**Quantification of neck deepening, local deepening speed and local curvature**

Requirements

- Install **Fiji/ImageJ** (this pipeline was tested using ImageJ 1.51h, Java 1.8.0\_66 (64bit), website: <http://imagej.nih.gov/ij>)

- Install **Matlab** (this pipeline was tested using 2018b Matlab version

**A) Preprocessing**

(projections, neck tracking, drift correction, space and time alignment)

1. Input data: raw imaging data

For each imaging session, the initial dataset is a list of hyperstacks*(x,y,z)* for many timepoints *(t)* and several animals.

1. Top projection and tracking of neck leading edge on the *(x,y)* plane
   1. Top projection (‘**0\_BatchMaxProjection.ijm’**)

In Fiji, run ‘**0\_BatchMaxProjection.ijm’**.

Indicate the folder where the data of the imaging session is located (contains a list of hyperstacks*(x,y,z)* for many timepoints and several animals). This program will automatically perform the maximal intensity projection of each hyperstack and save the projections*(x,y)* in an output folder.

* 1. Assemble top projections movies (manual)

For each animal, assemble all top projections*(x,y,t)* in a single file named ["Max\_s” + *i* + “.tif”], *i* being the number of the considered stage matching the numbering of the microscopy data (1,2,…,*i*). For all animals, save the top projection movie*(x,y,t)* in a temporary folder called ‘Analysis’.

* 1. Tracking of the neck position on the *(x,y)* plane (manual)

Using Fiji, for each top projection movie*(x,y,t)*, manually track the position of the neck leading edge using the Segmented Line tool (add each tracked line to the ROIManager to create a list of ROIs containing all the positions of the leading edge). Save the ROIManager content as [“Leading-edge\_s” + *i* + “.zip”] in the ‘Analysis’ folder for each analyzed animal.

* 1. Segmentation of the midline for calculating the x-y tilt of the stage (manual)

Using Fiji, for each top projection movie*(x,y,t)*, draw a line on the midline on the first frame of the movie, from head to thorax, and add the ROI to the ROIManager. Save the ROI as [“Reslice\_s” + *i* + “.roi”] in the ‘Analysis’ folder for each analyzed animal.

1. Generation the transverse view movie*(x,z,t)* (‘**1\_TransverseViewGenerator.ijm’**)

In Fiji, open ‘**1\_TransverseViewGenerator.ijm**’. Fill in the name of the hyperstacks and the considered stages in the Parameter section.

Run ‘**1\_TransverseViewGenerator.ijm**’. Indicate both the path where the raw imaging data is stored and the ‘Analysis’ folder, which contains top projections, *x-y* tracking of neck leading edge and midline positions.

This program will generate the transverse view*(x,z,t)* of neck leading edge, correcting for the *x-y* rotation of the stage and taking into consideration only the data centered around the *x-y* tracking of the neck leading edge. It will as well generate a sagittal resliced and transverse views movie of the head and thorax sides.

[Observed bug: the transverse views are sometimes generated without rotating the image, i.e. without correcting the *x-y* tilt of the stage. To correct for this lack of correction of the *x-y* tilt, patches were added in subsequent code (in grey in this ReadMe doc)].

1. Reorganization of the data in animal subfolders (manual)

For each stage/animal, create a folder named [NameCondition + *i*] (ex: Control1, Control2, …). In this folder, drop the top projection, the tracking ROIs, the midline ROI, the 3 transverse views and the sagittal view.

1. Tracking of the neck leading edge on transverse view movie*(x,z,t)* (manual)

Using Fiji: For each animal, open the neck leading edge transverse view [“SideView\_s” + i + “.tif”]. For each timepoint in which neck leading edge is still visible, manually track the apical position the neck leading edge using the Segmented Line tool. Save the obtained ROIs list as [“Leading-edge\_s” + *i* + “.zip”].

(Similarly, when possible, basal position of the neck leading edge is tracked and saved in [“Basal\_s” + *i* + “.zip”].)

1. Get the coordinates of the tracked points (‘**2\_GetCoordinates.ijm**’)

In Fiji, run ‘**2\_GetCoordinates.ijm**’, selecting as an input the master folder containing all the animal subfolders. The code will read the manual tracks and save the clicked coordinates so that manual tracking can be processed by Matlab in subsequent steps.

(For basal tracking, run ‘**2bis\_GetCoordinatesBasal.ijm**’.)

1. Determination of the *x-z* and *y-z* tilt (‘**3\_XZMidlineAngleDetermination.ijm**’and manual)

In Fiji, run ‘**3\_XZMidlineAngleDetermination.ijm**’, selecting as an input the master folder containing all the animal subfolders. This program will open the transverse projection midline movie*(x,z,t)* of the head, and assist the user in measuring the *x-z* tilt of the midline on this movie (the result is saved in the output vector ‘CenterCalculation.csv’).

In Fiji, open the midline reslice, measure the tilt angle (usually this angle=0, but it can be ≠0 for imaging without coverslip flattening), and save it as ‘Angle.csv’.

1. Spatial alignment on top projections (‘**4\_SpatialAlignment.ijm**’)

In Fiji, run ‘**4\_SpatialAlignment.ijm**’, selecting as an input the master folder containing all the animal subfolders. This program will assist the user in clicking on the spatial landmarks enabling to align data along the ML axis (2 landmark macrochaetes on the head, and midline position), and save the coordinates of the clicked landmarks in ‘alignement.csv’.

[Patch:

Run ‘4bis\_PATCHSpatialAlignmentWithoutRotation.ijm’ and ‘4ter\_PATCHGetRotationAngle.ijm’. These programs assist the user in clicking on spatial landmarks in images that were not corrected for the *x-y* tilt of the stage, and allow to collect the *x-y* tilt of the stage. These values, saved in ‘alignementWithoutRotation.csv’ and ‘AngleRotation.csv’ will be used instead of the values stored in ‘alignement.csv’, to correctly align in space data in which *x-y* tilt was not corrected for due to the bug that sometimes appear in **1\_TransverseViewGenerator.ijm’**]

1. Drift correction (Manual and ‘**5\_DriftCsvWriter.ijm’**)

A small *x-z* drift can be observed on transverse views, mainly due to instabilities in *z* as a result of the use of an autofocus program during acquisition and due to the natural drift of the sample.

For each animal folder, track a non-moving point located on the apical ECM on transverse view (either on the head or the thorax side, or possibly merge head and thorax views to track a single point), using the Point tool in Fiji. Save the obtained ROI list as ‘Drift.zip’. It is not necessary to track the reference particle for timepoints at which the particle is not moving as compared to previous tracked timepoint (i.e. not all timepoints need to be tracked). Up to 3 particles can be tracked for a movie, to cope with the fact that the tracked particle sometimes disappears during the course of the movie. In that case, make sure that the last timing tracked for the particle 1 is the same as the first timing tracked for particle 2, and save the tracking as ‘Drif1.zip’ and ‘Drift2.zip’ (and if needed ‘Drift3.zip’).

Run ‘**5\_DriftCsvWriter.ijm**’. This program will read ‘Drift.zip’ (or ‘Drift1.zip’, ‘Drift2.zip’ and ‘Drift3.zip’), calculate the drift for each frame and write it in ‘DriftX.csv’ and ‘DriftY.csv’.

1. Time alignment (manual)

For each animal, using the top projection, the user detects the frame at which microchaetes do their final division in the anterior thorax: this frame is defined as 22hAPF. Using this reference timing, the user then creates a ‘timing.csv’ file, containing a column listing the different timing of each frame, and save it in the animal folder.

**B) Calculation of deepening, local deepening speed and local curvature**

Calculation of deepening, deepening speed and curvature in time and along the M-L axis *(calculation time for the provided example without parallelization ≈ 20-30min)*

In Matlab, open ‘**NormalSpeedCurvatureCalculation.m**’. In the code, fill the pathway containing all the animal folders, the conditions names and the indexes of the animal for each condition (an example ‘Control1’ is provided in ‘Data’ folder). Define an output folder. Add ‘Functions’ folder to Matlab path.

For each animal, this program will use the tracking, drift and spatio-temporal landmark data to:

1. Interpolate the tracking to get an evenly spaced tracking for each timepoint, and pseudo-track each point of neck leading edge in time, and associate to it a coordinate PositionML (%),
2. calculate neck deepening as a function of PositionML and time,
3. calculate local curvature and normal deepening speed as a function of PositionML and time (these values are smoothed in time for reducing noise).

For each condition, the output is 3 3d matrices: and Depths*(PositionML,time,animal)*, Curvatures*(PositionML,time,animal)* and Speeds*(PositionML,time,animal)*.

**C) Plots**

1. Plot deepening curves of mutant/ablation and associated controls

In Matlab, open ‘**PlotDeepeningCurves.m**’. In the code, indicate the path were the deepening values are stored: ‘Data\Values’ (this folder contains the output of ‘**NormalSpeedCurvatureCalculation.m**’ for all mutant/ablation and control conditions). Add ‘Functions’ folder to Matlab path.

For each condition, this code will read deepening matrices*(PositionML,time,animal)*. For each animal, it will generate a deepening curve*(time)* averaged along the MLaxis. The program will then calculate the average curve (+-sem) among the animals of a same condition. Finally, it will plot and do statistical comparison for each timepoint (Welch tests) for the mutant/ablation conditions and their associated control condition.

1. Plot local normal deepening speed as a function of local curvature (or as a function of the tension × local curvature product). Extract slopes of *speed*=f(*tension×curvature*) plots, and plot deepening speed and curvature curves

In Matlab, open ‘**SpeedCurvatureTensionPlots.m**’. In the code, indicate the path were the speed, curvature and recoil velocities values are stored: ‘Data\Values’ (this folder contains the output of ‘**NormalSpeedCurvatureCalculation.m**’ for all mutant/ablation and control conditions as well as recoil velocity after laser ablation results). Add ‘Functions’ folder to Matlab path.

For each condition, this code will read deepening speed and curvature matrices*(PositionML,time,animal)*, as well as recoil velocity after ablation vectors*(time)*. For each condition, it will then plot (i) the average curvature*(PositionML, time)* against the average deepening speed*(PositionML,time)*, (ii) the curvature × recoil velocity product*(PositionML, time)* against the average deepening speed*(PositionML,time)* for the apical and basal sides, (iii) it will calculate the slope of the plot [deepening speed = f(curvature × recoil velocity) ].

This code also enables to plot and compare (i) the average speed*(time)* and curvature*(time)* of flattened and non-flattened animals, (ii) the average speed*(time)* and curvature*(time)* of the flattened side and the non-flattened side of animals flattened on one side only, (iii) plot the average speed*(time)* and curvature*(time)* of animals in which medial tissue is mechanically isolated from lateral tissue by laser ablations.