This document describes the steps to run "conditionalColoc," a package implementing the conditional colocalization analysis scheme described in "Analysis of conditional colocalization relationships and hierarchies from three-color microscopy images" by Vega-Lugo, da Rocha-Azevedo et al., JCB 2022 (https://doi.org/10.1083/jcb.202106129).

Software installation

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

Required Matlab toolboxes:
- Statistics and Machine Learning Toolbox
- Image Processing Toolbox
- Curve Fitting Toolbox

The conditionalColoc package contains all functions (but not the Matlab toolboxes) necessary to run conditional colocalization analysis, except for those in the u-track package, needed for punctate object detection and the image analysis management infrastructure. The u-track package will require additional toolboxes (as explained with the u-track download).

- To download u-track: https://github.com/DanuserLab/u-track.

Please add the conditionalColoc and u-track packages (all folders and subfolders) to your MATLAB path.

Example data

The folder "exampleData" in the download contains example datasets to test the provided software and ensure that it is installed correctly. "exampleData" contains three subfolders:

a. "Pt2Pt2Pt": provides two examples for the case of three punctate channels. For each example, the following data are provided:
   - "CS" tiff file: Three-channel image. Can be opened in ImageJ for visualization and in Matlab for analysis.
   - "BF" tiff file: Corresponding brightfield image. Can be opened in ImageJ for visualization and in Matlab for analysis.
   - "cellMask" tiff file: ROI cell mask. Note that this is a logical array, and can only be opened in Matlab.
   - "detection" mat file: The detection results (i.e. object coordinates) for all channels, in the form needed for conditional colocalization analysis. Can be only opened in Matlab.

b. "Pt2Pt2Blob": provides two examples for the case of two punctate channels and one non-punctate channel. For each example, the following data are provided:

- "CS" tiff file: Three-channel image. Can be opened in ImageJ for visualization and in Matlab for analysis.
- "cellMask" tiff file: ROI cell mask. Note that this is a logical array, and can only be opened in Matlab.
- "detectionAndSeg" mat file: The detection results (i.e. object coordinates) for the two punctate channels, and the patch segmentation output ("maskBlobs" variable; a logical array) for the non-punctate channel.
- This dataset has no brightfield as the fluorescence images were used for drawing ROI cell masks.

c. "Blob2Blob2Blob": provides two examples for the case of three non-punctate channels. For each example, the following data are provided:
   - "CS" tiff file: Three-channel image. Can be opened in ImageJ for visualization and in Matlab for analysis.
   - "cellMask" tiff file: ROI cell mask. Note that this is a logical array, and can only be opened in Matlab.
   - "segmentations" mat file: The patch segmentation output ("maskBlobs" variable; a logical array) for each channel.
   - This dataset has no brightfield as the fluorescence images were used for drawing ROI cell masks.

Instructions

There are three instructions sections:

I. Instructions for running the core conditional colocalization analysis functions, which would allow integration with any detection and segmentation workflows if desired.
II. Instructions for running detection, segmentation and then conditional colocalization analysis using the workflow provided by u-track and conditionalColoc.
III. Instructions for running the code that tests the conditional colocalization analysis on simulated data.

I. Instructions for running core conditional colocalization functions

The descriptions below explain the core function(s) of conditional colocalization analysis. They also explain the format of the detection and segmentation input arguments for the conditional colocalization analysis functions.

The core functions for conditional colocalization analysis are:

- **condColocPt2Pt2Pt:** when all channels are punctate.
- **condColocPt2Pt2Blob:** when one channel is non-punctate.
- **condColocGeneral:** for any combination of punctate and non-punctate channels.

Before running conditional colocalization analysis, user must decide which objects are going to be defined as target, which as reference, and which as condition (see paper). See functions' documentation for instructions on where to input target, reference, and condition information. If user wants to explore different permutations of target, reference and condition, functions must be rerun with the appropriate permutations of input arguments.

*** For three punctate channels, run the command:

```
[colocalMeasure, numOfObjects] = condColocPt2Pt2Pt(nameOfStructRef,
nameOfStructTar, nameOfStructCond, mask, distThreshToColocWithCond,
colocDistThresh, numRandomizations)
```

The first three input arguments, nameOfStructRef, nameOfStructTar, nameOfStructCond, are structures containing the coordinates of punctate objects in the reference, target and condition channels. These structures must have the following format:

```
nameOfStruct.xCoord = column vector of x-coordinates
nameOfStruct.yCoord = column vector of y-coordinates
```

The length of the column vector equals the number of objects detected in the image of interest. See supplied test data for examples.

See function help for explanation of all other input parameters listed above and output arguments.

*** For two punctate channels and one non-punctate channel (reference or condition, but NOT target), run the command:

```
[colocalMeasure, numOfObjects] = condColocPt2Pt2Blob(nameOfStruct1,
nameOfStruct2, segmentationData, mask, distThreshToColocWithCond,
colocDistThresh, numRandomizations)
```

The first and second input arguments, nameOfStruct1 and nameOfStruct2, are structures containing the coordinates of punctate objects, as above.

The third input argument, segmentationData, is as follows:

After segmenting the objects (through whichever function/algorithm is relevant), obtain the mask of the segmented objects. The mask is a logical image with 1 where there is an object and 0 where there isn't. Let's call the mask "maskBlobs"; see for example test data for case of Pt2Pt2Blob. The, from the mask, get "segmentationData" - which is the list of pixels making the segmented objects - using the following commands:

```
SegIdx = regionprops(logical(maskBlobs),'PixelList','Centroid');
    nObjects = length(SegIdx);
    segmentationData = cell(nObjects,1);
```

```
for i = 1:nObjects
   segmentationData{i,1} = SegIdx(i).PixelList;
   segmentationData{i,2} = SegIdx(i).Centroid;
end
```

See function help for explanation of all other input parameters listed above and output arguments.

*** For any combination of punctate and non-punctate channels, run the command:

```
colocalMeasure = condColocGeneral(refInfo,tarInfo,condInfo,cellMask,
distThreshToColocWithCond, colocDistThresh,numTarRefRand,numCondRand)
```

The first three input arguments, refInfo, tarInfo and condInfo, are either binary images with the segmentations of non-punctate objects, or structures containing the coordinates of punctate objects as described for the case of three punctate objects. Note that, for this function, the mask of segmented objects is input directly, and there is no need to convert it to a list of pixels (as needed for the Pt2Pt2Blob case).

See function help for explanation of all other input parameters listed above and output arguments.

NOTE: The current version of this function is computationally expensive. To speed up the calculations, some parts of the function are parallelized (using parfor loops). The parallelization happens automatically inside the function, using the maximum number of workers available. It is recommended to run this function on a cluster.

II. Instructions for using integrated packages and workflow

These instructions provide a fully integrated workflow to detect and/or segment images, define ROI masks, and then perform conditional colocalization analysis. It employs a movie management infrastructure that helps keep track or and organize the images and their analysis.

The descriptions below briefly explain the function(s) for each step of the analysis. Further detailed descriptions are provided in the help section of each function.

1. **Setting up images and analysis**

   a. Save images as three-channel TIFF files.

      See example images supplied with software. If the images are not saved as three-channel TIFF files during acquisition, they must be put together as three-channel TIFF files before the analysis. This can be done in ImageJ.

   b. Create a movieList for the images to be analyzed together as a dataset.

Follow the instructions in "movieSelectorGUI," as explained here in brief:

i. Create a folder for each three-channel image to be analyzed. For ease of reference, it is highly recommended to give the analysis folder a name that clearly identifies which image it belongs to. **This folder will contain all analysis results for that image.** The image itself should NOT be inside this folder.
ii. Enter the command "movieSelectorGUI" in the Matlab command window.
iii. Click on "new." A new window will appear.
iv. Click on "Select Path" and select the corresponding folder from step i for the three-channel image.
v. Click on "Import movie using Bio-Formats" and select the three-channel image to be analyzed. Then press save. This will create a "movieData" file for this three-channel image, which will be saved in the image's analysis folder, and will be shown in the top left column of the "movieSelectorGUI" interface.
vi. Repeat Steps iii-v for all three-channel images to be analyzed.
vii. Once all the three-channel images within one dataset have been set up, click on "Save as movie list" at the bottom left of movieSelectorGUI and follow instructions. With this, a movieList file will be saved in a user-specified folder. This movieList file, or the individual movieData files, are the starting point for all ensuing analysis.
viii. If this is the end of analysis/set up, exit movieSelectorGUI.

After these steps are run once, they do not need to be run again. Following the initial set up, movieData and movieList files can be opened again using the "open" button instead of "new" button in movieSelectorGUI. Note that there are two open buttons, top one will open individual movieData files, and bottom one will open individual movieList files (which will load all movieData files associated with it).

If any questions arise, you can find more information by clicking on the help buttons of movieSelectorGUI.

2. Specifying Region-Of-Interest (ROI) masks

While not required, it is highly recommended to have an ROI for conditional colocalization analysis (it defines analysis area) and for segmenting non-punctate objects (it improves the quality of segmentation).

a. If you already have ROI masks (as tif files), associate them with their three-channel images as follows:

i. In the matlab command window, load the movieList file of the three-channel images (as generated in Step 1b above) by running ML = MovieList.load('movieListName.mat').

ii. Run the command importCellMaskMLMD(ML) to associate the ROIs with their respective three-channel images.
iii. When the window opens, select for each three-channel image its corresponding ROI mask tif file.

b. If you do not have ROI masks, you can generate ROI masks manually using the following procedure, and then associate them with their three-channel images:

<u>For masks using brightfield images:</u>

i. Create a movieList of the brightfield images of the cells as described in Step 1b above. If this was made earlier, just load it using the "open" button.
ii. Click "Tools" at the top left of movieSelectorGUI, then select "Add region of interest."
iii. Then press "Draw new region of interest."
iv. Use the mouse to draw the ROI, and then click "Save."
v. Once done creating the ROIs, exit movieSelectorGUI.
vi. Follow the steps described in (a) above to associate the masks with their three-channel images.

<u>For masks using the three-channel images:</u>

Follow same instructions as above using the movieList of the three-channel images. The only difference is that before drawing the ROI, you should select which channel to use for drawing the ROI.

Each ROI mask will be automatically saved in its three-channel image's folder, in a subfolder called "ImportedCellMask".

3. Punctate object detection

a. Enter the command "movieSelectorGUI" in the Matlab command window.
b. Load the movieData or movieList files to be analyzed (using the "open" buttons, as explained in Step 1b above).
c. Select u-track, click continue, then, select "[2D] Single particles."
d. Click on detection step, then, select "settings" to choose the detection method from the drop-down menu. For conditional colocalization analysis, only "Gaussian Mixture-Model Fitting" and "Point source detection" are recognized. Please do not choose any other detection method.
e. To set parameters, click on "Setting," and follow GUI instructions. Together with setting the parameters, also specify which channels to analyze. Then, click on "Apply".

NOTE related to the three-channel nature of these images:

Gaussian Mixture-Model Fitting applies the same parameters to all channels analyzed at the same time. Thus, if different channels require different parameters, the detection must be run once for each channel separately, each with its optimal parameters.

Point source detection allows the use of different parameters for different channels in the same run, as the parameters are defined per channel. You MUST click on the "save" button after setting parameters for each channel, otherwise the settings will be lost.

f. To run analysis on all images from a movieList click "Run all movies/image", then press "Run"

Detection output will be automatically saved in each three-channel image's folder, in a subfolder called "TrackingPackage/GaussianMixtureModel" or "TrackingPackage/point_sources" (depending on which detection method was used).

Note that if both methods are run on a dataset, the conditional colocalization analysis functions will only recognize the last method that was run.

4. Non-punctate object segmentation

These commands will be run in the Matlab command line.

Use the function **segmentBlobsPlusLocMaxMLMD**.

a. Load a movieData or movieList file by running one of the below lines in the command window:
   i. MD = movieData.load('nameOfMovieDataFile.mat');
   ii. ML = movieList.load('nameOfMovieListFile.mat');
   Note that "nameOfMovieDataFile" and "nameOfMovieListFile" must be replaced by actual file name.

b. Run segmentation by running the below line in the command window: segmentBlobsPlusLocMaxMLMD(ML, blobChannels) (for movieList) or segmentBlobsPlusLocMaxMLMD(MD, blobChannels) (for movieData).

   This will run the segmentation code with its default parameters. To run with non-default parameters, see function help for detailed explanation of all input parameters. Optional parameters are input as name-value pairs, as in example below:

   segmentBlobsPlusLocMaxMLMD(ML, [1 3], 'thresholdMethod', {'rosin','otsu'}) will run segmentation using rosin method on channel 1 and otsu method on channel 3.

   NOTE: If an ROI is available in the "ImportedCellMask" subfolder, it will be

automatically used. If no ROI is available, then the whole image will be used for segmentation.

Segmentation output will be automatically saved in each three-channel image's folder, in a subfolder called "SegmentBlobsPlusLocMaxSimple".

5. Conditional colocalization analysis

NOTE: Step 9 below extends the instructions in Steps 5-8 in order to run conditional colocalization analysis for a range of colocalization radii.

****** For conditional colocalization analysis of three punctate objects, use the function:***

condColocAnalysisPt2Pt2PtMLMD

Function will make all permutations of target, reference, and condition.

a. Load a movieData or movieList file as explained in Step 4a above.

b. To run conditional colocalization analysis using default parameters, enter below line on the command window:
   - condColocAnalysisPt2Pt2PtMLMD(ML (or MD))

c. To run conditional colocalization analysis using different parameters, enter below line on the command window:
   - condColocAnalysisPt2Pt2PtMLMD(ML(orMD), 'detectionProcessID', {'SubRes','PointSource', 'PointSource'}, 'colocDistThresh', [4 4 4])

     This example will use coordinates from Gaussian mixture-model fitting ('SubRes') for channel 1 and coordinates from point source detection ('PointSource') for channels 2 and 3. Colocalization threshold (colocDistThresh) is set to be 4 pixels for all pairs of channels.

See function documentation for input parameters description.

NOTE: If an ROI is available in the "ImportedCellMask" subfolder, it will be automatically used. If no ROI is available, then the whole image will be used for analysis.

Conditional colocalization analysis output will be automatically saved in each three-channel image's folder, in a subfolder called "ConditionalColocPt2Pt2Pt".

****** For conditional colocalization of two punctate objects and one non-punctate object, use the function:***

condColocAnalysisPt2Pt2BlobMLMD.

Function will make all possible permutations of target, reference, and condition (reminder: only condition and reference can be non-punctate in this case).

a. Load a movieData or movieList file as explained in Step 4a above.

b. To run conditional colocalization analysis using default parameters, enter the below line on the command window
    - condColocAnalysisPt2Pt2BlobMLMD(ML (or MD), [A B], C)

    A and B are the indices of the punctate channels. C is the index of the non-punctate channel.

    Example: If punctate objects are in channels 2 and 3 and the non-punctate objects are in channel 1, then you call:
    - condColocAnalysisPt2Pt2BlobMLMD(ML (or MD), [2 3], 1).

c. To run conditional colocalization analysis using different parameters, enter additional parameters as name-value pairs, as in the example above in the condColocPt2Pt2PtMLMD section.

See function documentation for input parameters description.

NOTE: If an ROI is available in the "ImportedCellMask" subfolder, it will be automatically used. If no ROI is available, then the whole image will be used for analysis.

Conditional colocalization analysis output will be automatically saved in each three-channel image's folder, in a subfolder called "ColocalizationPt2Pt2Blob".

****** For conditional colocalization of any combination of punctate and non-punctate objects, use the function:***

condColocGeneralAnalysisMLMD.

Function will make all possible permutations of target, reference, and condition.

a. Load a movieData or movieList file as explained in Step 4a above.

b. To run conditional colocalization analysis using default parameters, enter the below line on the command window:
    - condColocGeneralAnalysisMLMD(ML (or MD), ptOrBlob)

    ptOrBlob is a vector to identify each channel as punctate or non-punctate. Enter 0 for non-punctate, 1 for punctate.

    Example: If objects in all channels are non-punctate, then you call:

- condColocGeneralAnalysisMLMD (ML (or MD), [0 0 0]).

c. To run conditional colocalization analysis using different parameters, enter additional parameters as name-value pairs, as in the example above in the condColocPt2Pt2PtMLMD section.

See function documentation for input parameters description.

NOTE: If an ROI is available in the "ImportedCellMask" subfolder, it will be automatically used. If no ROI is available, then the whole image will be used for analysis.

Conditional colocalization analysis output will be automatically saved in each three-channel image's folder, in a subfolder called "ConditionalColocGeneral"

NOTE: As with the core function above, the current version of this function is computationally expensive. To speed up the calculations, some parts of the function are parallelized (using parfor loops). The parallelization happens automatically inside the function, using the maximum number of workers available. It is recommended to run this function on a cluster.

6. Compilation and visualization of results

To plot and/or save the conditional colocalization analysis results from one or more datasets of the same condition, use the function:

plotSaveConditionalColoc

This function compiles colocalization values and various object statistics (e.g. number of objects, non-punctate object sizes, etc.) so as to save them and/or plot them.

Run the below line in the command window to plot and/or save the conditional colocalization analysis results:

plotSaveConditionalColoc(PtOrBlob, plotting, saveRes, ML1, ML2, …)

See function documentation for a detailed explanation of input parameters. Note that at least one movieList must be input.

Function will compile the results of all movieList files into the same plot when plotting and into the mat file when saving. For example, if 3 movieList files with 10 movies each are input, the plots will include 30 data points and the mat files will include 30 rows in each array. See function documentation for more details.

7. Statistical tests

To run statistical tests on conditional colocalization results use the function:

condColocStatTest.

Before using this function, Step 6 must be run and results must be saved.

See function help for description of input and output.

8. **-log$_{10}$(p-value) tables**

   To plot -log$_{10}$(p-value) tables use the function: **plotCondColocPvalues.**

   Before using this function, Step 7 must be run and results must be saved.

   See function help for description of input and output.

9. **Using a range of colocalization radii**

   The colocalization analysis functions described in Step 5 above can employ only one colocalization radius at a time.

   To perform conditional colocalization analysis using multiple colocalization radii, the functions in Steps 5-8 must be run for each radius separately.

   However, each time the functions of Step 5 are rerun, the new output from Step 5 overwrites the old output.

   Thus, to store the results of each run, Steps 5-8 must be run sequentially for one colocalization radius before moving on to the next radius.

   In Step 6, make sure to save the plots and colocalization values for each colocalization radius in a separate folder that is clearly identifiable.

III. Instructions for testing conditional colocalization analysis on simulated data

To set up simulations and run conditional colocalization analysis on the simulations, use the function:

runColocalizationAnalysisSimulation

This function generates simulated data using the given input parameters and runs the conditional colocalization code on those data. However, it does not output the simulated data. Rather, it outputs the simulation input parameters and the resulting conditional colocalization measures.

See function documentation for details on how to run different types of simulations and their input parameters.