Documentation
Arrested and Gelated Structures Lab

Version 0.9

Jasper Immink

May 30, 2021

Contents

1   Introduction

This document is a manual for users of Arrested and Gelated Structures Laboratory, or ArGSLab for short. ArGSLab is a tool for extracting backbone skeletons from gelated or arrested network structures, consisting of individual segments. ArGSLab extracts a skeleton from .tiff image stacks or coordinate diagrams, which is then in turn transformed into a graph of nodes and links. ArGSLab provides various methods to visualize the skeleton, the graph, and the eventual data. Furthermore, ArGSLab allows for a thorough quantitative analysis, and exports the graph for analyses tailored to the specific user.

In this document, we discuss each parameter and how it affects the results, we show the output data generated by ArGSLab, we show a flowchart of the code pipeline, and then we discuss most scripts individually.

Should you use ArGSLab for your own publications, please cite our accompanying article.

2   Individual Parameters

We will first discuss each parameter for the Standalone version, and then discuss the same for the GUI version. The latter are often identical to a parameter in the Standalone version.

2.1   Standalone Parameters

Switches

`switch_tortuosity` Turns on and off whether the tortuosity and related parameters are calculated.

`switch_linklength` Turns on and off whether the link length probability and related parameters are calculated.

`switch_visualize` Turns on and off whether a set of standard visualizations are produced and saved. More details are found in Section 3.

`switch_outputlabeling` Turns on and off whether timestamps and process updates are printed.

`switch_optimizationtools` Turns on and off whether a set of visualizations are produced and saved, which are intended for optimizing ArGSLab parameters. More details are found in Section 3.

`switch_exporttxt` Turns on and off whether output files are created in .txt format.

Main input variables

`projectpath` The folder path wherein the to be analyzed data resides.

`exclude` Disregards all files in `projectpath` that exist in `exclude`.

`zframes` Allows the user to specify a section of the .tiff stacks to be analyzed, with `zframes` the last slice to be analyzed. If `'all'`, every slice is analyzed.

`realpx` Sets the voxel pitch in units of μm.

`particlediameter` Sets diameter of a single particle or gel segment in units of μm.

`BoxSize_input` In the case of input data being coordinate diagrams, this sets the x, y, and z positions of the input data box edges, in units of the input data. In the input data, the box should start at 0.

`BoxSize_input` In the case of input data being coordinate diagrams, this sets the x, y, and z positions of the box edges of the generated image, in units of σ.

`Point_spread_function` Allows for the optional introduction of a generated point spread function with the given x, y, and z dimensions for generated images. In units of σ.

Pre-skeletonization parameters

`sigmablur` Sets the edge size of the cubic Gaussian blur kernel, with the kernel being 2*`sigmablur` +1 voxels.

`binThres` Sets the binarization threshold. If `'auto'`, Otsu's method is used; otherwise a number from 0 to 1 should be input. The precise process is described in more detail later (Section 5.3) in the Article.

`sesize` Sets the radius of the 3D structuring element for morphological closing, in units of single voxels.

`minfrac` Removes all structures in the image that are unconnected, and have a volume smaller than `minfrac` * volume of the largest structure.

Skeletonization parameters

`term_skel_thr` Sets the terminal branch pruning threshold. All branches that are only once connected to a structure, and are smaller than `term_skel_thr` are removed. Units of σ. More detailed information can be found in Section 5.2, and in the Article.

`coll_skel_thr` Sets the node collection threshold. All nodes are collected in a set of nodes, with nodes that exist in the vicinity (cut-off distance `coll_skel_thr`) of another node collected in the same set. These sets are then replaced with a single node. Units of σ. More detailed information can be found in Section 5.2, and in the Article.

Tortuosity parameters

`edgenode_dist` Defines a distance from the box edge, such that if nodes lie close than this distance from a box edge, they will be considered to be crossing the box edge. More detailed information on our definition of the tortuosity can be found in the Article.

Link length parameters

`binsize` Defines the amount of bins in the output of the $\tilde{N}(\Lambda)$.

`maxbin` Defines the maximum bin in the output of the $\tilde{N}(\Lambda)$, in units of σ.

Visualization parameters

`visualize_edges` Turns on or off whether skeleton nodes and links in the edge region (defined with `edgenode_dist`) are visualized.

`output_type` Defines whether the rotatable visualizations are exported as matlab files, or as .vtk files, for use in a program like ParaView.

`switch_cubify` Switches whether visualizations are exported with the voxel pitch set to unity, or whether to use the voxel pitch as defined by `realpx`.

`swellskel` Dilates visualized skeleton links with a cubic structural element with edges of `swellskel` voxels.

`nodesize` Dilates visualized skeleton nodes with a cubic structural element with edges of `nodesize` voxels.

2.2 GUI Parameters

General Settings

Figure 1: General Settings tab.

Project Folder button: Equivalent to `projectpath`.

Exclude files from project button: Equivalent to `exclude`.

Max z-frame: Equivalent to `zframes`.

Pixel size; X, Y and Z: Equivalent to `realpx`.

Particle diameter: Equivalent to `particlediameter`.

Reset default values: Resets all values to values described in the article.

Coordinate Diagram

Figure 2: Coordinate Diagram tab.

Input box sizes; X, Y, and Z: Equivalent to `BoxSize_input`.

Output box sizes; X, Y, and Z: Equivalent to `BoxSize_output`.

Point spread function; X, Y, and Z: Equivalent to `Point_spread_function`.

Full Analysis
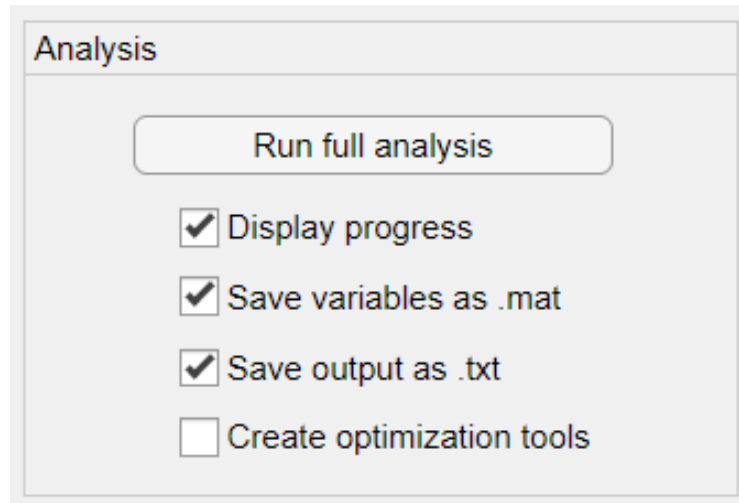

Figure 3: Full analysis tab.

Run full analysis button: Runs all of ArGSLab functionalities.

Display progress: Equivalent to `switch_outputlabeling` set to `'yes'`.

Save variables as .mat: Saves all relevant variables in a .mat file.

Save output as .txt: Saves all relevant output as .txt files.

Create optimization tools: Equivalent to `switch_optimizationtools` set to `'yes'`.

Preprocessing

Figure 4: Full analysis tab.

Sigmablur: Equivalent to `sigmablur`

Automatic binarization threshold: Equivalent to `binThres` set to `'auto'`.
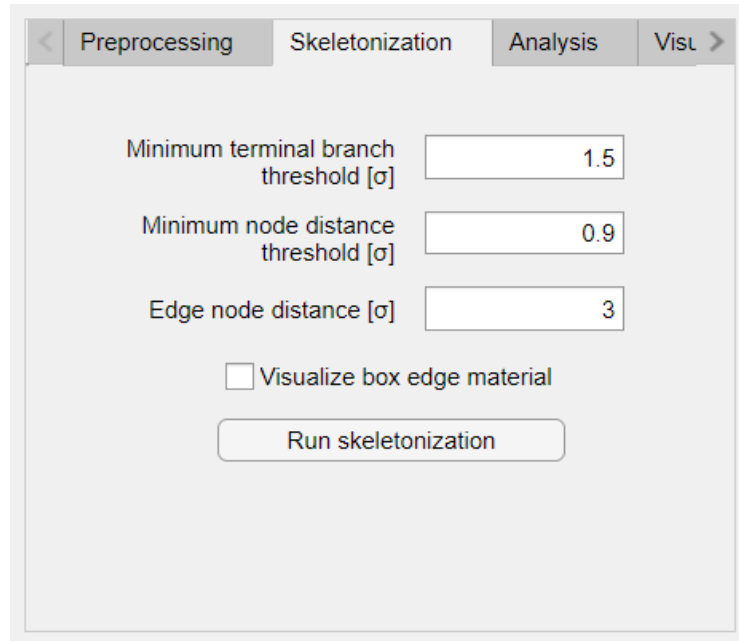
Binarization threshold: Equivalent to `binThres` not set to `'auto'`, and a value between 0 and 1.

Structuring element size: Equivalent to `sesize`.

Minimum fraction: Equivalent to `minfrac`

Run preprocessing: Run only the preprocessing stage on all the to be analyzed files.

Skeletonization

Figure 5: Skeletonization tab.

Minimum terminal branch threshold: Equivalent to `term_skel_thr`.
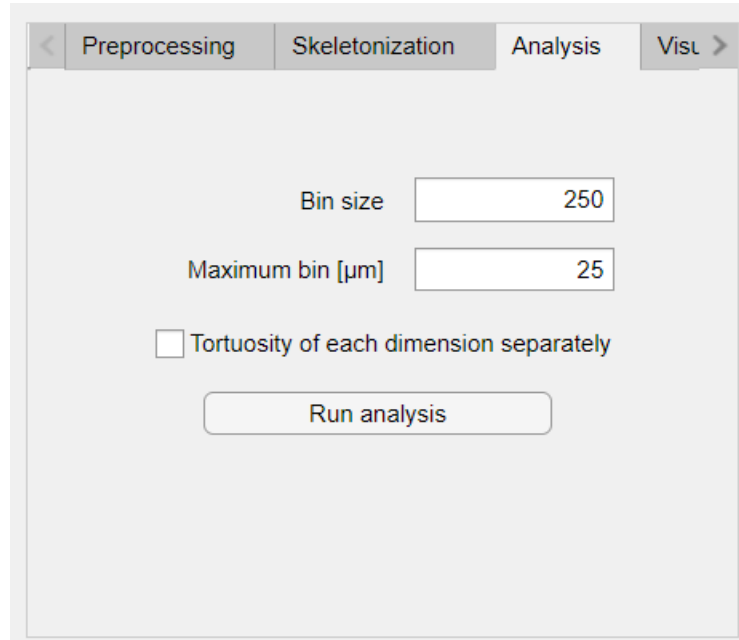
Minimum node distance threshold: Equivalent to `coll_skel_thr`.

Edge node distance: Equivalent to `edgenode_dist`.

Visualize box edge material: Equivalent to `visualize_edges`.

Run skeletonization: Run only the skeletonization stage on all the to be analyzed files.

Analysis

Figure 6: Full analysis tab.
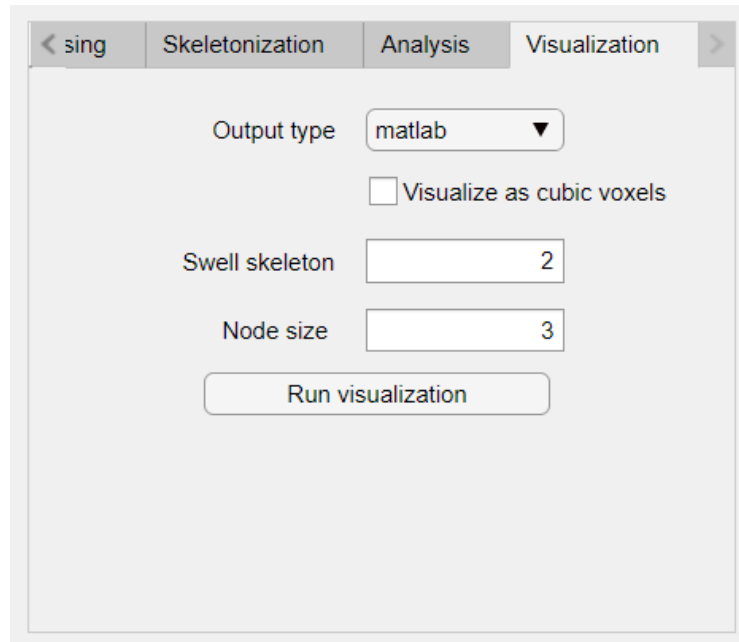
Bin size: Equivalent to `binsize`.

Maximum bin: Equivalent to `maxbin`.

Tortuosity of each dimension separately: Exports the tortuosity output for each dimension separately. Done by default in the Standalone version.

Run skeletonization: Run only the analysis stage on all the to be analyzed files.

Visualization Output

Figure 7: Visualization configuration tab.

Output type: Equivalent to `output_type`.

Visualize as cubic voxels: Equivalent to `switch_cubify`.

Swell skeleton: Equivalent to `swellskel`.

Node size: Equivalent to `nodesize`.

Run visualization: Run only the visualization stage on all the to be analyzed files.
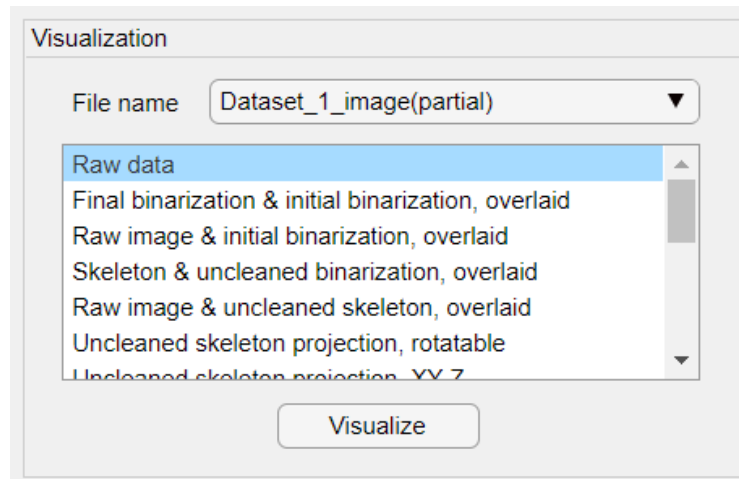
Visualization

Figure 8: Visualization tab.

visualizations will appear in the menu. Under file name, you can choose a specific file, in the drop down menu one can choose a specific visualization, and by pressing the *Visualize* button, ArGSLab will depict the requested visualization.

3   Output data

ArGSLab produces a folder for each data file treated, which contains a number of files discussed below. It also produces some files with average values for all treated systems. One can turn their production on and off with switches discussed in section 2.1.

Files generated for individual input files

Matlab variable files

These files are only generated as optimization tools. *Analysis_variables.mat*: Contains link length, tortuosity and density variables.

Skeleton_variables.mat: Contains the skeleton array, the `Nodelist` and the `Linklist` variables. The skeleton array is a logical array with the skeleton voxels denoted by 1. The `Nodelist` and `Linklist` variables constitute the graph. Each node and link are given an id, and `Nodelist` contains the node position index, the links it is connected to, the nodes that it is directly connected to, the x-, y- and z-coordinates, whether it is an endpoint or not, and the center of mass of the node. `Linklist` contains the id's of the nodes it connects, the voxel indices it comprises, and the length of the link in units of µm.

Preparation_variables.mat: Contains the logical binarized arrays from before and after the preprocessing step. Tool for optimization.

Plot files

Nodedistance.eps: A plot of the $\tilde{N}(\Lambda)$.

Nodedistance.fig: A plot of the $\tilde{N}(\Lambda)$ as a Matlab .fig file.

Intensity_alongz.fig: Shows the cut-off intensity as a function for the x-axis, above which pixels are considered to contain material, and below which they are considered to not contain material. Included are the intensity above which the 10% most intense pixels fall, and the intensity below which the 10% least intense pixels fall (I_{90} and I_{10}, respectively, as described in Section 5.3). Tool for optimization.

Filledvoxels_alongz.fig: Overlay of the raw image with the uncleaned skeleton. Tool for optimization.

Visualization files

Binarized_cleaning.tif: Figure 3C in the article. Shows the effects of further preprocessing after binarization, i.e. the morphological closing, hole filling, and the removal of smaller unconnected structures. White voxels contain material that is unaffected by preprocessing, green areas contain material that was added by preprocessing, red areas no longer contain material after by preprocessing, and black areas never contained material.

Binarized_skeleton.tif: Shows an overlay of the preprocessed, binarized image with the final skeleton in green, and the final nodes in blue.

Post-skeletonization_uncleaned.tif: identical to Binarized_skeleton.tif, but with the skeleton before cleaning steps. Tool for optimization.

Rawimage_binarized.tif: Figure 3B in the article. Raw input image overlaid with a green and red color filter, produced in the binarization step. Green indicates voxels determined to contain material (material voxels), and red to contain no material (non-material voxels). Black voxels had little to no intensity in the original image.

Rawimage_skeleton.tif: Figure 3D in the article. Shows an overlay of the raw image with the final skeleton in green, and the final nodes in blue.

Rawimage_skeletoncleaning.tif: Overlay of the raw image with the uncleaned and cleaned skeletons. Skeleton remaining after cleaning in green, removed in red; nodes remaining after cleaning in blue, removed in purple. Tool for optimization.

Rawimage_uncleaned-skeleton.tif: Overlay of the raw image with the uncleaned skeleton. Tool for optimization.

Projection visualization files (Matlab)

Projection_cleaned.fig: Rotatable 3D projection of the complete, cleaned skeleton, as a matlab .fig file.

Projection_xyz_cleaned.png: 2D image of the cleaned 3D skeleton, projected along the xyz axis.

Projection_xy-z_cleaned.png: 2D image of the cleaned 3D skeleton, projected along the xy-z axis.

Projection_-xyz_cleaned.png: 2D image of the cleaned 3D skeleton, projected along the -xyz axis.

Projection_-xy-z_cleaned.png: 2D image of the cleaned 3D skeleton, projected along the -xy-z axis.

Projection_uncleaned.fig: Rotatable 3D projection of the complete, uncleaned skeleton, as a matlab .fig file. Tool for optimization.

Projection_xyz_uncleaned.png: 2D image of the uncleaned 3D skeleton, projected along the xyz axis. Tool for optimization.

Projection_xy-z_uncleaned.png: 2D image of the uncleaned 3D skeleton, projected along the xy-z axis. Tool for optimization.

Projection_-xyz_uncleaned.png: 2D image of the uncleaned 3D skeleton, projected along the -xyz axis. Tool for optimization.

Projection_-xy-z_uncleaned.png: 2D image of the uncleaned 3D skeleton, projected along the -xy-z axis. Tool for optimization.

Projection visualization files (VTK)

nodes.vtk: VTK file containing the polygon object projection for the nodes of the cleaned skeleton, for use in a reader like ParaView.

skeleton.vtk: VTK file containing the polygon object projection for the cleaned skeleton, for use in a reader like ParaView.

nodes_uncleaned.vtk: VTK file containing the polygon object projection for the nodes of the uncleaned skeleton, for use in a reader like ParaView. Tool for optimization.

skeleton_uncleaned.vtk: VTK file containing the polygon object projection for the uncleaned skeleton, for use in a reader like ParaView. Tool for optimization.

Text files

Skeleton.txt: Text file containing the logical array of cleaned skeleton.

Tortuosity_summarized.txt: Text file containing the calculated tortuosity parameters.

Skel_parameters.txt: Text file containing the calculated skeleton parameters, such as node density and link/node ratio.

Linklength_distribution.txt: Text file containing the calculated link length probability table.

Linklist.txt: Text file containing the graph information for the links. Equivalent to `Linklist` in *Skeleton_variables.mat*.

Nodelist.txt: Text file containing the graph information for the nodes. Equivalent to `Nodelist` in *Skeleton_variables.mat*.

Tortuosity_xyz.txt: Text file containing all shortest possible paths between two points on opposite box edges, and their Euclidean counterparts, for the x, y and z-directions.

Files generated once for all input files

Analysis_output_combined.mat: Contains all important parameters that are calculated.

Linklength_averaged.eps: A plot of the $\tilde{N}(\Lambda)$, averaged over all data files.

Linklength_averaged.fig: A plot of the $\tilde{N}(\Lambda)$, averaged over all data files, as Matlab .fig file.

Linklength_perimage.txt: Text file containing all calculated link length probabilities, combined in a single table.

Summary_combined.txt: Valuable global parameters, exported as text file.

Summary_linklength.txt: Valuable link length parameters, exported as text file.

Tortuosity_combined.txt: Valuable tortuosity parameters, exported as text file.

4 Pipeline

On the next pages are the pipelines depicted for the Standalone and the GUI versions. Note that not all scripts are described here, and undescribed scripts are commented sufficiently. In-built Matlab scripts are not included either.

4.1   Standalone version

```
main
  parameters
  preloop_initialization
  for each image individually:
    coord2tif
    project_prep:
      binarize
      smallout

    skeleton3D
    wrapper_firstclean:
      skel2graph_firstclean
      iterate until no more change:
        Graph2skel3D
        skel2graph_firstclean
      add_lengths

    project_tortuosity:
      tortuosity

    visualize_preclean:
      vis_raw_skel
      vis_prepost_skel
      vis_projections
      vtkwrite

    wrapper_thoroughclean:
      skel2graph_thoroughclean
      iterate until no more change:
        Graph2skel3D
        skel2graph_thoroughclean
      add_lengths

    wrapper_edgeclean:
      skel2graph_edgeclean
      iterate until no more change:
        Graph2skel3D
        skel2graph_edgeclean
      add_lengths

    project_linklength
    visualize_skeletons:
      vis_raw_skel
      vis_projections
      vtkwrite

    visualize_optimizations:
      vis_skel_cleaning
      vis_bin_raw
      vis_prepost_skel
      vis_bin_bin
    exporttxt

  wrapup
```

4.2   GUI version

Run full analysis
- **Run_preprocessing**, for each image individually:
  - coord2tif
  - project_prep:
    - binarize
    - smallout

- **Run_skeletonization**, for each image individually:
  - skeleton3D
  - wrapper_firstclean:
    - skel2graph_firstclean, iterate until no more change:
      - Graph2skel3D
      - skel2graph_firstclean
    - add_lengths

- visualize_preclean:
  - vis_raw_skel
  - vis_prepost_skel
  - vis_projections
  - vtkwrite

- **Run_analysis**, for each image individually:
  - project_tortuosity:
    - tortuosity
  - wrapper_thoroughclean:
    - skel2graph_thoroughclean iterate until no more change:
      - Graph2skel3D
      - skel2graph_thoroughclean
    - add_lengths
  - wrapper_edgeclean:
    - skel2graph_edgeclean iterate until no more change:
      - Graph2skel3D
      - skel2graph_edgeclean
    - add_lengths

- project_linklength
- exporttxt
- visualize_optimizations:
  - vis_skel_cleaning
  - vis_bin_raw
  - vis_prepost_skel
  - vis_bin_bin

- **Run_visualize**, for each image individually:
  - visualize_skeletons:
    - vis_raw_skel
    - vis_projections
    - vtkwrite

- wrapup

5   Individual Scripts

5.1   General

parameters.m

In the Standalone version, this file is the parameter file for ArGSLab This is the only file that should be changed in the Standalone version. After the parameters are filled in, save the file and run 'main.m'.

5.2   Graph manipulation

wrapper_firstclean.m

A wrapper script around skel2graph_firstclean. This creates graph files from a skeleton, and applies a first cleaning. This is meant to serve for a cleaning procedure before tortuosity determination.

skel2graph_firstclean.m

Creates graph files from a skeleton, and applies a first cleaning routine that cleans nodes only canal nodes, non-connected nodes, and unnecessarily kept node voxels. Note that single-connected nodes and their links are removed in the later used skel2graph_thoroughclean.

add_lengths.m

Adds the real lengths (in units of µm, or the units of `realpx` for each link to the `Linklist` variable.

tortuosity.m

Calculates the direct length divided by the pathlength for all nodes. It calculates this by obtaining all nodes in an edge region defines as being `realpx` $\times$ `particlediameter` $\times$ `edgenode_dist` from the box edge, and calculating the distance to the nodes found in the opposite side. Is performed seperately along the three directions.

wrapper_thoroughclean.m

A wrapper script around skel2graph_thoroughclean.

skel2graph_thoroughclean.m

Creates graph files from a skeleton, and applies the full cleaning routine. It creates graph files from a skeleton, and applies the following cleaning steps. Unphysically small side chains, arising from irregular structure surfaces, are excluded by removing all side chains smaller than a certain threshold value, by default set to 1.5σ with σ the particle diameter. A more detailed depiction of
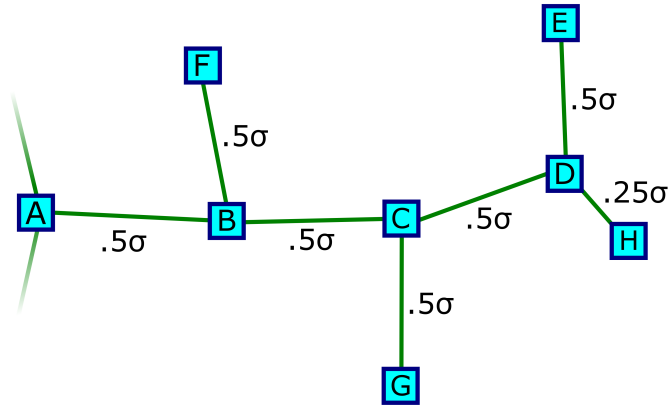

Figure 9: Depiction of a possible branched terminal node/link system. Nodes are in blue, links in green. In this case, each link is 0.5σ in length.

the inner workings of the terminal connection removal subroutine is depicted with Figure 9. In this Figure, let A be a node that is connected to two other nodes, neither of them considered for removal. Also, all depicted links have a length of 0.5σ or 0.25σ, and the `term_skel_thr` is defined as 1.5σ. The subroutine first finds the longest possible path length from A in this side branch, which will be $A - B - C - D - E$, which has a total path length of 2.0σ. This full path will therefore be preserved in this subroutine. However, side branches from the preserved branch are still under consideration for removal. This means that the $B - F$ branch, the $C - G$ branch and the $D - H$ branch are considered a side branch shorter than `term_skel_thr`, and will be removed. Furthermore *canal nodes* are removed, *i.e.* nodes that are connected to exactly two links. Such nodes and their connecting links are effectively one single link and therefore merged. Then, nodes that are in close proximity are collected into a single node, as dense sections of gels with irregular edges often lead to complex and unphysical skeleton sections with many interconnected nodes and links. A set of nodes to be collected contains all nodes that are closer than a threshold value to another node in the same set. The threshold value is by default set to 0.9σ.

wrapper_edgeclean.m

A wrapper script around skel2graph_edgeclean.

skel2graph_edgeclean.m

Creates graph files from a skeleton, applies the full cleaning routine, and removes any nodes or links leading to the box edge. The cleaning routines is necessary to remove any artefacts arising from this removal. The full cleaning routine is identical to the one described in skel2graph_thoroughclean.

5.3   Image manipulation

binarize.m

Binarizes a stack of images automatically, or with a set threshold. The automatic threshold works via Otsu's method is taken if `binThres` is 'auto'. The set threshold method determines, for each slice, the intensities I_{10} and I_{90}, which are the pixel intensity values below which 10% and 90% of individual pixel intensities fall, respectively. With `binThres` defined as V_u, set between 0 and 1, then defines the threshold intensity I_T, with

$$I_T = I_{10} + V_u(I_{90} - I_{10}). \tag{1}$$

The threshold determination is performed on each slice individually in order to compensate for differences in signal intensities between image slices: This is helpful when investigating microscopy data where a scattering length density mismatch leads to the average fluorescence intensity being dependent on the focal depth in the sample, or where fluorescence bleaching effects play a role.

smallout.m

Removes objects with a contiguous volume smaller than `minfrac×maxobject`. `maxobject` is defined as the volume of the largest contiguous object found in the data file. Throughout ArGSLab, all contiguity is defined as 26-connected connectivity.

5.4 Macroloops

preloop_initialization.m

Prepares several tables for output parameters to be stored in, reads the folder for file path strings, and other preparatory tasks.

project_prep.m

Wrapper for importing and preprocessing the raw image data. Preprocessing consists of a Gaussian blur step (matlab routine), binarize, morphological closing (matlab routine), smallout, and hole filling (matlab routine).

project_prep.m

Wrapper for performing the link length analysis.

exporttxt.m

Exports the requested analysis data as text files.

wrapup.m

Several scripts to wrap up the variables for exporting as .mat or .txt files.

5.5 Simulationtomatrix

coord2tif.m

Projects particle coordinates with a particle size onto a stack of slices in the form of a .tiff stack. Prepares this images stack to be imported by the following step (project_prep). Tiff stack size parameters are set by parameters such as `BoxSize_input`, `BoxSize_output`, and `particlediameter`.

project_tortuosity.m

A wrapper script around tortuosity.m.

5.6 Skeletonization

skeleton3D.m

Calculate the 3D skeleton of the precleaned gel structure using parallel medial axis thinning. This process consecutively removes material voxels bordering non-material voxels, with the condition that they are not end points, non-Euler invariant points, and are Euler simple points. Removal of a voxel considers the $3{\times}3$ voxel cube $D_3(i)$ surrounding voxel i, $i.e.$ a cube of 27 voxels with voxel i at the center. An end point is defined as the case where $D_3(i)$ contains in total two material voxels, of which one is i. Euler invariant points are defined as voxels that, upon removal, do not lead to a change of the Euler characteristic of the total structure. We define an Euler simple point as a voxel i whose removal neither changes the connectivity nor the Euler characteristic of $D_3(i)$. The transformation of a skeleton into a graph of nodes and links is then performed by assuming that (i) each voxel i whose surrounding $D_3(i)$ contains exactly 2 material voxels is an end node, (ii) each voxel i whose $D_3(i)$ contains exactly 3 material voxels belongs to a link and (iii) each voxel i whose $D_3(i)$ contains more than 3 material voxels is a branching node.
This routine is based on MATLAB code from Kerschnitzki, Kollmannsberger *et al.*, J. Bone. Miner. Res., 28(8):1837-1845, 2013. This code is based on algorithms published in Lee et al., Comput. Gr. Image Process., 56(6):462-478, 1994.

Graph2skel3D.m

Creates a skeleton array from a set of nodes (`Nodelist`) and links (`Linklist`).

5.7 Visualization

visualize_preclean.m

A wrapper script around the visualizations before the thorough cleaning routines. Tool for optimization.

vis_raw_skel.m

Creates a tiff image of the raw data, with a skeleton overlaid upon it.

vis_prepost_skel.m

Creates a tiff image where a skeletonization is overlaid with the binarized image.

vis_projections.m

Creates a matlab .fig file with a fully rotatable version of the current skeleton with nodes, along with four .png files of projections along a certain axis.

vtkwrite.m

Vtkwrite translates a 3D Matlab array as a VTK file format. Written by J. Yeh, used as imported from here.

visualize_skeletons.m

A wrapper script around the visualizations after the thorough cleaning routines.

visualize_optimizations.m

A wrapper script around the optimization visualizations after the thorough cleaning routines. Tool for optimization.

vis_skel_cleaning.m

Creates a tiff image where the skeletonization is overlaid with the binarized image.

vis_bin_raw.m

Creates a tiff image of the raw data, overlaid with green if it was determined above the threshold, and red if below.

vis_bin_bin.m

Creates a tiff image of the unpreprocessed binarized data, overlaid with the preprocessed binarized data. The sections are white if both images were white, black if both images were black, green if the preprocessing added material, and red if it removed it.