tracr Documentation

Release 2018

H. Geerlings

Apr 05, 2018

CONTENTS

This is the documentation page for Tomographic Reconstruction Analysis and Characterization Routines (TRACR).

Contents:

CHAPTER

ONE

REQUIRED PYTHON PACKAGES

h5py==2.7.0
matplotlib==2.0.2
numpy==1.13.3
Pillow==3.1.1
pydicom==0.9.9
pytest==3.0.7
scikit-image==0.12.3
scikit-learn==0.17.1
scipy==0.17.1

CHAPTER

TWO

DOCUMENTATION

class `tracr.base.`**`Feature`**

Generation of our ‘Feature’ object. This class accepts a numpy array of the raw intensity data (read in from tracr.io.read). It extends the ndarray class by further attributing: pixel size, (more to come).

INPUT: array (np array): numpy array representation of intensty data. pixelsize (float): pixel dimension (um/px)

OUTPUT: obj (Feature): Feature object combining numpy array and pixelsize dataself.

USAGE: feat = Feature(intensity_array, pixelsize=4.5)

`tracr.io.read.`**`read`**(*ifile*, ***kwds*)

Read in TIF or DCM intensity data and send to appropriate reader depending on format. If keyword arg specified for ‘format’, this is just a wrapper for the readers. If not, we guess. So far, this assumes that a folder input contains data frames directly below (i.e. there are no nested folders.) and that contained files are listed sequentially. Unspecified pixel size is set to 1.

INPUT:

ifile (list/str): raw image file(s). Formats include:

- TIF: file (single or multilayer), or user specified list of frames
- DICOM: file (single), or list of dcm frames
- HDF5: file, assumes a ‘tomograph’ dataset, may use ‘pixel size’ also

OUTPUT:

feat (feature): Feature object of intensity data, with ‘pixelsize’ attribute. See specific readers for more.

USAGE: e.g. read(glob.glob(‘myDataFolder/*tif’)) e.g. read(‘path/to/sampleX.dcm’, format=’dcm’), (no guessing here) e.g. read(‘path/to/sampleX.tif’, pixelsize=4.5)

`tracr.io.tif.read.`**`read`**(*ifile*, ***kwds*)

Converts TIF format data to a Feature object. Root reading functions:

- Check if list is single file (multi/single layer), or list of frames
- **Call appropriate reader**
  - Tranpose data for upwards-z indexing
- Initiate Feature class using output numpy array and pixelsize kwd

INPUT: ifile (list): List of either single/multi layer tif file, or list of individual files **pixelsize (float): has been set as a **kwds argument as per tracr.io.read

OUTPUT: feat (Feature): numpy array data representation with ‘pixelsize’ attribute.

USAGE: e.g. feat = read(glob.glob()'path/to/dataFolder/*tif'), pixelsize=px_size) e.g. feat = read('path/to/tifFrame', pixelsize=px_size) * pixelsize has been set in general reading function.

`tracr.io.tif.read.`**`read_multilayer`**`(`*ifile*`)`
File reader for single file, multilayer TIF images (3D data) e.g. 'sampleX_multilayer.tif'

`tracr.io.tif.read.`**`read_single`**`(`*ifile*`)`
File reader for single file, single layer TIF images (2D data). We also ensure that RGB images are greyscale before converting (ITU-R 601-2). e.g. 'frameX.tif'

`tracr.io.dcm.read.`**`read`**`(`*ifile*, ***kwds*`)`
Converts .dcm format data to a Feature object.

INPUT: ifile (list): List of either single (2D) or multiple frame (3D) dcm files. **pixelsize (float): voxel dimension (um/pixel).

OUTPUT:

feat (Feature): array (either 2D or 3D) representation of .dcm intensity data and 'pixelsize' in um/pixel

USAGE: e.g. intensity_array = read('path/to/frame.dcm') e.g. intensity_array = read('path/to/DCM_folder/')

`tracr.io.hdf5.read.`**`read`**`(`*ifile*, ***kwds*`)`
Assume single file hdf5 file, iterate through list. Converts HDF5 format data to a Feature object.

INPUT: ifile (list): single multilayer hdf5 file of raw intensity data.

The assumed structure of the HDF5 file is: GROUP "/" {

DATASET "tomograph" { DATATYPE numeric ATTRIBUTE "pixel size" {

DATATYPE float

} ATTRIBUTE "pixel units" {

DATATYPE string

}

}

}

By default, the "tomograph" dataset is tranposed from *[layer, width, height]* order to *[x, y, z]*, where *x <- width*, *y <- height* and *z <- layer*.

OUTPUT: feat (Feature): numpy array data representation with 'pixelsize' attribute.

USAGE: e.g. feat = read('path/to/hdf5_file', pixelsize=px_size) * pixelsize has been set in general reading function.

`tracr.filters.threshold.otsu.`**`otsu`**`(`**args*, ***kwds*`)`
Otsu wrapper… See skimage.filters docs for other methods.

`tracr.filters.threshold.otsu.`**`tracr_otsu`**`(`**args*, ***kwds*`)`
Calculates the threshold levels for an image using the multilevel otsu implementation from Deng-Yuan Huang and Chia-Hung Wang, "Optimal multi-level thresholding using a two-stage Otsu optimization approach", Pattern Recognition Letters 30 (2009) 275-284, doi:10.1016/j.patrec.2008.10.003.

This is an implementation of the recursive algorithm (lookup table) from Liao (2001), and referenced by Huang and Wang.

INPUT: image (array-like): Image intensity data nclasses (int): (optional) Number of classes into which the data should be subdivided. Default: 2 (two classes –> one threshold)

nbins (int): (optional) A histogram of *nbins* will be made from the intensity values in *image*. This provides the number of bins in that histogram. Default: 256

*trim (optional): ignores nonzero entries of dataset

OUTPUT: thresholds (tuple): Threshold levels

EDGE FILTER:

Generates an array of feature edges from raw intensity data.

INPUT: array (np array): numpy array of raw intensity data. | *sigma (float): Gaussian width parameter (for boolean filters).

OUTPUT: edge_array (np array)

USAGE: edges = canny(array, sigma=3)

LABELING FILTER:

Converts array of thresholded data to labeled representation i.e. with regions of interest enumerated rather than simply 'True' valuedself.

INPUT:

binary_arr (np array): numpy array of already thresholded data, typically with logical entries.

OUTPUT: labeled_arr (np array): labeled representation of intensity data. num (int): number of distinct features.

`tracr.metrics.com.`**`com`**(*arr*, *larr*)

INPUT: arr (ndarray): Array of ra intensity data. Shape must match 'larr'. labeled_arr (ndarray): A labeled array with 'num' clusters of integer

values and '0' cluster for ignored material (background).

OUTPUT:

coms (list): An 'num+1' length list of labeled cluster centroids. This includes the '0' cluster (passed signal).

USAGE: centroids = com(array, labels)

`tracr.metrics.volume.`**`volume`**(*larr*)

INPUT:

label_arr (ndarray): A labeled array with 'num' clusters of integer values and '0' cluster for passed material (background).

OUTPUT: vols (list): An 'n+1' list of voxel volumes corresponding to labels from 'arr'.

`tracr.metrics.sing_vals.`**`sing_vals`**(*arr*, *labels*, *num*)

Generates semiaxial scalings for bounding ellipsoids pertaining to each feature of interest.

INPUT: arr (np array): raw intensity data. labels (np array): labeled intensity data. num (int): feature count

OUTPUT: S (list): list of singular values for each feature of interest.

`tracr.metrics.eccentricity.`**`eccentricity`**(*s*)

A module for generating a scalar description [0,1] of eccentricity generated from the semiaxial scalings of features obtained through the SVD.

INPUT: s (list): list of singular values for each feature of interest

OUTPUT:

ecc (np array): An num x 1 array of ratios of the lowest/highest sigmas for each feature in labeled array.

CHAPTER

THREE

USAGE DEMONSTRATION WITH VISUALIZATION

```
from __future__ import division
import numpy as np
from glob import glob
from matplotlib import pyplot as plt

from tracr.io import read
from tracr.filters.threshold import skimage_otsu as skotsu
from tracr.filters import edge, label
from tracr.metrics import com, volume

# Read TIF data (uXCT slice, in this case)
feat = read('path/to/your/image/data')

# Filter (ignoring zeros, optional) and perform a low-pass (pores in metal) analysis
finite = feat[np.nonzero(feat)]
thresh_val = skotsu(finite)
binary_arr = (feat < thresh_val)
labels, num = label(binary_arr)

# Obtain metrics
coms = com(feat, labels)
volumes = volume(labels)

# Visualize
f, axarr = plt.subplots(2, 2)
f.suptitle('Pore Count: {}, Average Pore Volume: {}'.format(num, np.mean(volumes[2:])))
axarr[0,0].imshow(feat, cmap='bone')
axarr[0,0].set_title('Raw Intensity Array')
axarr[1,0].hist(finite, bins=128)
axarr[1,0].axvline(thresh_val, color='r')
axarr[1,0].set_title('Intensity Histogram and Threshold Value')
axarr[0,1].imshow(binary_arr, cmap='bone')
axarr[0,1].set_title('Thresholded Data')
```

axarr[1,1].imshow(labels)
axarr[1,1].set_title('Labeled Array')
plt.show()

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX