pycellfit

Release 0.1.0

Aug 03, 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.**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*, *cells_set*)

Bases: object

add_edge(*edge_label*)

Adds edge label to set of edge labels

cell_labels

coordinates

degree

edges
set of labels of edges connected to this node

id_iter = count(0)

plot(*label=False*)

plot_unit_vectors()

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*, *intermediate_points*, *cells*)
Bases: object

angular_position(*coordinates*)
given a (x,y) coordinate, the angular position in radians from 0 to 2*pi around the fit circle is returned

Parameters **coordinates** –

Returns

calculate_edge_points()

center

circle_fit()

corresponding_tension_vector

cw_around_circle
true if the edge (start_node to end_node) goes clockwise around the fit circle, false if ccw

Returns

end_node

end_tangent_angle(*method='chord'*)

id_iter = count(0)

length

linear

location

map_unit_vectors_to_junctions()

outside(*background*)

plot(*label=False*)

plot_circle()

plot_tangent(*c='y'*)

radius

split_line_multiple(*length=None*, *n_pieces=None*)
Splits a ogr wkbLineString into multiple sub-strings, either of a specified <<length>> or a specified <<n_pieces>>.

line should be an ogr LineString Geometry Length should be a float or int. n_pieces should be an int. Either length or n_pieces should be specified.

Returns a list of ogr wkbLineString Geometries.

split_line_single(*line*, *length*)
Returns two ogr line geometries, one which is the first length <<length>> of <<line>>, and one one which is the remainder.

line should be a ogr LineString Geometry. length should be an integer or float.

start_node

start_tangent_angle(*method='chord'*)

xc

yc

pycellfit.edge.**collinear**(*points*, *epsilon=0.01*)
determine if three points are collinear

Parameters

- **epsilon** –
- **points** (*list of 3 2-member tuples*) – list of 3 point tuples that might be collinear

Returns boolean to tell if the points are collinear or not

Return type bool

pycellfit.edge.**distance**(*p1*, *p2*)

1.1.5 pycellfit.tension_vector module

1.1.6 pycellfit.cell module

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

add_edge_point(*edge_point*, *neighboring_cell_label*)

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

generate_maze(*neighboring_cell_label*)

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

Returns

make_edges(*master_set*)

neighboring_cell_labels

number_of_edge_points
returns the number of edge points in edge_point_list

Returns number of edge points

plot()

1.1.7 pycellfit.mesh module

class pycellfit.mesh.**Mesh**(*array_of_pixels*)
Bases: object

add_cell(*cell_pixel_value*)

add_edge_points_and_junctions(*array_of_pixels*)

circle_fit_all_edges()

find_cells_from_array()

generate_mesh(*average_nodes_per_edge=4*)

make_edges_for_all_cells()

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_quad_junctions
returns the number of quad junctions in the mesh

Returns number of edges 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

plot()

plot_tensions()

remove_cell(*cell_pixel_value*)

solve_tensions()

solve_tensions_new()

solve_tensions_new2()

1.1.8 pycellfit.constrained_circle_fit module

1.1.9 pycellfit.circle_fit_helpers module

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: **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 then 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

Install from PyPI:

```
pip install pycellfit
```

Or compile from source:

```
git clone https://github.com/NilaiVemula/pycellfit.git
cd pycellfit
python setup.py install
```

Full documentation for this package can be found on readthedocs.

Dependencies

This project is written in Python and has been tested on Python 3.7 and 3.8 on Linux and Windows. This project primarily depends on numpy, scipy, matplotlib, and other common python packages common in scientific computing. Additionally, Pillow is required for reading in input image files. 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

Currently, pycellfit supports the following features in the cellular force inference pipeline:

- [ ] converting raw images into segmented images
  - see SeedWaterSegmenter or neural_net_cell_segmenter (work in progress).
- [x] read in segmented images
- [x] convert between watershed and skeleton segmented images
- [x] identify triple junctions
- [ ] identify quad junctions
- [x] generate a mesh
- [x] fit cell edges to circular arcs
- [ ] calculate tangent vectors using circle fits, nearest segment, and chord methods
  - circle fit is incorrect, others have not been added
- [x] calculate tensions
- [ ] calculate pressures
- [x] visualize all of the above steps

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 pycellfit-web)

2.1.7 References

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index