FIB Tortuosity Documentation

Release 0.2.5.1

Joshua Taillon

Aug 01, 2018

Table of Contents

PyPI:		
Help!		

PyPI:		
Help!		

CHAPTER 1

Introduction

This module is used to calculate the geometric tortuosity of a three-dimensional structure, based on J. Taillon's 2018 *Ultramicroscopy* paper. Useful for FIB/SEM reconstructions of SOFCs, but probably useful in other situations as well.

These methods have been written for the specific software and methods used in our lab, and as such, might need some tweaking to work with your data. In general, .tiff files used as inputs are expected in the format that is output by Avizo. There is part of the code that depends on the `ImageDescription` tiff tag that Avizo saves. This tag saves information about the physical dimensions of the bounding box, and is of the format: `'BoundingBox <minX> <maxX> <minY> <maxY> <minZ> <maxZ>\n'` For the sample files in this documentation, the value is: `'BoundingBox 3231.15 6462.29 0 4100.38 0 4000\n'` If you cannot write this information into the tiff file that you are trying to provide, the code will have to be modified.

The expected inputs are binary (0 and 1) or ternary (0, 1, or 2) labeled 3D tif files representing the phases to calculate. Because the module is focused on SOFC calculations primarily, it is expected that 1 index represents the bulk electrolyte phase. If present, the 2 index is the phase for which to calculate the tortuosity. The 0 index are the areas that cannot be traversed (i.e. the hard boundaries that create the tortuosity).

For convenience, two small test 3D tiff files are provided for the example calculated in the *General Instructions (for SOFCs)* section. Those can be downloaded here: `Electrolyte` and `Cathode`.

PyPI:		
Help!		

CHAPTER 2

Installation

2.1 Requirements

The following dependencies are needed to run the code (as-written). Some of these could be pretty easily removed, but the code was written with my research results in mind, and as such, has some dependencies that make things match my personal preferences (such as `seaborn`). Some details on installing them are given below:

Package	Source	PyPI
`numpy`	Source	PyPI
`matplotlib`	Source	PyPI
`libtiff`	Source	N/A
`pylibtiff`	Source, Python 3 port	PyPI
`scikit-fmm`	Source	PyPI
`seaborn`	Source	PyPI
`hyperspy` *(optional)*	Source	PyPI

2.2 Python 2.7

To get a full set-up on Python 2.7 (hopefully) pretty easily, run the following:

```
$ pip install numpy matplotlib libtiff scikit-fmm seaborn fibtortuosity hyperspy
```

2.3 Python 3.5

On Python 3.5, things are a little bit trickier because one of the dependencies (`libtiff`) has not yet been ported to Python 3. I have done my best to port this package myself (currently awaiting inclusion into the main `libtiff` package). I cannot guarantee it will work, but it seems alright on my machine. To install my fork, open up a command line and run the following:

```
$ git clone https://github.com/jat255/pylibtiff.git
$ cd pylibtiff
$ git checkout ENH_py3_upgrade
$ pip install .
```

This library depends on a compiled version of the `tiff` library as well. On a linux system (like Ubuntu) this library is probably available in the package repository, and can be installed with the following:

```
$ sudo apt-get install libtiff5 libtiff5-dev
```

On Windows, things are just a little bit harder. I was able to get it built on my Windows box by follow the instructions here.

Some tips:

- Make sure to compile the version (32 or 64 bit) that matches your Python installation
- This page will explain how to get the Visual C++ compiler to work on the command line
- Once you have compiled the `libtiff.dll` file, add the directory containing it to your system path, and then restart any Python environments you have open. The library should now be available…

Once all that is done, run the same code as for 2.7, but take out *libtiff* from the `pip` list.

2.4 "Stable" Version

Although it shouldn't really be considered stable, there are released versions available for installation through `pip` on PyPI.

Note: Note: due to some recent bugs with PyPI, the package will likely not show up if you use `pip search`, but should install as normal with `pip install`.

To install the latest release:

```
$ pip install --user fibtortuosity
```

2.5 Development Version

The latest version of the code should be available in the Bitbucket repository. To get this version installed on your system, clone the repository, and then install with `pip`:

```
$ git clone https://bitbucket.org/jat255/fibtortuosity.git
$ cd fibtortuosity
$ pip install -e ./
```

PyPI:		
Help!		

CHAPTER 3

General Instructions (for SOFCs)

3.1 Generic analysis

These instructions will walk through calculating the tortuosity for the two example files `Electrolyte` and `Cathode`. In general, the low-level methods `_geo_dist()`, `_calc_interface()`, `_calc_tort()`, and `_calc_euc_x()` do all the "heavy" lifting, while modules like `tortuosity_from_labels_x()` and `run_full_analysis_lsm_ysz()` wrap these methods into convenient higher-level interfaces.

1. Put the downloaded files into a directory accessible to Python, and fire up a Python session (Jupyter, notebook, etc.)

2. Import the module:

```
>>> import fibtortuosity as ft
```

3. The following code will use the higher order function `tortuosity_from_labels_x()` to calculate the tortuosity in the *x* direction (perpendicular to the electrolyte boundary) The *x* direction function includes some additional code (compared to the *y* and *z* versions) that calculates the euclidean distance from the boundary of the bulk electrolyte, rather than simply from the edge of the volume.

   `geo`, `euc`, `tort`, and `desc` will contain the results of this calculation afterwards:

```
>>> geo, euc, tort, desc = ft.tortuosity_from_labels_x('example_
↪electrolyte.tif',
...                                                      'example_
↪electrolyte_and_cathode.tif',
...                                                      'LSM',
...                                                       units='nm',
...                                                       print_output=True,
...                                                       save_output=False)
    Starting calculation on LSM in x direction.
    Loaded electrolyte data in 0:00:00.257833 seconds.
    Loaded cathode data in 0:00:00.252949 seconds.
    ImageDescription is: b'BoundingBox 3231.15 6462.29 0 4100.38 0 4000\n'
```

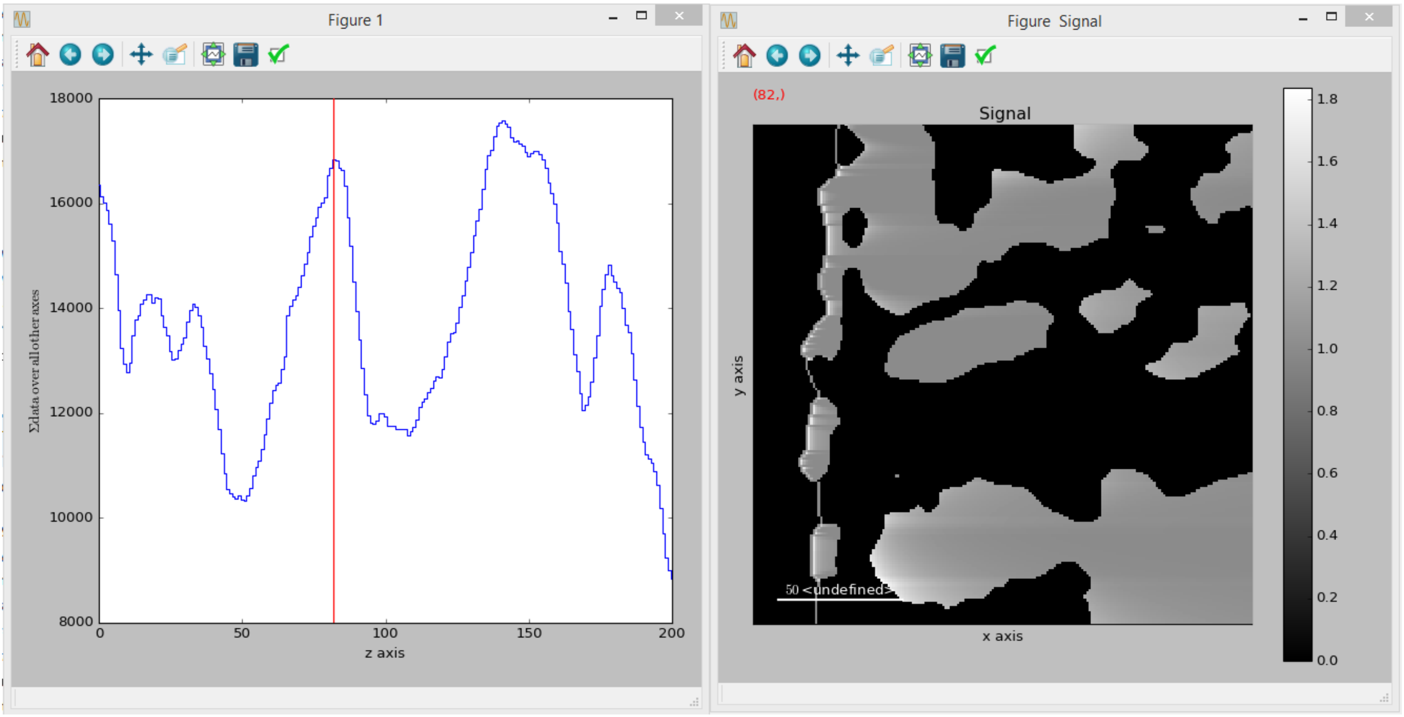
(continues on next page)

(continued from previous page)

```
    Bounding box dimensions are: (3231.14, 4100.38, 4000.00) nm
    Voxel dimensions are: (16.16, 20.50, 20.00) nm
    Starting geodesic calculation at: 2016-04-05 15:52:57.412840
    Geodesic calculation took: 0:00:07.159744
    Calculating zero distance interface took: 0:00:00.207048
    Calculating euclidean distance took: 0:00:00.378937
    Calculating tortuosity took: 0:00:00.295921
    Total execution time was: 0:00:08.597359
```

4. If HyperSpy is installed, it can be used to easily visualize the three-dimensional data that is produced as a result (example below is in Jupyter):

```
>>> %matplotlib qt4
>>> import hyperspy.api as hs
>>> t_s = hs.signals.Image(tort)
>>> for i, n in enumerate(['z', 'x', 'y']):
...     t_s.axes_manager[i].name = n
>>> t_s.plot()
```

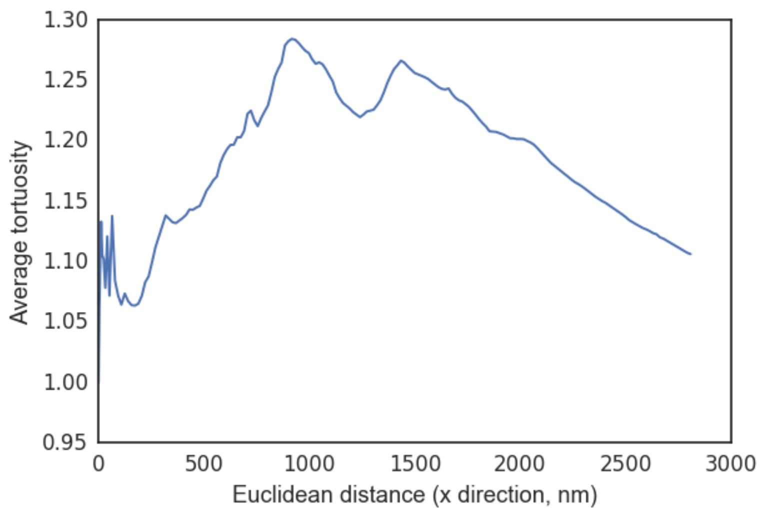


5. The `tortuosity_profile()` and `plot_tort_prof()` methods can be used to visualize the average tortuosity over a dimension:

```
>>> t_avg, e_avg = ft.tortuosity_profile(tort, euc, axis='x')
>>> ft.plot_tort_prof(t_avg, e_avg, 'x')
```

6. To save the results, a variety of export options are available. The average profiles can be easily saved using `save_profile_to_csv()`. The 3D tortuosity (or euclidean/geodesic distance) arrays can be saved as a 3D tiff using the `save_as_tiff()` method. Also, if HyperSpy is being used, it can save the data in the `.hdf5` format (see its documentation for details). Furthermore, any of the Numpy methods for saving (such as `save()` or `savez()`) can be used directly on the resulting arrays.

   (a) To save profile:



```
>>> ft.save_profile_to_csv('profile.csv', e_avg, t_avg, 'x', 'LSM')
```

(b) To save tiff file:

```
>>> # Using `desc` allows the file to be opened directly in Avizo
>>> print(desc)
    b'BoundingBox 3231.15 6462.29 0 4100.38 0 4000\n'
>>> ft.save_as_tiff('tortuosity.tif', tort, 'float32', desc)
    Writing TIFF records to tort_results.tif
    filling records:   100% done (12Mi+1003Ki+361 bytes/s)
    resized records: 31Mi+46Ki+716 bytes -> 8Mi+717Ki+565 bytes (compression:␣
↪3.59x)
```

This gives you the following output: `Tortuosity-profile` and `Tortuosity-data`

3.2 Full Analysis

In my personal research, I more fully combined these methods into the highest-level function `run_full_analysis_lsm_ysz()`, so I could just run one function, walk away, and let it run. It has some additional useful features, like texting a number when it's done or showing a browser notification. Also, it's written with specific materials in mind, so it may take some tweaking to get it running for your specific needs, but I include it here as an idea of what can be done with the underlying code.

To do a full analysis of a sample (all phases and all directions), I would run something like the following, making sure that each of the files specified are in the current directory:

```
>>> import itertools
>>> import fibtortuosity as ft
>>> phases = ['LSM','YSZ','Pore']
>>> directions = ['x', 'y', 'z']
>>> for p, d in itertools.product(phases, directions):
...     ft.run_full_analysis_lsm_ysz(electrolyte_file="bulkYSZ.tif",
...                                  electrolyte_and_pore_file="bulkYSZandPRE.tif",
...                                  electrolyte_and_lsm_file="bulkYSZandLSM.tif",
```

(continues on next page)

(continued from previous page)

```
...                                          electrolyte_and_ysz_file="bulkYSZandYSZ.tif",
...                                          date='2016-04-05',
...                                          phase=p,
...                                          direction=d,
...                                          npzfile=None,
...                                          units='nm',
...                                          delay=0,
...                                          calculate_all=True,
...                                          load_from_prev_run=False,
...                                          create_hspy_sigs=False,
...                                          save_avizo_tiff=True,
...                                          tort_profile=True,
...                                          save_tort_prof=True,
...                                          in_ipython=False)
```

After this has run, the directory will contain all the tortuosity data saved as `.tif` files, as well as all the average profiles for each phase and each direction, which can then be plotted/analyzed however is necessary.

PyPI:		
Help!		

CHAPTER 4

API Summary

`fibtortuosity.tortuosity_from_labels_x`
`fibtortuosity.tortuosity_from_labels_y`
`fibtortuosity.tortuosity_from_labels_z`
`fibtortuosity.calculate_geodesic_distance`
`fibtortuosity.save_results`
`fibtortuosity.load_results`
`fibtortuosity.run_full_analysis_lsm_ysz`
`fibtortuosity.save_as_tiff`
`fibtortuosity.tortuosity_profile`
`fibtortuosity.plot_tort_prof`
`fibtortuosity.save_profile_to_csv`
`fibtortuosity.load_profile_from_csv`
`fibtortuosity._geo_dist`
`fibtortuosity._calc_interface`
`fibtortuosity._calc_tort`
`fibtortuosity._calc_euc_x`

CHAPTER 5

Full API Documentation