
Programs for leaf shape analysis

The programs listed herein have been written in Matlab 2013 (The Mathworks, Inc.) to analyze leaf surface shape in 3D as well as shape deformations over the course of development. They relate to the manuscript by Rolland-Lagan AG, Remmler L and Girard-Bock C, entitled 'Quantifying shape changes and tissue deformation in leaf development', submitted to Plant Physiology.

In order to run the programs, one has to have obtained surface and landmark data in a format compatible with the one presented in Remmler and Rolland-Lagan (2012, *Plant Physiol.* 159:27-39).

We provide, together with the programs, a 'Reference data' folder, which contains surface and tracked beads data from Remmler and Rolland-Lagan (2012, *Plant Physiol.* 159:27-39). This will enable a user to test run the set of programs 'as is' using those data.

If the user wants to use the same programs with another dataset, the programs provided here will likely have to be adapted (for instance with respect to the time points analyzed, which are in most cases hard coded in the programs). Also, the user will have to ensure surface and landmark data used are saved in a compatible way (same variable names and format, same folder hierarchy), or adapt the programs to handle the data accordingly. Programs were written and used on computers with Microsoft Windows as operating system.

Programs do computations, save quantitative data and make figures when relevant. They only require user input to prompt the user for the location of folder or files to use to do the computations or generate the figures, or to select different types of computations options. Programs have to be run in order. To run a program, type its name at the command prompt in the Matlab command window. To view source code, type edit followed by the name of the program at the command prompt in Matlab, or open with a text editor.

Note: when a .fig file is open, the color scale can be displayed using the command colorbar (type colorbar at the command prompt). Axes can be displayed using the command axis on (type axis on at the command prompt).

For any questions, please email arolland@uottawa.ca.

Important:

- Before running the programs, make sure you have set the path to the programs folder.

This can be done by typing the following at the Matlab command line:

```
>>addpath(uigetdir(cd,'Select folder containing the programs'));
```

- If you get the error
??? Undefined function or variable 'FunctionName'
Check that you did not make a typo in the function name. Note that function names are case sensitive.

Contents

Part 1: 3D LEAF SHAPE AND CURVATURE ANALYSIS	3
MeanLeaf_extract	3
Sample_cross_sections.....	3
Mean_cross_sections	3
Show_cross_sections.....	3
Localcurvature	3
Display_Characteristicleaf_heights_in2D.....	3
Display_local_curvature_data	3
Disp_3Dmeshes_characteristicleaves	3
Makemovie_3Dmeshes	4
Part 2: DEFORMATION GRIDS PATTERNS AND DEFORMATION GRID GROWTH CALCULATIONS	4
Compute_deformation_patterns	4
Display_Deformations_DaytoNext	4
warpleaves_ShapePaper	4
MeanLeaf_extract_fordeformation	5
Display_deformations_and_growth_daytonext3D	5
Display_deformations_forgridmovie3D	5
Makemovie_3Dgriddeformations	6
Deform_across_days	6
NOTES ON VISUALIZATION	6

Part 1: 3D LEAF SHAPE AND CURVATURE ANALYSIS

MeanLeaf_extract

Program to compute a true mean and a characteristic leaf shape for each time point from all samples available at that time point. Uses samples warped to the mean leaf shape computed from all samples available at each time point.

From the mean and characteristic leaf fit shapes we can then calculate cross sections along proximodistal or mediolateral axis and compare those to cross sections of all samples to see how mean and best fit leaf shapes fit the dataset. We can also compute spatial patterns of local curvature. All those are computed in the programs listed below.

Sample_cross_sections

Program to compute cross sections for individual samples (samples are warped to the mean shape of their relevant time point for alignment purposes)

Mean_cross_sections

Program to compute cross sections for the mean and the mode leaves.

Show_cross_sections

Program to display cross sections along the proximo-distal or medio-lateral axes (user is prompted to choose which to display). Displays cross sections for all samples, and superimpose cross sections for mean leaf shape and characteristic leaf shape.

Localcurvature

Program to compute curvature patterns for characteristic leaf surface (principal curvatures, mean curvature, Gaussian curvature). Part of this program uses code slightly adapted from code written by Dumais and Kwiatkowska (Dumais J and Kwiatkowska D (2002) The Plant Journal 31: 229-241).

Display_Characteristicleaf_heights_in2D

Used for paper to make supplemental figure of characteristic leaf surface in 2D colour coded for height of leaf heights in 2D.

Display_local_curvature_data

Program to make figures of local patterns of Gaussian and Mean curvature.

Disp_3Dmeshes_characteristicleaves

Program to make figures of 3D leaves as meshes.

The program has to be run as `Disp_3Dbestfitshape(colines,colbkg,opt);` where `colines` is the line color, `colbkg` is the background color, and `opt = 1` overrides the chosen line color to display lines based on z-value (i.e. leaf height). If not input parameters are provided, default values are used to show a black grid on a grey background.

Examples:

```
>> Disp_3Dmeshes_characteristicleaves([0 0 0],[0.7 0.7 0.7],0);  
>> Disp_3Dmeshes_characteristicleaves([0 0 0],[1 1 1],1);
```

Makemovie_3Dmeshes

Program to create a movie of leaves in 3D. User is prompted to give the days to include in the movie. For instance if the user gives [7,19], this will make a movie of characteristic leaves at days 7 and 19 (both times points will be included in the same movie).

Part 2: DEFORMATION GRIDS PATTERNS AND DEFORMATION GRID GROWTH CALCULATIONS

Compute_deformation_patterns

(or use `Compute_deformation_patterns_lwm`: same but computes a lwm transformation instead of a third order polynomial transformation)

Program to compute mean deformation patterns, based on samples for which we have bead displacements from 1 day to the next. It also saves the mean outlines of samples used to compute the deformation: saves mean outlines predeformation and post deformation.

`Avxpredef{t}` and `Avypredef{t}`: mean of samples at time `t` used to compute deformation from `t` to `t+1`

`Avxpostdef{t}` and `Avypredef{t}`: mean of samples at time `t` used when computing deformation from `t-1` to `t`.

Note: Run `Compute_deformation_patterns(1)` to show intermediate figures as the program is running.

NOTE: In subsequent programs, the user is prompted to indicate whether `Compute_deformation_patterns` or `Compute_deformation_patterns_lwm` was used.

Display_Deformations_DaytoNext

Program used to display deformation grids from 1 day to the next user is prompted to indicate whether `Compute_deformation_patterns` or `Compute_deformation_patterns_lwm` was used at the previous step.

warpleaves_ShapePaper

Program used to warp (time `t` samples used to compute deformation patterns from `t` to `t+1`) to the average leaf shape of those samples (`Avxpredef{t}`, `Avypredef{t}`). It saves warped outlines in `warpedleaf3D_preddef.mat` in each relevant sample's folder in the Results folder of the growth experiment dataset generated for growth analysis (Remmler and Rolland-Lagan, 2012).

This later makes it possible to compute the 3D characteristic leaf shape for each deformation grid at time t within a t_t+1 deformation pair.

In the same way, this program also warps (time t samples used to compute deformation patterns from $t-1$ to t) to the average leaf shape of those samples (`Avxpostdef{t}`, `Avypostdef{t}`), and saves warped outline in `warpedleaf3D_postdef.mat` in each relevant sample's folder in the Results folder of the growth experiment.

This later makes it possible to compute the 3D characteristic leaf shape for each deformation grid at time $t+1$ within a t_t+1 deformation pair.

(Note: user is prompted to indicate whether `Compute_deformation_patterns` or `Compute_deformation_patterns_lwm` was used earlier - note that this is only needed because those programs both save the average outlines from samples pre and post deformation, and the matrices in which those are saved have different names depending on which program was used. However the outlines are the same with both methods, so if one wants to test the 2 types of transformations, `warpleaves_ShapePaper` does not have to be run twice).

MeanLeaf_extract_fordeformation

Program used to compute characteristic and mean leaf shapes based on sets of leaves pre and post deformation, respectively. This saves the true average and characteristic leaf shape for each deformation grid at time t within a t_t+1 deformation pair, and the true average and characteristic leaf shape for each deformation grid at time $t+1$ within a t_t+1 deformation pair.

Results are stored in cell array format such that `modeleaf{day,1}` = leaf at day t within a $t,t+1$ deformation pair, `modeleaf{day,2}`=leaf at time t within a $t-1,t$ deformation pair.

(Note both true mean and characteristic leaf shapes are saved but only the characteristic leaf shape is used in subsequent programs).

The data can then be used to show, for each deformation pair, the mean leaf grid in 3D before and after deformation. We can then compute, for each square of the grid, its growth from time t to time $t+1$ using the deformation of the square in 3D space. We can then compare the resulting growth patterns to the growth patterns computed from 3D particles positions in Remmler and Rolland-Lagan, 2012.

(Note: user is prompted to indicate whether `Compute_deformation_patterns` or `Compute_deformation_patterns_lwm` was used earlier - note that this is only needed because those programs both save the average outlines from samples pre and post deformation, and the matrices in which those are saved have different names depending on which program was used. However the outlines are the same with both methods, so if one wants to test the 2 types of transformations, `warpleaves_ShapePaper` does not have to be run twice).

Display_deformations_and_growth_daytonext3D

Display program to show grid deformation from one day to the next in 3D and/or to show growth maps from 3D data in 2D or 3D - Used to make growth maps and to make 3D figures. User is prompted to answer a few questions regarding display options.

Display_deformations_forgridmovie3D

Modified version of the above to save figures of 3D grid deformations in format good to make a movie (those figures are used by the next program to make a movie).

Makemovie_3Dgriddeformations

Program to make a movie of grid deformations from multiple 3D figures.

Each figure opens one at a time and the user is prompted to click where he/she wants the figure label to appear (for instance, click at the bottom left so that the label appears there on the movie - label position is user-determined separately for each loaded figure).

Deform_across_days

To display grid deformations across time points without resetting the grid (the grid from the first time point is deformed through all subsequent time points).

Figures named Figdef0.fig, Figdef1.fig etc... show a display in 2D of the deformation across time points. The green outline shows how the original outline (of the 1st time point) deforms through subsequent time points. It is normal that over time this outline shifts away from the outline of the leaf at each time point for 2 reasons: 1) small errors accumulate from time point to time point (i.e. in the day to day deformations the deformed outline is very close to the actual outline of the next time point but it is not exactly the same); 3) the deformed outline originate from the deformation of the sample outlines from the 1st time point. The samples at a later time point *n* may not be derived from the same plants (unless the exact same set of leaves can be followed through all time points). Moreover, part of the leaf outline at the 1st time point may be missing (e.g. at DAS 7 the very bottom of the blade may not have been extracted, this 'missing part' will still be missing in the deformed outline across time points).

Figures named Figure1.fig, Figure2.fig etc... will be displayed in 2D or 3D depending on the user's choice. Figures with odd numbers are colour-coded for growth and show relative growth to the next time point (relative growth is defined as in Remmler and Rolland-Lagan, 2012). Figures with even numbers show the grid post deformation and are not coloured but show the position of the deformed outline.

Choices given to the user include:

- the choice of days to display deformations across (the more time points are included, the more errors will accumulate).
- whether to display deformations in 2D or 3D (this will only apply to Figure1.fig, Figure2.fig... etc...).
- if display is in 2D, whether to display all grid lines or a subset of grid lines on the odd-numbered figures (this is to make it easier to visualize growth data - all grid lines are displayed in the even-numbered figures as those are not coloured for growth).
- if display is in 3D, whether to display grid lines or no grid lines on the odd-numbered figures.

NOTES ON VISUALIZATION

Figures in the .fig format can easily be modified in terms of visualization options within Matlab.

For instance, the colour axis can be changed with the command `caxis`.

For instance, if a figure is open and displays growth colour-coded from red to blue for values going from 0 to 180 and the user wants to see growth colour coded from 0 to 90, he/she can type the following in the command window:

```
>>caxis([0 90]);
```

The colormap can also easily be changed, which may be useful for colour blind people.

For instance, to change the colormap to greyscale, type the following in the command window:

```
>>colormap(gray);
```

The command:

```
>>axis equal
```

Ensures that all axes are to the same scale

Axis limits can easily be reset. For instance the following command will set axes limits from 0 to 100 on the x axis, from 0 to 200 on the y axis, and from -50 to 50 on the z axis:

```
>>axis([0 100 0 200 -50 50]);
```

```
axis([0 100 0 200 -50 50]);
```

Last, figures can be rotated in 3D by clicking on the following icon of the Figure menu and moving the cursor of the mouse across the figure to move it around:

