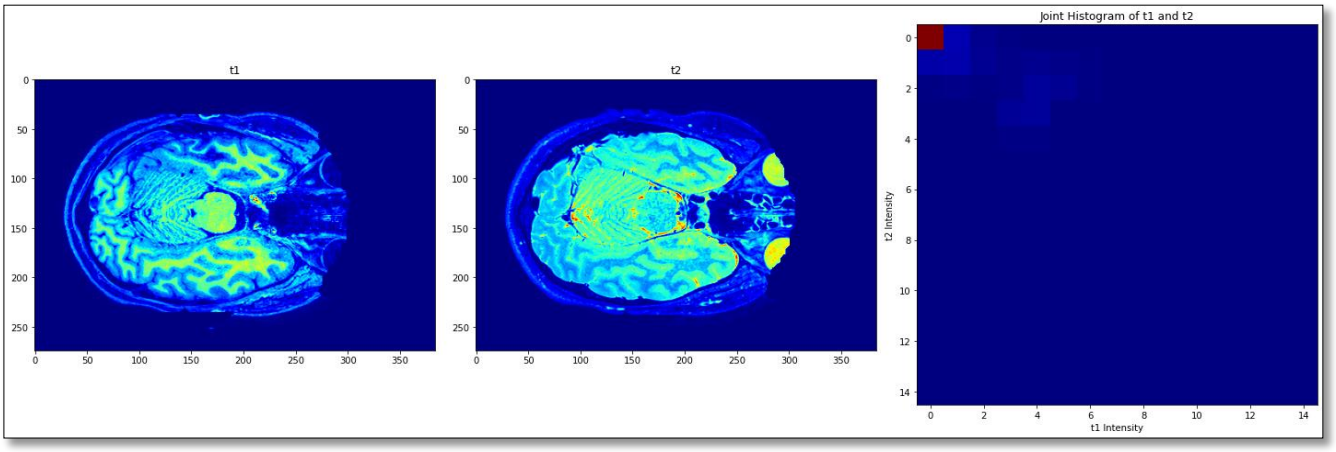
Part 1: Joint Histogram

a) Create python function JointHist(I, J, bin) calculating the joint histogram of two images of the same size. Pre-existing functions not to be used.

Solution:

In this task we have written a python function to create a joint histogram of two images of same size using histogramdd of numpy and plotting using matplotlib

Result:



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

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

Solution:

Calculated sum of joint histogram created from two images t1 and t2 using np.sum().
Also calculated the product of each dimension of one of the image used. Both the result came out to be same.
Note: In the below implementation the joint histogram was created out of two images of size (n,p,r). Hence the above formula is proved to be true for n*p*r as well.

Result:

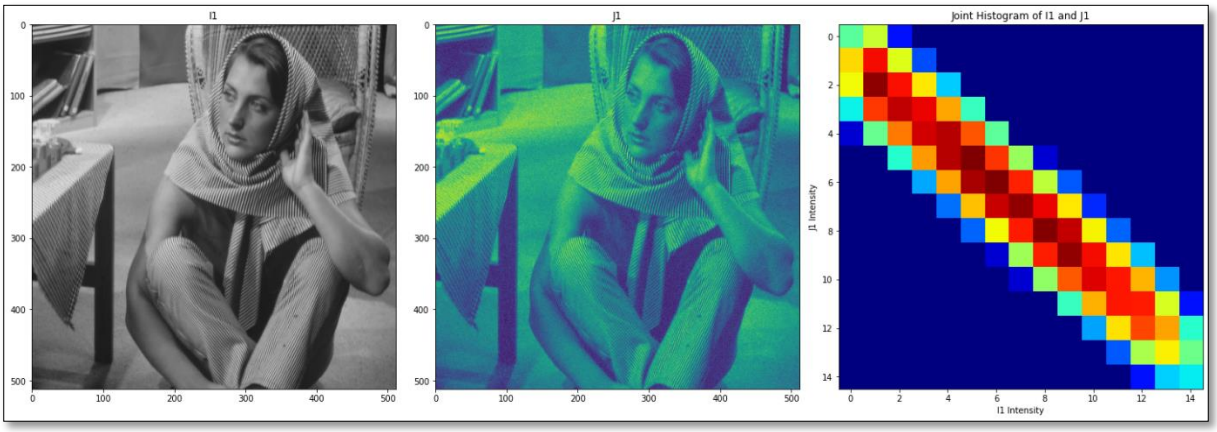
```
In [314]: runfile('C:/Users/scorp/OneDrive/Documents/Bishops Course/Sem2/
VolumetricImageAnalsis/Assignment2/Part1.py', wdir='C:/Users/scorp/OneDrive/
Documents/Bishops Course/Sem2/VolumetricImageAnalsis/Assignment2')
The shape of the t1 is (274, 384, 384)
The shape of the t2 is (274, 384, 384)
Sum of joint histogram of t1 and t2 is 40402944
Product of each dimension (274* 384* 384) of t1 is 40402944
Sum of joint histogram of two images is equal to product of each dimension of the
size of an image used

In [315]:
```

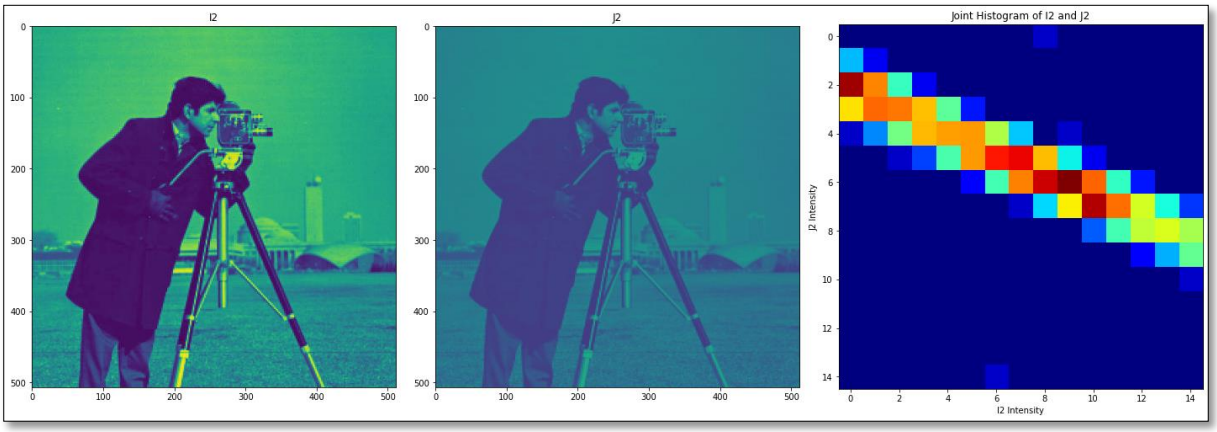
c) Calculate and show the joint histogram of different pairs of images given (I1,J1), (I2,J2)…etc. briefly describing the observation.

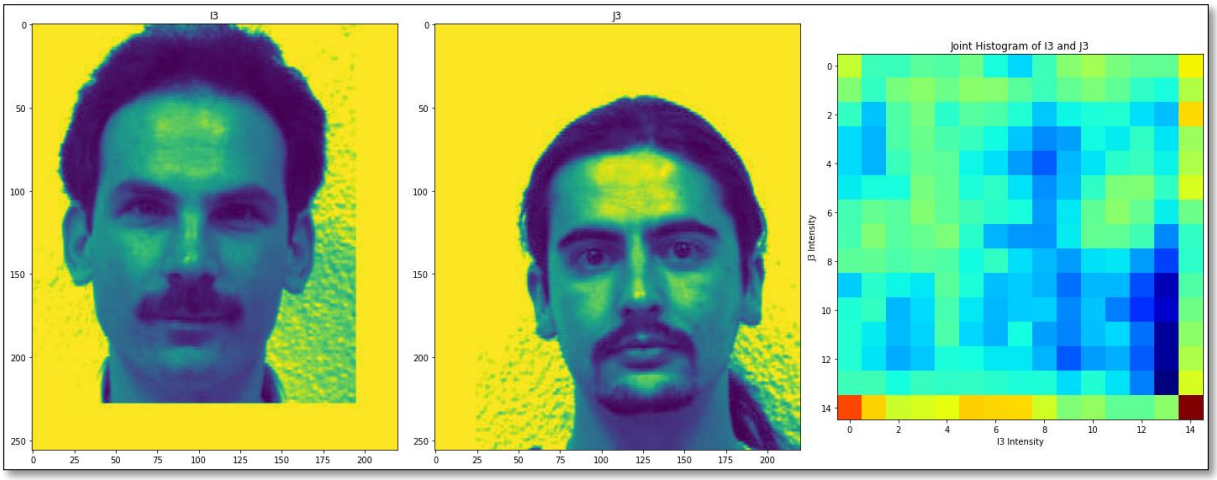
Solution:

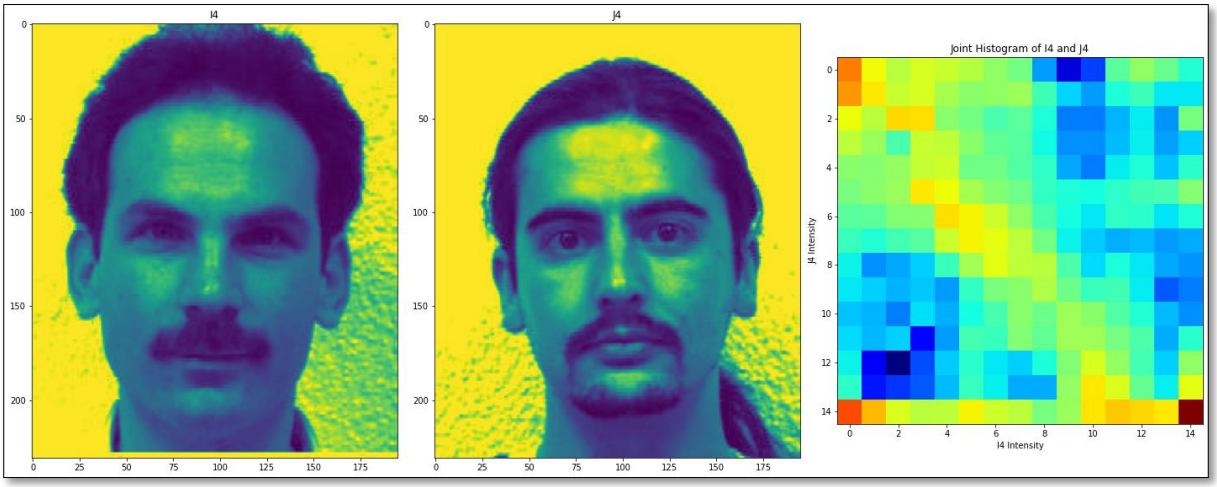
We have used the function created in part a to display the joint histogram of each pair of images I and J. We have used the logarithmic scale too to visualize the histogram more clearly.

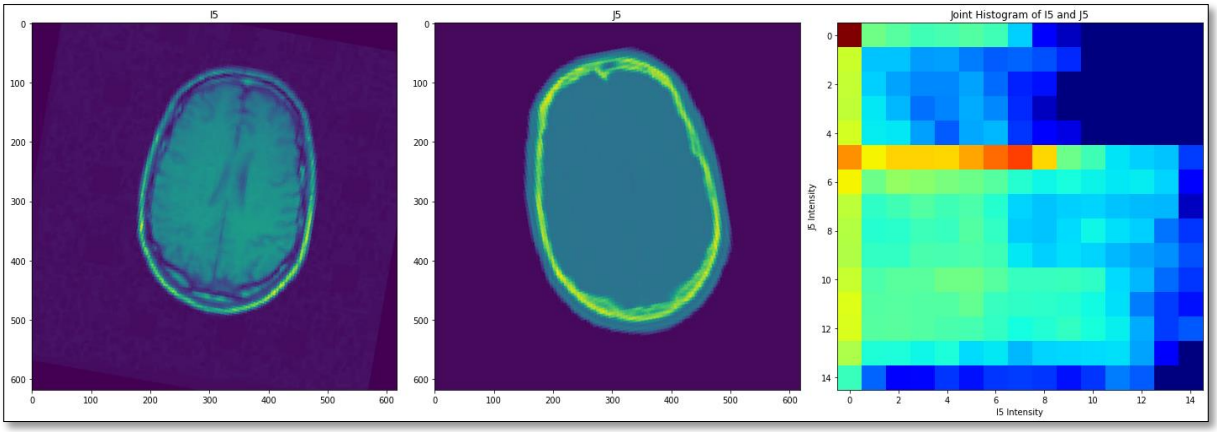
Result:

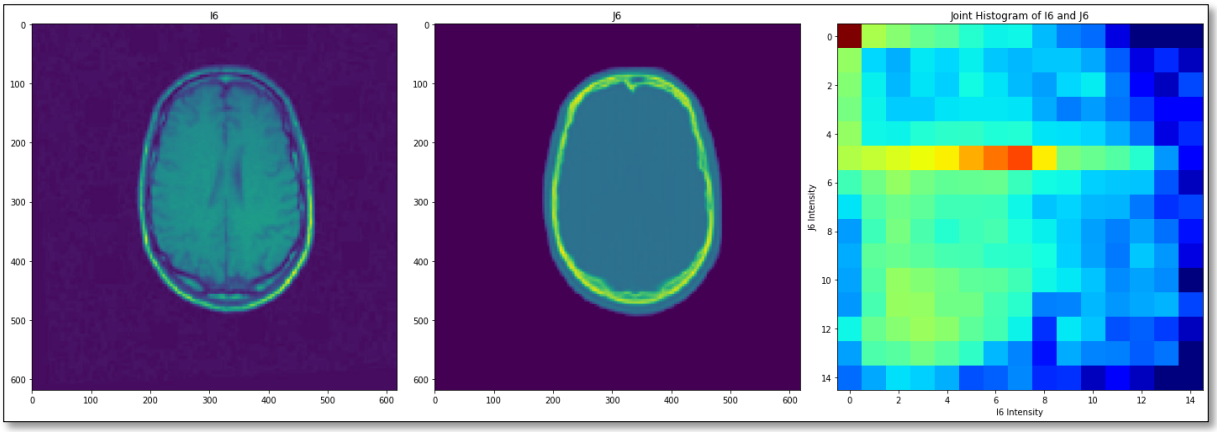





I3
J3
Joint Histogram of I3 and J3
I3 Intensity
J3 Intensity


I4
J4
Joint Histogram of I4 and J4
I4 Intensity
J4 Intensity


I5
J5
Joint Histogram of I5 and J5
I5 Intensity
J5 Intensity



Part 2: Similarity Criteria

For two images I and J of the same size :

a) Create python function SSD(I,J) calculating the sum squared difference
b) Create python function CORR(I,j) calculating the pearson correlation coefficient
c) Create python function MI(I,J) calculating the mutual information
d) Compare results of the above three functions for different pair of images providing along with observations.

Observation:

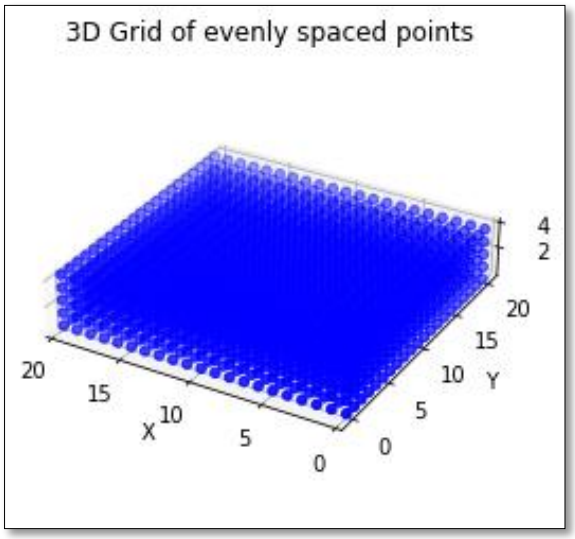
Result:

```
In [269]: runfile('C:/Users/scorp/OneDrive/Documents/Bishops Course/Sem2/VolumetricImageAnalsis/Assignment2/Part2.py',
VolumetricImageAnalsis/Assignment2')
Calculating the sum squared difference between two images I and J:
Images I1 and J1 are not of same shape
SSD of Image I2 and Image J2 is 30364219
SSD of Image I3 and Image J3 is 4401503
SSD of Image I4 and Image J4 is 3848301
SSD of Image I5 and Image J5 is 20998301
SSD of Image I6 and Image J6 is 42510000


Calculating the pearson correlation coefficient between two images I and J:
Images I1 and J1 are not of same shape
CORR of Image I2 and Image J2 is [[2.49969168 0.99621727]
 [0.99621727 0.40005242]]
CORR of Image I3 and Image J3 is [[1.0212648  0.14339325]
 [0.14339325 0.97921275]]
CORR of Image I4 and Image J4 is [[0.96004816 0.56404675]
 [0.56404675 1.04166066]]
CORR of Image I5 and Image J5 is [[0.84324697 0.65643844]
 [0.65643844 1.18589841]]
CORR of Image I6 and Image J6 is [[0.86555368 0.78025159]
 [0.78025159 1.1553359 ]]


Calculating the mutual information between two images I and J:
Images I1 and J1 are not of same shape
MI of Image I2 and Image J2 is 2.228748876135049
MI of Image I3 and Image J3 is 0.6687586245620836
MI of Image I4 and Image J4 is 1.0241532987504278
MI of Image I5 and Image J5 is 0.55488206712046
MI of Image I6 and Image J6 is 0.6748563215819839

In [270]:
```

Part 3: Spatial Transforms

a) Generate a 3d grid of evenly spaced points

Result:



b) Create a python 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

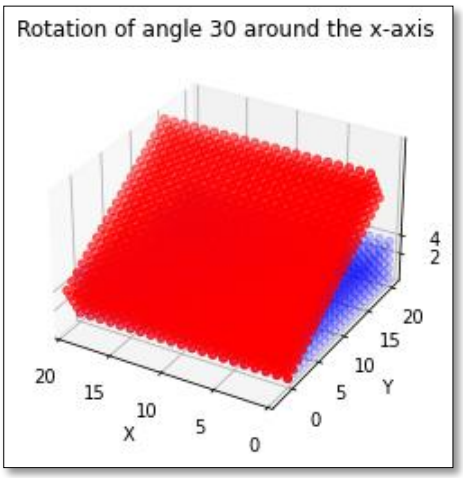
ii. Rotation of angle omega around the y-axis

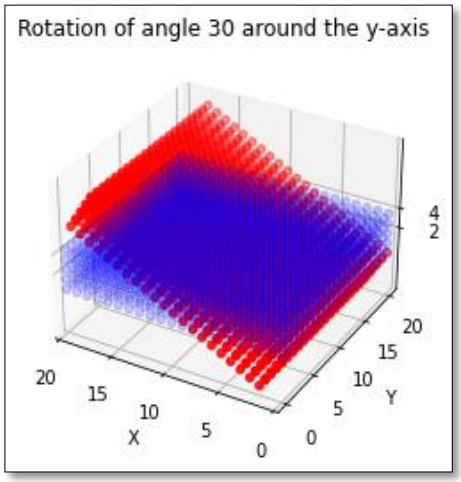
iii. Rotation of angle phi around the z-axis

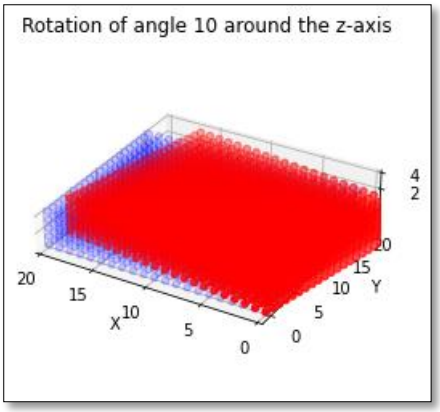
iv. Translation of vector t=(p,q,r) test your function on the 3d point cloud from (a) and show the result.

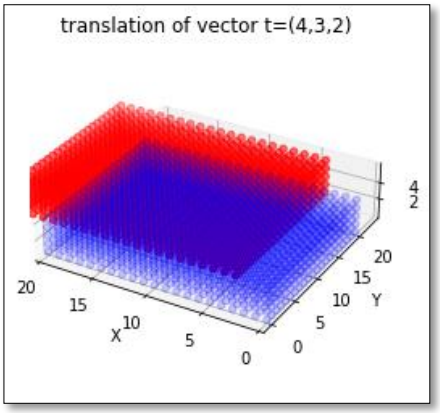
Result:

Note: Blue 3D point cloud is original and Red is the transformed one





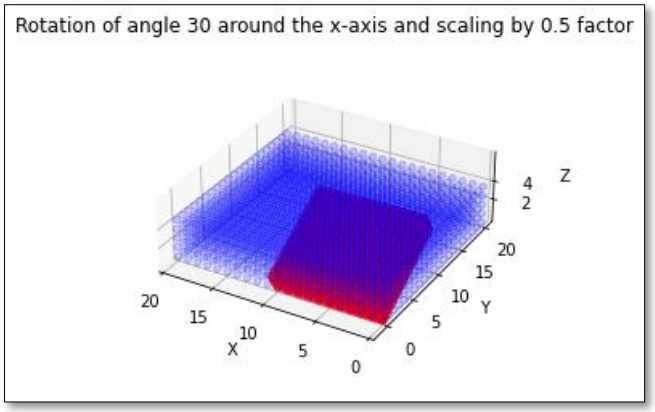


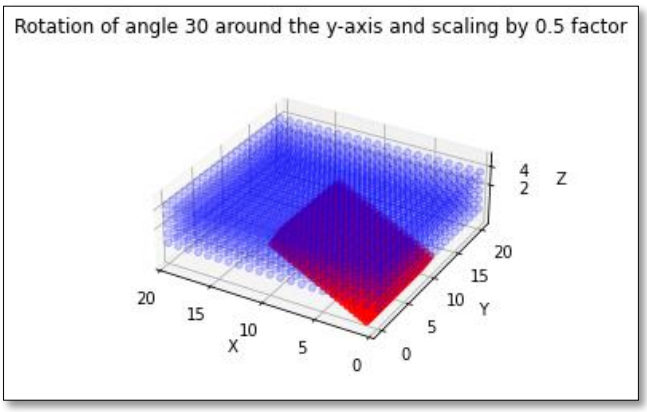


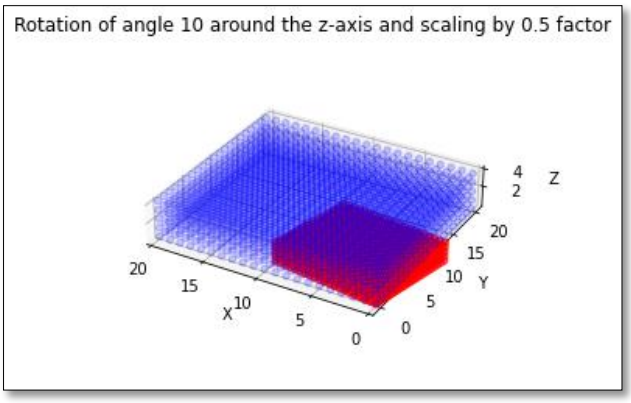
c) Write a function affine_transform(s, theta, omega, phi, p, q, r) that does the same as above (b) and adds a scaling factor s. test and show this function, as in (b)

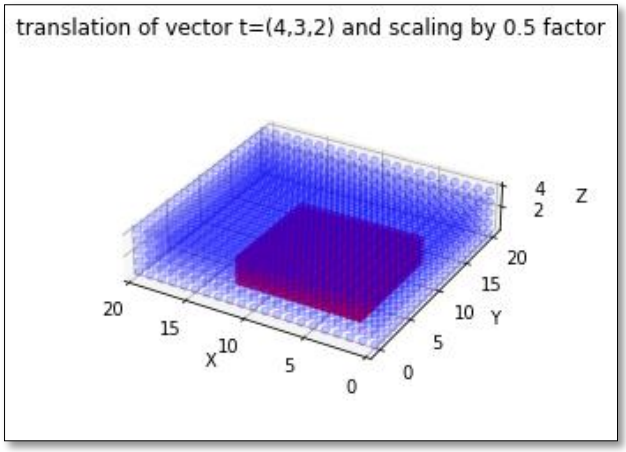
Result:

Note: Blue 3D point cloud is original and Red is the transformed one


Rotation of angle 30 around the x-axis and scaling by 0.5 factor
Z
4
2
20
15
10
5
0
X
Y


Rotation of angle 30 around the y-axis and scaling by 0.5 factor
Z
4
2
20
15
10
5
0
X
Y


Rotation of angle 10 around the z-axis and scaling by 0.5 factor
Z
4
2
20
15
10
5
0
X
Y



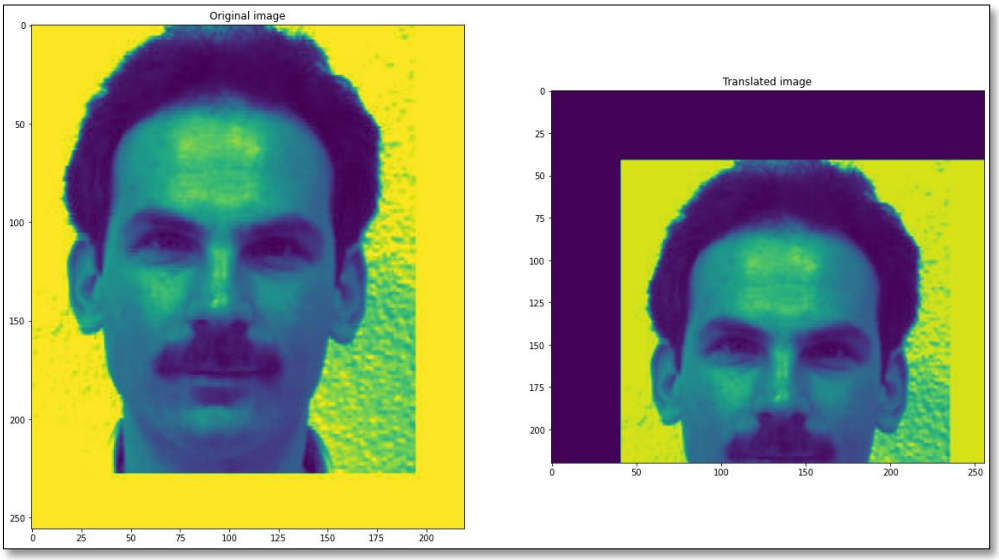
d) Given the 3 following matrices M1, M2, M3, determine the type of transformation corresponding to each matrix. Justify.

Result:

Part 4 : Simple 2d registration

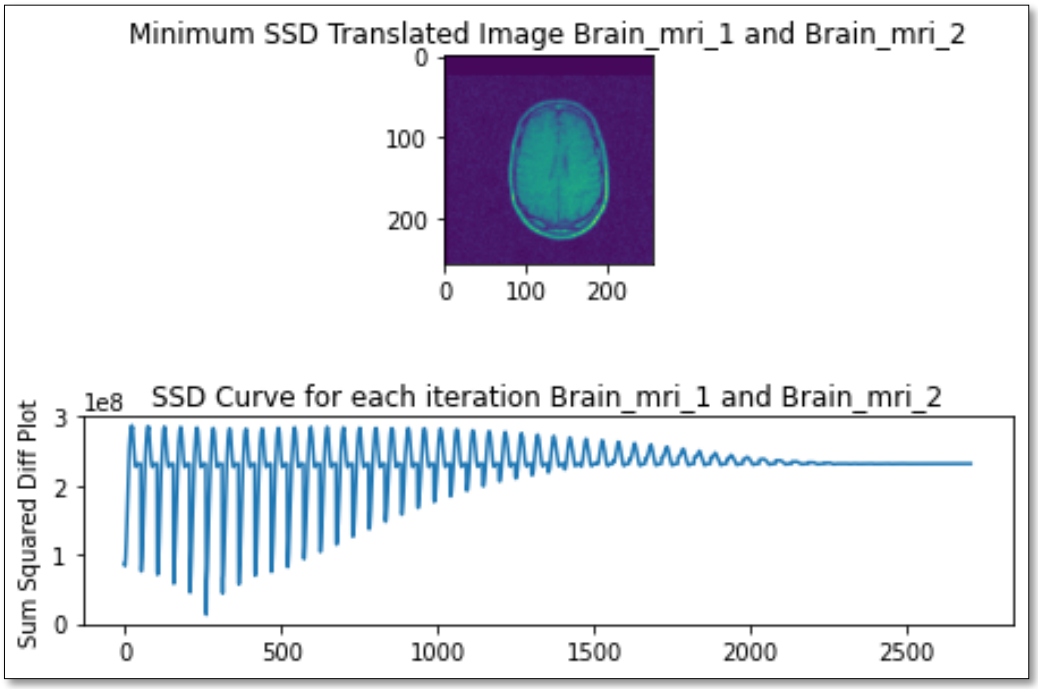
a) Translate an image by given vector t=(p,q) in x and y direction using pre-existing interpolation python function incase p and q values used are floats

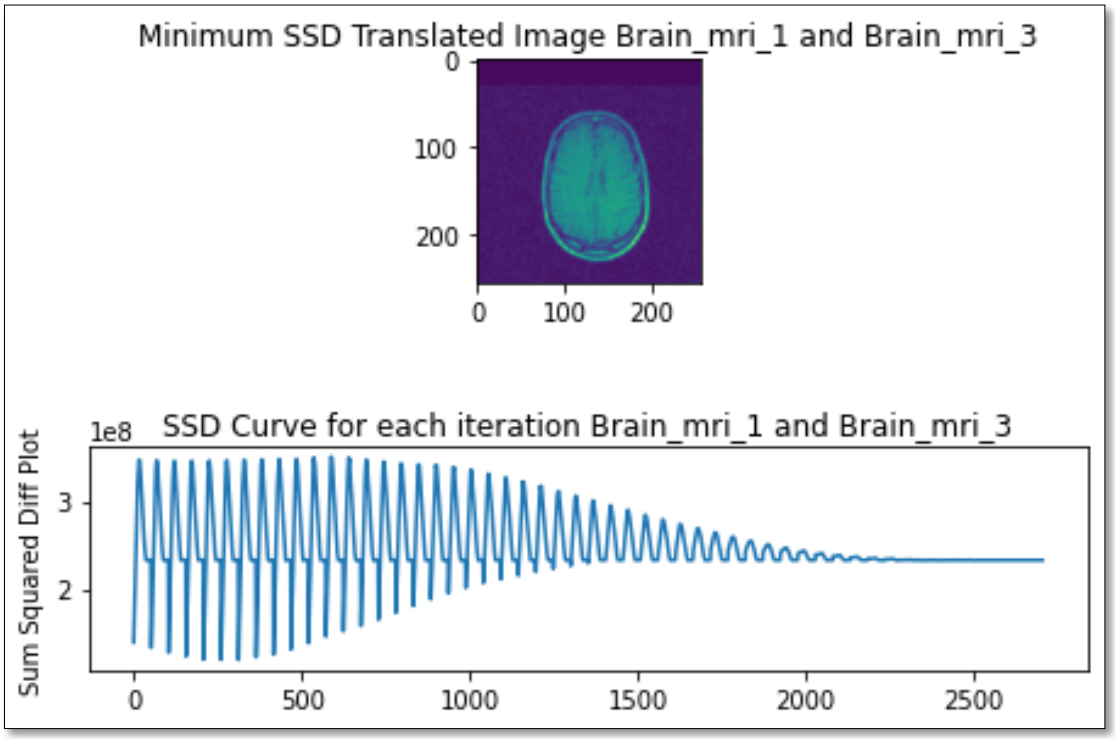
Result:

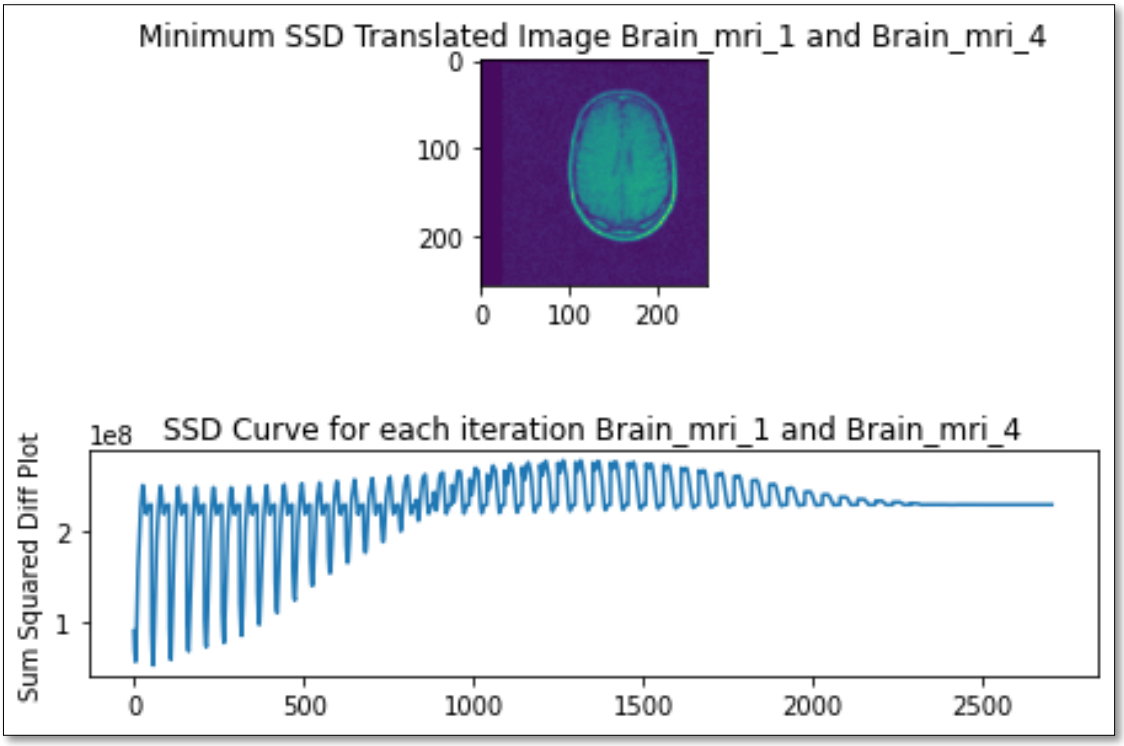


b) 2D registration minimizing SSD and considering only translations. The function written should search for the alignment, SSD for each iteration to be saved and plotted. The function to be tested on the 3 different translations provided for image Brain_MRI_1.png (BrainMRI_2,3,4).

Result:


Minimum SSD Translated Image Brain_mri_1 and Brain_mri_2
0
100
200
0
100
200
SSD Curve for each iteration Brain_mri_1 and Brain_mri_2
1e8
Sum Squared Diff Plot
3
2
1
0
0
500
1000
1500
2000
2500

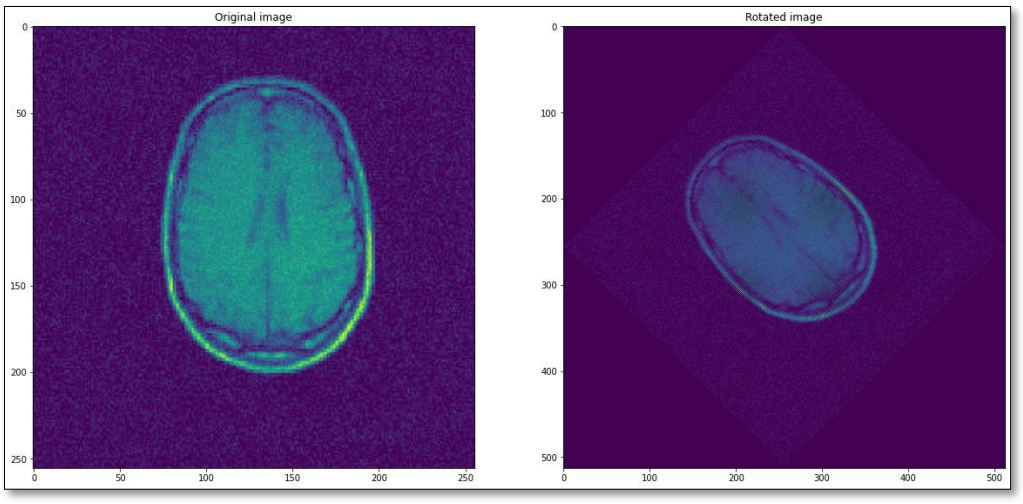

Minimum SSD Translated Image Brain_mri_1 and Brain_mri_3
0
100
200
0
100
200
SSD Curve for each iteration Brain_mri_1 and Brain_mri_3
1e8
Sum Squared Diff Plot
3
2
0
500
1000
1500
2000
2500



c) Rotate an image by an angle theta, around the top left of the image. Pre-existing python rotate functions is not to be used. Pre-existing interpolation python function can be used.

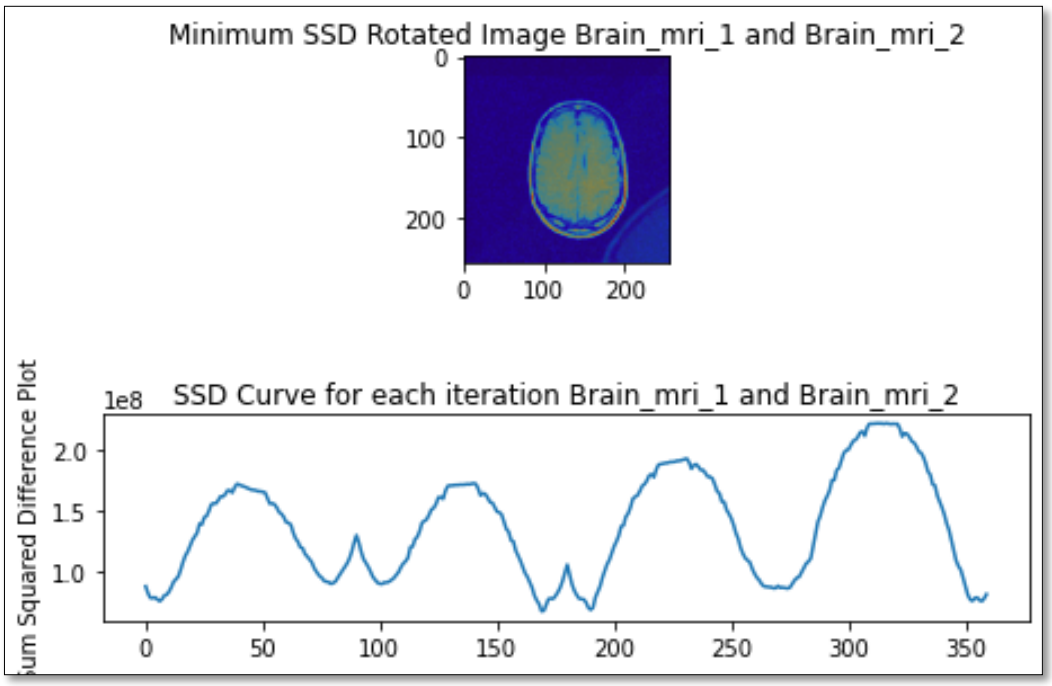
Result:

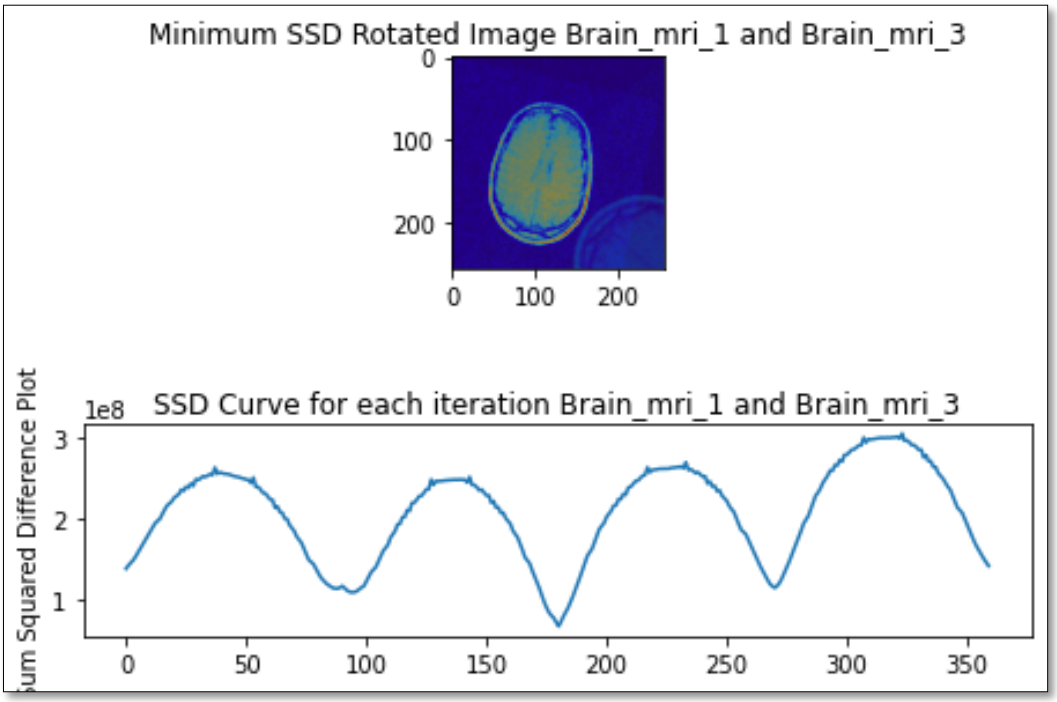
Rotate image by 45 degrees around the top left of the image. Used math and numpy libraries for implementation. Pre-existing interpolation python function is used.

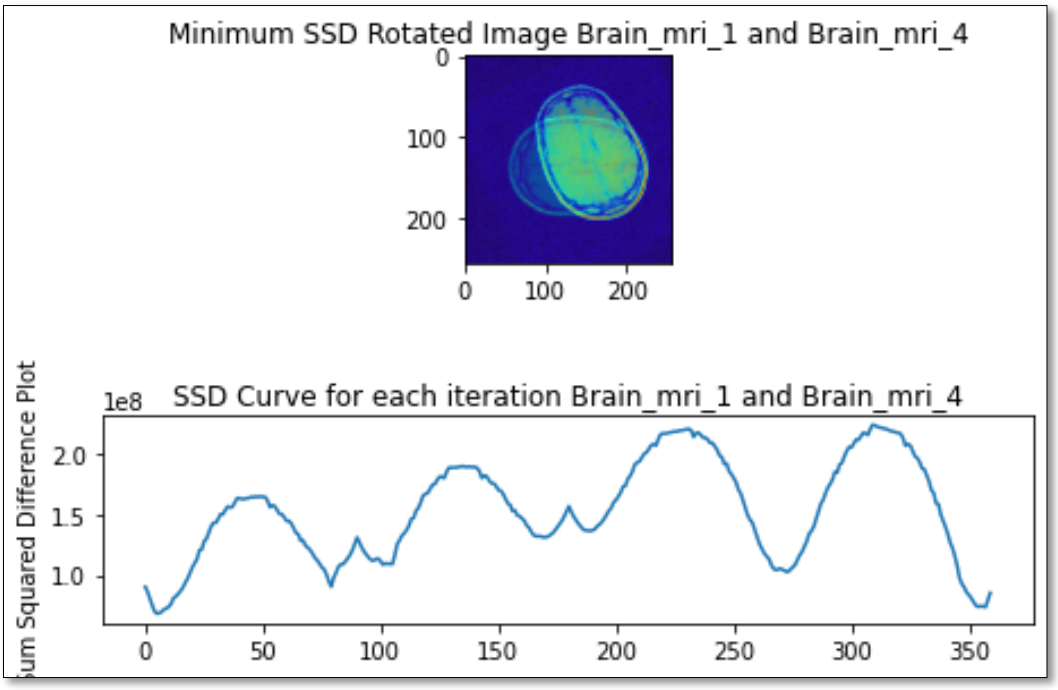


d) 2d registration of an image minimizing SSD and considering only rotations. For each iteration, SSD values are to be saved and plotted. The function to be tested on the 3 different translations provided for image Brain_MRI_1.png (BrainMRI_2,3,4).

Result:



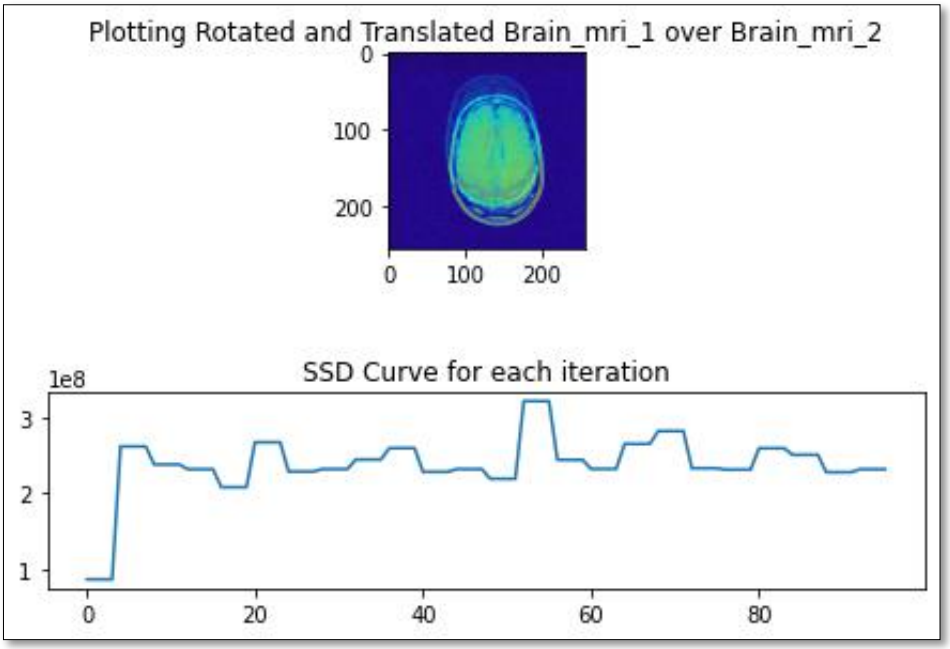


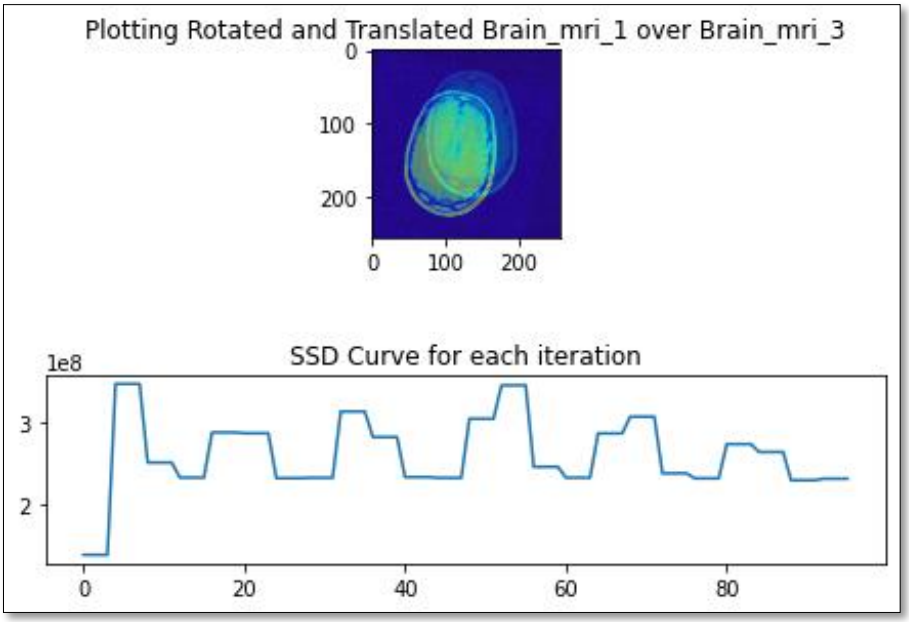


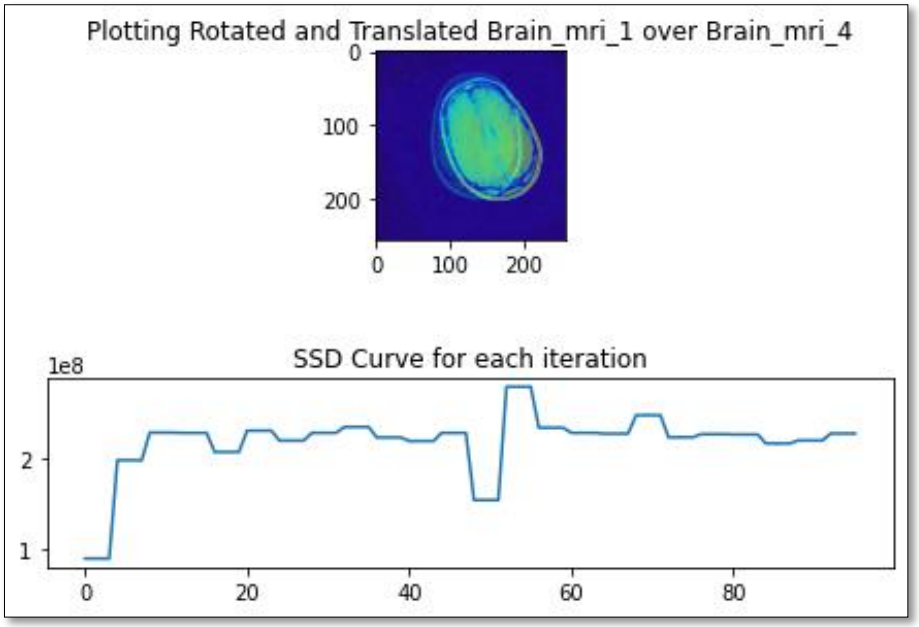
e) Implement a gradient descent for minimizing SSD, considering both translation and rotation. Register BrainMRI_2,3,4 onto Brain_MRI_1. Which registrations converge? Which do not converge? Why do you think some fail to converge?

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)

Result:


Plotting Rotated and Translated Brain_mri_1 over Brain_mri_2
0
100
200
0
100
200
SSD Curve for each iteration
1e8
3
2
1
0
20
40
60
80

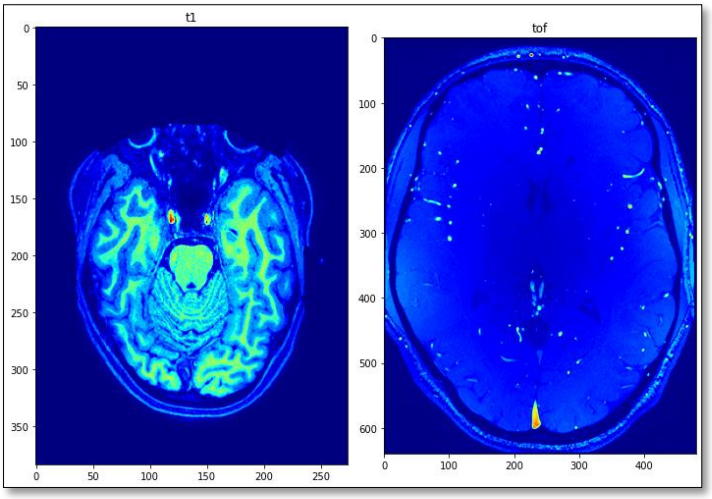

Plotting Rotated and Translated Brain_mri_1 over Brain_mri_3
0
100
200
0
100
200
SSD Curve for each iteration
1e8
3
2
0
20
40
60
80



Part 4: Practical Application

a) Align tof image on t1 image

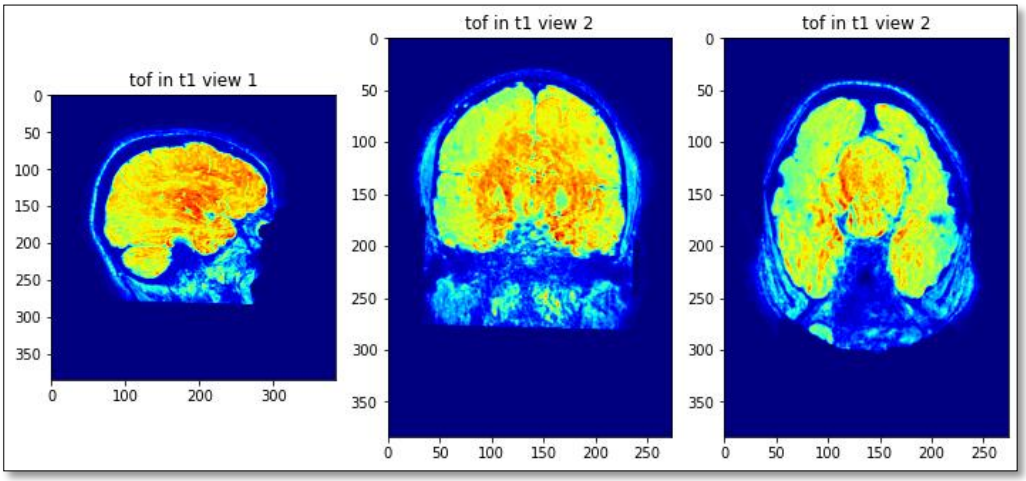
- Plotting images t1 and tof



Command 1:
flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -in t2.nii -ref t1.nii -out t2_in_t1_aligned.nii.qz

Result:

```
shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$ flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -in tof.nii -ref t1.nii -
out tof_in_t1_aligned.nii

Final result:
1.003097 0.002155 0.021235 24.655329
-0.000491 1.000615 -0.162569 30.259245
-0.018402 0.168151 0.997762 119.223024
0.000000 0.000000 0.000000 1.000000

shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$
```

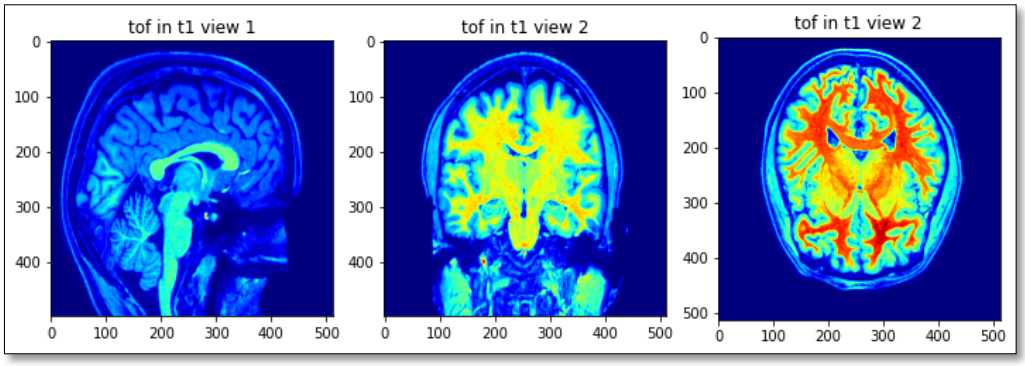


Command 2:

flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -cost mutualinfo -in t1.nii -ref tof.nii -out t2_in_t1_aligned_mi.nii.gz

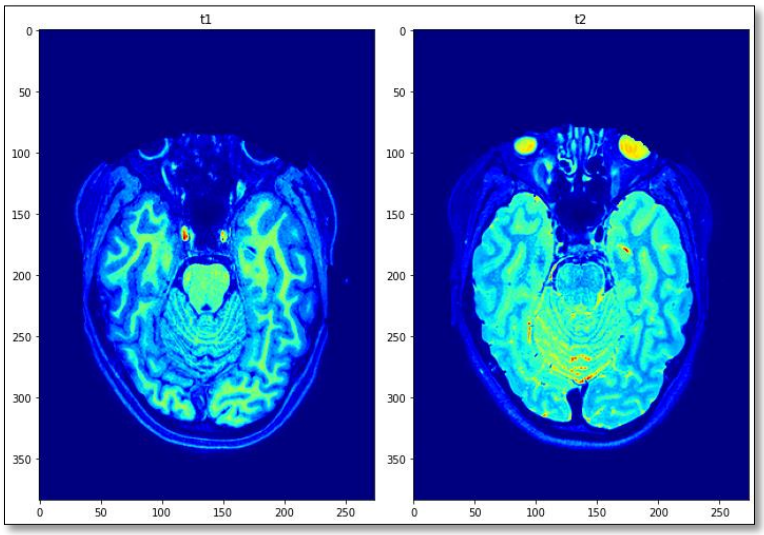
Result:

```
shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$ flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -cost mutualinfo -in t1.n
ii -ref tof.nii -out tof_in_t1_aligned_mi.nii

Final result:
0.989157 0.000548 -0.020939 -21.340638
0.005349 0.974639 0.163742 -49.553381
0.020660 -0.163527 0.978271 -112.374511
0.000000 0.000000 0.000000 1.000000

shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$
```



b) Align t2 image on t1 image

- Plotting images t1 and t2

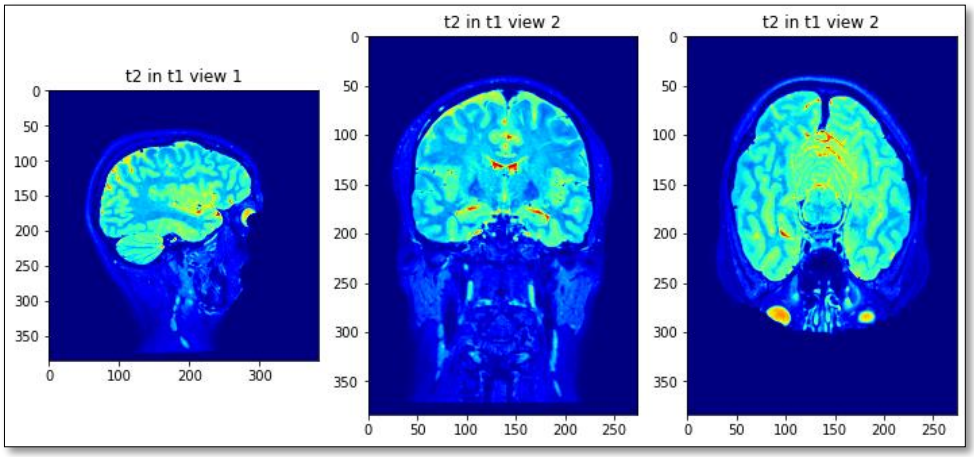


Command1:

flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -in t2.nii -ref t1.nii -out t2_in_t1_aligned.nii

Result:

```
shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$ flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -in t2.nii -ref t1.nii -o
ut t2_in_t1_aligned.nii

Final result:
1.002575 0.002773 -0.007889 0.371326
0.000935 0.986044 0.005607 -0.545707
0.002766 0.021569 0.934856 4.455892
0.000000 0.000000 0.000000 1.000000

shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$
```

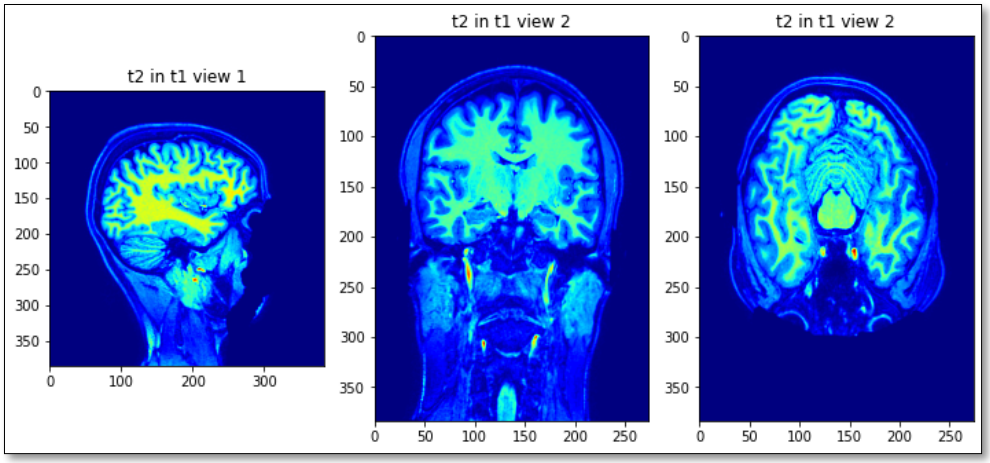


Command2:

flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -cost mutualinfo -in t1.nii -ref t2.nii -out t2_in_t1_aligned_mi.nii

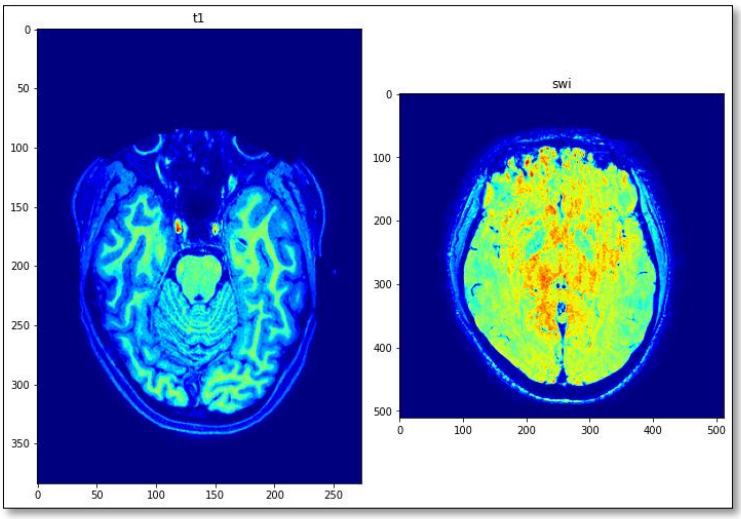
Result:

```
shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$ flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -cost mutualinfo -in t1.n
ii -ref t2.nii -out t2_in_t1_aligned_mi.nii

Final result:
1.001019 -0.003405 0.003635 -0.261618
0.003188 1.001203 -0.011500 1.151275
-0.004590 0.008408 1.000390 -0.926505
0.000000 0.000000 0.000000 1.000000

shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$
```



c) Align swi image on t1 image

- Plotting images t1 and swi

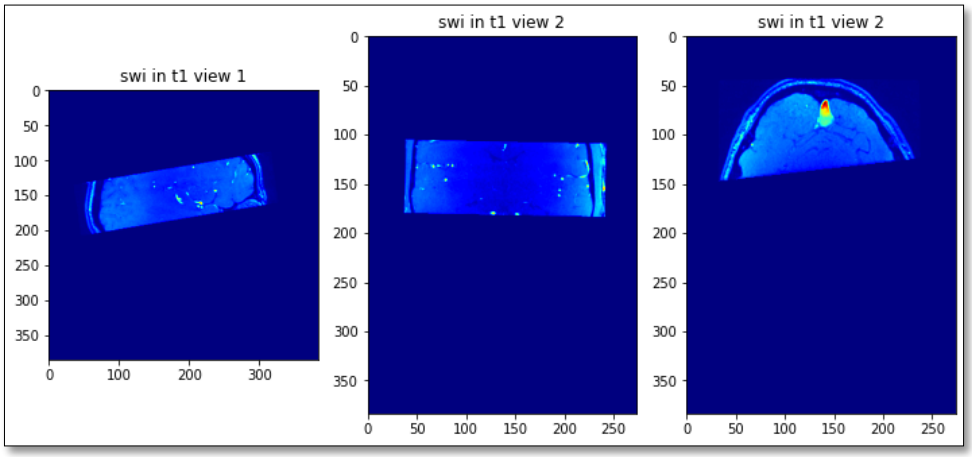


Command1:

flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -in swi.nii -ref t1.nii -out swi_in_t1_aligned.nii

Result:

```
shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$ flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -in swi.nii -ref t1.nii -
out swi_in_t1_aligned.nii

Final result:
1.000715 0.025382 0.025492 -17.587788
-0.026411 1.003367 0.029355 18.168634
-0.025488 -0.024846 0.993789 75.455441
0.000000 0.000000 0.000000 1.000000
```

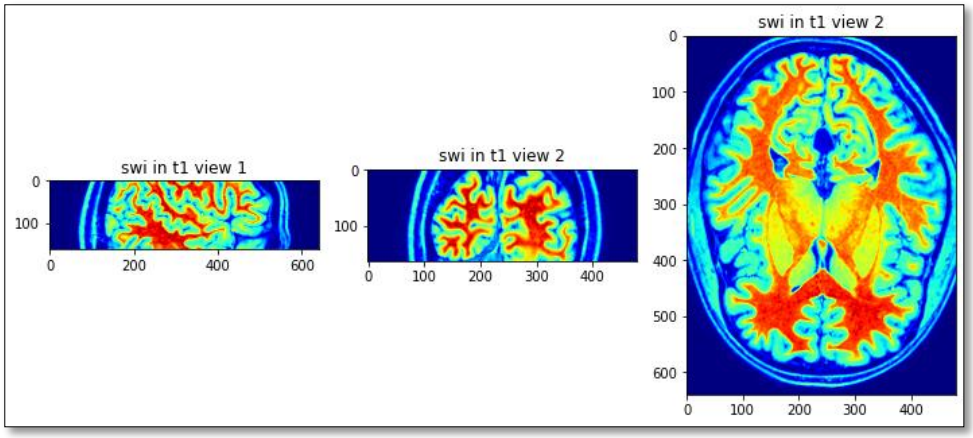


Command2:

flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -cost mutualinfo -in t1.nii -ref swi.nii -out swi_in_t1_aligned_mi.nii

Result:

```
shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$ flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -cost mutualinfo -in t1.n
ii -ref swi.nii -out swi_in_t1_aligned_mi.nii

Final result:
1.001252 -0.021938 -0.026944 19.498028
0.021193 1.001724 -0.036457 -14.677683
0.026310 0.024546 1.001597 -75.630932
0.000000 0.000000 0.000000 1.000000

shweta@LAPTOP-31OF2A8A:/mnt/c/Users/scorp/Downloads/images$
```


swi in t1 view 1
swi in t1 view 2
swi in t1 view 2