Prof. Dr. Thomas Schultz
Rasha Sheikh (rasha@cs.uni-bonn.de)
Johannes Grün (gruen@cs.uni-bonn.de)

University of Bonn
B-IT / Institute of CS II
November 26, 2020

Winter term 2020/21

Image Acquisition and Analysis in Neuroscience

Assignment Sheet 3

Solution has to be uploaded by December 10, 2020, 8:00 a.m., via eCampus

If you have questions concerning the exercises, please use the forum on eCampus.

- Please work on this exercise in **small groups** of 3 students. Submit each solution only once, but clearly indicate who contributed to it by forming a team in eCampus. Remember that all team members have to be able to explain all answers.
- Please submit your answers in PDF format, and your scripts as *.py/*.ipynb files. If you are using Jupyter notebook, please also export your scripts and results as PDF.

Exercise 1 (Image Registration, *12 Points*)

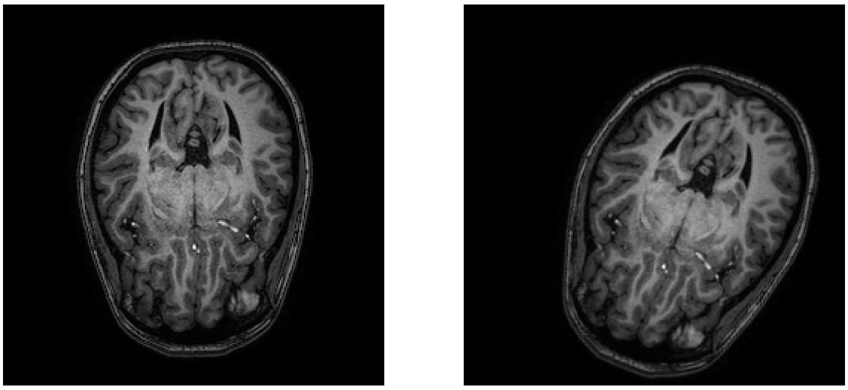

Figure 1: The original and transformed brain image.

Fig. 1 shows `axial.png`, a 2D slice of a brain MR scan, and `axial_transformed.png`, a translated and rotated version of the same slice. Both images are available from eCampus. Please take the following steps to develop a simple registration algorithm that brings them back into alignment:

a) Write a routine that can translate and rotate an input image by arbitrary amounts. Feel free to use the predefined functions offered by the package `scipy.ndimage`. (2P)

b) Implement a routine that evaluates the L2 cost function for a pair of input images. Test it by creating a plot that shows the L2 cost as a function of translating `axial.png` in x and y direction, as well as rotating it, and comparing it to the unchanged image. (2P)
Hint: In this example, it is safe to compute the cost over the fixed image, and pad the moving image with zeros if needed. You do not have to treat partial overlaps.

c) Implement a simple optimization method based on the golden ratio rule that uses your routines for image transformation and cost function evaluation to correctly align `axial_transformed.png` back to `axial.png`. Please submit the code, the resulting image, and the parameters for translation and rotation that you found.

(a) Write code that finds an initial bracket. (3P)
(b) Write code that, given a bracket, refines it until the bracket width is below some specified level of precision. (3P)
(c) Use that code to iterate over alternating optimization of translation and rotation until convergence. (2P)

Exercise 2 (Mutual Information, *4 Points*)

Implement a routine that evaluates the Mutual Information cost function. Compare its results to the L2 cost function by testing it in the same way as in exercise 1 b).

Hint: You do not have to implement fuzzy binning.

Exercise 3 (2D Rotation Around an Arbitrary Point, *4 Points*)

a) Compute the transformation matrix that corresponds to a rotation by 90 degrees clockwise around the point (4,-2). *Hint:* Start with a translation first to the origin. (3P)
b) What are the results of applying your transformation to the points (-3,2), (-5,3), (-3,5)? (1P)

Exercise 4 (Gradient Descent Stepsize, *5 Points*)

Assume that you attempt to find a local minimum of a scalar function $f(x)$ using a basic gradient descent scheme with stepsize λ^{-1}.

a) How much progress $f(x_k) - f(x_{k+1})$ do you expect to make in a given iteration if the stepsize is small enough that it is safe to rely on a first-order Taylor approximation? (1P) How could your implementation of gradient descent react if the ratio between actual and expected progress is low? (1P)
b) How much progress do you expect to make if you scale the independent variable, i.e., you optimize $f(\alpha x)$ with $\alpha > 0$, but keep the same stepsize λ^{-1}? (1P)
c) Argue why setting the stepsize to $\lambda^{-1} = \left(\tilde{\lambda} f''\right)^{-1}$ makes the expected progress independent of the scaling factor α. (1P) Argue why this independence is also achieved if $f(x) = [h(x)]^2$ and we set the stepsize to $\lambda^{-1} = \left(\tilde{\lambda}[h']^2\right)^{-1}$. (1P)

Good Luck!