Calculate Distance

Get files

In [149]:

```
import os
import nipype.interfaces.freesurfer as fs
import pymeshlab as ml
import collections
import csv
import numpy as np

# !pip install numpy --upgrade
# !pip install nipype
```

In [150]:

```
def get_files(folder_name = 'output_lh'):
    folders = []
    wfiles = []
    pfiles = []
    for folder in os.listdir(folder_name):
        if folder[0].isdigit():
            folders.append(folder)
            wfile = folder_name  + '/' + folder + '/'+folder+'_lh_white_Df2.white'
            pfile = folder_name + '/'+folder+'/'+folder+'_lh_pial_Df2.pial'
            wfiles.append(wfile)
            pfiles.append(pfile)

    print('There are ',len(folders), ' folders.','\n')
    print('First 2 folders:\n', folders[:2],'\n')

    print('First 2 white Df2 files:\n', wfiles[:2],'\n')

    print('First 2 pial Df2 files:\n', pfiles[:2])
    return folders,wfiles, pfiles
```

In [152]:

```
## get files
folders, wfiles, pfiles = get_files(folder_name = 'output_lh')
```

```
There are  107  folders.

First 2 folders:
 ['200008', '200109']

First 2 white Df2 files:
 ['output_lh/200008/200008_lh_white_Df2.white', 'output_lh/200109/2001
09_lh_white_Df2.white']

First 2 pial Df2 files:
 ['output_lh/200008/200008_lh_pial_Df2.pial', 'output_lh/200109/200109
_lh_pial_Df2.pial']
```

Convert Files

In [153]:

```
def convert_files(wfiles):
    for f in wfiles:
        mris = fs.MRIsConvert()
        mris.inputs.in_file = f
        mris.inputs.out_datatype = 'stl'
        mris.run()
```

In [155]:

```
## ---------------------------Convert Files----------------------------------
## uncomment following 2 lines to convert.
# convert_files(wfiles)
# convert_files(pfiles)
```

Calculate Distance

In [65]:

```
## ------notice: calculating distance TAKES TIME----------------------------
truth_folder_pre = '../../../../washbee/speedrun/deepcsr-preprocessed/'
w_distance = collections.defaultdict(dict)
p_distance = collections.defaultdict(dict)

cnt = 0
print("There are ", len(folders), " folders in total.\n")
print('\n Start calculating distance ...')
for f in folders:
    cnt += 1
    if cnt % 10 == 0:
        print('processing {}th folder ... '.format(cnt))
    wf_cvt = f + '_lh_white_Df2.white_converted.stl'
    pf_cvt = f + '_lh_pial_Df2.pial_converted.stl'
    lh_p = truth_folder_pre + f + '/lh_pial.stl'
    lh_w = truth_folder_pre + f + '/lh_white.stl'

    ms = ml.MeshSet()
    file_truth, file = lh_w, wf_cvt
    ms.load_new_mesh(file_truth)
    ms.load_new_mesh(file)
    w_distance[f] = ms.get_hausdorff_distance()

    ms = ml.MeshSet()
    file_truth, file = lh_p, pf_cvt
    ms.load_new_mesh(file_truth)
    ms.load_new_mesh(file)
    p_distance[f] = ms.get_hausdorff_distance()
```

```
There are  107  folders in total.


 Start calculating distance ...
processing 10th folder ...
processing 20th folder ...
processing 30th folder ...
processing 40th folder ...
processing 50th folder ...
processing 60th folder ...
processing 70th folder ...
processing 80th folder ...
processing 90th folder ...
processing 100th folder ...
```

Save Distance to CSV file

Save distance for white

helper functions

In [144]:

```
#-----------------------Save values to a Dictionary-----------------
def save_to_dic(w_distance, folders):
    dic = collections.defaultdict(list)
    measure_keys = list(w_distance['200008'].keys())
    idx = []
    for k in w_distance:
        idx.append(k)
        for key in measure_keys:
            dic[key].append(w_distance[k][key])
    ## add folder to w_distance
    for f in folders:
        w_distance[f]['folder'] = f
    return dic, measure_keys

#-----------------------Save Dictionary to CSV-----------------------------

def save_to_csv(measure_keys, w_distance, csv_file_name = 'w_distance.csv'):
    csv_columns = measure_keys
    dict_data = list(w_distance.values())
    csv_file = csv_file_name
    try:
        with open(csv_file, 'w') as csvfile:
            writer = csv.DictWriter(csvfile, fieldnames=csv_columns)
            writer.writeheader()
            for data in dict_data:
                writer.writerow(data)
    except IOError:
        print("I/O error")
```

Save distance for pial

In [147]:

```
dic_p, measure_keys_p = save_to_dic(p_distance, folders)
save_to_csv(measure_keys_p, p_distance, 'p_distance.csv')
```

Save distance for white

In [160]:

```
dic_w, measure_keys_w = save_to_dic(w_distance, folders)
save_to_csv(measure_keys_w, w_distance, 'w_distance.csv')
```

Preview statistics of the measurements.

In [161]:

```
#
print('--------pial model---------')
dic = dic_p

print('CorticalFlow Sample Counts {}'.format(len(dic['max'])))
for k in measure_keys:
    if k not in ['folder','n_samples']:
        print(k.rjust(15),' mean', round(np.mean(dic[k]),2), ' median', round(np.mec

print('--------white model---------')
dic = dic_w

print('CorticalFlow Sample Counts {}'.format(len(dic['max'])))
for k in measure_keys:
    if k not in ['folder','n_samples']:
        print(k.rjust(15),' mean', round(np.mean(dic[k]),2), ' median', round(np.mec
```

```
--------pial model---------
CorticalFlow Sample Counts 107
            RMS  mean 0.56  median 0.55  std 0.05
    diag_mesh_0  mean 232.76  median 232.6  std 3.0
    diag_mesh_1  mean 231.21  median 230.95  std 2.86
            max  mean 5.69  median 5.18  std 1.87
           mean  mean 0.37  median 0.37  std 0.03
            min  mean 0.0  median 0.0  std 0.0
--------white model---------
CorticalFlow Sample Counts 107
            RMS  mean 0.53  median 0.5  std 0.09
    diag_mesh_0  mean 221.94  median 222.11  std 2.76
    diag_mesh_1  mean 221.73  median 221.61  std 2.87
            max  mean 6.27  median 6.02  std 2.06
           mean  mean 0.34  median 0.34  std 0.04
            min  mean 0.0  median 0.0  std 0.0
```