# CS463/516 Medical Imaging

Contents

[CS463/516 Medical Imaging 1](#_Toc74394275)

[Assignment 2 – image registration (alignment) 2](#_Toc74394276)

[Part 1: Joint histogram 10/100 2](#_Toc74394277)

[a) Write a python function JointHist(I, J, bin) which calculates the joint histogram of two images of the same size. 2](#_Toc74394278)

[b) For images of size n x p, verify that: 3](#_Toc74394279)

[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). 4](#_Toc74394280)

[Part 2: similarity criteria 20/100 4](#_Toc74394281)

[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! 4](#_Toc74394282)

[b) Write a python function corr(I,J) that calculates the pearson correlation coefficient between two images of the same size. No ‘for’ loops! 5](#_Toc74394283)

[c) Write a function MI(I,J) that calculates the mutual information between two images of the same size. 6](#_Toc74394284)

[d) Compare the results of the three functions above on the different pairs of images provided. Describe briefly what you observe. 6](#_Toc74394285)

[Part 3: spatial transforms 20/100 6](#_Toc74394286)

[a) Generate a 3d grid of evenly spaced points (see Figure 1a) 6](#_Toc74394287)

[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. 7](#_Toc74394288)

[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). 9](#_Toc74394289)

[d) Given the 3 following matrices M1, M2, M3, determine the type of transformation corresponding to each matrix. Justify. 13](#_Toc74394290)

[Part 4: simple 2d registration 40/100 16](#_Toc74394291)

[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. 16](#_Toc74394292)

[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? 16](#_Toc74394293)

[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). 16](#_Toc74394294)

[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. 16](#_Toc74394295)

[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) 16](#_Toc74394296)

[Part 5: practical application: 20/100 (10%, +10% bonus) 16](#_Toc74394297)

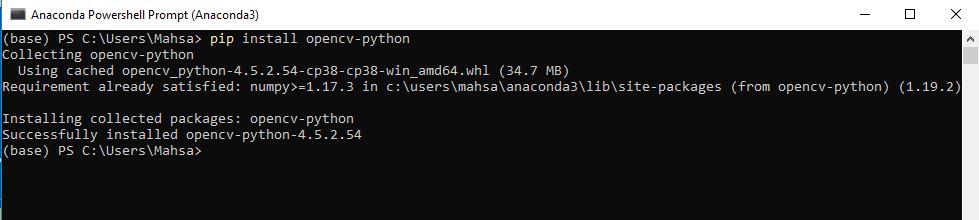
## Assignment 2 – image registration (alignment)

## Part 1: Joint histogram 10/100

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

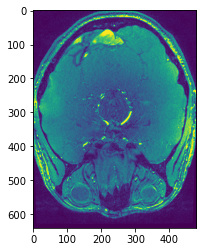
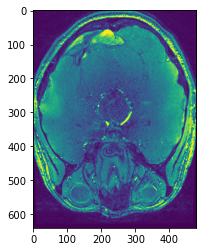
Histogram is as a plot which is related to frequency of pixels in a Gray Scale Image with pixel values (0 - 255). A grayscale image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information where the pixel value is different from 0 to 255. Images are as black-and-white, they are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest where a pixel can be considered as an every point in an image.

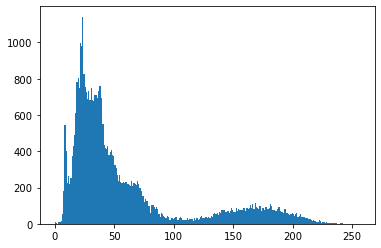
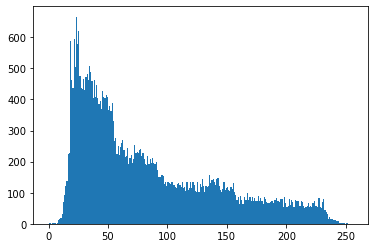
I firstly installed opency-python by execute "pip install opency-python" in anaconda prompt.



I selected two images as jpg extension and tried to open them by the aid of PIL library, then I resized them 220\*180 and saved them.

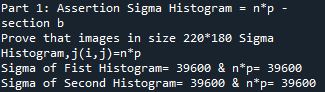
Furthermore, I illustrated them and tried to convert them to gray and with calcHist I calculated its histogram.





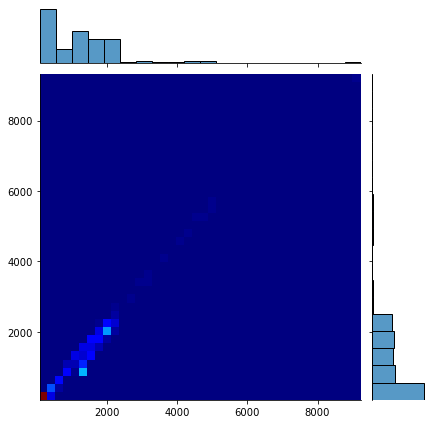
### For images of size n x p, verify that:

Then I tried if those images in the size of n\*p=220\*180 is equal to Sigma Histogram,j(i,j).



### 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).

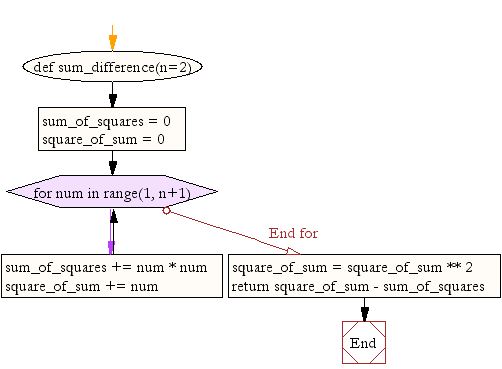
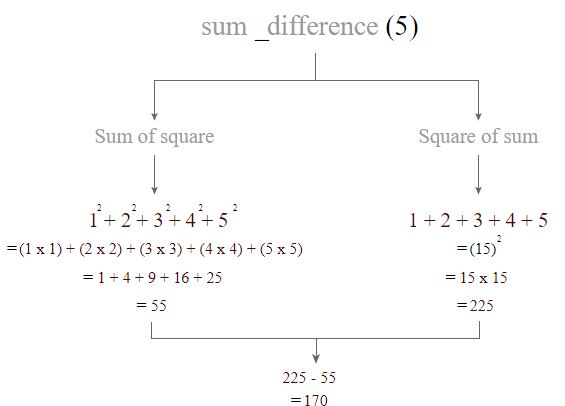
Last part is to visualize joint hist by using the logarithmic scale:



### Part 2: similarity criteria 20/100

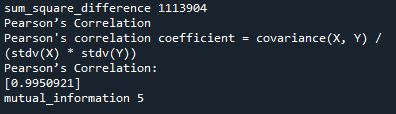
### Write a python function SSD(I,J) that calculates the sum squared difference between two images I and J of the same size. No ‘for’ loops!

I used “ssd=sum((x - y)\*\*2)” to compute sum\_square\_difference



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

Computation of Pearson's correlation coefficient = covariance(X, Y) / (stdv(X) \* stdv(Y)).

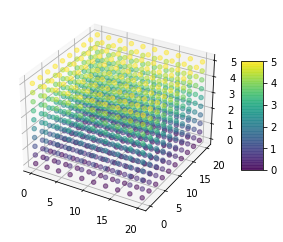


### c) Write a function MI(I,J) that calculates the mutual information between two images of the same size.

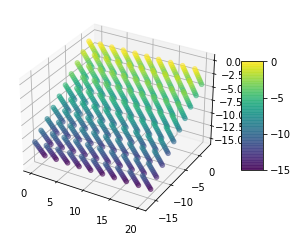
### d) Compare the results of the three functions above on the different pairs of images provided. Describe briefly what you observe.

## Part 3: spatial transforms 20/100

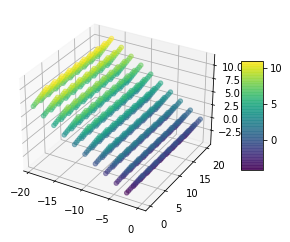
### Generate a 3d grid of evenly spaced points (see Figure 1a)



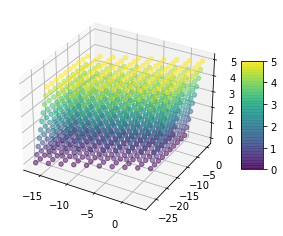
### Write a function rigid\_transform(theta, omega, phi, p, q, r) that returns the matrix (in homogenous coordinates) of the rigid transform corresponding to: i. Rotation of angle theta around the x-axis ii. rotation of angle omega around the y-axis iii. rotation of angle phi around the z-axis iv. translation of vector t=(p,q,r) test your function on the 3d point cloud from (a) and show the result.



Rotate around X



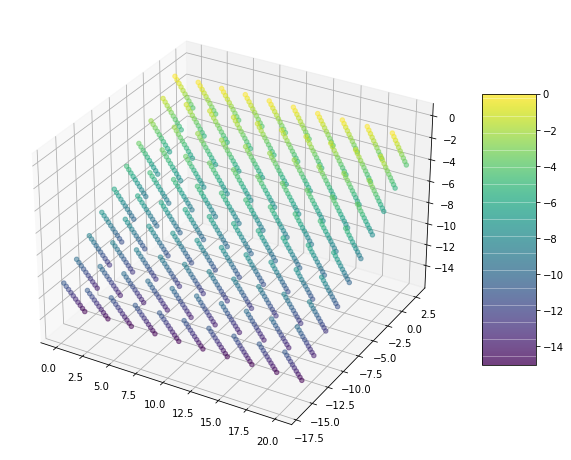
Rotate around Y

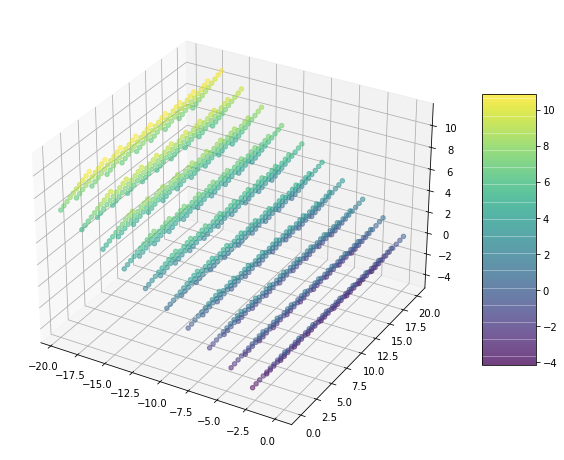


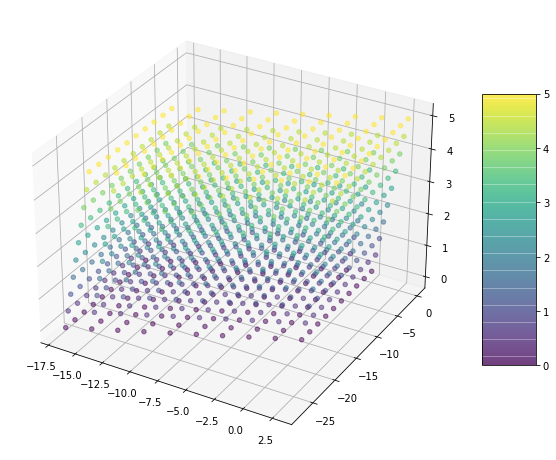
Rotate around Z

### 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).

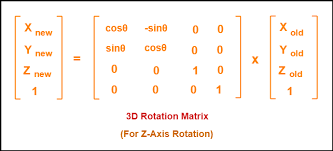
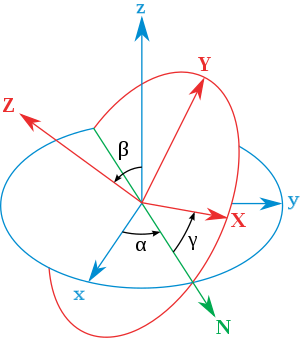
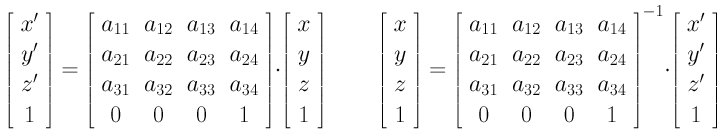
Scaling param=10

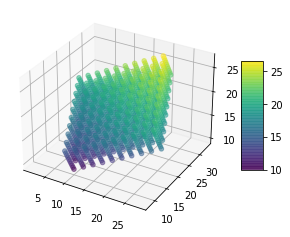




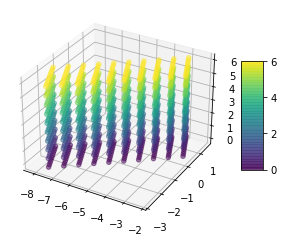


### Given the 3 following matrices M1, M2, M3, determine the type of transformation corresponding to each matrix. Justify.

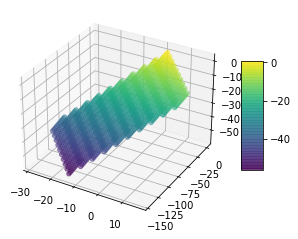




Matrix M1



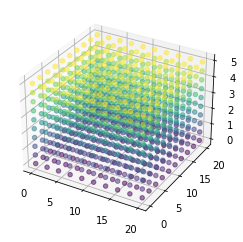
Matrix M2



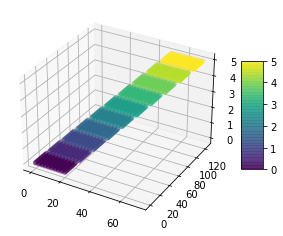
Matrix M3

## Part 4: simple 2d registration 40/100

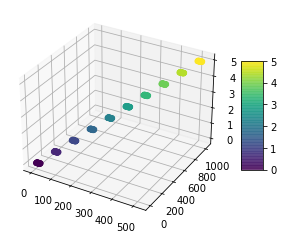
### Write a function translation(I,p,q) that returns a new image corresponding to image I translated by vector t=(p,q). p and q may be floats, therefore you must manage the interpolation. Call existing python interpolation functions.



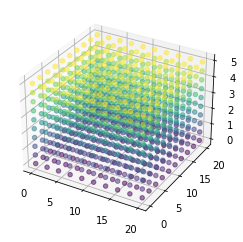
Original image



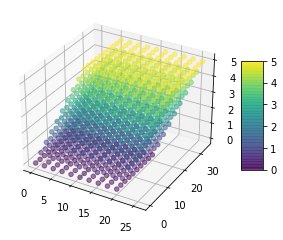
p=10.2, q=20.79



p=100.2, q=200.79



p=0.1, q=0.2



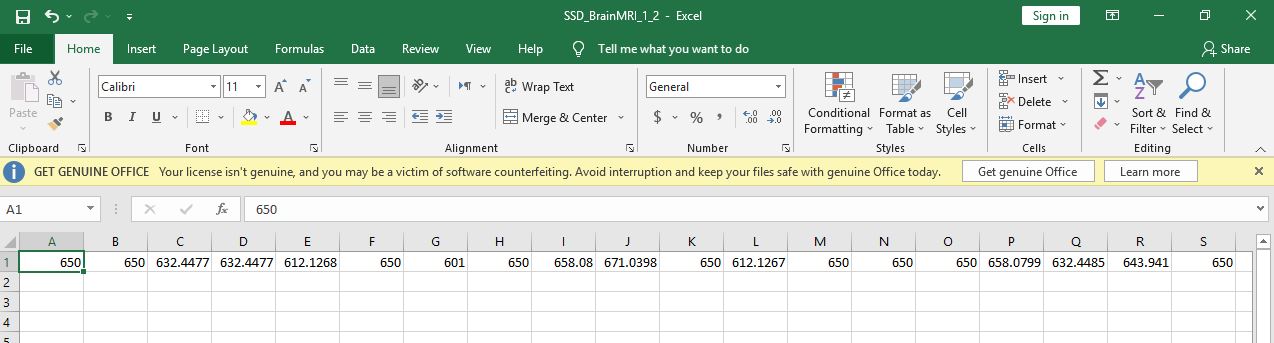
p=1.2, q=2.79

Translation changes the position and however the p,q is smaller the position stays stable while for greater value the position gets smaller.

### 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?



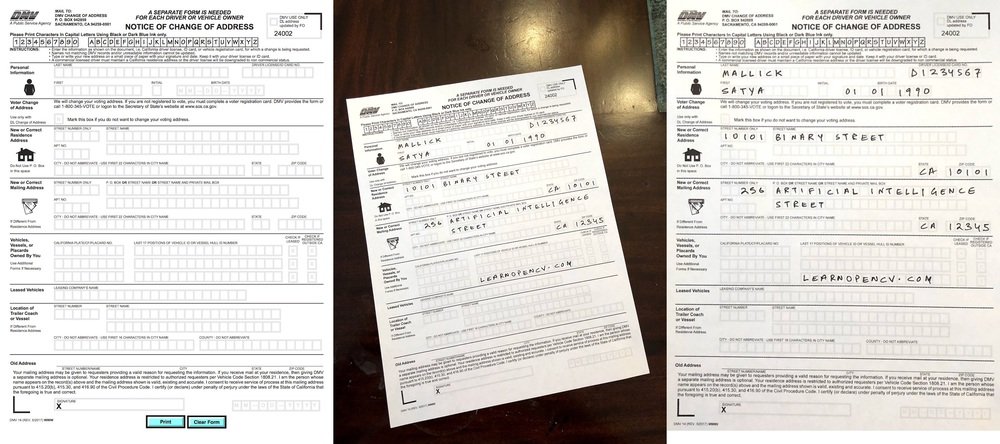
registration\_BrainMRI\_1\_and\_2

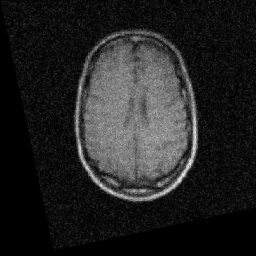


SSD \_ Brain 1\_2

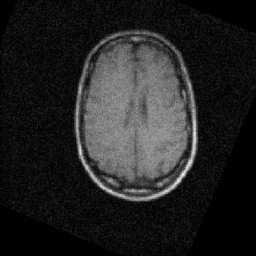
SSD curve, is strictly decreasing, and if not, why?

SSD is not always decreasing as it obvious on the csv file. Sometimes increasing and sometimes decreasing, due to in each iteration it is aligning into new point with different coordination.

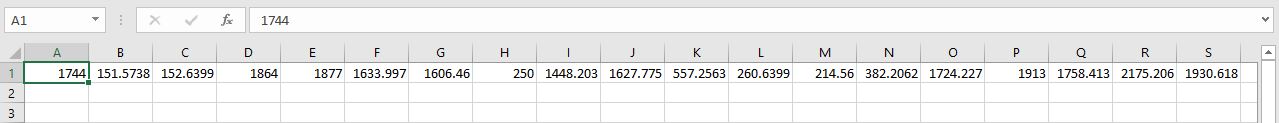




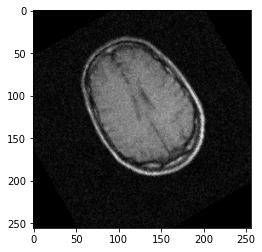
registration\_BrainMRI\_1\_and\_4



registration\_BrainMRI\_1\_and\_4



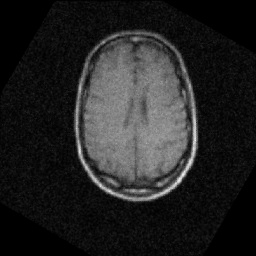
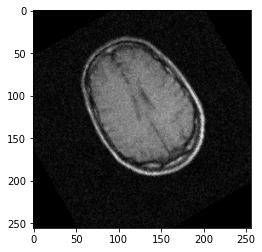
### Write a function rotation(I, theta) which returns an image I’ that has been rotated by an angle theta, around the top left of the image. Do not use existing python rotate functions, you must create a grid corresponding to the image, rotate the grid, and interpolate (can use existing python interpolation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



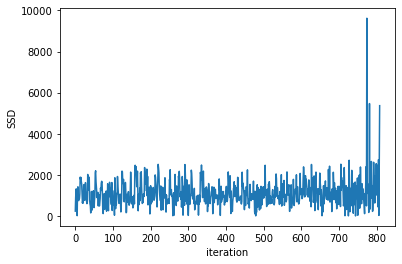
Rotate by the angle 30 degree

### 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.

I gave rotated picture by the angle 30 degree from above assignment in order to align it into the straight picture as it was before: Here I used built in function **hypot** for computing ssd and saved it into csv for each iteration then I plot ssd and the picture after function for visualizing.



Before using function After using function



SSD values in each iteration

### 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)

There is implementation for gradient descent for minimizing SSD and in order to optimize minimizing SSD in a better solution I have selected ADAM optimizer because its accuracy and speed it much more better Gradient Descent.

The answer is Adam is much more near to the real ssd rather than gradient descent:

Assignment 4 - part e: Gradient descent for minimizing SSD, considering both translation and rotation.

Iteration for gradient descent 796

X optimal value is 66.6907830766504

The local minimum with gradient descent happens at 66.6907830766504

Iteration for Adam Optimizer 796

X optimal value is 393.50307687947003

The local minimum with Adam Optimizer happens at 393.50307687947003

## Part 5: practical application: 20/100 (10%, +10% bonus)

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:

Reference:

https://www.mathplanet.com/education/geometry/transformations/transformation-using-matrices