```python
# -*- coding: utf-8 -*-
"""
Created on Tue Jul 25 16:22:50 2017
@author: davidblacher
"""

import FunITK as fun
from FunITK import Volume
import datetime
import numpy as np
import matplotlib.pyplot as plt
import os

idxSlice = 130
ph3_CT = Volume(path="../data/phantom3_MR_v2/ph3_CT_x1", method="CT",
                resample=1, ref=idxSlice)
ph3_CT_x4 = Volume(path="../data/phantom3_MR_v2/ph3_CT_x4", method="CT",
                   resample=4, ref=idxSlice)
ph3_CT_x9 = Volume(path="../data/phantom3_MR_v2/ph3_CT_x9", method="CT",
                   resample=9, ref=idxSlice)
ph3_CT_x25 = Volume(path="../data/phantom3_MR_v2/ph3_CT_x25", method="CT",
                    resample=25, ref=idxSlice)
ph3_CT_x100 = Volume(path="../data/phantom3_MR_v2/ph3_CT_x100", method="CT",
                     resample=100, ref=idxSlice)

ph3_MR_v2 = Volume(path="../data/phantom3_MR_v2/ph3_MR_v2_x1", method="MR",
                   resample=1, ref=idxSlice)
ph3_MR_v2_x4 = Volume(path="../data/phantom3_MR_v2/ph3_MR_v2_x4", method="MR",
                      resample=4, ref=idxSlice)
ph3_MR_v2_x9 = Volume(path="../data/phantom3_MR_v2/ph3_MR_v2_x9", method="MR",
                      resample=9, ref=idxSlice)
ph3_MR_v2_x25 = Volume(path="../data/phantom3_MR_v2/ph3_MR_v2_x25", method="MR",
                       resample=25, ref=idxSlice)
ph3_MR_v2_x100 = Volume(path="../data/phantom3_MR_v2/ph3_MR_v2_x100", method="MR",
                        resample=100, ref=idxSlice)

vol_list = [[ph3_CT, ph3_CT_x4, ph3_CT_x9, ph3_CT_x25, ph3_CT_x100,],
            [ph3_MR_v2, ph3_MR_v2_x4, ph3_MR_v2_x9, ph3_MR_v2_x25, ph3_MR_v2_x100]]
modality, sets = np.shape(vol_list)

length = ph3_CT_x100.zSize
lspacing = ph3_CT_x100.zSpace
sliceNumbers = np.arange(length, dtype=int)

# for data centered around iso-centre, this is real x-axis:
iso = 361
dist = ( (sliceNumbers - iso ) ).round(2)

warp_simple = np.zeros((sets, length, 2))
warp_iter = np.zeros((sets, length, 2))
warpMagnitude_simple = np.zeros((sets, length, 1))
warpMagnitude_iter = np.zeros((sets, length, 1))
lows_CT = np.zeros((sets, 2))
radii_CT = np.zeros((sets, 2))
lows_MR = np.zeros((sets, 4))
radii_MR = np.zeros((sets, 4))

# 2 DC for CT, 3 DC for MR (2 using MR.centroid, 1 using CT.centroid!)
DC_CT = np.zeros((sets, length, 2))
DC_CT_average = np.zeros((sets, 2))
DC_MR = np.zeros((sets, length, 4))
DC_MR_average = np.zeros((sets, 4))

for i in range(sets):
    vol_list[0][i].getCentroid()
    CT_DC_simple = vol_list[0][i].getDice()
    CT_DC_simple_average = vol_list[0][i].diceAverage
    CT_lower_simple = vol_list[0][i].lower
    CT_radius_simple = vol_list[0][i].bestRadius

    vol_list[1][i].getCentroid()
    MR_DC_simple = vol_list[1][i].getDice()
```

```
73      MR_DC_simple_average = vol_list[1][i].diceAverage
74      MR_lower_simple = vol_list[1][i].lower
75      MR_radius_simple = vol_list[1][i].bestRadius
76      vol_list[1][i].getMask()
77      MR_DC_simple_CT_COM = vol_list[1][i].getDice(centroid=vol_list[0][i].centroid)
78      MR_DC_simple_CT_COM_average = vol_list[1][i].diceAverage
79      MR_lower_simple_CT_COM = vol_list[1][i].lower
80      MR_radius_simple_CT_COM = vol_list[1][i].bestRadius
81      # this calculates the coordinate difference of MR.centroid relative to CT.centroid
82      warp_simple[i] = fun.sitk_coordShift(vol_list[0][i].centroid, vol_list[1][i].centroid)
83      # this calculates the norm (=absolute distance) between the centroids in each slice
84      warpMagnitude_simple[i] = fun.sitk_coordDist(warp_simple[i])
85
86
87      vol_list[0][i].getCentroid(percentLimit='auto', iterations=5, top=0.20)
88      CT_DC_iter = vol_list[0][i].dice
89      CT_DC_iter_average = vol_list[0][i].diceAverage
90      CT_lower_iter = vol_list[0][i].lower
91      CT_radius_iter = vol_list[0][i].bestRadius
92
93      vol_list[1][i].getCentroid(percentLimit='auto', iterations=6, top=0.20)
94      MR_DC_iter = vol_list[1][i].dice
95      MR_DC_iter_average = vol_list[1][i].diceAverage
96      MR_lower_iter = vol_list[1][i].lower
97      MR_radius_iter = vol_list[1][i].bestRadius
98      vol_list[1][i].getMask()
99      MR_DC_iter_CT_COM = vol_list[1][i].getDice(centroid=vol_list[0][i].centroid)
100     MR_DC_iter_CT_COM_average = vol_list[1][i].diceAverage
101     MR_lower_iter_CT_COM = vol_list[1][i].lower
102     MR_radius_iter_CT_COM = vol_list[1][i].bestRadius
103
104     # this calculates the coordinate difference of MR.centroid relative to CT.centroid
105     warp_iter[i] = fun.sitk_coordShift(vol_list[0][i].centroid, vol_list[1][i].centroid)
106     # this calculates the norm (=absolute distance) between the centroids in each slice
107     warpMagnitude_iter[i] = fun.sitk_coordDist(warp_iter[i])
108
109
110     DC_CT[i] = np.column_stack((CT_DC_simple,CT_DC_iter))
111     DC_CT_average[i] = CT_DC_simple_average,CT_DC_iter_average
112
113     DC_MR[i] = np.column_stack((MR_DC_simple, MR_DC_iter,
114                                 MR_DC_simple_CT_COM, MR_DC_iter_CT_COM))
115     DC_MR_average[i] = (MR_DC_simple_average, MR_DC_iter_average,
116                         MR_DC_simple_CT_COM_average, MR_DC_iter_CT_COM_average)
117     lows_CT[i] = CT_lower_simple, CT_lower_iter
118     lows_MR[i] = MR_lower_simple, MR_lower_iter, MR_lower_simple_CT_COM, MR_lower_iter_CT_COM
119     radii_CT[i] = CT_radius_simple, CT_radius_iter
120     radii_MR[i] = MR_radius_simple, MR_radius_iter, MR_radius_simple_CT_COM,
    MR_radius_iter_CT_COM
121
122
123 now = datetime.datetime.now()
124
125 COLUMNS  = ('sliceNo dist warp_x  warp_y  warpMagnitude  DC_CT  DC_MR '
126             ' DC_MR_CT-COM  warp_x*  warp_y*  warpMagnitude*   DC_CT*'
127             ' DC_MR*  DC_MR_CT-COM*')
128 for i in range(sets):
129     DATA = np.column_stack((sliceNumbers.astype(str),
130                             dist.astype(str),
131
132                             warp_simple[i].round(4).astype(str),
133                             warpMagnitude_simple[i].round(4).astype(str),
134                             DC_CT[i,:,0].round(4).astype(str),
135                             DC_MR[i,:,0].round(4).astype(str),
136                             DC_MR[i,:,2].round(4).astype(str),
137
138                             warp_iter[i].round(4).astype(str),
139                             warpMagnitude_iter[i].round(4).astype(str),
140                             DC_CT[i,:,1].round(4).astype(str),
141                             DC_MR[i,:,1].round(4).astype(str),
142                             DC_MR[i,:,3].round(4).astype(str)))
143  #   text = np.row_stack((NAMES, DATA))
```

```
144       head0 = ("{}_x{}\n path: {}\n thresholds:\n lower (simple): {},\n"
145       " lower (iter): {}\n upper: {}\n DC-average (simple): {} (bestRadius: {})\n"
146       " DC-average (iter): {} (bestRadius: {})\n".format(vol_list[0][i].method,
147       vol_list[0][i].resample, vol_list[0][i].path, lows_CT[i][0], lows_CT[i][1],
148       vol_list[0][i].upper, DC_CT_average[i][0], radii_CT[i][0],
149       DC_CT_average[i][1], radii_CT[i][1]))
150
151       head1 = ("{}_x{}\n path: {}\n thresholds:\n lower (simple): {},\n"
152       " lower (iter): {}\n lower (simple_CT-COM): {}\n lower (iter_CT-COM): {}\n"
153       " upper: {}\n DC-average (simple): {} (bestRadius: {})\n DC-average (iter): {}"
154       " (bestRadius: {})\n DC-average (CT-COM, simple): {} (bestRadius: {})\n"
155       " DC-average (CT-COM, iter): {} (bestRadius: {})".format(vol_list[1][i].method,
156        vol_list[1][i].resample, vol_list[1][i].path, lows_MR[i][0], lows_MR[i][1],
157        lows_MR[i][2], lows_MR[i][3], vol_list[1][i].upper, DC_MR_average[i][0],
158        radii_MR[i][0], DC_MR_average[i][1], radii_MR[i][1], DC_MR_average[i][2],
159        radii_MR[i][2], DC_MR_average[i][3], radii_MR[i][3]))
160
161       head = str(now) + '\n'+ head0 + head1 + '\n' + COLUMNS
162       np.savetxt('../data/output_txt/phantom3_out_txt/CT-MR_v2_x{}_{}_{}.txt'
163                  .format(vol_list[0][i].resample, now.date(), now.time()), DATA,
164                  delimiter="   &  ", header=head, comments="# ", fmt='%3s')
165
166
167   for i in range(sets):
168
169   # creates CT.masked using CT.mask,
170   # but assigns each slice the centroid distance*1000*spacing as pixel value
171       vol_list[0][0].applyMask(replaceArray=warp_simple[i][:,0])
172   # exports 3D image as .mha file
173       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
174                      "ph3_v2_out_x{}_warpX_simple.mha".format(vol_list[0][i].resample))
175       vol_list[0][0].applyMask(replaceArray=warp_simple[i][:,1])
176       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
177                      "ph3_v2_out_x{}_warpY_simple.mha".format(vol_list[0][i].resample))
178
179       vol_list[0][0].applyMask(replaceArray=warp_iter[i][:,0])
180       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
181                      "ph3_v2_out_x{}_warpX_iter.mha".format(vol_list[0][i].resample))
182       vol_list[0][0].applyMask(replaceArray=warp_iter[i][:,1])
183       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
184                      "ph3_v2_out_x{}_warpY_iter.mha".format(vol_list[0][i].resample))
185
186       vol_list[0][0].applyMask(replaceArray=warpMagnitude_simple[i])
187       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
188                      "ph3_v2_out_x{}_warpMagnitude_simple.mha".format(vol_list[0][i].resample))
189
190       vol_list[0][0].applyMask(replaceArray=warpMagnitude_iter[i])
191       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
192                      "ph3_v2_out_x{}_warpMagnitude_iter.mha".format(vol_list[0][i].resample))
193
194       vol_list[0][0].applyMask(replaceArray=DC_MR[i,:,0])
195       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
196                      "ph3_v2_out_x{}_DC_MR_simple.mha".format(vol_list[0][i].resample))
197
198       vol_list[0][0].applyMask(replaceArray=DC_MR[i,:,2])
199       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
200                      "ph3_v2_out_x{}_DC_MR_CT-COM_simple.mha".format(vol_list[0][i].resample))
201
202
203       vol_list[0][0].applyMask(replaceArray=DC_MR[i,:,1])
204       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
205                      "ph3_v2_out_x{}_DC_MR_iter.mha".format(vol_list[0][i].resample))
206
207       vol_list[0][0].applyMask(replaceArray=DC_MR[i,:,3])
208       fun.sitk_write(vol_list[0][0].masked, "../data/output_img/ph3_MR_v2_out_img/mha_files",
209                      "ph3_v2_out_x{}_DC_MR_CT-COM_iter.mha".format(vol_list[0][i].resample))
210
211
212   # instead of opening the created file manually, you can use this lines in
213   # the IPython console to start 3D Slicer and open it automatically:
214   # %env SITK_SHOW_COMMAND /home/david/Downloads/Slicer-4.5.0-1-linux-amd64/Slicer
215   # sitk.Show(CT.masked)
```