pycellfit

Release 0.1.0

Jun 09, 2020

Contents

CHAPTER 1

pycellfit

1.1 pycellfit package

1.1.1 Submodules

1.1.2 pycellfit.utils module

Top-level utility functions for pycellfit module.

pycellfit.utils.**contains_triple_junction**(*linestring*, *list_of_points*)
helper function that tells you if a shapely LineString contains a triple junction

Parameters

- **linestring** (*shapely.geometry.LineString*) – the LineString of interest
- **list_of_points** (*list*) – list containing a bunch of points of type shapely.geometry.Point

Returns if the linestring contains a point in the list_of_points, the first point is returned; else, None.

Return type shapely.geometry.Point

pycellfit.utils.**extract_all_points**(*dataframe_of_cells*, *visualize=False*)
generate list of all points in mesh

Parameters

- **visualize** (*bool*) – boolean to indicate if plot of all points should be made or not
- **dataframe_of_cells** (*geopandas dataframe*) – a dataframe where each row contains a unique cell

Returns list_of_all_points: list of all points in the mesh

Return type list

pycellfit.utils.**locate_triple_junctions**(*dataframe_of_cells*, *visualize=False*)
generate list of triple junctions in mesh

Parameters

- **visualize** (*bool*) – boolean to indicate if plot of all points should be made or not
- **dataframe_of_cells** (*geopandas dataframe*) – a dataframe where each row contains a unique cell

Returns list_of_tjs: list of all triple junctions in the mesh

Return type list

pycellfit.utils.**make_segments**(*cell_boundary*, *list_of_triple_junctions*)
recursive function that splits up a cell boundary based on triple junctions

Parameters

- **cell_boundary** (*shapely.geometry.LineString*) – boundary of a cell that needs to be broken up into segments
- **list_of_triple_junctions** (*list*) – list of all triple junctions (of type shapely.geometry.Point) in the mesh

Return results list of segments (of type shapely.geometry.LineString) that make up the cell boundary

Return type list

pycellfit.utils.**read_segmented_image**(*file_name*, *visualize=False*)
Displays the segmented image using matplotlib

Parameters

- **file_name** (*str*) – file name of a segmented image in .tif format
- **visualize** (*bool*) – if true, then image will be plotted using matplotlib.pyplot

Raises **TypeError** – only accepts tif/tiff files as input

Returns array of pixel values

Return type numpy.ndarray

1.1.3 pycellfit.junction module

class pycellfit.junction.**Junction**(*coordinates*)
Bases: object

add_edge(*edge_label*)
Adds edge label to set of edge labels

coordinates

degree

edges
set of labels of edges connected to this node

id_iter = count(0)

remove_edge(*edge_label*)
Remove an edge and tension vector connected to this node

Parameters

- **self** –
- **edge_label** –

tension_vectors
returns list of Tension vectors connected to this node

x

y

1.1.4 pycellfit.edge module

class pycellfit.edge.**Edge**(*start_node*, *end_node*, *radius*, *center*)
Bases: object

center

corresponding_tension_vector

end_node

id_iter = count(0)

radius

start_node

xc

yc

1.1.5 pycellfit.tension_vector module

A class to define tension vectors in CellFIT

class pycellfit.tension_vector.**TensionVector**(*edge*)
Bases: object

corresponding_edge
The corresponding Edge for this tension vector

Returns corresponding edge

Return type *edge.Edge*

direction
returns the direction of the tension vector in radians (or degrees) from the horizontal

Parameters **units** (*str*) – units for the direction, either 'rad' or 'deg'

Raises **ValueError** – is units is not 'rad' or 'deg

Returns direction

Return type float

id_iter = count(0)

label
returns the label (id number) for this tension vector

Returns label

Return type int

magnitude
Magnitude of the tension vector

Returns magnitude

x_component
returns the x-component of the tension vector

Returns x-component of the vector

Return type float

y_component
returns the y-component of the tension vector

Returns y-component of the vector

Return type float

1.1.6 pycellfit.cell module

class pycellfit.cell.**Cell**(*pixel_value*)
Bases: object

add_edge_point(*edge_point*)

approximate_cell_center()
approximates the coordinates of the center of the cell by averaging the coordinates of points on the perimeter (edge) of the cell

:return approximate center of the cell :rtype: tuple

clockwiseangle_and_distance(*point*)
helper function used in sorting edge points. Calculates the clockwise angle and the distance of a point from the approximate center of the cell. Source: https://stackoverflow.com/questions/41855695/sorting-list-of-two-dimensional-coordinates-by-clockwise -angle-using-python

Returns direction (clockwise angle), length vector (distance from center)

Return type tuple

edge_points_cw
sort all edge points in clockwise order and return the sorted list

Returns sorted list of all edge points (tuples)

Return type list

label
the label of a Cell is it's unique pixel value. It is assigned when the Cell object is created.

Returns

number_of_edge_points
returns the number of edge points in edge_point_list

Returns number of edge points

1.1.7 pycellfit.mesh module

class pycellfit.mesh.**Mesh**
Bases: object

add_cell(*cell_pixel_value*)

number_of_cells
returns the number of cells in the mesh

Returns number of cells in mesh

Return type int

number_of_edges
returns the number of edges in the mesh

Returns number of edges in the mesh

Return type int

number_of_junctions
returns the number of junctions in the mesh

Returns number of junctions in the mesh

Return type int

number_of_triple_junctions
counts and outputs the number of triple junctions in the mesh

:return number of triple junctions in mesh :rtype: int

remove_cell(*cell_pixel_value*)

1.1.8 pycellfit.constrained_circle_fit module

Constrained Circle Fit. Fit points to a circular arc when the two end points are fixed.

pycellfit.constrained_circle_fit.**algebraic_circle_fit**(*point_1*, *point_2*, *point_3*)
finds center and radius of a circle that contains three points on its edge

Parameters

- **point_1** –
- **point_2** –
- **point_3** –

Returns

pycellfit.constrained_circle_fit.**center_point_to_t_alpha_beta**(*center*, *point_p*)
converts from standard form (center, point on circle) to parametric form (t, alpha, beta)

Parameters

- **center** –
- **point_p** –

Returns

pycellfit.constrained_circle_fit.**constraint**(*ans*)
first constraint for the solver: alpha^2 + beta^2 = 1

Parameters **ans** –

Returns

pycellfit.constrained_circle_fit.**constraint2**(*ans*, *point_a*, *point_p*)
second constraint for the solver: distance from center to point_a = distance from center to point_p

Parameters

- **ans** –
- **point_a** –
- **point_p** –

Returns

pycellfit.constrained_circle_fit.**f**(*ans*, *x*, *y*, *point_a*, *point_p*)
cost function that needs to be minimized See https://arxiv.org/pdf/1504.06582.pdf (page 9)

Parameters

- **ans** –
- **x** –
- **y** –
- **point_a** –
- **point_p** –

Returns

pycellfit.constrained_circle_fit.**fit**(*x*, *y*, *start_point*, *end_point*)
performs a constrained circle fit based on points around an arc and the start and end points of the arc

Parameters

- **x** –
- **y** –
- **start_point** –
- **end_point** –

Returns

pycellfit.constrained_circle_fit.**plot_data_and_circle_fit**(*x*, *y*, *xc*, *yc*, *radius*, *start_point*, *end_point*)
plots the results of the circular fit

Parameters

- **x** – nparray of all x coordinates of data
- **y** – nparray of all y coordinates of data
- **xc** – float of x coordinate of center of fit circle
- **yc** – float of y coordinate of center of fit circle
- **radius** – radius of fit circle
- **start_point** –
- **end_point** –

Returns None

pycellfit.constrained_circle_fit.**r**(*point_a*, *point_p*, *alpha*, *beta*, *t*)
calculate radius of circle

Parameters

- **point_a** –

- **point_p** –
- **alpha** –
- **beta** –
- **t** –

Returns

pycellfit.constrained_circle_fit.**t_alpha_beta_to_center_radius**(*ans*, *point_a*, *point_p*)

converts from parametric form (t, alpha, beta) to standard form (radius, center)

Parameters

- **ans** –
- **point_a** –
- **point_p** –

Returns

pycellfit.constrained_circle_fit.**test1**()

pycellfit.constrained_circle_fit.**test2**()

1.1.9 pycellfit.circle_fit_helpers module

pycellfit.circle_fit_helpers.**M**(*g*, *h*, *x*, *y*)

pycellfit.circle_fit_helpers.**a0**(*point_a*, *point_p*, *x*, *y*)

pycellfit.circle_fit_helpers.**a1**(*point_a*, *point_p*, *alpha*, *beta*, *x*, *y*)

pycellfit.circle_fit_helpers.**a2**(*point_a*, *alpha*, *beta*, *x*, *y*)

pycellfit.circle_fit_helpers.**b0**(*point_a*, *point_p*)

pycellfit.circle_fit_helpers.**b1**(*point_a*, *point_p*, *alpha*, *beta*)

pycellfit.circle_fit_helpers.**b2**()

pycellfit.circle_fit_helpers.**distance**(*point_1*, *point_2*)

calculates the euclidian distance between two points

Parameters

- **point_1** –
- **point_2** –

Returns

pycellfit.circle_fit_helpers.**q**(*point_a*, *x*, *y*)

pycellfit.circle_fit_helpers.**qx**(*point_a*, *x*, *y*)

pycellfit.circle_fit_helpers.**qxx**(*point_a*, *x*, *y*)

pycellfit.circle_fit_helpers.**qxy**(*point_a*, *x*, *y*)

pycellfit.circle_fit_helpers.**qy**(*point_a*, *x*, *y*)

pycellfit.circle_fit_helpers.**qyy**(*point_a*, *x*, *y*)

1.1.10 pycellfit.segmentation_transform module

functions to convert between watershed and skeleton segmented images

pycellfit.segmentation_transform.**skeleton_to_watershed**(*skeleton_image_array*, *region_value=0*, *boundary_value=255*, *keep_boundaries=False*)

converts a segmented skeleton image (all regions are same value with region boundaries being a second value) to a watershed segmented image (each region has a unique value and there are no boundary pixels, background region has value of zero)

Parameters

- **skeleton_image_array** (*np.ndarray*) – 2D numpy array with pixel values of a skeleton segmented image
- **region_value** (*float*) – value of pixels in regions in skeleton_segmented images (default is 0)
- **boundary_value** (*float*) – value of boundary pixels of regions in skeleton_segmented images (default is 255)
- **keep_boundaries** (*bool*) – if True, watershed image will keep boundaries in returned result

Returns watershed_image_array

Rtype watershed_image_array np.ndarray

pycellfit.segmentation_transform.**watershed_to_skeleton**(*watershed_image_array*, *region_value=0*, *boundary_value=255*)

converts a watershed segmented image (no boundaries between regions and each region has a different pixel value, background region has value of zero) to a skeleton segmented image (each region has the same pixel value and are separated by boundaries of a second value)

Parameters

- **watershed_image_array** (*numpy.ndarray*) – 2D numpy array with pixel values of a watershed segmented image
- **region_value** (*float*) – desired value of all regions in the output (skeleton segmented) array
- **boundary_value** (*float*) – desired value of boundary pixels in the output (skeleton segmented) array

Returns skeleton_image_array

Rtype skeleton_image_array np.ndarray

1.1.11 pycellfit.segmentation_transform_utils module

pycellfit.segmentation_transform_utils.**fill_region**(*array_of_pixels*, *position*, *new_value*)

fills a region of a 2D numpy array with the same value

Parameters

- **array_of_pixels** (*np.ndarray*) – 2D numpy array of all pixel values
- **position** (*tuple*) – tuple with (row, col) location of pixel in the region to modify

- **new_value** (*float*) – new value for pixel at *position* and all pixels in same region

Returns None

pycellfit.segmentation_transform_utils.**pad_with**(*vector*, *pad_width*, *iaxis*, *kwargs*)
helper function that is called by np.pad to surround a nparray with a constant value Example: [[0,0],[0,0]] becomes [[-1,-1,-1, -1],[-1, 0, 0, -1],[-1, 0, 0, -1],[-1,-1,-1, -1]]

1.1.12 Module contents

CHAPTER 2

README

2.1 pycellfit

2.1.1 Project Description

pycellfit: an open-source Python implementation of the CellFIT method of inferring cellular forces developed by Brodland et al.

Author: Nilai Vemula, Vanderbilt University (working under Dr. Shane Hutson, Vanderbilt University)

Project Goal: To develop an open-source version of CellFIT, a toolkit for inferring tensions along cell membranes and pressures inside cells based on cell geometries and their curvilinear boundaries. (See[1].)

Project Timeline: Initial project started in August 2019 with work based off of XJ Xu. This repository was re-made in May 2020 in order to restart repository structure.

Project Status: **Early development**

2.1.2 Getting Started

This project is available on PyPI and can be installed using pip.

It recommended that users make a virtual environment and install the package as such:

[1] Brodland GW, Veldhuis JH, Kim S, Perrone M, Mashburn D, et al. (2014) CellFIT: A Cellular Force-Inference Toolkit Using Curvilinear Cell Boundaries. PLOS ONE 9(6): e99116. https://doi.org/10.1371/journal.pone.0099116

```
pip install pycellfit
```

Full documentation for this package can be found on readthedocs.

Dependencies

One of the goals of this project is to avoid dependencies that are difficult to install such as GDAL. This project primarily depends on numpy, scipy, matplotlib, and other common python packages common in scientific computing. A full list of dependencies is available in the requirements.txt file. All dependencies should be automatically installed when running pip install.

2.1.3 Development

This project is under active development and not ready for public use. The project is built using Travis CI, and all tests are run with every commit or merge.

2.1.4 Features

This section will include a list of features available in the package and maybe a check-list of things to add...

2.1.5 Examples

A example walk-through of how to use this module is found in quickstart.

2.1.6 Future Goals

The final implementation of pycellfit will be as a web-app based on the Django framework. See (add link to django-pycellfit repo).

2.1.7 References

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index