data for the unexpected beginning bacterium.
- **link_probability** (*float*) – Probability associated with the proposed link between the source and unexpected beginning bacterium. A valid link is established if *1 - link_probability > 0.5*.
- **source_bac_idx** (*int*) – Index of the source bacterium being linked to the unexpected beginning bacterium.
- **link_type** (*str*) –

  **Type of link being evaluated:**

  - 'division': Division link (source bacterium divides into two daughters, including the unexpected beginning bacterium).
  - 'continuity': Continuity link (unexpected beginning bacterium continues the life history of the source bacterium).
- **unexpected_begging_bac_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria present in the time step of the unexpected beginning bacterium.
- **maintenance_cost** (*pandas.DataFrame*) – Cost DataFrame representing the cost of maintaining existing links for source bacteria.
- **parent_image_number_col** (*str*) – Column name in the DataFrame for parent image numbers.
- **parent_object_number_col** (*str*) – Column name in the DataFrame for parent object numbers.
- **center_coord_col** (*dict*) – Dictionary containing column names for x and y center coordinates.
- **redundant_link_division_df** (*pandas.DataFrame*) – DataFrame containing redundant links between candidate source bacteria and their current targets. These links are removed to allow the unexpected beginning bacterium to link to the candidate bacterium.

**Returns**

pandas.DataFrame:

Updated DataFrame with resolved links for the unexpected beginning bacterium.

### 1.1.17 Module: unExpectedEnd

unExpectedEnd.**handle_unexpected_end_bacteria**(*df*, *neighbors_df*, *neighbor_matrix*, *interval_time*, *doubling_time*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *division_vs_continuity_model*, *continuity_links_model*, *division_links_model*, *color_array*, *coordinate_array*)

Handle unexpected end bacteria by finding and validating candidate targets, and creating new links.

This function processes bacteria flagged with *unexpected_end* by identifying candidate target bacteria in the next frame. It validates potential links using statistical conditions, biological plausibility, and machine learning models. If validated, it creates new links to resolve tracking gaps.

The function handles two scenarios: - **continuity**: The bacterium's life history continues to the next frame. - **division**: The bacterium divides, linking to two daughter bacteria.

**Parameters**

- **df** (*pd.DataFrame*) – Dataframe containing bacterial measured bacterial features.
- **neighbors_df** (*pd.DataFrame*) – Dataframe containing neighbor relationship information for bacteria.
- **neighbor_matrix** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **interval_time** (*float*) – Time interval between frames in the tracking data.
- **doubling_time** (*float*) – Estimated doubling time for bacteria, used to determine plausible division events.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centers. Example: {'x': 'Center_X', 'y': 'Center_Y'}
- **division_vs_continuity_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria.
- **continuity_links_model** (*sklearn.Model*) – Machine learning model used to evaluate candidate links for continuity events.
- **division_links_model** (*sklearn.Model*) – Machine learning model used to validate division links.
- **color_array** (*np.ndarray*) – Array representing the colors of objects in the tracking data. This is used for mapping bacteria from the dataframe to the coordinate array for spatial analysis.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating candidate links.

**Returns**

pd.DataFrame

Updated dataframe with resolved links for unexpected end bacteria.

unExpectedEnd.**resolve_unexpected_end_bacteria**(*df*, *neighbors_df*, *neighbor_matrix*, *link_type*, *unexpected_end_bac_idx*, *target_bac_idx*, *target_bac*, *daughter_bac_idx*, *daughter_bac*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *next_frame_bacteria*, *target_link_probability*, *daughter_link_probability*)

Resolve unexpected end bacteria by creating new tracking links.

This function handles unexpected end bacteria by creating new tracking links to the next frame. It supports two types of links: - *'continuity'*: The source bacterium's life history continues to the target bacterium in the next frame. - *'division'*: The source bacterium divides, creating links to both a target bacterium and a daughter bacterium.

The function uses probabilities from an ML model to validate the links and removes previous links if necessary.

**Parameters**

- **df** (*pd.DataFrame*) – Dataframe containing bacterial tracking data, including IDs, life histories, and coordinates.

- **neighbors_df** (*pd.DataFrame*) – Dataframe containing neighbor relationship information for bacteria.
- **neighbor_matrix** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **link_type** (*str*) – Type of link to establish: - *'continuity'*: Continuity link to the target bacterium in the next frame. - *'division'*: Division link to the target bacterium and a daughter bacterium in the next frame.
- **unexpected_end_bac_idx** (*int*) – Index of the unexpected end bacterium (source bacterium).
- **target_bac_idx** (*int*) – Index of the target bacterium in the next frame.
- **target_bac** (*pd.Series*) – Data for the target bacterium in the next frame.
- **daughter_bac_idx** (*int*) – Index of the daughter bacterium, used for division links.
- **daughter_bac** (*pd.Series*) – Data for the daughter bacterium, used for division links.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for x and y coordinates of bacterial centers. Example: {'x': 'Center_X', 'y': 'Center_Y'}
- **next_frame_bacteria** (*pd.DataFrame*) – Dataframe containing bacteria in the next frame relative to the source bacterium.
- **target_link_probability** (*float*) – Probability value from the ML model for the link to the target bacterium. A valid link is considered if *1 - target_link_probability > 0.5*.
- **daughter_link_probability** (*float*) – Probability value from the ML model for the link to the daughter bacterium (division). A valid link is considered if *1 - daughter_link_probability > 0.5*.

**Returns**

pd.DataFrame

Updated dataframe with new links added and previous links removed.

### 1.1.18 Module: restoringTrackingLinks

restoringTrackingLinks.**add_tracking_link**(*df*, *neighbors_df*, *neighbor_list_array*, *source_bac_idx*, *target_bac_idx*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *all_bac_in_target_bac_time_step*, *prob_val*)

Adds a new tracking link between a source bacterium and a target bacterium if conditions are met.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame containing tracking information for bacteria.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame representing the neighbor relationships between bacteria.
- **neighbor_list_array** (*scipy.sparse.csr_matrix*) – Sparse matrix representing neighborhood connectivity for all bacteria.

- **source_bac_idx** (*int*) – Index of the source bacterium in the current DataFrame.
- **target_bac_idx** (*int*) – Index of the target bacterium in the current DataFrame.
- **parent_image_number_col** (*str*) – Column name for the parent bacterium's image number.
- **parent_object_number_col** (*str*) – Column name for the parent bacterium's object number.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for the x and y centroid coordinates, e.g., *{'x': 'Center_X', 'y': 'Center_Y'}*.
- **all_bac_in_target_bac_time_step** (*pandas.DataFrame*) – Subset of the DataFrame containing all bacteria in the target bacterium's time step.
- **prob_val** (*float*) – Probability value indicating the likelihood of the new link being valid. A lower probability (*1 - prob_val*) must exceed a threshold (0.5) for the link to be considered.

**Returns:**

**pandas.DataFrame:**
The updated DataFrame with the new tracking link added if the conditions are met.

restoringTrackingLinks.**restore_tracking_links**(*df*, *neighbors_df*, *neighbor_list_array*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *df_raw_before_rpl_errors*, *continuity_links_model*, *division_links_model*, *coordinate_array*)

Attempts to restore previously removed tracking links for bacteria by re-evaluating conditions and costs. This function focuses on resolving links for unexpected beginning bacteria while maintaining tracking integrity.

**Process:**

- Identifies bacteria with *unexpected_beginning* status.
- Cross-checks these bacteria against the raw tracking data to find potential links that were removed.
- **Evaluates potential links based on:**
  - Continuity (single-link bacteria).
  - Division (bacteria dividing into daughters).
- Applies machine learning models to calculate costs for restoring links and uses an optimization algorithm (Hungarian method) to finalize the link restoration.
- Updates the DataFrame with restored links if conditions are satisfied.

**Key Features:**

- **Continuity Links:**
  - Checks whether the bacterium in question can continue its life history based on its source link.
- **Division Links:**
  - Ensures division links meet criteria, such as the daughter-to-mother length ratio being below a threshold.
- **Optimization:**
  - Uses a cost matrix and optimization algorithm to assign links with minimal cost.

**Parameters**

- **df** (*pandas.DataFrame*) – The current DataFrame containing bacteria tracking data.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame representing the neighbor relationships between bacteria.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix representing neighborhood connectivity for all bacteria.
- **parent_image_number_col** (*str*) – Column name for the parent bacterium's image number.
- **parent_object_number_col** (*str*) – Column name for the parent bacterium's object number.
- **center_coord_cols** (*dict*) – Dictionary containing the column names for x and y centroid coordinates, e.g., *{'x': 'Center_X', 'y': 'Center_Y'}*.
- **df_raw_before_rpl_errors** (*pandas.DataFrame*) – Original tracking data before resolving RPL errors, used to verify previously existing links.
- **continuity_links_model** (*sklearn.Model*) – Machine learning model to evaluate the validity of non-divided bacteria continuity links.
- **division_links_model** (*sklearn.Model*) – Machine learning model to evaluate division-based links for bacteria.
- **coordinate_array** (*numpy.ndarray*) – Array containing precomputed spatial coordinates for bacteria.

**Returns:**

**pandas.DataFrame:**
The updated DataFrame with restored tracking links where applicable.

### 1.1.19 Module: candidateBacteriaFeatureSpace

candidateBacteriaFeatureSpace.**find_candidates_for_unexpected_beginning**(*neighbors_df*, *unexpected_begging_bac*, *unexpected_beginning_bac_time_step_bacteria*, *previous_time_step_bacteria*, *center_coord_cols*, *color_array*, *coordinate_array*)

Finds candidate bacteria from the previous time step to link with unexpected beginning bacteria.

**Parameters**

- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing neighboring relationships between bacteria, including their image and object numbers.
- **unexpected_begging_bac** (*pandas.DataFrame*) – DataFrame containing information about unexpected beginning bacteria.
- **unexpected_beginning_bac_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria in the time step of the unexpected beginning bacteria.
- **previous_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria from the previous time step.

- **center_coord_cols** (*dict*) – Dictionary mapping x and y center coordinates to their respective column names.
- **color_array** (*numpy.ndarray*) – Array storing color-coded information for bacterial objects.
- **coordinate_array** (*csr_matrix*) – Sparse boolean array representing encoded spatial coordinates of bacteria.

**Returns**

**tuple:**

- **final_candidate_bacteria_info** (*pandas.DataFrame*): Subset of bacteria from the previous time step that can potentially link to unexpected beginning bacteria and are selected for further analysis.
- **final_candidate_bacteria** (*pandas.DataFrame*): DataFrame containing detailed information about candidate bacteria and their neighbors, combined with unexpected beginning bacteria data.

candidateBacteriaFeatureSpace.**find_candidates_for_unexpected_end**(*neighbors_df*, *unexpected_end_bac*, *unexpected_end_bac_time_step_bacteria*, *next_time_step_bacteria*, *center_coord_cols*, *color_array*, *coordinate_array*)

Finds candidate bacteria from the next time step to link with unexpected end bacteria.

**Parameters**

- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing neighboring relationships between bacteria, including their image and object numbers.
- **unexpected_end_bac** (*pandas.DataFrame*) – DataFrame containing information about unexpected end bacteria.
- **unexpected_end_bac_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria in the time step of the unexpected end bacteria.
- **next_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria from the next time step.
- **center_coord_cols** (*dict*) – Dictionary mapping x and y center coordinates to their respective column names.
- **color_array** (*numpy.ndarray*) – Array storing color-coded information for bacterial objects.
- **coordinate_array** (*csr_matrix*) – Sparse boolean array representing encoded spatial coordinates of bacteria.

**Returns**

**tuple:**

- **final_candidate_bacteria_info** (*pandas.DataFrame*): Subset of bacteria from the previous time step that can potentially link to unexpected end bacteria and are selected for further analysis.
- **final_candidate_bacteria** (*pandas.DataFrame*): DataFrame containing detailed information about candidate bacteria and their neighbors, combined with unexpected end bacteria data.

### 1.1.20 Module: calculateInitialCostTerms

calculateInitialCostTerms.**calc_distance_and_overlap_matrices**(*source_time_step_df*, *sel_source_bacteria*, *bacteria_in_target_time_step*, *sel_target_bacteria*, *center_coord_cols*, *color_array*, *daughter_flag=False*, *maintain=False*, *coordinate_array=None*)

Constructs initial distance and overlap matrices for tracking bacteria across consecutive time steps.

**Distance Calculation:**

- **When *maintain=True*:**
  - **Calculates distances to maintain both division and continuity links:**
    - Division links: Evaluates spatial relationships between parent and daughter bacteria.
    - Continuity links: Tracks life history continuity of the same bacterium across time steps.
- **When *maintain=False*:**
  - **Computes overlaps and distances between bacteria in the source and target time steps:**
    - Centroid distances (*center_distance*).
    - **Endpoint distances:**
      - Between first endpoints (*endpoint1_1_distance*).
      - Between second endpoints (*endpoint2_2_distance*).
    - **Additional distances for daughter tracking:**
      - Between mismatched endpoints (*endpoint1_2_distance* and *endpoint2_1_distance*).
      - Between centroids and endpoints (*center_endpoint1_distance* and *center_endpoint2_distance*).
  - Determines the minimum distance for each source-target pair and pivots the results into a distance matrix.

**Parameters**

- **source_time_step_df** (*pandas.DataFrame*) – DataFrame containing tracking data for bacteria in the source time step.
- **sel_source_bacteria** (*pandas.DataFrame*) – Subset of source bacteria selected for analysis.
- **bacteria_in_target_time_step** (*pandas.DataFrame*) – DataFrame containing tracking data for bacteria in the target objects time step.
- **sel_target_bacteria** (*pandas.DataFrame*) – Subset of target bacteria selected for analysis.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **color_array** (*np.ndarray*) – Array representing the colors of objects in the tracking data. This is used for mapping bacteria from the dataframe to the coordinate array for spatial analysis.

- **daughter_flag** (*bool*) – Whether to include additional distances related to parent-daughter relationships.
- **maintain** (*bool*) – If *True*, computes distance terms for maintaining division and continuity links.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating candidate links.

**Returns**

**tuple: (overlap_df, distance_df)**

- *overlap_df*: Overlap matrix indicating the existence of relationships between bacteria.
- *distance_df*: Distance matrix with minimum distances for source-target pairs.

calculateInitialCostTerms.**calc_distance_term_for_maintain_continuity_links**(*continuity_df*, *center_coord_cols*)

Calculates a distance matrix to maintain continuity links.

**Distance Calculation:**

**For each source-target pair, the following distances are calculated:**

- The Euclidean distance between the centroids of the source object and the target object.
- The Euclidean distance between the first endpoints of the source and target objects (located on the same side).
- The Euclidean distance between the second endpoints of the source and target objects (located on the opposite side).

For each source-target pair, the smallest of these three distances is selected as the min_distance.

**Parameters**

- **continuity_df** (*pandas.DataFrame*) – A DataFrame containing information about objects in consecutive time steps. The DataFrame must include centroid and endpoint coordinates for calculating distances between source and target objects.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of object centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

**Returns**

**pandas.DataFrame: A pivoted distance matrix where:**

- Rows correspond to source objects (*index_1*).
- Columns correspond to target objects (*index_2*).
- Values represent the minimum distance between a source and target.

calculateInitialCostTerms.**calc_distance_term_for_maintain_division**(*division_df*, *center_coord_cols*)

Calculates a distance matrix to maintain relationship between parent and daughter objects in a division event.

**Distance Calculation:**

**For each parent-daughter pair, the following distances are calculated:**

- The Euclidean distance between the centroids of the parent and daughter objects (*center_distance*).

- The Euclidean distance between the first endpoints of the parent and daughter objects (*endpoint1_1_distance*).
- The Euclidean distance between the second endpoints of the parent and daughter objects (*endpoint2_2_distance*).
- The Euclidean distance between the first endpoint of the parent and the second endpoint of the daughter (*endpoint12_distance*).
- The Euclidean distance between the second endpoint of the parent and the first endpoint of the daughter (*endpoint21_distance*).
- The Euclidean distance between the centroid of the parent and the first endpoint of the daughter (*center_endpoint1_distance*).
- The Euclidean distance between the centroid of the parent and the second endpoint of the daughter (*center_endpoint2_distance*).

For each parent-daughter pair, the smallest of these distances is selected as the *min_distance*.

**Parameters**

- **division_df** (*pandas.DataFrame*) – A DataFrame containing information about parent-daughter relationships in division events. The DataFrame must include centroid and endpoint coordinates for both parent and daughter objects.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of object centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

**Returns**

**pandas.DataFrame: A pivoted distance matrix where:**

- Rows correspond to parent objects (*index_parent*).
- Columns correspond to daughter objects (*index_daughter*).
- Values represent the minimum distance between a parent and daughter.

### 1.1.21 Module: calculateCreateLinkCost

calculateCreateLinkCost.**calc_continuity_link_cost**(*df*, *neighbors_df*, *neighbor_list_array*, *source_bac_with_can_target*, *center_coord_cols*, *col_source*, *col_target*, *parent_image_number_col*, *parent_object_number_col*, *continuity_links_model*, *division_vs_continuity_model*, *maintain_exist_link_cost_df*, *check_maintenance_for='target'*, *coordinate_array=None*)

Calculates the cost of establishing continuity link between a source bacterium and its potential target bacterium. This function evaluates continuity links by analyzing spatial, geometric, and neighborhood-based features. It determines the likelihood that a bacterium in one time step corresponds to the same bacterium in the next time step.

**Behavior**: - Calculates features like intersection-over-union (IoU), centroid distance, length ratios, and neighbor overlap. - Predicts continuity probabilities using a pre-trained model (*continuity_links_model*). - Compares continuity and division probabilities using a comparison model (*division_vs_continuity_model*). - Adjusts costs based on whether maintaining existing links is preferable.

**Parameters**

- **df** (*pandas.DataFrame*) – Full DataFrame containing bacterial features for the current time step.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame describing relationships between neighboring bacteria.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **source_bac_with_can_target** (*pandas.DataFrame*) – DataFrame containing the source bacteria and their candidate targets in the next time step.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **col_source** (*str*) – Suffix for source bacterium columns.
- **col_target** (*str*) – Suffix for target bacterium columns.
- **parent_image_number_col** (*str*) – Column name for the parent image number.
- **parent_object_number_col** (*str*) – Column name for the parent object number.
- **continuity_links_model** (*sklearn.Model*) – Machine learning model to predict continuity probabilities.
- **division_vs_continuity_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria.
- **maintain_exist_link_cost_df** (*pandas.DataFrame*) – Cost of maintaining current links between bacteria.
- **check_maintenance_for** (*str*) – Specifies whether maintenance costs are evaluated for the *target* or *source*. Default is *'target'*.
- **coordinate_array** (*csr_matrix*) – spatial coordinate matrix for neighborhood and overlap calculations.

**Returns**

pandas.DataFrame:

A cost matrix for potential continuity links, with rows as source bacteria and columns as their candidate targets.

calculateCreateLinkCost.**calc_continuity_link_cost_for_restoring_links**(*df*, *neighbors_df*, *neighbor_list_array*, *source_bac_with_can_target*, *center_coord_cols*, *col_source*, *col_target*, *parent_image_number_col*, *parent_object_number_col*, *continuity_links_model*, *coordinate_array=None*)

Calculates the cost of restoring continuity link between a source bacterium and its potential UB target bacterium. This function evaluates continuity links by analyzing spatial, geometric, and neighborhood-based features. It determines the likelihood that a bacterium in one time step corresponds to the same bacterium in the next time step.

**Behavior**: - Calculates features like intersection-over-union (IoU), centroid distance, length ratios, and neighbor overlap. - Predicts continuity probabilities using a pre-trained model (*continuity_links_model*).

**Parameters**

- **df** (*pandas.DataFrame*) – Full DataFrame containing bacterial features for the current time step.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame describing relationships between neighboring bacteria.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **source_bac_with_can_target** (*pandas.DataFrame*) – DataFrame containing the source bacteria and their candidate UB target in the next time step.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **col_source** (*str*) – Suffix for source bacterium columns.
- **col_target** (*str*) – Suffix for target bacterium columns.
- **parent_image_number_col** (*str*) – Column name for the parent image number.
- **parent_object_number_col** (*str*) – Column name for the parent object number.
- **continuity_links_model** (*sklearn.Model*) – Machine learning model to predict continuity probabilities.`.
- **coordinate_array** (*csr_matrix*) – spatial coordinate matrix for neighborhood and overlap calculations.

**Returns**

pandas.DataFrame:

A cost matrix for potential continuity links, with rows as source bacteria and columns as their candidate targets (UB bacteria).

calculateCreateLinkCost.**calc_division_link_cost**(*df*, *neighbors_df*, *neighbor_list_array*, *df_source_daughter*, *center_coord_cols*, *col_source*, *col_target*, *parent_image_number_col*, *parent_object_number_col*, *division_links_model*, *division_vs_continuity_model*, *maintain_exist_link_cost_df*, *check_maintenance_for='target'*, *coordinate_array=None*)

Calculates the cost of establishing division links between a source bacterium and its potential daughter bacteria. This function evaluates potential division links by calculating various spatial, geometric, and neighborhood-based features. It uses machine learning models to assess the likelihood of a division event based on these features.

**Behavior**: - Calculates features like intersection-over-union (IoU), centroid distance, length ratios, and neighbor overlap. - Predicts division probabilities using a pre-trained model (*division_links_model*). - Compares division and continuity probabilities using a comparison model (*division_vs_continuity_model*). - Adjusts costs based on whether maintaining existing links is preferable.

**Parameters**

- **df** (*pandas.DataFrame*) – Full DataFrame containing bacterial features for the current time step.

- **neighbors_df** (*pandas.DataFrame*) – DataFrame describing relationships between neighboring bacteria.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **df_source_daughter** (*pandas.DataFrame*) – DataFrame containing the source bacteria and their candidate daughters.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **col_source** (*str*) – Suffix for source bacterium columns.
- **col_target** (*str*) – Suffix for target bacterium (daughter) columns.
- **parent_image_number_col** (*str*) – Column name for the parent image number.
- **parent_object_number_col** (*str*) – Column name for the parent object number.
- **division_links_model** (*sklearn.Model*) – Machine learning model to predict division probabilities.
- **division_vs_continuity_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria.
- **maintain_exist_link_cost_df** (*pandas.DataFrame*) – Cost of maintaining current links between bacteria.
- **check_maintenance_for** (*str*) – Specifies whether maintenance costs are evaluated for the *target* or *source*. Default is *'target'*.
- **coordinate_array** (*csr_matrix*) – spatial coordinate matrix for neighborhood and overlap calculations.

**Returns**

pandas.DataFrame:

A cost matrix for potential division links, with rows as source bacteria and columns as their candidate daughters.

calculateCreateLinkCost.**calc_division_link_cost_for_restoring_links**(*df*, *neighbors_df*, *neighbor_list_array*, *df_source_daughter*, *center_coord_cols*, *col_source*, *col_target*, *parent_image_number_col*, *parent_object_number_col*, *division_links_model*, *coordinate_array*)

Calculates the cost of restoring division links between a source bacterium and its potential daughter bacteria (UB bacteria). This function evaluates potential division links by calculating various spatial, geometric, and neighborhood-based features. It uses machine learning model to assess the likelihood of a division event based on these features.

**Behavior**: - Calculates features like intersection-over-union (IoU), centroid distance, length ratios, and neighbor overlap. - Predicts division probabilities using a pre-trained model (*division_links_model*).

**Parameters**

- **df** (*pandas.DataFrame*) – Full DataFrame containing bacterial features for the current time step.

- **neighbors_df** (*pandas.DataFrame*) – DataFrame describing relationships between neighboring bacteria.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix representing neighbor relationships.
- **df_source_daughter** (*pandas.DataFrame*) – DataFrame containing the source bacteria and their candidate daughters.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **col_source** (*str*) – Suffix for source bacterium columns.
- **col_target** (*str*) – Suffix for target bacterium (daughter) columns.
- **parent_image_number_col** (*str*) – Column name for the parent image number.
- **parent_object_number_col** (*str*) – Column name for the parent object number.
- **division_links_model** (*sklearn.Model*) – Machine learning model to predict division probabilities.
- **coordinate_array** (*csr_matrix*) – spatial coordinate matrix for neighborhood and overlap calculations.

**Returns**

pandas.DataFrame:

A cost matrix for potential division links, with rows as source bacteria and columns as their candidate daughters.

calculateCreateLinkCost.**optimize_assignment_using_hungarian**(*cost_df*)

Optimizes the assignment of rows to columns in a cost matrix using the Hungarian algorithm.

**Parameters**

**cost_df** (*pandas.DataFrame*) – A cost matrix represented as a DataFrame

**Returns**

pandas.DataFrame

A DataFrame containing the optimized assignments and their associated costs:

### 1.1.22 Module: calculateMaintainingLinkCost

calculateMaintainingLinkCost.**calc_maintain_exist_link_cost**(*sel_source_bacteria_info*, *sel_target_bacteria_info*, *center_coord_cols*, *divided_vs_non_divided_model*, *coordinate_array*)

Calculates the cost of maintaining existing links between bacteria in consecutive time steps. This function calculates features such as IOU, distance, neighbor relationships, and motion alignment for each source-target pair. These features are used as inputs to a predictive model to calculate the cost of maintaining each link.

**Parameters**

- **sel_source_bacteria_info** (*pandas.DataFrame*) – Subset of bacteria in the source time step selected for analysis.
- **sel_target_bacteria_info** (*pandas.DataFrame*) – Subset of bacteria in the target time step selected for analysis.

- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **divided_vs_non_divided_model** (*sklearn.Model*) – Machine learning model used to evaluate the probability of division vs. continuity links.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating spatial relationships between bacteria.

**Returns**

pandas.DataFrame:

A cost matrix for maintaining existing links between bacteria. Rows correspond to source bacteria, columns correspond to target bacteria, and values represent the cost of maintaining each link.

### 1.1.23 Module: calculateCreateLinkCostMCC

calculateCreateLinkCostMCC.**create_continuity_link_cost**(*df*, *neighbors_df*, *neighbor_list_array*, *mcc_target_neighbors_features*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *non_divided_bac_model*, *divided_vs_non_divided_model*, *maintain_exist_link_cost_df*, *coordinate_array*)

Calculate the cost of creating a new continuity link between a source bacterium with a missing connectivity (MCC link) and candidate target bacteria, and replacing the MCC link with this new link.

**Parameters**

- **df** (*pandas.DataFrame*) – Main DataFrame containing tracking data for bacteria.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing neighboring relationships between bacteria.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix representing neighboring relationships for fast lookups.
- **mcc_target_neighbors_features** (*pandas.DataFrame*) – DataFrame containing bacterial measured bacterial features of MCC target neighbors (candidate bacteria)
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates.
- **parent_image_number_col** (*str*) – Column name representing parent image numbers.
- **parent_object_number_col** (*str*) – Column name representing parent object numbers.
- **non_divided_bac_model** (*sklearn.Model*) – Machine learning model used to evaluate candidate links for non-divided bacteria.
- **divided_vs_non_divided_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria.
- **maintain_exist_link_cost_df** (*pandas.DataFrame*) – DataFrame containing maintenance costs for existing links.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating candidate links.

**Returns**

pandas.DataFrame:

DataFrame containing the cost of replacing MCC links with new continuity links.

calculateCreateLinkCostMCC.**division_detection_cost**(*df*, *neighbors_df*, *neighbor_list_array*, *candidate_source_daughter_df*, *min_life_history_of_bacteria_time_step*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *divided_bac_model*, *divided_vs_non_divided_model*, *maintain_exist_link_cost_df*, *coordinate_array*)

Evaluates the cost of maintaining a missing connectivity link (MCC link) while adding a division link between the source bacterium and candidate daughter bacteria.

**Workflow:**

**•Checks if the source bacterium can support a valid division event by:**

- Ensuring biological feasibility based on length ratios.
- Evaluating statistical boundaries for daughter-to-mother length relationships.
- Considering the age and unexpected beginning status of the source bacterium.

- Identifies candidate target bacteria that meet division conditions.
- Calculates the cost of adding division links using the *calc_division_link_cost* function.

**Parameters**

- **df** (*pandas.DataFrame*) – Main DataFrame containing tracking data for bacteria.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing neighboring relationships between bacteria.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix representing neighboring relationships for fast lookups.
- **candidate_source_daughter_df** (*pandas.DataFrame*) – DataFrame containing features of source bacteria and their potential daughter bacteria.
- **min_life_history_of_bacteria_time_step** (*int*) – Minimum life history time steps required for a bacterium to be considered valid.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates.
- **parent_image_number_col** (*str*) – Column name representing parent image numbers.
- **parent_object_number_col** (*str*) – Column name representing parent object numbers.
- **divided_bac_model** (*sklearn.Model*) – Machine learning model used to validate division links.
- **divided_vs_non_divided_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria.
- **maintain_exist_link_cost_df** (*pandas.DataFrame*) – DataFrame containing maintenance costs for existing links.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating candidate links.

**Returns**

pandas.DataFrame:

DataFrame containing the cost of maintaining MCC links while adding division links.

### 1.1.24 Module: calculateCreateLinkCostUB

calculateCreateLinkCostUB.**compute_create_link_to_unexpected_beginning_cost**(*df*, *unexpected_begging_bac_time_step_bact*, *unexpected_begging_bacteria*, *previous_time_step_bacteria*, *neighbors_df*, *neighbor_list_array*, *min_life_history_time_step*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*, *divided_vs_non_divided_model*, *non_divided_model*, *division_model*, *color_array*, *coordinate_array*)

Computes the cost of linking unexpected beginning bacteria to bacteria from a previous time step, evaluating continuity link cost, division costs, and maintenance the current links of candidate bacteria costs.

**Parameters**

- **df** (*pandas.DataFrame*) – DataFrame containing bacterial measured bacterial features.
- **unexpected_begging_bac_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria present in the time step of unexpected beginnings.
- **unexpected_begging_bacteria** (*pandas.DataFrame*) – DataFrame containing unexpected beginning bacteria.
- **previous_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria from the previous time step.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing neighboring relationships between bacteria, including their image and object numbers.
- **neighbor_list_array** (*lil_matrix*) – Array storing neighbor connections for bacteria.
- **min_life_history_time_step** (*int*) – Minimum number of life history frames required for a bacterium to be considered
- **parent_image_number_col** (*str*) – Column name in the DataFrame for parent image numbers.
- **parent_object_number_col** (*str*) – Column name in the DataFrame for parent object numbers.

- **center_coord_cols** (*dict*) – Dictionary containing column names for x and y center coordinates.
- **divided_vs_non_divided_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria
- **non_divided_model** (*sklearn.Model*) – Machine learning model used to evaluate candidate links for continuity events.
- **division_model** (*sklearn.Model*) – Machine learning model used to validate division links.
- **color_array** (*np.ndarray*) – Array representing the colors of objects in the tracking data. This is used for mapping bacteria from the dataframe to the coordinate array for spatial analysis.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating candidate links.

**Returns**

**tuple:**

- **continuity_link_cost_df** (*pandas.DataFrame*): Cost DataFrame for creating links to continue the life histories of bacteria across time steps.
- **division_cost_df** (*pandas.DataFrame*): Cost DataFrame for creating division links.
- **redundant_link_division_df** (*pandas.DataFrame*): DataFrame containing redundant links between candidate bacteria and one of their current daughter bacteria. These links are flagged for removal to allow unexpected beginning bacteria to establish new link with the candidate bacteria.
- **maintain_exist_link_cost_df** (*pandas.DataFrame*): Cost DataFrame for maintaining the current links of candidate bacteria

### 1.1.25 Module: evaluateDivisionLinkForUB

evaluateDivisionLinkForUB.**evaluate_division_link_feasibility**(*df_source_daughter_cost*, *division_cost_dict*, *source_bac_ndx*, *source_bac*, *target_bac_ndx*, *target_bac*, *source_bac_next_time_step*, *sum_daughter_len_to_mother_ratio_boundary*, *max_daughter_len_to_mother_ratio_boundary*, *min_life_history_of_bacteria*)

Evaluates the feasibility of replacing a continuity link with a division link between a source bacterium and a target bacterium by checking specific biological and statistical conditions. Updates the division cost dictionary with the evaluated cost.

**Logic:**

- Computes the length ratios between the source bacterium and its potential daughters to validate whether a division event is biologically plausible.
- Compares these ratios against statistically derived boundaries to ensure the division adheres to expected patterns.
- Evaluates the age of the source bacterium.
- If all conditions pass, retrieves the precomputed division cost from the input DataFrame.

- If any condition fails, assigns a high cost (*1*), indicating that the link is highly improbable.

**Parameters**

- **df_source_daughter_cost** (*pandas.DataFrame*) – DataFrame containing the precomputed division costs between source and daughter bacteria.
- **division_cost_dict** (*dict*) – Dictionary storing division costs for source-target bacterium pairs.
- **source_bac_ndx** (*int*) – Index of the source bacterium in the DataFrame.
- **source_bac** (*pandas.Series*) – Attributes of the source bacterium.
- **target_bac_ndx** (*int*) – Index of the target bacterium in the DataFrame.
- **target_bac** (*pandas.Series*) – Attributes of the target bacterium.
- **source_bac_next_time_step** (*pandas.DataFrame*) – Attributes of the source bacterium in the next time step.
- **sum_daughter_len_to_mother_ratio_boundary** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation) for the sum of daughter-to-mother length ratios.
- **max_daughter_len_to_mother_ratio_boundary** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation) for the maximum daughter-to-mother length ratio.
- **min_life_history_of_bacteria** (*int*) – Minimum required life history frames for a valid bacterium.

**Returns**

dict:

Updated *division_cost_dict* with the evaluated division cost for the source-target pair.

evaluateDivisionLinkForUB.**evaluate_new_division_link**(*maintenance_cost_df*, *existing_daughters_idx*, *target_bac_ndx*, *source_bac_ndx*, *new_daughter_prob*, *redundant_link_dict_division*)

Evaluates whether an existing daughter link should be replaced with a new link to minimize the overall cost.

**Logic:**

- Compares the probabilities associated with maintaining the existing daughter links and the new link.
- Identifies the "worst" existing daughter (the one with the minimum probability).
- If the "worst" daughter is not the target bacterium, updates the cost to add the new link and marks the redundant link in the *redundant_link_dict_division*.
- If the target bacterium is already the "worst" link, assigns a high cost (*1*) to indicate an improbable replacement.

**Parameters**

- **maintenance_cost_df** (*pandas.DataFrame*) – DataFrame containing the maintenance costs for existing daughter links.
- **existing_daughters_idx** (*list*) – Indices of the existing daughters.
- **target_bac_ndx** (*int*) – Index of the target bacterium (the new daughter candidate).
- **source_bac_ndx** (*int*) – Index of the source bacterium.

- **new_daughter_prob** (*float*) – Probability of establishing a new daughter link with the target bacterium.
- **redundant_link_dict_division** (*dict*) – Dictionary storing redundant links for division events.

**Returns**

**tuple:**

- float: Cost of adding the new daughter link.
- dict: Updated *redundant_link_dict_division* with the identified redundant link.

evaluateDivisionLinkForUB.**evaluate_replacement_daughter_link**(*maintenance_cost_df*, *source_bac_ndx*, *source_bac*, *target_bac_ndx*, *target_bac*, *new_daughter_cost*, *source_bac_daughters*, *max_daughter_len_to_mother_ratio_boundary*, *sum_daughter_len_to_mother_ratio_boundary*, *redundant_link_dict_division*)

Evaluates whether an existing division link can be replaced with a new link to minimize the overall cost, while ensuring the replacement adheres to biological constraints.

**Logic:**

- Computes the new length ratios for the potential division link, incorporating the source bacterium, its existing daughters, and the target bacterium.
- Compares the new ratios against statistical boundaries (*max_daughter_len_to_mother_ratio_boundary* and *sum_daughter_len_to_mother_ratio_boundary*) to ensure the new division link is biologically plausible.
- If only one daughter satisfies the constraints, compares the probability of the target bacterium against the least probable existing daughter link. If the new link is more probable, it replaces the least probable link.
- If multiple daughters satisfy the constraints, delegates the decision to *evaluate_replacement_daughter_link* to identify the best replacement.
- If no daughters meet the constraints, assigns a high cost (*1*), indicating the link is improbable.

**Parameters**

- **maintenance_cost_df** (*pandas.DataFrame*) – DataFrame containing the maintenance costs for existing daughter links.
- **source_bac_ndx** (*int*) – Index of the source bacterium in the DataFrame.
- **source_bac** (*pandas.Series*) – Attributes of the source bacterium.
- **target_bac_ndx** (*int*) – Index of the target bacterium in the DataFrame.
- **target_bac** (*pandas.Series*) – Attributes of the target bacterium.
- **new_daughter_cost** (*float*) – Cost of establishing a new daughter link with the target bacterium.
- **source_bac_daughters** (*pandas.DataFrame*) – DataFrame containing attributes of the source bacterium's existing daughters.

- **max_daughter_len_to_mother_ratio_boundary** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation) for the maximum daughter-to-mother length ratio.
- **sum_daughter_len_to_mother_ratio_boundary** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation) for the sum of daughter-to-mother length ratios.
- **redundant_link_dict_division** (*dict*) – Dictionary storing redundant links for division events.

**Returns**

**tuple:**

- float: Cost of adding the new daughter link.
- dict: Updated *redundant_link_dict_division* with the identified redundant link.

evaluateDivisionLinkForUB.**handle_two_daughter_links**(*division_cost_df*, *maintenance_cost_df*, *division_cost_dict*, *source_bac_ndx*, *source_bac*, *target_bac_ndx*, *target_bac*, *redundant_link_dict_division*, *source_bac_daughters*, *max_daughter_len_to_mother_ratio_boundary*, *sum_daughter_len_to_mother_ratio_boundary*)

Handles scenarios where a source bacterium already has two daughter links and evaluates the feasibility of replacing one of the existing links with a new division link to minimize the overall cost.

**Logic:**

- Retrieves the cost of adding a new daughter link from the *division_cost_df*.
- Delegates the evaluation of replacing an existing link to *evaluate_replacement_daughter_link*, which ensures biological constraints are satisfied and identifies the most redundant link if applicable.
- Updates the *division_cost_dict* with the resulting cost for the new link.
- Updates the *redundant_link_dict_division* to mark any redundant links that were replaced.

**Parameters**

- **division_cost_df** (*pandas.DataFrame*) – DataFrame containing precomputed division costs for source-target links.
- **maintenance_cost_df** (*pandas.DataFrame*) – DataFrame containing maintenance costs for existing daughter links.
- **division_cost_dict** (*dict*) – Dictionary storing division costs for source-target bacterium pairs.
- **source_bac_ndx** (*int*) – Index of the source bacterium in the DataFrame.
- **source_bac** (*pandas.Series*) – Attributes of the source bacterium.
- **target_bac_ndx** (*int*) – Index of the target bacterium in the DataFrame.
- **target_bac** (*pandas.Series*) – Attributes of the target bacterium.
- **redundant_link_dict_division** (*dict*) – Dictionary storing redundant links for division events.
- **source_bac_daughters** (*pandas.DataFrame*) – DataFrame containing attributes of the source bacterium's existing daughters.

- **max_daughter_len_to_mother_ratio_boundary** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation) for the maximum daughter-to-mother length ratio.
- **sum_daughter_len_to_mother_ratio_boundary** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation) for the sum of daughter-to-mother length ratios.

**Returns**

**tuple:**

- dict: Updated *division_cost_dict* with the cost for the new link.
- dict: Updated *redundant_link_dict_division* with any redundant links replaced.

evaluateDivisionLinkForUB.**update_division_cost_and_redundant_links**(*division_cost_df*, *maintenance_cost_df*, *division_cost_dict*, *source_bac_ndx*, *source_bac*, *target_bac_ndx*, *target_bac*, *redundant_link_dict_division*, *sum_daughter_len_to_mother_ratio_boundary*, *max_daughter_len_to_mother_ratio_boundary*, *min_life_history_of_bacteria*, *source_bac_next_time_step*, *source_bac_daughters*)

**Evaluates two scenarios for link feasibility:**

1. **Single Link Scenario**: The source bacterium has one existing link. The function evaluates the feasibility of adding a new link from the target bacterium to the source bacterium.
2. **Double Link Scenario: The source bacterium has two existing links (daughter links).** The function evaluates the feasibility of removing one of the existing links and replacing it with a new link from the target bacterium.

**Parameters**

- **division_cost_df** (*pandas.DataFrame*) – DataFrame containing costs for potential division links.
- **maintenance_cost_df** (*pandas.DataFrame*) – DataFrame containing costs for maintaining current links of candidate source bacteria.
- **division_cost_dict** (*dict*) – Dictionary storing updated division costs for all bacteria.
- **source_bac_ndx** (*int*) – Index of the source bacterium being evaluated.
- **source_bac** (*pandas.Series*) – Data for the source bacterium being evaluated.
- **target_bac_ndx** (*int*) – Index of the target bacterium being evaluated.
- **target_bac** (*pandas.Series*) – Data for the target bacterium being evaluated.
- **redundant_link_dict_division** (*dict*) – Dictionary storing redundant links for division relationships.
- **sum_daughter_len_to_mother_ratio_boundary** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation) for the sum of daughter-to-mother length ratios.
- **max_daughter_len_to_mother_ratio_boundary** (*dict*) – Dictionary containing 'avg' (average) and 'std' (standard deviation) for the maximum daughter-to-mother length ratio.

- **min_life_history_of_bacteria** (*int*) – Minimum number of life history frames required for a bacterium to be considered valid.
- **source_bac_next_time_step** (*pandas.DataFrame*) – DataFrame containing the next time step's candidate for the source bacterium.
- **source_bac_daughters** (*pandas.DataFrame*) – DataFrame containing the daughters of the source bacterium.

**Returns**

**tuple:**

- **division_cost_dict** (*dict*): Updated dictionary of division costs for source-target pairs.
- **redundant_link_dict_division** (*dict*): Updated dictionary of redundant links.

### 1.1.26 Module: calculateCreateLinkCostUE

calculateCreateLinkCostUE.**compute_create_link_from_unexpected_end_cost**(*df*, *neighbors_df*, *neighbor_list_array*, *unexpected_end_bac_time_step_bacteria*, *unexpected_end_bacteria*, *next_time_step_bacteria*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*, *min_life_history_time_step*, *divided_vs_non_divided_model*, *non_divided_bac_model*, *division_model*, *color_array*, *coordinate_array*)

Computes the cost of linking unexpected end bacteria to bacteria from a next time step, evaluating continuity link cost, division costs, and maintenance the current links of candidate bacteria costs.

**Parameters**

- **df** (*pandas.DataFrame*) – DataFrame containing bacterial measured bacterial features.
- **unexpected_end_bac_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria present in the time step of unexpected end.
- **unexpected_end_bacteria** (*pandas.DataFrame*) – DataFrame containing unexpected end bacteria.
- **next_time_step_bacteria** (*pandas.DataFrame*) – DataFrame containing all bacteria from the previous time step.
- **neighbors_df** (*pandas.DataFrame*) – DataFrame containing neighboring relationships between bacteria, including their image and object numbers.
- **neighbor_list_array** (*lil_matrix*) – Array storing neighbor connections for bacteria.
- **min_life_history_time_step** (*int*) – Minimum number of life history frames required for a bacterium to be considered

- **parent_image_number_col** (*str*) – Column name in the DataFrame for parent image numbers.
- **parent_object_number_col** (*str*) – Column name in the DataFrame for parent object numbers.
- **center_coord_cols** (*dict*) – Dictionary containing column names for x and y center coordinates.
- **divided_vs_non_divided_model** (*sklearn.Model*) – Machine learning model used to compare divided and non-divided states for bacteria
- **non_divided_bac_model** (*sklearn.Model*) – Machine learning model used to evaluate candidate links for continuity events.
- **division_model** (*sklearn.Model*) – Machine learning model used to validate division links.
- **color_array** (*np.ndarray*) – Array representing the colors of objects in the tracking data. This is used for mapping bacteria from the dataframe to the coordinate array for spatial analysis.
- **coordinate_array** (*csr_matrix*) – Array of spatial coordinates used for evaluating candidate links.

**Returns**

**tuple: A tuple containing the following DataFrames:**

**•filtered_continuity_link_cost_df (*pandas.DataFrame*):**
Filtered continuity link costs after applying thresholds and validations.

**•validated_division_cost_df (*pandas.DataFrame*):**
A DataFrame containing division costs after resolving conflicts and selecting the most plausible division links. Conflicts are resolved in three main scenarios:

1. **Overlapping Division Scenarios for a Single Source:**
   - When a single source bacterium is involved in multiple division scenarios with overlapping daughter candidates, the function evaluates each scenario's probabilities. The division pair with the highest likelihood is retained, and other conflicting scenarios are removed.
2. **Overlapping Daughters Across Different Sources:**
   - When different source bacteria propose overlapping daughters in their division scenarios, the utils compares the average division probabilities for each source. The highest likely source-daughter link is retained.
3. **Conflict Between Division and Continuity:**
   - When a candidate bacterium is involved in both division and continuity scenarios, either for a single source or multiple sources, the function compares probabilities, retains the more likely link (division or continuity), and removes the less likely one, ensuring biologically plausible assignments.

**•final_division_cost_df (*pandas.DataFrame*):**
Cleaned and finalized division costs for further analysis.

### 1.1.27 Module: bacteriaTrackingUpdate

bacteriaTrackingUpdate.**bacteria_modification**(*df*, *source_bac*, *target_bac_life_history*, *target_frame_bacteria*, *neighbor_df*, *neighbor_list_array*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*)

Modifies bacterial tracking data by assigning target bacterium to source bacterium, either as its child or the same bacterium, and recalculates features to ensure consistency in the tracking data.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame containing tracking data and measured bacterial features for all bacteria. This DataFrame is updated in place.
- **source_bac** (*dict*) – Dictionary representing the parent or the bacterium to which target is being assigned. Contains keys such as *ImageNumber*, *ObjectNumber*, and *id*. If *None*, target bacterium is treated as an unexpected beginning.
- **target_bac_life_history** (*pandas.DataFrame*) – Subset of the DataFrame containing the lifecycle data of target bacterium.
- **target_frame_bacteria** (*pandas.DataFrame*) – DataFrame containing bacteria present in the same time step as the target bacterium.
- **neighbor_df** (*pandas.DataFrame*) – DataFrame containing information about neighboring bacteria for calculating neighbor-specific features.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix indicating neighbor relationships between bacteria.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the DataFrame.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the DataFrame.
- **center_coord_cols** (*dict*) – Dictionary specifying column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

**Returns**

pandas.DataFrame:

The updated DataFrame *df* with modified bacterial tracking data.

bacteriaTrackingUpdate.**calc_modified_features**(*df*, *selected_bacteria*, *neighbor_df*, *neighbor_list_array*, *center_coord_cols*, *parent_image_number_col*, *parent_object_number_col*)

Modifies and updates bacterial tracking data by calculating various features related to bacterial life history.

**Calculated Features:**

- *id*: Unique identifier for each bacterium.
- *divideFlag*: Whether a bacterium has divided.
- *unexpected_end* and *unexpected_beginning*: Flags indicating unexpected lifecycle events.
- *daughters_index*: List of indices for daughter bacteria.
- *division_time*: Time of division.
- *LengthChangeRatio*: Ratio of daughter to mother length.
- *bacteria_movement*: Distance traveled between time steps.

- *TrajectoryX* and *TrajectoryY*: Components of movement trajectory.
- *MotionAlignmentAngle*: Angle of motion alignment.
- *LifeHistory*: Duration of a bacterium's lifecycle.
- *direction_of_motion*: Direction of movement in degrees.
- Neighbor-related features such as *difference_neighbors*.
- Slope and trajectory information (*bacteria_slope*, *prev_bacteria_slope*).
- Parent-daughter relationships and alignment angles.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame containing tracking data and measured bacterial features for all bacteria. This DataFrame is updated in place with calculated features.
- **selected_bacteria** (*pandas.DataFrame*) – A DataFrame containing the subset of rows (bacteria) to be modified. It includes tracking and measured bacterial features for selected bacteria.
- **neighbor_df** (*pandas.DataFrame*) – A DataFrame containing information about neighboring bacteria. Used for calculating neighbor-related features.
- **neighbor_list_array** (*lil_matrix*) – Sparse matrix indicating neighboring relationships between bacteria.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of bacterial centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **parent_image_number_col** (*str*) – Column name for the parent image number in the dataframe.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the dataframe.

**Returns**

pandas.DataFrame:

The updated DataFrame *df* with modified features.

bacteriaTrackingUpdate.**remove_redundant_link**(*dataframe*, *incorrect_target_bac_life_history*, *neighbor_df*, *neighbor_list_array*, *parent_image_number_col*, *parent_object_number_col*, *center_coord_cols*)

Removes redundant or incorrect link in the bacterial tracking data by modifying the lifecycle information of a target bacterium.

**Parameters**

- **dataframe** (*pandas.DataFrame*) – The main DataFrame containing tracking data and measured bacterial features for all bacteria. This DataFrame is updated in place.
- **incorrect_target_bac_life_history** (*pandas.DataFrame*) – Subset of the DataFrame containing the lifecycle data of the target bacterium with incorrect link.
- **neighbor_df** (*pandas.DataFrame*) – DataFrame containing information about neighboring bacteria for calculating neighbor-specific features.

- **neighbor_list_array** (*lil_matrix*) – Sparse matrix indicating neighbor relationships between bacteria.
- **parent_image_number_col** (*str*) – Column name for the parent image number in the DataFrame.
- **parent_object_number_col** (*str*) – Column name for the parent object number in the DataFrame.
- **center_coord_cols** (*dict*) – Dictionary specifying the column names for bacterial centroid coordinates (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

**Returns**

pandas.DataFrame:

The updated DataFrame with redundant link removed.

### 1.1.28 Module: helper

helper.**calc_distance_matrix**(*target_bac_coords_df*, *reference_bac_coords_df*, *x_col*, *y_col*)

Calculates a distance matrix between two sets of bacteria coordinates.

This function computes the pairwise Euclidean distances between the coordinates of target bacteria and reference bacteria

**Parameters**

- **target_bac_coords_df** (*pandas.DataFrame*) – A DataFrame containing the coordinates of the target bacteria.
- **reference_bac_coords_df** (*pandas.DataFrame*) – A DataFrame containing the coordinates of the reference bacteria.
- **x_col** (*str*) – The column name for the x-coordinate in both DataFrames.
- **y_col** (*str*) – The column name for the y-coordinate in both DataFrames.

**Returns**

A pandas DataFrame representing the distance matrix, where rows correspond to *target_bac_coords_df* indices and columns correspond to *reference_bac_coords_df* indices.

helper.**calc_movement**(*bac2*, *bac1*, *center_coordinate_columns*)

This function computes the displacement of a bacterium (*bac1* (source) to *bac2* (target)) using multiple spatial points.

**The computed movements include:**

- Centroid movement.
- Endpoint movements (first and second endpoints).
- Cross-movements between endpoints.

The minimum movement among all these calculations is returned to represent the most plausible movement distance.

**Parameters**

- **bac2** (*pandas.Series*) – Data for the bacterium in the next time step.
- **bac1** (*pandas.Series*) – Data for the bacterium in the current time step.
- **center_coordinate_columns** (*dict*) – A dictionary specifying the column names for the x and y coordinates of the bacterial centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

**Returns:**
float:

The minimum movement distance between *bac1* and *bac2*.

helper.**calc_neighbors_dir_motion_all**(*df*, *neighbor_df*, *division_df*, *selected_rows=None*)

Calculates motion alignment features for bacteria based on neighbor trajectories.

**Parameters**

- **df** (*pandas.DataFrame*) – Main DataFrame containing bacterial motion and trajectory data.
- **neighbor_df** (*pandas.DataFrame*) – DataFrame containing neighbor information, including image and object numbers for neighbors.
- **division_df** (*pandas.DataFrame*) – DataFrame containing division-related indices for bacteria.
- **selected_rows** (*pandas.DataFrame*) – Optional DataFrame for updating specific selected rows. Default is None.

**Returns**
pandas.DataFrame: The updated DataFrame with motion alignment features added.

helper.**calc_normalized_angle_between_motion_for_df**(*df*, *target_bac_motion_x_col*, *target_bac_motion_y_col*, *neighbors_avg_dir_motion_x*, *neighbors_avg_dir_motion_y*)

Calculates the normalized angles between motion vectors for each row in a DataFrame.

This function computes the angle between a motion vector and its neighbors' average motion vector for each row in the DataFrame. The angles are normalized by dividing by 180 degrees.

**Mathematical Explanation**:

- **Dot Product**: The dot product of two vectors is a measure of how aligned they are. For two vectors:

$$\mathbf{v_1} = (x_1, y_1), \quad \mathbf{v_2} = (x_2, y_2)$$

The dot product is given by:

$$\mathbf{v_1} \cdot \mathbf{v_2} = x_1 \cdot x_2 + y_1 \cdot y_2$$

- **Magnitude of a Vector**: The magnitude (length) of a vector is calculated as:

$$\|\mathbf{v}\| = \sqrt{x^2 + y^2}$$

- **Cosine of the Angle**: The cosine of the angle ( theta ) between two vectors is computed as:

$$\cos(\theta) = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{\|\mathbf{v_1}\|\|\mathbf{v_2}\|}$$

If either vector has zero magnitude, the cosine is undefined.

- **Angle in Radians**: The angle ( theta ) in radians is obtained using the arccosine function:

$$\theta = \arccos(\cos(\theta))$$

To handle numerical precision, the cosine value is clipped to the range ([-1, 1]).

- **Convert Radians to Degrees**: The angle is converted from radians to degrees using the formula:

$$\text{Angle (degrees)} = \frac{\theta \cdot 180}{\pi}$$

- **Normalized Angle**: Finally, the angle is normalized by dividing the degrees by 180:

$$\text{Normalized Angle} = \frac{\text{Angle (degrees)}}{180}$$

**Parameters**

- **df** (*pandas.DataFrame*) – A DataFrame containing direction of motion of objects and average direction of motion of neighbors.
- **target_bac_motion_x_col** (*str*) – The column name for the x-component of the object's motion vector.
- **target_bac_motion_y_col** (*str*) – The column name for the y-component of the object's motion vector.
- **neighbors_avg_dir_motion_x** (*str*) – The column name for the x-component of the neighbors' average motion vector.
- **neighbors_avg_dir_motion_y** (*str*) – The column name for the y-component of the neighbors' average motion vector.

**Returns**

np.ndarray: A 1D array of normalized angles (0 to 1) for each row.

helper.**calculate_all_bac_endpoints**(*df*, *center_coord_cols*, *angle_tolerance=1e-06*)

Calculates the endpoints of bacterial major axes based on orientation, length, and position.

This function determines the two endpoints of a bacterium's major axis by considering its orientation (parallel to vertical, horizontal, or otherwise) and center coordinates.

**Mathematical Explanation**:

- **Condition 1**: Bacteria parallel to the vertical axis $\pi/2$. For bacteria with orientation close to $\pm\pi/2$ (absolute difference less than $10^{-6}$), the endpoints are calculated as:

$$(x_{end1}, y_{end1}) = (x_{center}, y_{center} + L)$$
$$(x_{end2}, y_{end2}) = (x_{center}, y_{center} - L)$$

  where $L$ is the major axis length.

- **Condition 2**: Bacteria parallel to the horizontal axis. For bacteria with orientation close to 0 (absolute difference less than $10^{-6}$):

$$(x_{end1}, y_{end1}) = (x_{center} + L, y_{center})$$
$$(x_{end2}, y_{end2}) = (x_{center} - L, y_{center})$$

- **Condition 3**: Non-axis-aligned bacteria: "For bacteria with arbitrary orientation $\theta$, the endpoints are computed using the following formulas:"

$$x_{end1} = x_{center} + \frac{L}{2} \cdot \cos(\theta)$$
$$y_{end1} = y_{center} + (x_{end1} - x_{center}) \cdot \tan(\theta)$$
$$x_{end2} = x_{center} - \frac{L}{2} \cdot \cos(\theta)$$
$$y_{end2} = y_{center} + (x_{end2} - x_{center}) \cdot \tan(\theta)$$

- Endpoint selection ensures correct ordering along the major axis.

**Parameters**

- **df** (*pandas.DataFrame*) – A DataFrame containing bacterial data, including orientation, center coordinates, and major axis length.
- **center_coord_cols** (*dict*) – A dictionary with keys 'x' and 'y', specifying the column names for center coordinates.
- **angle_tolerance** (*float*) – The tolerance for detecting vertical or horizontal orientations (default: 1e-6).

**Returns**

pandas.DataFrame: The modified DataFrame with additional columns for endpoints: - *endpoint1_X*, *endpoint1_Y*: Coordinates of the first endpoint. - *endpoint2_X*, *endpoint2_Y*: Coordinates of the second endpoint.

helper.**calculate_all_bac_slopes**(*df*)

Calculates the slopes of lines formed by bacterial endpoints in the input DataFrame.

This function determines the slope for each bacterium based on its two endpoints (*endpoint1_X*, *endpoint1_Y*, *endpoint2_X*, *endpoint2_Y*) provided in the input DataFrame. The slope is calculated for all bacteria, with special handling for vertical lines (where the slope is undefined).

**Parameters**

**df** (*pandas.DataFrame*) – A DataFrame containing columns for bacterial endpoints: - *endpoint1_X*, *endpoint1_Y*: Coordinates of the first endpoint. - *endpoint2_X*, *endpoint2_Y*: Coordinates of the second endpoint.

**Returns**

pandas.DataFrame: The input DataFrame with an additional column:

- **bacteria_slope**: The slope of the line formed by the bacterial endpoints. NaN is assigned for vertical lines.

helper.**calculate_angles_between_slopes**(*slope_a*, *slope_b*)

Calculates the orientation angles between two slopes in degrees.

This function computes the angle in degrees between two lines defined by their slopes. NaN values are replaced with 90 degrees to handle undefined cases (e.g., perpendicular lines).

**Mathematical Explanation**:

- **Angle Between Two Slopes**: The angle $\theta$ between two lines with slopes $m_1$ and $m_2$ is given by:

$$\theta = \arctan\left(\left|\frac{m_1 - m_2}{1 + m_1 m_2}\right|\right)$$

**where:**

$m_1$ is the slope of the first line. $m_2$ is the slope of the second line.

- **Convert Radians to Degrees**: The angle in degrees is calculated using:

$$\text{Angle (degrees)} = \frac{\theta \cdot 180}{\pi}$$

- **Handling Undefined Cases**: When $1 + m_1 m_2 = 0$, the angle is undefined (lines are perpendicular). In such cases, NaN values are replaced with 90 degrees.

**Parameters**

- **slope_a** (*np.ndarray*) – An array of slope values for the first set of lines.
- **slope_b** (*np.ndarray*) – An array of slope values for the second set of lines.

**Returns**

An array of angles in degrees between the lines.

helper.**calculate_bac_endpoints**(*center*, *major*, *angle_rotation*, *angle_tolerance=1e-06*)

Calculates the endpoints of the major axis of a bacterium based on its center, length, and orientation.

This function computes the two endpoints of the major axis for a bacterium given its center coordinates, major axis length, and orientation angle. It handles three cases: bacteria aligned with the vertical axis, bacteria aligned with the horizontal axis, and bacteria with arbitrary orientations.

**Mathematical Explanation**:

- **Condition 1**: Bacteria parallel to the vertical axis $\pi/2$. For bacteria with orientation close to $\pm\pi/2$ (absolute difference less than $10^{-6}$), the endpoints are calculated as:

$$(x_{end1}, y_{end1}) = (x_{center}, y_{center} + L)$$
$$(x_{end2}, y_{end2}) = (x_{center}, y_{center} - L)$$

  where $L$ is the major axis length.

- **Condition 2**: Bacteria parallel to the horizontal axis. For bacteria with orientation close to 0 (absolute difference less than $10^{-6}$):

$$(x_{end1}, y_{end1}) = (x_{center} + L, y_{center})$$
$$(x_{end2}, y_{end2}) = (x_{center} - L, y_{center})$$

- **Condition 3**: Non-axis-aligned bacteria: "For bacteria with arbitrary orientation $\theta$, the endpoints are computed using the following formulas:"

$$x_{end1} = x_{center} + \frac{L}{2} \cdot \cos(\theta)$$
$$y_{end1} = y_{center} + (x_{end1} - x_{center}) \cdot \tan(\theta)$$
$$x_{end2} = x_{center} - \frac{L}{2} \cdot \cos(\theta)$$
$$y_{end2} = y_{center} + (x_{end2} - x_{center}) \cdot \tan(\theta)$$

- Endpoint selection ensures correct ordering along the major axis.

**Parameters**

- **center** (*list*) – A list containing the x and y coordinates of the bacterium's center (e.g., *(x_center, y_center)*).
- **major** (*float*) – The length of the major axis of the bacterium.
- **angle_rotation** (*float*) – The orientation angle of the bacterium in radians.
- **angle_tolerance** (*float*) – The tolerance for detecting vertical or horizontal orientations (default: 1e-6).

**Returns**

list: A list containing two sub lists for the endpoints: - *[x_end1, y_end1]*: Coordinates of the first endpoint. - *[x_end2, y_end2]*: Coordinates of the second endpoint.

helper.**calculate_bacterial_life_history_features**(*dataframe*, *calc_all_features*, *neighbor_df*, *division_df*, *center_coord_cols*, *use_selected_rows*, *original_df=None*)

Calculates features related to the continuous life history of bacteria.

This function computes various features for bacteria across time steps, such as changes in length, movement patterns, and directional motion.

**Parameters**

- **dataframe** (*pandas.DataFrame*) – A DataFrame containing bacterial features across time steps.
- **calc_all_features** (*bool*) – If True, computes additional features such as length change ratio, trajectory direction, and updates neighbor interactions.
- **neighbor_df** (*pandas.DataFrame*) – A DataFrame containing information about neighboring bacteria.
- **division_df** (*pandas.DataFrame*) – A DataFrame containing division-related information for bacteria.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for x and y center coordinates. Expected keys: - *'x'*: Base column name for x-coordinate. - *'y'*: Base column name for y-coordinate.
- **use_selected_rows** (*bool*) – If True, updates features based on selected rows from the original DataFrame.
- **original_df** (*pandas.DataFrame*) – The original DataFrame (optional) for updating selected rows when *flag_selected_rows* is True.

**Returns**

pandas.DataFrame: The updated DataFrame with the following newly computed features:

- **Length Change Ratio** (*LengthChangeRatio*): The ratio of the major axis length between the current and previous time steps.
- **Movement** (*bacteria_movement*): The minimum movement of the bacterium across time steps, calculated as the smallest displacement among the center and endpoints.
- **Direction of Motion**:
  - *direction_of_motion*: The angle (in radians) representing the trajectory direction.
  - *TrajectoryX* and *TrajectoryY*: The x and y components of the trajectory displacement.
- **Neighbor Interactions**: Updates direction of motion (*direction_of_motion*) based on neighboring bacteria's trajectories and interactions.

helper.**calculate_normalized_angle_between_motion_vectors**(*neighbor_motion_vectors*, *target_motion_vectors*)

Calculates the normalized angle (0 to 1) between two sets of direction of motion vectors.

This function computes the angles between corresponding direction of motion vectors in two arrays, normalizes the angles by dividing by 180, and returns the result.

**Mathematical Explanation**:

- **Dot Product**: The dot product of two vectors is a measure of how aligned they are. For two vectors:

$$\mathbf{v_1} = (x_1, y_1), \quad \mathbf{v_2} = (x_2, y_2)$$

The dot product is given by:

$$\mathbf{v_1} \cdot \mathbf{v_2} = x_1 \cdot x_2 + y_1 \cdot y_2$$

- **Magnitude of a Vector**: The magnitude (length) of a vector is calculated as:

$$\|\mathbf{v}\| = \sqrt{x^2 + y^2}$$

- **Cosine of the Angle**: The cosine of the angle ( theta ) between two vectors is computed as:

$$\cos(\theta) = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{\|\mathbf{v_1}\|\|\mathbf{v_2}\|}$$

  If either vector has zero magnitude, the cosine is undefined.

- **Angle in Radians**: The angle ( theta ) in radians is obtained using the arccosine function:

$$\theta = \arccos(\cos(\theta))$$

  To handle numerical precision, the cosine value is clipped to the range ([-1, 1]).

- **Convert Radians to Degrees**: The angle is converted from radians to degrees using the formula:

$$\text{Angle (degrees)} = \frac{\theta \cdot 180}{\pi}$$

- **Normalized Angle**: Finally, the angle is normalized by dividing the degrees by 180:

$$\text{Normalized Angle} = \frac{\text{Angle (degrees)}}{180}$$

**Parameters**

- **neighbor_motion_vectors** (*np.ndarray*) – A 2D array where each row represents a direction of motion vector for neighbors.
- **target_motion_vectors** (*np.ndarray*) – A 2D array where each row represents a vector for target motions.

**Returns**

A 1D array of normalized angles (0 to 1) between the vectors.

helper.**calculate_trajectory_angles**(*df*, *center_coord_cols*, *suffix1=''*, *suffix2=None*)

Calculates the angles of trajectory directions between two objects.

This function computes the direction vector between two bacteria and calculates the angle of that direction in radians. The angles are measured counterclockwise from the positive x-axis.

**Parameters**

- **df** (*pandas.DataFrame*) – A DataFrame containing the x and y coordinates for two positions of objects. These positions are used to calculate the trajectory direction.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of object centers.
- **suffix1** (*str*) – A suffix added to the base column names in *center_coord_cols* to identify the first set of positions. Default is an empty string.

- **suffix2** (*str*) – An optional suffix added to the base column names in *center_coord_cols* to identify the second set of positions. If not provided, the function assumes the second set of positions corresponds to columns prefixed with *prev_time_step*.

**Returns**

np.ndarray: A 1D array of angles (in radians) corresponding to the direction of trajectory for each object.

helper.**calculate_trajectory_displacement**(*df*, *center_coord_cols*, *axis*)

Calculates the displacement of objects along a specified axis between consecutive positions.

This function computes the difference (displacement) between the current and previous positions of objects along a specified axis (x or y). It is useful for analyzing movement patterns or trajectories over time.

**Parameters**

- **df** (*pandas.DataFrame*) – A DataFrame containing the current and previous positions of objects.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of the current position. Expected keys: - *'x'*: Base column name for the x-coordinate. - *'y'*: Base column name for the y-coordinate.
- **axis** (*str*) – The axis along which to calculate the displacement. Must be one of: - *'x'*: Calculates displacement along the x-axis. - *'y'*: Calculates displacement along the y-axis.

**Returns**

pandas.Series: A Series containing the displacements along the specified axis.

helper.**convert_angle_to_radian**(*df*)

Converts bacterial orientation angles from degrees to radians and modifies the values.

The orientation is adjusted using the formula:

$$Orientation(radian) = -\frac{(\text{Orientation (degrees)} + 90) \cdot \pi}{180}$$

This ensures the angles are transformed to a range suitable for downstream computations.

**Parameters**

**df** (*pandas.DataFrame*) – A DataFrame containing a column named "AreaShape_Orientation" with orientation angles in degrees.

**Returns**

pandas.DataFrame: The modified DataFrame with the "AreaShape_Orientation" column converted to radians.

helper.**convert_end_points_um_to_pixel**(*end_points*, *um_per_pixel=0.144*)

Converts endpoint coordinates from micrometers (um) to pixel units based on a specified conversion factor.

**Parameters**

- **end_points** (*list or np.array*) – A list or array of endpoint coordinates in micrometers to be converted.
- **um_per_pixel** (*float*) – The conversion factor representing micrometers per pixel (default: 0.144).

**Returns**

A numpy array of endpoint coordinates converted to pixel units.

helper.**convert_pixel_to_um**(*df*, *pixel_per_micron*, *all_rel_center_coord_cols*)

Converts distance measurements from pixel units to micrometers (um) within a DataFrame based on a specified conversion factor.

**Parameters**

- **df** (*pandas.DataFrame*) – The input DataFrame containing spatial and shape measurements in pixel units.
- **pixel_per_micron** (*float*) – The conversion factor representing micrometers per pixel.
- **all_rel_center_coord_cols** (*dict*) – A dictionary containing the column names for x and y center coordinates, e.g., {'x': ['Location_Center_X', 'AreaShape_Center_X'], 'y': ['Location_Center_Y', 'AreaShape_Center_Y']}.

**Returns**

The modified DataFrame with measurements converted to micrometers.

helper.**convert_um_to_pixel**(*major_len*, *radius*, *endpoints*, *center_pos*, *pixel_per_micron=0.144*)

Converts physical measurements in pixel to micrometers (um) based on a specified conversion factor.

**Parameters**

- **major_len** (*float*) – The length measurement in um to be converted.
- **radius** (*float*) – The radius measurement in um to be converted.
- **endpoints** (*list or np.array*) – A list or array of endpoint coordinates in um.
- **center_pos** (*list or np.array*) – A list or array of center coordinate in um.
- **pixel_per_micron** (*float*) – The conversion factor representing pixel per micrometers(default: 0.144).

**Returns**

A tuple containing the converted major_len, radius, endpoints, center_pos in pixel units.

helper.**extract_bacteria_features**(*df*, *center_coord_cols*)

Extracts key geometric and spatial features of bacteria from the input DataFrame.

This function retrieves the major axis length, minor axis length, radius, orientation, and center coordinates of bacteria, organizing them into a dictionary for further use.

**Parameters**

- **df** (*pandas.DataFrame*) – A DataFrame containing bacterial data with required columns for shape and position.
- **center_coord_cols** (*dict*) – A dictionary with keys 'x' and 'y', specifying the column names for center coordinates.

**Returns**

dict: A dictionary containing the extracted features: - 'major': Major axis length of bacteria. - 'minor': Minor axis length of bacteria. - 'orientation': Orientation of the bacteria (e.g., angle or direction). - 'center_x': X-coordinate of the bacteria's center. - 'center_y': Y-coordinate of the bacteria's center.

helper.**extract_bacteria_info**(*bacteria_data*, *pixels_per_micron*, *center_coord_cols*, *major_axis_len_col*, *orientation_col*)

Extracts and converts bacterial object information and convert from micrometers (μm) to pixel.

This function processes spatial and shape information of bacterial objects from a given DataFrame. The center coordinates and major axis length are converted from micrometers to pixel using the provided conversion factor.

**Parameters**

- **bacteria_data** (*pandas.DataFrame*) – A DataFrame containing information about bacterial objects, including spatial measurements and orientation.
- **pixels_per_micron** (*float*) – The conversion factor representing the number of pixels per micrometer.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for x and y center coordinates. Example: {'x': 'Location_Center_X', 'y': 'Location_Center_Y'}.
- **major_axis_len_col** (*str*) – The name of the column representing the major axis length of the objects (in pixels).
- **orientation_col** (*str*) – The name of the column representing the orientation of the objects (e.g., in degrees or radians, depending on the dataset).

**Returns**

**tuple: A tuple containing the following elements:**

- objects_center_x (pandas.Series): The x-coordinates of object centers in pixel.
- objects_center_y (pandas.Series): The y-coordinates of object centers in pixel.
- objects_major_axis (pandas.Series): The major axis length of objects in pixel.
- objects_orientation (pandas.Series): The orientation of objects.

helper.**find_bacteria_neighbors**(*dataframe*, *neighbor_df*)

Constructs a neighbor adjacency matrix for bacterial objects based on their spatial relationships.

This function processes a DataFrame containing bacterial object information and a second DataFrame specifying neighbor relationships. It generates an adjacency matrix where an entry (i, j) is True if bacterium i has bacterium j as a neighbor.

**Parameters**

- **dataframe** (*pandas.DataFrame*) – A DataFrame containing bacterial object information, including their unique indices, ImageNumber and ObjectNumber.
- **neighbor_df** (*pandas.DataFrame*) – A DataFrame specifying neighbor relationships between bacteria, including object numbers and their corresponding image numbers. It contains at least the following columns: - 'First Image Number': Image number of the first bacterium in the pair. - 'First Object Number': Object number of the first bacterium. - 'Second Image Number': Image number of the second bacterium in the pair. - 'Second Object Number': Object number of the second bacterium.

**Returns**

scipy.sparse.lil_matrix: A sparse adjacency matrix (LIL format) where rows and columns correspond to bacterial objects (indexed by *index*), and a True value at position (i, j) indicates that bacterium i has bacterium j as a neighbor.

helper.**identify_important_columns**(*df*)

Identifies key columns related to center coordinates, labels, and parent objects from the provided DataFrame.

**Parameters**

**df** (*pandas.DataFrame*) – Input DataFrame containing columns with bacterial feature data, such as tracking information and spatial coordinates.

**Returns**

tuple: A tuple containing:

- **center_coord_cols** (*dict*): Primary center coordinate columns (e.g., {'x': 'Location_Center_X', 'y': 'Location_Center_Y'}).
- **all_rel_center_coord_cols** (*dict*): All available center coordinate columns (e.g., {'x': ['Location_Center_X', 'AreaShape_Center_X'], 'y': ['Location_Center_Y', 'AreaShape_Center_Y']}).
- **parent_image_number_col** (*str*): Name of the column for parent image number.
- **parent_object_number_col** (*str*): Name of the column for parent object number.
- **label_col** (*str*): Name of the column for object labels.

helper.**update_dataframe_with_selected_rows**(*df*, *selected_rows*)

Updates specific rows in a DataFrame based on another DataFrame of selected rows.

This function takes an input DataFrame and updates specific rows (identified by the indices of the *selected_rows* DataFrame) with the values from the corresponding columns in *selected_rows*.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame to be updated.
- **selected_rows** (*pandas.DataFrame*) – A DataFrame containing the selected rows and their updated values. The indices of *selected_rows* must match the indices in *df* that are to be updated.

**Returns**

pandas.DataFrame: The updated DataFrame with the following columns modified or added:

- **`prev_time_step_MajorAxisLength`**: Major axis length of the bacterium from the previous time step.
- **`LengthChangeRatio`**: The ratio of the major axis length between the current and previous time steps.
- **Center Coordinates**: - *prev_time_step_center_x*: X-coordinate of the bacterium's center from the previous time step. - *prev_time_step_center_y*: Y-coordinate of the bacterium's center from the previous time step.
- **Endpoint Coordinates**: - *prev_time_step_endpoint1_X*: X-coordinate of the first endpoint from the previous time step. - *prev_time_step_endpoint1_Y*: Y-coordinate of the first endpoint from the previous time step. - *prev_time_step_endpoint2_X*: X-coordinate of the second endpoint from the previous time step. - *prev_time_step_endpoint2_Y*: Y-coordinate of the second endpoint from the previous time step.
- **`bacteria_movement`**: The minimum movement of the bacterium across time steps, calculated as the smallest displacement among the center and endpoints.
- **Trajectory Features**: - *direction_of_motion*: The angle (in radians) representing the trajectory direction. - *TrajectoryX*: X-component of the trajectory displacement. - *TrajectoryY*: Y-component of the trajectory displacement.

### 1.1.29 Module: calculateOverlap

calculateOverlap.**calculate_frame_existence_links**(*division_df*, *continuity_df*, *coordinate_array*)

Calculates overlaps source & target object of existence links in consecutive frames, distinguishing between objects involved in divisions and those maintaining continuity.

**Parameters**

- **division_df** (*pandas.DataFrame*) – DataFrame containing objects involved in division, with parent and daughter indices for overlap computation.
- **continuity_df** (*pandas.DataFrame*) – DataFrame containing objects maintaining continuity between frames, with indices for overlap computation.
- **coordinate_array** (*csr_matrix*) – Sparse boolean array where each row corresponds to the coordinates of an object.

**Returns**

pandas.DataFrame

A combined pivoted DataFrame containing IoU values: - For objects involved in division, it shows overlaps between parent and daughter objects. - For continuous objects, it shows overlaps between objects in the two frames.

calculateOverlap.**compute_intersections_and_unions**(*df*, *col1*, *col2*, *coordinate_array*)

Computes intersections and unions between two sets of objects based on their spatial coordinates, and calculates the Intersection over Union (IoU) for each pair.

**Parameters**

- **df** (*pandas.DataFrame*) – Input DataFrame containing object indices in columns *col1* and *col2*.
- **col1** (*str*) – Column names representing indices of two sets of objects.
- **col2** (*str*) – Column names representing indices of two sets of objects.
- **coordinate_array** (*csr_matrix*) – Sparse boolean array where each row corresponds to the coordinates of an object.

**Returns**

pandas.DataFrame: Updated DataFrame with a new column *iou* containing the IoU values for each pair of objects.

calculateOverlap.**compute_intersections_and_unions_division**(*df*, *col1*, *col2*, *coordinate_array*)

Computes intersections and unique areas for a pair of objects, typically for a parent-daughter relationship, and calculates Intersection over Union (IoU) with unique area consideration.

**Parameters**

- **df** (*pandas.DataFrame*) – Input DataFrame containing object indices in columns *col1* and *col2*.
- **col1** (*str*) – Column names representing indices of two sets of objects.
- **col2** (*str*) – Column names representing indices of two sets of objects.
- **coordinate_array** (*csr_matrix*) – Sparse boolean array where each row corresponds to the coordinates of an object.

**Returns**

pandas.DataFrame: Updated DataFrame with a new column *iou* containing the IoU values.

calculateOverlap.**find_overlap_oad_mother_daughters**(*mother_bad_daughters_df*, *coordinate_array*)

Identifies overlaps between mother objects of OAD (mothers with over assign daughters status) types and their associated daughter objects by calculating IoU.

**Parameters**

- **mother_bad_daughters_df** (*pandas.DataFrame*) – Input DataFrame containing indices of mother and daughter objects.

- **coordinate_array** (*csr_matrix*) – Sparse boolean array where each row corresponds to the coordinates of an object.

**Returns**

pandas.DataFrame: Pivoted DataFrame showing the IoU values for each mother-daughter pair.

calculateOverlap.**track_object_overlaps_to_next_frame**(*current_df*, *selected_objects*, *next_df*, *selected_objects_in_next_time_step*, *center_coordinate_columns*, *color_array*, *daughter_flag=False*, *coordinate_array=None*)

Tracks overlaps between objects in the current frame and candidate objects in the next frame using IoU, with optional division-specific calculations.

**Parameters**

- **current_df** (*pandas.DataFrame*) – DataFrame containing current frame objects.
- **selected_objects** (*pandas.DataFrame*) – Subset of objects selected from the current frame.
- **next_df** (*pandas.DataFrame*) – DataFrame containing next frame objects.
- **selected_objects_in_next_time_step** (*pandas.DataFrame*) – Subset of objects selected from the next frame.
- **center_coordinate_columns** (*dict*) – Dictionary containing column names for x and y center coordinates.
- **color_array** (*numpy.ndarray*) – Array storing colors for each object.
- **daughter_flag** (*bool*) – Whether to calculate unique areas for division analysis.
- **coordinate_array** (*csr_matrix*) – Sparse boolean array where each row corresponds to the coordinates of an object.

**Returns**

tuple: Pivoted DataFrame showing IoU values for overlaps and a product DataFrame with detailed pairwise data.

### 1.1.30 Module: neighborAnalysis

neighborAnalysis.**compare_neighbor_sets**(*df*, *neighbor_list_array*, *parent_image_number_col*, *parent_object_number_col*, *selected_rows_df=None*, *selected_time_step_df=None*, *return_common_elements=True*, *col_target=''*, *index2=True*)

Compares neighbor sets of source and target bacteria for each link to calculate differences and commonalities.

**This function determines:**

- How many neighbors are different between the source and target bacteria.
- How many neighbors are shared (common) between the source and target bacteria.

The results are stored as new columns in the DataFrame (*difference_neighbors* and *common_neighbors*).

**Key Operations:**

1. **Neighbor Extraction**: - Extracts the neighbor sets for each bacterium (source and target) from the *neighbor_list_array*. - Accounts for neighbors in the current and previous time steps.
2. **Comparison**: - Calculates the difference in neighbors (i.e., neighbors present in one bacterium but not in the other). - Identifies common neighbors (i.e., neighbors shared between source and target).

3. **Handling Special Cases**: - Accounts for bacteria with unexpected beginnings or ends. - Ensures proper handling of bacteria flagged with parent-child relationships.

4.**Updating DataFrame:**

•**Updates *df* with calculated values for:**

- *difference_neighbors_<col_target>*: Number of differing neighbors between source and target.
- *common_neighbors_<col_target>*: Number of shared neighbors.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame containing bacterial data, including neighbor information and flags.
- **neighbor_list_array** (*scipy.sparse.lil_matrix*) – A sparse matrix representing neighbor relationships, where rows correspond to bacteria and columns represent their neighbors.
- **parent_image_number_col** (*str*) – The column name in *df* representing the parent bacterium's image number.
- **parent_object_number_col** (*str*) – The column name in *df* representing the parent bacterium's object number.
- **selected_rows_df** (*pandas.DataFrame*) – A subset of rows from *df* to analyze specific bacteria and their neighbors. If *None*, the entire DataFrame is considered. Default is *None*.
- **selected_time_step_df** (*pandas.DataFrame*) – A subset of rows representing bacteria from a specific time step for reference. If *None*, the entire DataFrame is used as the reference. Default is *None*.
- **return_common_elements** (*bool*) – Whether to calculate and return the number of shared neighbors between source and target bacteria. Default is *True*.
- **col_target** (*str*) – A suffix added to column names to differentiate data for specific bacteria subsets. Default is *''*.
- **index2** (*bool*) – Whether to use a secondary index (*index2*) for calculations. Default is *True*.

**Returns**

**pandas.DataFrame: Updated DataFrame with columns:**

- *difference_neighbors_<col_target>*: Number of neighbors differing between source and target bacteria.
- *common_neighbors_<col_target>*: Number of shared neighbors.

neighborAnalysis.**compare_neighbors_batch**(*df*, *neighbor_list_array*, *bac1*, *bac2_batch*, *return_common_elements=False*)

Compares the neighbor sets of a single bacterium (*bac1*) with multiple bacteria (*bac2_batch*) (candidate target bacteria) to calculate the number of differing and shared neighbors for each pair.

**Key Operations:**

- Extracts the neighbor indices for *bac1* and all bacteria in *bac2_batch* from *neighbor_list_array*.
- Filters neighbors based on conditions like unexpected beginnings and ends.
- Compares neighbor sets for each pair to compute the differences and commonalities.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame containing bacterial data.
- **neighbor_list_array** (*scipy.sparse.lil_matrix*) – Sparse matrix representing neighbor relationships for bacteria.
- **bac1** (*pandas.Series*) – Data for the first bacterium (source) in the relationship.
- **bac2_batch** (*pandas.DataFrame*) – Data for multiple target bacteria in the relationship.
- **return_common_elements** (*bool*) – If *True*, returns the number of shared neighbors in addition to differing neighbors. Default is *False*.

**Returns**

**tuple:**

- *diff_neighbor_list* (np.ndarray): Array of differing neighbor counts for each pair.
- *common_neighbor_list* (np.ndarray, optional): Array of shared neighbor counts for each pair if *return_common_elements=True*.

neighborAnalysis.**compare_neighbors_single_pair**(*df*, *neighbor_list_array*, *bac1*, *bac2*, *return_common_elements=False*)

Compares the neighbor sets of two bacteria (bac1 and bac2) to calculate the number of differing and shared neighbors.

**Key Operations:**

- Extracts the neighbor indices for *bac1* and *bac2* from *neighbor_list_array*.
- Filters neighbors based on conditions like unexpected beginnings and ends.
- Compares the neighbor sets to compute the differences and commonalities.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame containing bacterial data.
- **neighbor_list_array** (*scipy.sparse.lil_matrix*) – Sparse matrix representing neighbor relationships for bacteria.
- **bac1** (*pandas.Series*) – Data for the first bacterium (source) in the relationship.
- **bac2** (*pandas.Series*) – Data for the second bacterium (target) in the relationship.
- **return_common_elements** (*bool*) – If *True*, returns the number of shared neighbors in addition to differing neighbors. Default is *False*.

**Returns**

**tuple:**

- *diff_neighbor* (int): Number of differing neighbors.
- *common_neighbor* (int, optional): Number of shared neighbors if *return_common_elements=True*.

### 1.1.31 Module: calcDistanceForML

calcDistanceForML.**calc_min_distance_ml**(*df*, *center_coord_cols*, *postfix_source*, *postfix_target*, *link_type=None*, *df_view=None*)

Computes the minimum distance between source and target bacteria based on their centroid and endpoint coordinates. Supports multiple distance calculations, including distances for division-specific comparisons.

**Details:**

- **Centroid Distance**: Euclidean distance between the centroids of source and target bacteria.

**•Endpoint Distances:**

  - Endpoint1-to-Endpoint1 and Endpoint2-to-Endpoint2.
  - For division-specific calculations, additional distances are considered: - Endpoint1-to-Endpoint2 and vice versa. - Centroid-to-Endpoints for cross-movement.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame containing bacterial data.
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of bacterial centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).
- **postfix_source** (*str*) – Suffix used to identify source-related columns in the DataFrame.
- **postfix_target** (*str*) – Suffix used to identify target-related columns in the DataFrame.
- **link_type** (*str*) – Specifies the type of distance calculation: - *'div'*: Includes additional cross-movement distances (e.g., endpoint-to-endpoint). - *None*: Computes only basic distances (centroid-to-centroid and same-endpoint distances).
- **df_view** (*pandas.DataFrame*) – A subset of the main DataFrame for efficient computation (optional).

**Returns:**

pandas.DataFrame:

**The updated DataFrame with the following additional columns:**

- *'min_distance'*: Minimum distance between source and target considering all point pairs.
- *'min_distance_continuity'* (if *stat='div'*): Minimum distance excluding cross-movement distances.

### 1.1.32 Module: calMotionAlignmentAngle

calMotionAlignmentAngle.**calc_motion_alignment_angle_ml**(*df*, *neighbor_df*, *center_coord_cols*, *selected_rows=None*, *col_target=None*, *col_source=None*)

Calculates the motion alignment angle between the movement direction of a link (source-to-target) and the average motion direction of neighboring bacteria. The alignment is computed in degrees, representing the angular difference between the link's motion and its neighbors' average motion.

**Parameters**

- **df** (*pandas.DataFrame*) – The main DataFrame containing bacterial data.
- **neighbor_df** (*pandas.DataFrame*) – A DataFrame containing neighbor relationships (e.g., First and Second Image/Object Numbers).
- **center_coord_cols** (*dict*) – A dictionary specifying the column names for the x and y coordinates of bacterial centroids (e.g., *{"x": "Center_X", "y": "Center_Y"}*).

- **selected_rows** (*pandas.DataFrame*) – The subset of rows for which motion alignment angles need to be computed.
- **col_target** (*str*) – Suffix for target-related columns in *selected_rows*.
- **col_source** (*str*) – Suffix for source-related columns in *selected_rows*.

**Returns:**
pandas.DataFrame:

**The updated *selected_rows* DataFrame with a new column:**

- *"Motion_Alignment_Angle"* + *col_target*: The calculated motion alignment angle (degrees).

### 1.1.33 Module: iouCalForML

iouCalForML.**calculate_iou_ml**(*df*, *link_type*, *col_source='prev_index_prev'*, *col_target='prev_index'*, *df_view=None*, *coordinate_array=None*, *both=True*)

Wrapper function for calculating Intersection over Union (IoU) values using *compute_intersection_union* function.

**Parameters**

- **df** (*pandas.DataFrame*) – DataFrame containing the source (*col_source*) and target (*col_target*) indices.
- **link_type** (*str*) – The type of link to evaluate, passed to *calculate_intersections_and_unions*. Options: - *'continuity'* - *'div'*
- **col_source** (*str*) – Column name in *df* for source indices. Default is *'prev_index_prev'*.
- **col_target** (*str*) – Column name in *df* for target indices. Default is *'prev_index'*.
- **df_view** (*pandas.DataFrame*) – Optional subset of *df* to use for efficient computation. Defaults to *None*.
- **coordinate_array** (*scipy.sparse.csr_matrix*) – Sparse matrix representing the coordinate of bacteria.
- **both** (*bool*) – If *True*, computes additional IoU metrics for continuity in division links.

**Returns:**
pandas.DataFrame:

**The updated DataFrame *df* with added columns:**

- *'iou'*: Inverted IoU values (1 - IoU).
- *'iou_continuity'* (optional): Inverted IoU for continuity when *link_type='div'* and *both=True*.

iouCalForML.**compute_intersection_union**(*df*, *col1*, *col2*, *link_type='continuity'*, *df_view=None*, *coordinate_array=None*, *both=True*)

Calculates intersections and unions for sparse matrices represented bacteria coordinates in *coordinate_array* and derives Intersection over Union (IoU) metrics for tracking links.

**Parameters**

- **df** (*pandas.DataFrame*) – DataFrame containing the indices (*col1* and *col2*) used to query the binary matrices.
- **col1** (*str*) – Column name in *df* representing the source indices.
- **col2** (*str*) – Column name in *df* representing the target indices.

- **link_type** (*str*) – The type of link to evaluate. Options: - *'continuity'*: Evaluates IoU using standard intersection and union metrics. - *'div'*: Includes a modified IoU calculation for division links, where unique daughter areas are considered.
- **df_view** (*pandas.DataFrame*) – Optional subset of *df* to use for calculations. Defaults to *None*.
- **coordinate_array** (*scipy.sparse.csr_matrix*) – Sparse matrix representing the coordinate of bacteria.
- **both** (*bool*) – If *True*, computes additional IoU metrics for continuity in division links.

**Returns:**

pandas.DataFrame:

**The updated DataFrame *df* with added columns:**

- *'iou'*: Calculated IoU values.
- *'iou_continuity'* (optional): IoU for continuity when *link_type='div'* and *both=True*.

### 1.1.34 Module: lengthRatio

lengthRatio.**check_len_ratio**(*df*, *selected_rows*, *col_target*, *col_source*)

Calculates the length change ratio between target and source bacteria and computes the dynamic length change.

**Details:**

**•LengthChangeRatio:**

– Computed as the ratio of the major axis lengths of the target and source bacteria.

**•Dynamic Length Change:**

**–For bacteria where the target is smaller than the source (*LengthChangeRatio* < *1*):**

∗ If the source bacterium's age is less than 2, the dynamic length is *1* - *LengthChangeRatio*.

∗ If the source bacterium's age is 2 or greater, the dynamic length is adjusted using the average changes in the source bacterium's history.

– For bacteria where the target is larger than or equal to the source (*LengthChangeRatio* >= *1*), the dynamic length is set to *0*.

**Parameters**

- **df** (*pandas.DataFrame*) – The DataFrame containing all bacterial data.
- **selected_rows** (*pandas.DataFrame*) – Subset of *df* containing the rows to process.
- **col_target** (*str*) – Column suffix for the target bacterium.
- **col_source** (*str*) – Column suffix for the source bacterium.

**Returns:**

pandas.DataFrame:

The updated *selected_rows* DataFrame with a new column *length_dynamic* appended for the target bacteria.

### 1.1.35 Module: propagateBacteriaLabels

propagateBacteriaLabels.**propagate_bacteria_labels**(*df*, *parent_image_number_col*, *parent_object_number_col*, *label_col*)

Assign or propagate labels for bacteria based on parent-child relationships.

This function assigns unique labels to bacteria and propagates them across frames based on parent-child relationships in tracking data. It ensures consistent labeling of bacteria across multiple time steps by leveraging parent image and object identifiers.

**Parameters**

- **df** (*pd.DataFrame*) – Input dataframe containing bacterial tracking data, including parent-child relationships.
- **parent_image_number_col** (*str*) – Name of the column representing the parent image number.
- **parent_object_number_col** (*str*) – Name of the column representing the parent object number.
- **label_col** (*str*) – Name of the column to assign or propagate labels.

**Returns**

pd.DataFrame

Updated dataframe with assigned or propagated labels in the specified column. Also adds a *checked* column to indicate whether a row has been processed.

### 1.1.36 Module: bacterialLifeHistoryAnalysis

bacterialLifeHistoryAnalysis.**process_bacterial_life_and_family**(*processed_cp_out_df*, *interval_time*, *elongation_rate_method*, *assigning_cell_type*, *cell_type_array*, *label_col*, *center_coord_cols*)

Calculates additional features related to the life history of bacteria from a processed CellProfiler output DataFrame.

**Parameters**

- **processed_cp_out_df** (*pandas.DataFrame*) – Processed CellProfiler output DataFrame containing measured bacterial features.
- **interval_time** (*float*) – Time interval between consecutive images (in minutes).
- **elongation_rate_method** (*str*) – Method for calculating the Elongation rate. Options: - 'Average': Computes the average elongation rate. - 'Linear Regression': Estimates the elongation rate using linear regression.
- **assigning_cell_type** (*bool*) – If True, assigns cell types to objects based on intensity thresholds.
- **cell_type_array** (*numpy.ndarray*) – A boolean array indicating the channel in which the intensity exceeds the threshold for each bacterium.
- **label_col** (*str*) – Column name in the CellProfiler DataFrame that indicates the label of bacteria during tracking.

- **center_coord_cols** (*dict*) – A dictionary with keys 'x' and 'y', specifying the column names for center coordinates.

**Returns**

**processed_cp_out_df (pandas.DataFrame):**
A DataFrame including new features related to the life history and family of bacteria.

**data_frame_with_selected_col (pandas.DataFrame):**
A subset of *processed_cp_out_df* that includes specific columns to save in numpy output files.

### 1.1.37 Module: calcElongationRate

calcElongationRate.**calculate_average_elongation_rate**(*division_length*, *birth_length*, *t*)

Calculates the average elongation rate of a bacterium over its lifecycle based on initial and final logarithm lengths.

**Parameters**

- **division_length** (*float*) – The length of the bacterium at the time of division (final length).
- **birth_length** (*float*) – The length of the bacterium at birth (initial length).
- **t** (*float*) – The total time duration of the lifecycle (unit: minute).

**Returns**

float:

The calculated average elongation rate. This represents the logarithmic elongation rate per minute.

calcElongationRate.**calculate_elongation_rate**(*df_life_history*, *interval_time*, *elongation_rate_method*)

Calculates the elongation rate of a bacterium based on its life history data and the specified method.

**Parameters**

- **df_life_history** (*pandas.DataFrame*) – Input DataFrame containing the life history of bacteria
- **interval_time** (*float*) – Time interval between consecutive images (in minutes).
- **elongation_rate_method** (*str*) – The method used for calculating the elongation rate. Options include: - "Average": Calculates the average elongation rate based on initial and final lengths. - "Linear Regression": Fits a linear regression to the length vs. time data to estimate the elongation rate.

**Returns**

float:

The calculated elongation rate. If the bacterium is observed for only one time step, returns NaN.

calcElongationRate.**calculate_linear_regression_elongation_rate**(*time*, *length*)

Calculates the elongation rate of a bacterium using linear regression on the logarithm of length as a function of time.

**Parameters**

- **time** (*numpy.ndarray*) – Array of time points corresponding to the bacterium's lifecycle.
- **length** (*numpy.ndarray*) – Array of major length measurements for the bacterium at the corresponding time points.

**Returns**

float:

The calculated elongation rate (slope of the logarithm of length vs. time).

### 1.1.38 Module: calcVelocity

calcVelocity.**calc_average_velocity**(*df_life_history*, *center_coord_cols*, *interval_time*)

Calculates the velocity of a bacterium based on its life history data.

**Parameters**

- **df_life_history** (*pandas.DataFrame*) – Input DataFrame containing the life history of bacteria
- **center_coord_cols** (*dict*) – A dictionary with keys 'x' and 'y', specifying the column names for center coordinates.
- **interval_time** (*float*) – Time interval between consecutive images (in minutes).

**Returns**

float:

The calculated velocity. If the bacterium is observed for only one time step, returns NaN.

### 1.1.39 Module: fluorescenceIntensity

fluorescenceIntensity.**assign_cell_types**(*dataframe*, *intensity_threshold*)

Assigns binary cell type labels based on fluorescence intensity thresholds.

**Parameters**

- **dataframe** (*pandas.DataFrame*) – Input DataFrame containing intensity data and other features.
- **intensity_threshold** (*float*) – Threshold value above which a cell type is marked as present (1).

**Returns**

numpy.ndarray:

A 2D numpy array where each row represents a cell, and each column represents a cell type, marked as 1 or 0 based on intensity threshold.

fluorescenceIntensity.**determine_final_cell_type**(*dataframe*, *cell_type_array*)

Determines and assigns the final cell type classification for each bacterium based on fluorescence intensity data.

**Parameters**

- **dataframe** (*pandas.DataFrame*) – Input DataFrame containing bacterial data and fluorescence intensity columns.
- **cell_type_array** (*numpy.ndarray*) – A 2D numpy array where each row represents a bacterial cell, and each column represents a binary classification (1 or 0) of a specific cell type based on intensity thresholds.

**Returns**

pandas.DataFrame: Updated DataFrame with a new column 'cellType' representing the final cell type classification for each bacterium:

- 3: Assigned if the fluorescence intensities of at least two channels exceed the defined threshold.
- Channel name (e.g., 'GFP', 'RFP', etc.): Assigned if only one channel's intensity exceeds the threshold.
- 0: Assigned if none of the fluorescence intensities across all channels exceed the defined threshold.
- 1: Default cell type assigned when only one intensity column exists.

fluorescenceIntensity.**get_fluorescence_intensity_columns**(*dataframe_col*)

Retrieves a sorted list of fluorescence intensity column names from the provided DataFrame column names.

**Parameters**

**dataframe_col** (*pandas.Series*) – A Pandas Series containing column names from a DataFrame.

**Returns**

list:

A sorted list of column names that contain the substring *Intensity_MeanIntensity_*.

fluorescenceIntensity.**mark_cell_types_by_intensity**(*df*, *cell_type_array*, *cols*, *intensity_threshold*)

Marks cell types as present (1) or absent (0) based on intensity threshold values.

**Parameters**

- **df** (*pandas.DataFrame*) – Input DataFrame containing intensity data for each cell type across multiple columns.
- **cell_type_array** (*numpy.ndarray*) – A 2D numpy array where each row represents a bacterial cell, and each column represents a cell type. This array will be updated with binary values (0 or 1) based on the intensity threshold.
- **cols** (*list*) – List of column names in *df* corresponding to cell types whose intensities need to be checked.
- **intensity_threshold** (*float*) – Threshold value above which the cell type is considered present (marked as 1).

**Returns**

numpy.ndarray:

Updated 2D numpy array with binary marks indicating cell type presence for each bacterial cell.

### 1.1.40 Module: drawDistributionPlot

drawDistributionPlot.**calculate_frequency**(*array*, *feature_related_col_name*)

Calculate the frequency of unique values in a given array and return the result as a DataFrame.

This function computes the frequency of each unique value in the input array and returns the results in a pandas DataFrame with columns for the unique values and their respective frequencies.

**Parameters**

- **array** (*numpy.ndarray*) – A 1D array of values for which the frequency distribution is to be calculated.
- **feature_related_col_name** (*str*) – The name to use for the column representing unique values in the output DataFrame.

**Returns**

pandas.DataFrame: A DataFrame with the following columns: - '<feature_related_col_name>': The unique values from the array. - 'Frequency': The count of each unique value.

drawDistributionPlot.**draw_feature_distribution**(*df*, *features_dict*, *label_col*, *interval_time*, *doubling_time*, *output_path*)

Generate and plot the distribution of selected features from a DataFrame.

This function iterates through a dictionary of features, calculates their frequency distribution, assigns them to bins, and visualizes the distribution with bar plots.

**Parameters**

- **df** (*pandas.DataFrame*) – The input DataFrame containing feature data to analyze.
- **features_dict** (*dict*) – A dictionary mapping feature column names to descriptive labels and type of analysis for plotting.
- **label_col** (*str*) – Name of the column for object labels.
- **interval_time** (*float*) – Time interval to scale the 'LifeHistory' feature.
- **doubling_time** (*float*) – Threshold for filtering 'LifeHistory' values.
- **output_path** (*str*) – The directory path where the plots will be saved.

**Returns**

None. The plots are saved as JPG files in the specified output directory.

drawDistributionPlot.**plot_frequency_distribution**(*x_positions*, *grouped_sel_feature*, *bin_labels*, *feature*, *feature_name*, *y_axis_value*, *output_path*)

Plot the frequency distribution of a selected feature and save the output.

**Parameters**

- **x_positions** (*numpy.ndarray*) – An array of x-axis positions for the grouped frequency bars.
- **grouped_sel_feature** (*pandas.DataFrame*) – A DataFrame containing grouped frequency data to plot.
- **bin_labels** (*list*) – Labels for the x-axis bins.
- **feature** (*str*) – The name of the feature being plotted.
- **feature_name** (*str*) – A descriptive label for the feature (used for x-axis labeling).
- **y_axis_value** (*str*) – A descriptive label for y-axis labeling.
- **output_path** (*str*) – The directory path where the plot will be saved.

CHAPTER

TWO

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

# INDEX

## D



## E



## F