Image processing in Python: assignment 2

Part 1.

i. Write some useful information like a title and your names in some comments at the top of the code.

ii. Load in the file “IMG-0004-00001.dcm” and display it (you’ll need some libraries)

iii. Load the remaining three images, visualise them as well

iv. Modify your code from yesterday to allow automatic registration. To do this you need to:

   a. Change the shiftImages function so that it does not modify the global image

   b. Add a cost function to measure how well the registration is going

   c. Remove the automatic plot updating and links to the keyboard interface

```
def costFunction(image1, image2):
    #YOUR CODE HERE
```

v. Use your code, with a suitable optimizer from scipy.optimize, to register two images

   a. Try 00002 to 00001 and vice-versa – do the answers you get either way make sense?

```
Reg1 = brute(shiftImages, ((-100,100), (-100, 100)), args=(lungs1,
lungs2))
```

   1. What does ((-100,100), (-100, 100)) control?

   2. Are those sensible values?

   3. If you have time, try some other optimisers, e.g. dual_annealing. Are they better?

Part 2.

vi. Write some useful information like a title and your names in some comments at the top of the code.

vii. Copy your automatic registration code from the previous part and use it to register all images to IMG-0004-00001

   a. Write out the results as numpy arrays

   b. Bonus points: check for the existence of an existing registration result and only calculate if not found.

viii. Display pairs of images (e.g. 00004 over 00001) with an appropriate set of colourmaps and transparency. Can you see anything going on?

Part 2a.

ix. Create a figure and in it display the 00001 image

x. Find the code in the file “interface.py”

  a. Have a read of it. What do you think it is doing?

  b. Import the functions from the file

```
from interface import onMove,…
```

  c. Link the functions in the interface to the correct event handlers in matplotlib with some code like what you had yesterday

```
cid1 = fig2.canvas.mpl_connect('button_press_event', function name here)
cid2 = fig2.canvas.mpl_connect('motion_notify_event', function name here)
cid3 = fig2.canvas.mpl_connect('button_release_event', function name here)
cid4 = fig2.canvas.mpl_connect('key_press_event', function name here)
```

xi. Using the interface functions, move the red box over a region of interest

xii. You can now get the indices that define this region of interest with a line of code like this:

```
indices = [int(rect.get_y()), int(rect.get_y() + rect.get_height()),
int(rect.get_x()), int(rect.get_x() + rect.get_width())]
```

xiii. Using those indices, you can crop the image to just the region of interest, an example for the first image would be something like this:

```
baselineTumourRegion = lungs_1[indices[0]:indices[1],
indices[2]:indices[3]]
```

xiv. Extract the region of interest for each image in your series and choose a suitable feature of the image to look at (e.g. mean intensity)

  a. The feature should be a single number summarising the region of interest

  b. Have a look through some of the numpy documentation for ideas

xv. Plot your signal from the region as a function of time.

  a. What is happening?

  b. Does it correlate with what you see in the images?

General tips

- Always comment your code! (You are assessed on the amount and quality of comments)
- Run your script after every small change, make sure it still works!
- Don’t delete stuff if it doesn’t work! Ask for help.