1) Your python source code **in .py format**

Part 1: Joint histogram 10/100

a) Write a python function JointHist(I, J, bin) which calculates the joint histogram of two images of the same size.

b) For images of size n x p, verify that:

$$\sum_{i,j} H_{I,J}(i,j) = n \cdot p$$

c) Calculate and show the joint histogram of different pairs of images given in the handout (I1,J1 I2,J2, etc.). Describe briefly what you observe (you may want to use the logarithmic scale to visualize joint hist).

Part 2: similarity criteria 20/100 – do *not* use the pre-existing python libraries for this question!

a) Write a python function SSD(I,J) that calculates the sum squared difference between two images I and J of the same size. No 'for' loops!

b) Write a python function corr(I,J) that calculates the pearson correlation coefficient between two images of the same size. No 'for' loops!

c) Write a function MI(I,J) that calculates the mutual information between two images of the same size.

d) Compare the results of the three functions above on the different pairs of images provided. Describe briefly what you observe.

Part 3: spatial transforms 20/100

a) Generate a 3d grid of evenly spaced points (see Figure 1a)

b) Write a function rigid_transform(theta, omega, phi, p, q, r) that returns the matrix (in homogenous coordinates) of the rigid transform corresponding to:

   i. Rotation of angle theta around the x-axis
   ii. rotation of angle omega around the y-axis
   iii. rotation of angle phi around the z-axis
   iv. translation of vector **t**=(p,q,r)

   test your function on the 3d point cloud from (a) and show the result.

c) Write a function affine_transform(s, theta, omega, phi, p, q, r) that does the same as above (b) and adds a *scaling* factor s. test and show this function, as in (b) (example in figure 1c).

d) Given the 3 following matrices M1, M2, M3, determine the type of transformation corresponding to each matrix. Justify.

M1 =
0.9045	-0.3847	-0.1840	10.0000
0.2939	0.8750	-0.3847	10.0000
0.3090	0.2939	0.9045	10.0000
0	0	0	1.0000

M2 =
-0.0000	-0.2598	0.1500	-3.0000
0.0000	-0.1500	-0.2598	1.5000
0.3000	-0.0000	0.0000	0
0	0	0	1.0000

M3 =
0.7182	-1.3727	-0.5660	1.8115
-1.9236	-4.6556	-2.5512	0.2873
-0.6426	-1.7985	-1.6285	0.7404
0	0	0	1.0000

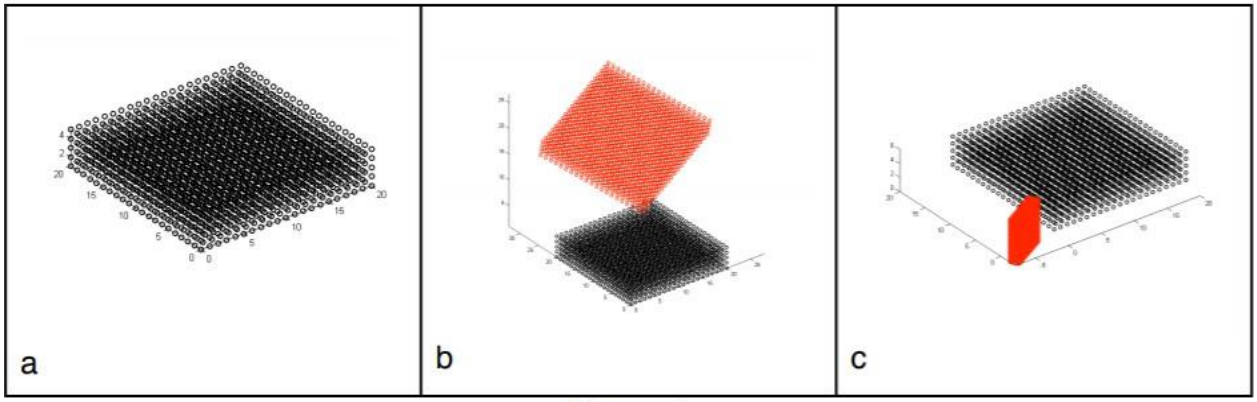


Figure 1

Part 4: simple 2d registration 40/100

a) Write a function translation(I,p,q) that returns a new image corresponding to image I translated by vector **t**=(p,q). p and q may be floats, therefore you must manage the interpolation. Call existing python interpolation functions.

b) Implement 2d registration minimizing SSD and considering only translations. As your function searches for the alignment, save the SSD for each iteration.
Test your function on the 3 different translations provided for image Brain_MRI_1.png (BrainMRI_2,3,4). show the registration you obtain, the SSD curve as a function of iteration, and discuss the quality of your registration. Describe the SSD curve, is it strictly decreasing, and if not, why?

c) Write a function rotation(I, theta) which returns an image I' that has been rotated by an angle theta, around the top left of the image. Do *not* use existing python rotate functions, you must create a grid corresponding to the image, rotate the grid, and interpolate (can use existing python interpolation functions).

d) Implement 2d registration minimizing SSD and considering only rotations. For each iteration, save the SSD. Test your function on the 3 differently rotated images supplied in the handout. Visualize the obtained registrations and SSD curve.

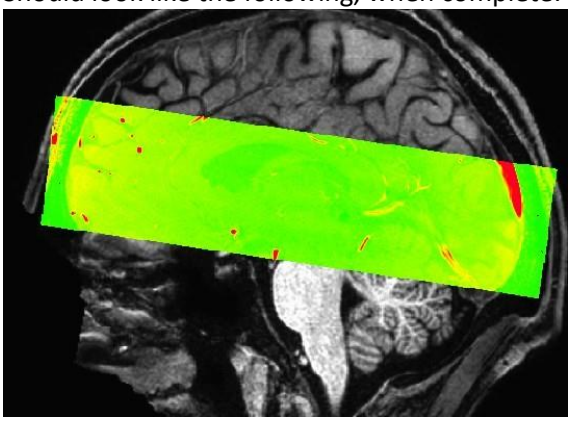
e) Implement a gradient descent for minimizing SSD, considering both translation and rotation. Register BrainMRI_2,3,4 onto Brain_MRI_1.
Which registrations converge? Which do not converge? Why do you think some fail to converge?

   i. To improve the performance of gradient descent, a more advanced optimization technique is needed. Improve your rigid registration with a better optimization technique (of your choice).
   Test for the 3 cases of rigid transformations given (BrainMRI_2,3,4)

Part 5: practical application: 20/100

I have provided you with two 3d images (tof.nii and t1.nii) at the link below. The images are from the same subject during the same scanning session, however, the images are not aligned. the spatial resolution (voxel size) and field of view are not the same (tof.nii is higher resolution, but captures only a slab of the brain, not the entire brain as for t1.nii). Your job is to align these two images using existing registration software. You may use either the FSL libraries or the Advanced Normalization Tools (ANTs). You will need a working Linux setup to run these tools. More detail will be provided later in the week during video lecture.

Should look like the following, when complete:



The colors are irrelevant, just a way to show the tof (green) has been properly aligned to the T1 (gray).

Link to data:
https://drive.google.com/file/d/17y1BLYJAzw_GFe1rv3JIQ5yHqLFPf9qF/view?usp=sharing