Temporal Analysis of Spheroid Imaging (TASI) Toolkit Instruction

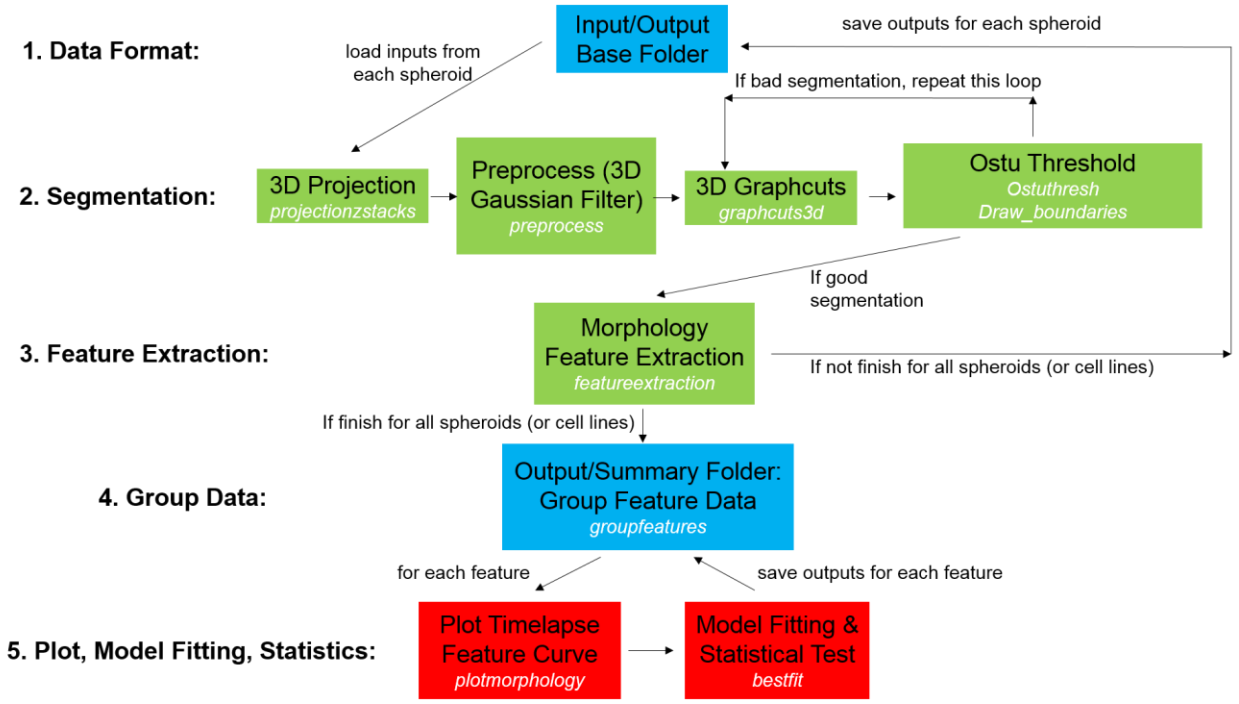


Figure 1. Workflow of TASI Toolkit. There are mainly 5 parts of the workflow. The blue boxes are folders for input/output data. The green and red boxes are steps of individual algorithms. Arrowheads indicate the running sequence of each step. Our novelty functions are highlighted in red.

Download TASI from GitHub

For step1 to 3, use “TASI_individual.m” as an example. Repeatedly run this algorithm for all individual spheroids.

Then for step 4 to 5, use “TASI_group.m” as an example. You can choose different sections according to how many groups you have.

Codes from External Sources

For the first time users, please download and include the following external codes under your TASI folder.

1. 3D Projection

Go to https://sites.google.com/site/brightfieldorstaining/downloads-1 and download "projstack.m"[1]. You can comment out line 27-42 to make the processing faster.

2. 3D Graphcuts

Go to http://www.mathworks.com/matlabcentral/fileexchange/34126-fast-continuous-max-flow-algorithm-to-2d-3d-image-segmentation, download and unzip the "CMFv1.0.zip" folder and put it under TASI folder. {Yuan Jing, 2010, A study on continuous max-flow and min-cut approaches}

Note: The first time you run the code, you need to change the current folder path to the unzipped "CMF v1.0" folder under TASI folder. Then compile the function CMF3D_mex using the following command:

```
>> mex CMF3D_mex.c
```

But after the first time running, you don't have to run the above command anymore.

3. ShadedErrorBar

Go to http://www.mathworks.com/matlabcentral/fileexchange/26311-shadederrorbar, download and unzip "shadedErrorBar" folder, and put it under TASI folder.

4. Set Path to Include the External Codes

After including all external codes/folders under your TASI folder, open Matlab, click "Home/Set Path" Tab, click "Add with Subfolders…" tab, it will pop up a window, choose your TASI folder, click "select folder", then click "save" and "close".

1. Data Format

The data format part is to construct the input and output folders/subfolders and rename the image sequences according to the description.

1.1 Input Data Format

There should be an input base folder. Under base folder, create one subfolder per cell line (or per spheroid, or per treatment). For example, I have 6 spheroids data, so I should generate 6 subfolders called: leader1, leader2, leader3, follower1, follower2 and follower3. And one of the subfolder is my **Input_directory** (e.g. "…\Spheroid\Inputs\Follower1").

Within each subfolder, the input series images should be named as "CellLine#_t#_z#_ch#.extension". "CellLine" is a placeholder for any length of string, including underscores and dashes. "#" is a placeholder for any length of number. "t" means timepoint. "z" means z slice or position in z direction for confocal images. "ch" means channel for multi-stained images. Increase the # in the sequence of "ch#" first, then "z#", and finally "t#". For example, if I have 2 channels, 2 z positions and 2 timepoints, my images should be in the sequence of:

cell1_t01_z1_ch1.tif

cell1_t01_z1_ch2.tif

cell1_t01_z2_ch1.tif

cell1_t01_z2_ch2.tif

cell1_t02_z1_ch1.tif

cell1_t02_z1_ch2.tif

cell1_t02_z2_ch1.tif

cell1_t02_z2_ch2.tif

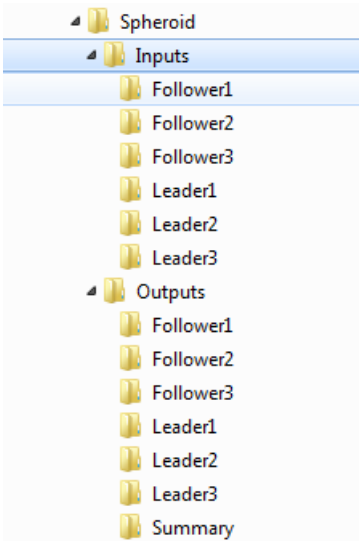


Figure 2. Example of Input and Output Data Format.

1.2 Output Data Format

Similar as input base folder, there should be an output base folder. Under output base folder, create one subfolder per cell line (or per spheroid, or per treatment) and also a subfolder called "Summary" (or any name you want), which contains the statistical test, model fitting results among all conditions (6 spheroids as my example). The **Output_directory** is one of the cell line subfolder (e.g. "…\Spheroid\Outputs\Follower1").

Within each cell line subfolder, there are output images (both ".fig" and ".tif" formats) and results files (both ".mat" and ".txt" formats). Under the **Summary** subfolder, there are comparison figures (both ".fig" and ".tif" formats) and statistical results files (both ".mat" and ".txt" formats).

2. Segmentation

The segmentation part is to project the 4D (x,y,z,t) images into 3D (x,y,t) to increase the contrast of dim branches of the spheroids, then perform 3D gaussian filter to smooth the image sequences in time and spatial domain. And then use 3D graphcuts and ostu thresholding to segment the images into binary masks. If you use other segmentation methods to generate the binary masks, you can skip this part. But make sure you put the binary masks (named as mask0001, mask0002, mask0003, etc.) under each Outputs/CellLine folder.

2.1 3D Projection

projection=projectionzstacks(Z,n_channel,channel,saveimage,Input_directory,Output_directory)

This *projectionzstacks* function is to enhance the contrast of images and convert time series images into 3d matrix for easier segmentation.

Example for this function:

projection = projectionzstacks(7,2,1,1,Input_directory,Output_directory);

You need to specify the input parameters: Z, n_channel, channel and saveimage. The "Input_directory" and "Output_directory" are defined at the Data Format Step, so don't change them. If there is no "Input_directory" and "Output_directory", a dialogue window will pop up to let you select the input/output folder.

Z: number of Z slices, eg. Z = 7.

n_channel: number of channels, eg. n_channel = 2 means you have two different channels

channel: indicate which channel you want to process, eg. channel = 2, means the second channel. Note: If you named your channel as ch0, ch1, ch2, then channel = 2 processing for ch1 images.

saveimage: 1 (save) or 0 (not save the projected images under Output_directory). If not specified, default is 1.

The output **projection** is a 3d M×N×T matrix saved under Output_directory. [M, N] is image size and T is number of timepoints. The function will also save all the projected images called proj00##.tif under Output_directory, ## is timepoints (see Fig2 as an example).

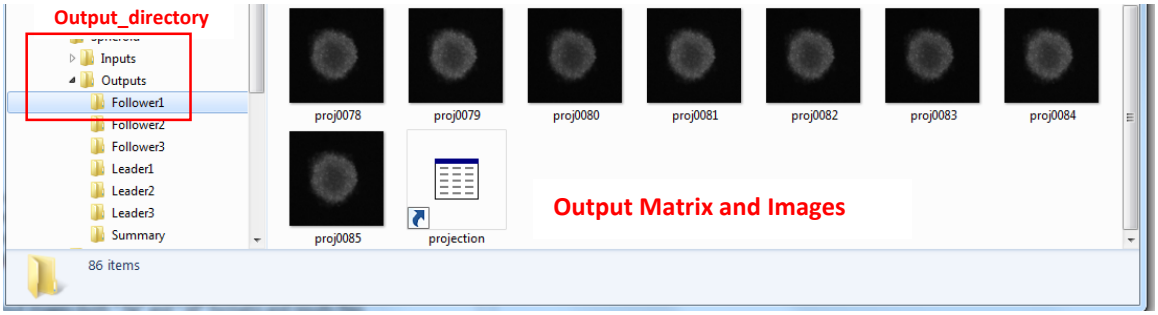


Figure 3. Example of output directory and output matrix for function *projectionzstacks*.

2.2 Preprocess: 3D gaussian filter and/or enhancing contrast

[proj_3dgauss, proj_adjust] = preprocess(Output_directory, projection, sigma, adjust, saveimage)

This *preprocess* function is to perform 3d gaussian filtering and then enhance contrast of the filtered images.

Example for this function:

[proj_3dgauss, proj_adjust] = preprocess(Output_directory, projection, [2.5, 2.5, 0.5], 1, 1);

You need to specify the input parameters: sigma, adjust, saveimage. The input "projection" is the output of the previous function projectionzstacks, so don't change it. If "projection" is not provided, a dialogue window will pop up to let you select the "projection.mat" matrix. The "Output_directory" is again defined at the Data Format Step, so don't change it. If there is no "Output_directory", a dialogue window will pop up to let you select the output folder.

sigma: 1*3 vector with sigma values in x, y, t(or z) direction for gaussian filter. The larger the sigma, more smooth and blurry filtering in that direction. If not specified, default is [2.5, 2.5, 0.5].

adjust: 0 or 1 scalar, whether to perform contrast enhancement on filtered images. 1 is adjust or enhance, 0 is not adjust. If not specified, default is 1.

saveimage: 0 or 1 scalar, whether to save individual filtered images or/and adjusted images under the Output_directory. 1 is save, 0 is not save. If not specified, default is 1.

projection: 3d M*N*T matrix, which is the output of *projectionzstacks* function. Don't change it.

The outputs proj_3dgauss and proj_adjust are both 3d M*N*T matrix saved under Output_directory. [M, N] is the image size. T is timepoint.

proj_3dgauss is the gaussian filtered matrix. If you set saveimage = 1, the function will automatically save all the filtered images as 'gaussian00##.tif' under Output_directory. ## is timepoint or frame number.

proj_adjust is the gaussian filtered and then enhanced contrast matrix. If adjust=1 and saveimage=1, the function will automatically save all the adjusted images as 'gadj00##.tif' under Output_directory.

This function will also automatically generate a 'preprocess_parameters.txt' file to save all the filtering settings (sigma, filter method, filter size) as references for future refining/adjustment/repeating of the 3d gaussian filtering.

2.3 3D Graphcuts

graphcuts = graphcuts3d(Output_directory, proj_adjust, alpha, u1, u2)

This function is to segment the 3d image matrix (either proj_adjust or proj_3dgauss) using 3d graphcuts methods{Yuan, 2010, A Study on Continuous Max-Flow and Min-Cut Approaches}.

Example for this function: *graphcuts = graphcuts3d(Output_directory, proj_adjust, 0.1, 0, 0.7);*

You need to specify the input parameters: proj_adjust, alpha, u1 and u2. The "Output_directory" is again defined at the Data Format Step, so don't change it. If there is no "Output_directory", a dialogue window will pop up to let you select the output folder.

proj_adjust: 3d M*N*T gaussian filtered and adjusted contrast matrix, the output of *preprocess* function. Note: If in the preprocess step you set adjust=0, then you should use proj-3dgauss as your input instead of proj-adjust. If “proj_adjust” is not provided, a dialogue window will pop up to let you select the “proj_adjust.mat” matrix (when adjust=1 at preprocess step) or select the “proj_3dgauss.mat” matrix (when adjust=0 at preprocess step).

alpha: scalar within [0,1]. Constant for penalty, smaller the better (or larger) segmentation. If not specified, default is 0.2. See [Yuan’s references] for definition and selection for alpha.

u1 and **u2**: scalars within [0,1]. Lower and upper limit for ulab (or u(x) in the CMF_README file). Smaller the better (or larger) segmentation. If not specified, default is u1=0.2, u2=0.7.

The output **graphcuts** is a 3d M*N*T grayscale single matrix saved under Output_directory. [M, N] is the image size. T is timepoint. This function will also automatically generate a 'graphcuts_parameters.txt' file to save all the alpha, u1, u2 values as references for future refining/adjustment/repeating of the graphcuts segmentation.

Then this section will automatically pop up a window, showing the original projected image and the segmented image at the middle timepoint (Fig3). In this way, you can know how the segmentation result looks like and choose the Ostuthresh settings for the next step.

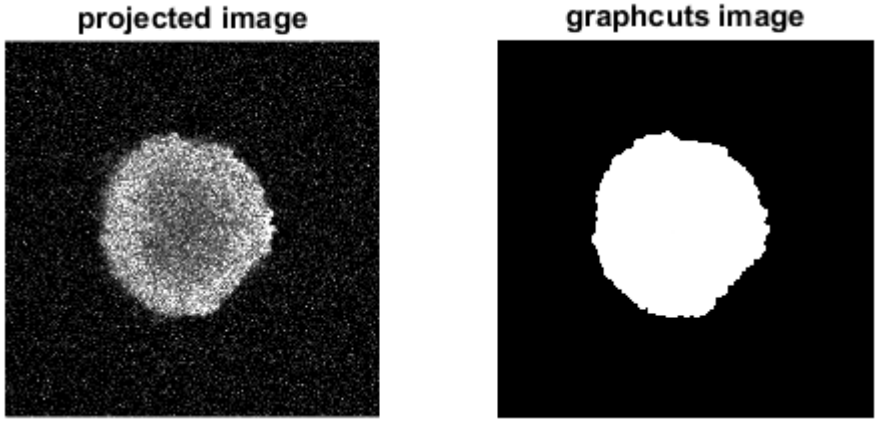


Figure 4. Example of 3d projection and graphcuts segmentation result.

2.4 Ostuthresh

a. *[] = ostuthresh(Output_directory, graphcuts, thresh, area, se)*

This function is to threshold the graphcuts results into binary masks, and save all the binary masks under the Output_directory. Then generate a text file "thresh_parameters.txt" saving all the input parameter settings.

Example for this function: *ostuthresh(Output_directory, graphcuts, [], 150, 1);*

You need to specify the inputs: thresh, area and se. The “Output_directory” is again defined at the Data Format Step, so don’t change it. If there is no “Output_directory”, a dialogue window will pop up to let you select the output folder. The input “graphcuts” is the output of the previous function graphcuts3d, so don’t change it. If “graphcuts” is not provided, a dialogue window will pop up to let you select the “graphcuts.mat” matrix.

thresh: thresholding value for Matlab function *im2bw*, range[0,1]. The intensity larger than thresh will be white, and other intensities will be black. If not specified, use Ostu graythresh method, automatically calculate the threshold value using Matlab function *graythresh*.

area: threshold area for removing background noises, remove all objects smaller than area value. If not specified, default = 10 pixel.

se: size of disk structure to close gaps. se must be integers, possible ranges for se is 3~8, the larger the gaps on the boundary, se should be larger. If not specified, default = 0, no close gaps.

There is no output matrix, but the function will automatically save all the binary images under

Output_directory as 'mask00##.tif', ## is timepoint. It will also automatically generate a 'thresh_parameters.txt' file to save all the thresholding parameters (thresh, area, se) for future refining/adjustment of the thresholding.

b. draw_boundaries(Output_directory, enhance, movie_name, Frame_rate, Linewidth, color)

After generating binary masks at each time point, this function is to create a movie to visualize the segmentation results. It will draw the segmented mask boundary onto projected image sequences as the movie and save it under Output_directory.

Example for this function:

draw_boundaries(Output_directory, 1, 'boundary.avi', 7, 2, [1 1 1]);

You need to specify the inputs: enhance, movie_name, Frame_rate, Linewidth and color. The Output_directory is the same as before, so don't change it.

enhance: whether to enhance the contrast of the original images for better visualization, but will slow down the speed. It must be either 1 or 0. 1 is to enhance, 0 is not. Default is 0, not enhance.

movie_name: 'xxxxx' any name you want to call the output movie. If not specified, default is "movie"

Frame_rate: scalar or integer. frame per second (fps) for playing the movie. If not specified, default is 10 fps.

Linewidth: scaler or integer. Line width for drawing boundaries. If not specified, default is 2.

color: must be in format [x x x], where x is any number between 0 and 1. For example, [0 0 1] is blue; [1 1 1] is white(default); [1 0 0] is red; [0 1 0] is green, etc.

3 Feature Extraction

This part is to smooth the segmented mask of a spheroid at each time point, then extract morphology features of filled spheroid, unfilled spheroid and single cells isolated from the spheroid, respectively from the same segmented binary masks. All the output results (matrix, csv files and images) will be saved under Output_directory folder.

[T_unfill, S_unfill, T_fill, S_fill, single] = featureextraction (ZSliceName, Resolution, SingleCellArea, smooth, plotsetting, Input_directory, Output_directory)

Note: If you skip part2-segmentation and use your own segmented binary images, you need to change line 104 according to your binary images' name. For example, if your segmented images are named as 'binary01.tif','binary02.tif', etc. then change line 117 to:

mask_list = dir([Output_directory filesep 'binary*.tif']); % originally it's 'mask*.tif'

This function will also automatically save all the smoothed and merge images under Output_directory. Smoothed images are binary masks with smoother boundaries. Merge images are binary masks with invasive radius, core radius, branch points and centroids labeled in different color, as a reference to check the morphology quantification. All the results matrix and structure will be saved into a "morphology.mat" file. And the tables will be saved both in excel and text format.

Example for this function: *[T_unfill, S_unfill, T_fill, S_fill, single] = featureextraction...*

('z3_ch00', 1.13, 200, 0, [1 5 5], Input_directory,Output_directory);

You need to specify the inputs: ZSliceName, Resolution, SingleCellArea, smooth and plotsetting.

ZSliceName: a string, part of the name of your original fluorescent images. e.g. If you want to select Z5 plane and channel 00, then ZSliceName = 'z5_ch00'. If you want to select Z1 and channel 1, then ZSliceName = 'z1_ch1', according to the name of your original images.

Resolution: resolution of your camera, unit: um/pixel

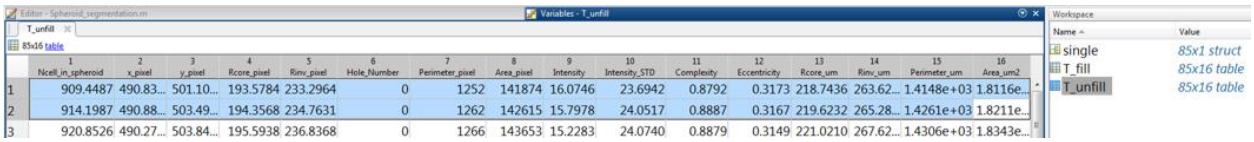
SingleCellArea: threshold value to remove debris smaller than SingleCellArea, unit: um^2; default is 200 um^2, if not specified.

smooth: whether have Ismooth image sequences or not. If not specified, default is 0, which means haven't smooth the binary images before, so the function will perform the smoothing using smoothboundaries function. If smooth is 1, then the function will automatically load the already saved smooth images from Output_directory.

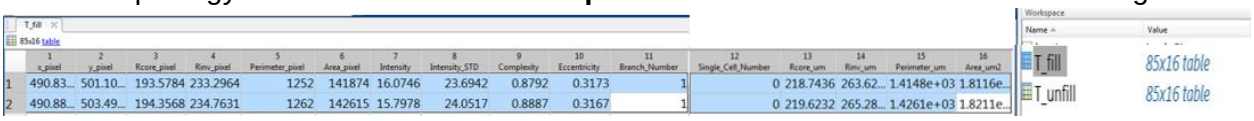
plotsetting: a 3*1 or 1*3 vector, [width szB szC], which are the setting for plot merge images. Default is [2 5 5]. Width is the line width for 2 radius circles, default = 2; szB is the marker size for branch points, default = 5; szC is the marker size for centroids, default = 5.

The outputs are: table (T_unfill) and matrix (S_unfill) for unfilled spheroid; table (T_fill) and matrix (S_fill) for filled spheroid; cell array (single) for single cells outside spheroid.

The morphology features for **unfilled spheroid** are shown in the following table:



The morphology features for **filled spheroid** are shown in the following table:



The morphology features for **single cells** isolated from the spheroid are: area in pixel, coordinates in pixel, mean intensity and mean STD of intensity.

4 Group Data

This part is to put the extracted morphology data of each spheroid into N treatment/cell line groups according to your experiment. The output "Group.mat" will be saved under the "Summary_directory" folder. For example, I have 3 leader spheroids and 3 follower spheroids, so I grouped them into leader vs. follower groups and each group contains 3 spheroids.

[Group, Summary_directory] = groupfeatures(N, Ngi, GroupName)

This function will first pop up a window and let you select the base output folder. For example, my base output is "…/Spheroid/Outputs" (Figure2). Then it will ask you to select the Summary folder for saving your outputs. For example, my summary folder is "…/Spheroid/Outputs/Summary" (Figure2). Finally it will automatically pop up sum(Ngi) window for you to select each morphology.mat file. For example, it will pop up 6 windows (3+3=6), asking morphology.mat for "Group1-#1", "Group1-#2", "Group1-#3", "Group2-#1", "Group2-#2" and "Group2-#3" spheroid.

Example for this function: *[Group, Summary_directory] = groupfeatures(2, [3 3], []);*

You need to specify the inputs: N and Ngi. GroupName is optional for specifying.

N: interger, number of groups. eg. N = 2.

Ngi: a 1*N or N*1 vector, [Ng1 Ng2 Ng3 ...]. eg. Ngi = [3 3]. Ng1/Ng2/Ng3 is number of spheroids in group1/2/3. They can be the same or different integers.

GroupName: a string, the name for saving 'Group.mat' output structure. If not specified, default is 'Group'.

The outputs are:

Group: N*1 structure with 3 fields: 'fill', 'unfill' and 'Ng'. Row1/2/3… is the filled or unfilled morphology data for each group.

- **fill:** For example, I have 3 spheroids in group1, then Group(1).fill is a 3*1 cell, each cell is the filled morphology matrix for each spheroid in group1. The morphology matrix is a matrix with size [timepoint, feature number]. Each column is a feature, and each row is a timepoint.
- **unfill:** Similarly format for 'unfill' field.
- **Ng:** each row is the number of spheroids in each group.

Summary_directory: Output folder for saving Group structure data.

5 Plot, Model Fitting, Statistics

This part is to plot the specific morphology feature among groups as a function of time, then perform curve fitting using 3 models (1st polynomial, 2nd polynomial and exponential), plot fitting curves and compare significant difference of fitting parameters among groups using student's t-test (2 groups) or ANOVA test (>2 groups).

5.1 Plot Timelapse Feature Curves

[F, H1legend] = plotmorphology(Summary_directory, Group, FieldName, FeatureCol, YLabel, FigName, TimeInterval, LegendLabel, Linestyle, Marker, fontsize, Width)

This function is to plot 3 figures for dynamic morphology features. Figure1 is the individual curve for each spheroid in each group. Figure2 is the mean feature curve with shaded STD for each group. Figure3 is the mean feature curve with shaded 95% CI for each group. Figure1 is named as [FigName]; Figure2 is named as [FigName-STD]; Figure3 is named as [FigName-95CI]. Each figure will saved in 3 formats, '.fig', '.tiff', and '.eps' formats under Summary_directory. The output data structure F is also saved under Summary_directory.

Example for this function:

[F, H1legend] = plotmorphology(Summary_directory, Group, 'fill', 14, 'Invasive Radius of Filled Spheroids (\mum)', 'Rinv', 10, {'Leader','Follower'}, {'-','--'}, {'x','none'}, [], []);

You need to specify the inputs: FieldName, FeatureCol, YLabel, FigName, TimeInterval, LegendLabel, Linestyle and Marker. The 1st and 2nd inputs 'Summary_directory' and 'Group' are outputs from the previous function *groupfeatures*, so don't change them. If Summary_directory and Group are not provided, a window will pop up and let you select them. The last 2 inputs 'fontsize' and 'Width' are optional to set, a default setting will be used if not specified.

The explanation for each inputs is:

Summary_directory: Input and Output folder for loading Grouped morphology data and saving output figures and data.

Group: N*1 structure with 3 fields: 'fill', 'unfill' and 'Ng'. See page 9 explanation of the output "Group". If not specified, a window will pop up and let you to select the Group.mat.

FieldName: a string, must be either 'fill' or 'unfill', indicating which types of spheroid feature you want to analyze.

FeatureCol: an integer, the column number of the feature you want to analyze. If not specified, an error will occur, but the feature name for each column will be displayed in the error for you to select the correct feature column.

YLabel: a string, the label for y-axis.

FigName: a string, the name you want to save the figure.

TimeInterval: a number, the time interval (in minute) for image sequence.

LegendLabel: 1*N or N*1 cell. Each cell contains a string, which is the legend label for each group. N is the number of groups. e.g. LegendLabel = {'Leader','Follower','Parental'}.

Linestyle: 1*N or N*1 cell. Each cell contains a string, which is the line style for plotting group1/2/3... curves. e.g. Linestyle = {'-','--',':'}.

Marker: 1*N or N*1 cell. Each cell contains a string, which is the marker for plotting group1/2/3... curves. e.g. Marker = {'x','none','o'}.

fontsize: 1*N or N*1 vector, the fontsize for [3 figures, figure1 legend, figure2/3 legend]. If not specified, the default = [14 10 14].

Width: Linewidth for all curves. If not specified, default = 2.

The outputs:

F: 1*N structure with 15 fields: 'Field', 'FeatureCol', 'feature', 'NormFeature', 'Mean', 'STD', 'CI', 'X', 'Ng', 'YLabel', 'FigName', 'LegendLabel', 'Linestyle', 'Marker', 'Width'.

F(1).field is group1 data, F(2).field is group2 data, F(3).Field is group3 data, etc.

- 'Field': 'fill' or 'unfill' for the feature plotted.
- 'FeatureCol': column of feature from morphology data.
- 'feature': a Nt*Ng matrix for the feature you want to plot. Nt is the timepoint of the sequence. Ng is the number of cell lines (or spheroids) in group1/2/3. eg: F(1).feature is feature data for group1.
- 'NormFeature': a Nt*Ng matrix with the normalized feature data for G1/2/3. The feature value at first timepoint was normalized to the same among all cell lines (or spheroids) within each group. G1_norm(t) = G1_feature(t) – G1_feature(t0) + G1_initial . F(2).NormFeature is normalized feature data for group2.
- 'Mean': a Nt*1 vector of the mean value for the feature you want to plot. eg: F(1).Mean is average feature data for group1 at each timepoint.
- 'STD': a Nt*1 vector of the STD for the feature you want to plot. eg: F(1).STD is std of feature data for group1 at each timepoint.
- 'CI': a Nt*1 vector of 95% confidence interval for the feature. eg: F(1).CI is 95% confidence interval for group1 feature at each timepoint.
- 'X': a 1*Ng vector for timepoints, will be used in the bestfit function as input.
- 'YLabel':label for y axis, will be used in the bestfit function as input.
- 'FigName': name to save the figure.
- 'LegendLabel': 1*N or N*1 cell, legend label string for group 1/2/3. F(1).LegendLabel is the legend label string for group1.
- 'Linestyle': 1*N or N*1 cell, each cell is a string for group 1/2/3 linestyle.
- 'Marker': 1*N or N*1 cell, each cell is a string for group 1/2/3 markers.
- 'Width': LineWidth for all curves.

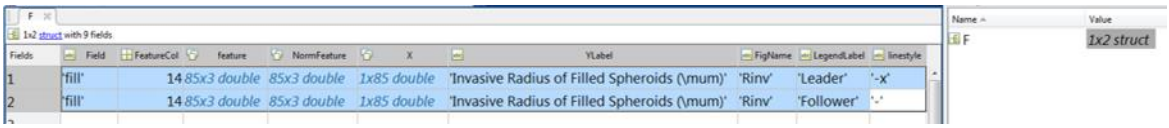


Figure 5. Example of Output Structure F.

H1legend: Ntotal*1 cell with legend for each individual spheroid

5.2 Model Fitting & Statistical Test

[FitParameters, T_rank, best, order, FitMeanParameters, TMean_results]...

= bestfit(Summary_directory, F, H1legend, fitline, fs)

This function is to plot 4 figures for model fitting of dynamic morphology features and save them under Summary_directory. Figure1/2/3 are fitting for all individual spheroids using 3 models: poly1/poly2/exp1. Then perform student's t-test (or ANOVA test) of the fitting parameters between 2 groups (or more than 2 groups) to calculate significant difference. Finally select best fitting models based on the following criteria:

score = adjusted_Rsquare/(mean_p_value_ttest).

Only use the pvalues of those fitting parameters measuring the slope, not intercept among groups, and if the pvalue <=0.05, convert it to 0.05 for calculating the mean_p_value. Highest score is the best model. The outputs of fitting parameters, goodness of fit and best models (for Fig1/2/3) will be saved as [FigName-model.mat];

Figure4 is the fitting of all 3 models for the mean curve+shaded STD for each group. The outputs of fitting parameters, goodness of fit (for Fig4) will be saved as [FigName-mean-models.mat] under Summary_directory.

Example for this function:

[FitParameters, T_rank, best, order, FitMeanParameters, TMean_results]...

= bestfit(Summary_directory, F, H1legend, {'-.','-.'}, []);

The only input you need to specify is: 'fitline'.

The first 3 inputs are the outputs from the previous function *plotmorphology*, you don't need to specify them, just leave them unchanged. If not specified, a window will pop up and let you select the summary directory and F (''Rinv.mat'' or any ''feature name.mat'' you want to fit models.). The last input is optional setting, if not specified, a default will be used.

Explanation for inputs:

Summary_directory: see page 10.

F and **H1legend:** is the same as the output F from previous function *plotmorphology*. See page 12.

fitline: 1*N or N*1 cell, each cell is a string for fitting models' linestyle. eg. fitline = {'-.', '-.', '-.'}.

fs: a 1*3 vector for fontsize for [all figures' fontsize; figure1/2/3 legends' fontsize; figure4 legend's fontsize]. If not specified, the default = [14 8 14].

Outputs for Figure1/2/3:

FitParameters: 1*1 structure with 3 fields, 'poly1', 'poly2', 'exp1'. Each field is a Ntotal * 5 table containing the fitting parameters (p1, p2, p3) and adjusted goodness of fit of each model for each spheroids. Each row is a spheroid, each column is a fitting parameter. Ntotal is the total number of spheroids. This output will also be saved in 3 csv files called 'FitParameters-poly1','FitParameters-poly2' and 'FitParameters-exp1.csv'.

T_rank: 3*6 table, each row is a model. Column A is the score, the higher the better fitting for the model. Column 2 and 3 are mean adjusted Rsquare and mean Rsquare for all spheroids. Column

4, 5, 6 are the ttest (or anova) pvalues for fitting parameters p1, p2, p3 among groups. The order of each row (or each model) is ranked from best to worst. This output will also be saved as a csv file: 'T_rank.csv'.

best: char, the best fitting model with the highest score.

order: 3*1 vector, the rank order number for 3 models (best to worst).

Outputs for Figure4:

FitMeanParameters: N*1 cell for each group. Each cell is a 3*5 matrix containing the fitting parameters for the mean Normal feature in each group. Row Labels are: 'poly1', 'poly2', 'exp1'. Column Labels are the fitting parameters (p1, p2, p3), adjusted Rsquare, and Rsquare for each model. This output will also be saved in 3 csv files called 'FitParameters-Mean-[Group1 Name]','FitParameters-Mean-[Group2 Name]' and 'FitParameters-Mean-[Group3 Name].csv'.

TMean_results: 3*3 table, adjusted Rsquare for each model for each group. Each row is a group and each column is a model. This output will also be saved as a csv file: 'Mean_gof.csv'. This function will also automatically generate 6 ANOVA table figures and 6 ANOVA box plots to show the statistical results, if N_group > 2. You can manually save those figures if you need them.

References

1. Selinummi, J., et al., *Bright Field Microscopy as an Alternative to Whole Cell Fluorescence in Automated Analysis of Macrophage Images.* Plos One, 2009. **4**(10).