University of Girona

Spain

Medical Imaging Analysis
Lab 4 - Image Registration

Submitted by :
Mr. Shubham WAGH
Mr. Nayee Muddin Khan
DOUSAI

1 Introduction

The main objective of this lab work is to get familiar with image registration framework. We have implemented a typical image registration algorithm based on rigid and affine transformation with few set of given examples. A simple MATLAB code is provided and an improvement is to be done for the existing methods as per the objective and goal from the guidelines mentioned in the lab guide for the better scope of the code. Here we also implement multi-resolution registration framework and compare it with single stage registration both qualitatively and quantitatively. In the below sections we discuss about the changes, observations and problems from the registration framework in detail.

2 Image Registration

Image registration framework is basically a procedure to combine two or more images for providing more details about the given images. This is one of the important topic in medical imaging as in medical imaging techniques we have to take many images of the patient in the same pose. But it is very hard to keep the patient in the same spot or location without moving. So we need image registration which is considered as the pre-processing for medical image analysis.

In the image registration we are usually referred by one image called as fixed image. The other input images we provide for the image registration are called as moving images which does a series of geometric transformations to get the best possible result as fixed image. The geometrical frame work to get the best results is explained in the following figure.

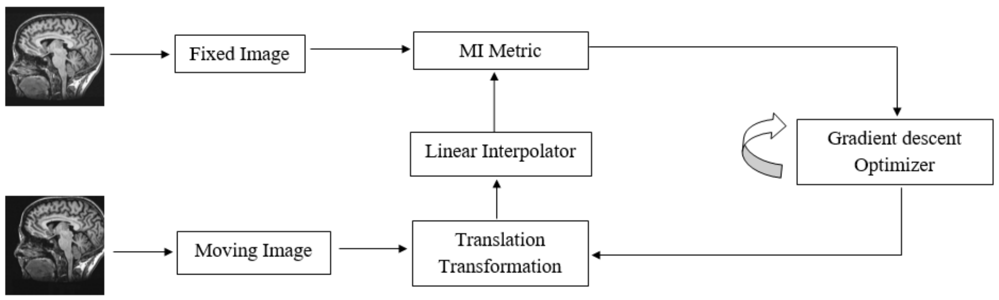

Figure 1: Image Registration Framework

Basically the image registration framework consists of Optimizer function, a geometric transformation and a linear interpolator. From the given sample code written by D.Kroon's from University of Twente we have four different scipt files. The main function is called as *affineReg2D.m*. By using this function we will take input and output for fixed and moving images. There are many other parameters to be given in this function like to call rigid or affine transformation, Sum of squared distances (SSD) and multi-resolution of the images. From the code we will use Gaussian smoothing to remove the noise and the blur of the images. Gaussian smoothing reduces the discrepancy and increases the chances for the better image registration. After this function we will use interpolator called as *image_interpolation.m* which is used to fill the blank spaces in the new image.The comparison of the fixed and the moving image is done in *affine_function.m.* we will use *fminunc* as optimizer here, which will minimise the similarity metric from the transformation of images.

- What is the function of the scale vector?
  The scale vector varies accordingly with repsect to the directions of x and y. As we move one pixel unit in one direction it won't effect much. While in rotation factor as we measures angle, even by the scale of one it takes longer time for computation than translation. Therefore, we need to scale different measures.

- Why is Gaussian smoothing used?
  Smoothing of the images is used in our code because the pixel values we have after interpolation is not same as before real values. So we use smoothing to register two images. This smoothing makes the neighbour values to be similar to make better image registration.

- Where is the center of rotation of the transformation?
  The center of rotation of the transformation is centre of images. From the code *affine_transform_2d_double*, the centre of images is set to $(0, 0)$ and then we calculate the transformed coordinates as the rotation matrix rotated the images at $(0, 0)$.

3 Similarity Metric

Initially we are given sum of squared distances (SSD) as a similarity metric in the code. A new similarity metric called entropy of the difference image is implemented in the code. It firstly calculates the difference of the two images and computes entropy over this difference. Lower entropy values indicates higher similarity between the images. The following equation defines the entropy of the difference image

$$S_{A,B} = \sum_{x=0}^{X} \sum_{y=0}^{Y} A(x,y) - B(x,y)$$

$$H(S_{A,B}) = -\sum_{i=0}^{I} p(i) log(i)$$

On implementing the entropy of the difference image as similarity metric, the registration results were not as good as compared to the results obtained from SSD. Hence for all further steps of this lab, we used SSD as our only similarity metric to check the performance of our registration.

4 Affine Transformation

Implementation of rigid transformation was provided in the code. However, rigid transformation involves scaling, rotation and translation whereas affine transformation involves shear and change in aspect ratio in addition to the above three geometric transformations. Affine transformation is governed by the following sets of equations:

$$f_x(x, y) = a_x x + a_y y + t_x$$
$$f_y(x, y) = b_x x + b_y y + t_y$$

Rigid transformation is less efficient and more prone to error as compared to affine transformation. The input image may be complex and can contain many geometric characteristics. Affine transformation includes all those complexities and outperforms rigid transformation. For better results, the parameters governing the image scale were set to 1 (initially) instead of 0 as the optimizer tries to optimize all the six parameters of the affine transformation. However, the results were not as good owing due to constraints of the optimizer and maximum number of iterations. Then we set to 1 only some parameters of affine transformation governing the image scale. A sample output can be seen below.

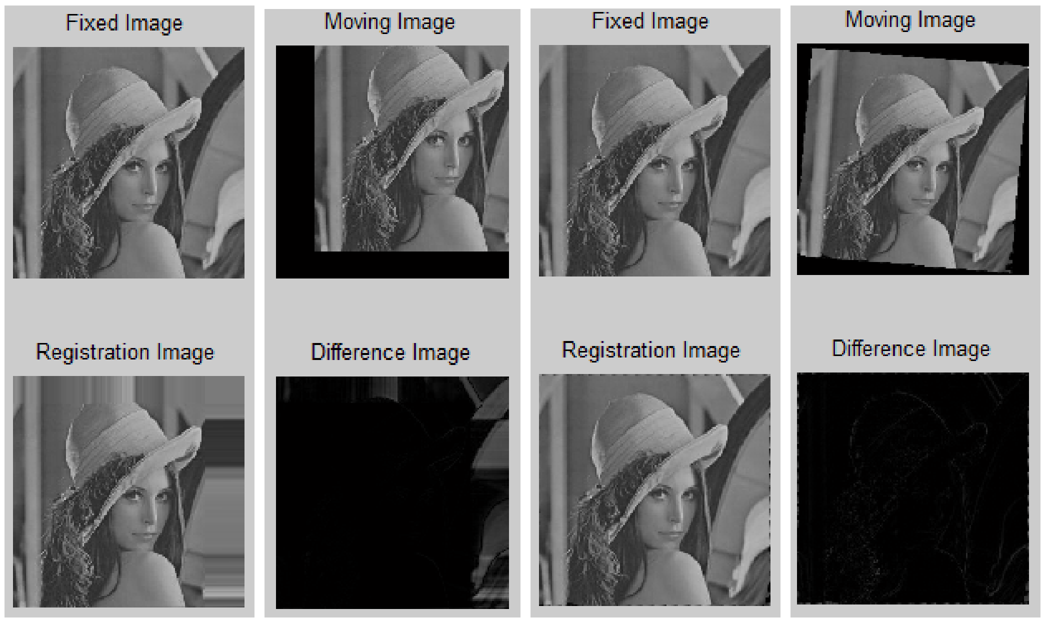

Figure 2: Image registration using SSD and an affine transformation. Fixed (Lena 2) and moving images (Lena 1 and Lena 3) (top row) and registered moving and its difference with fixed (bottom row)

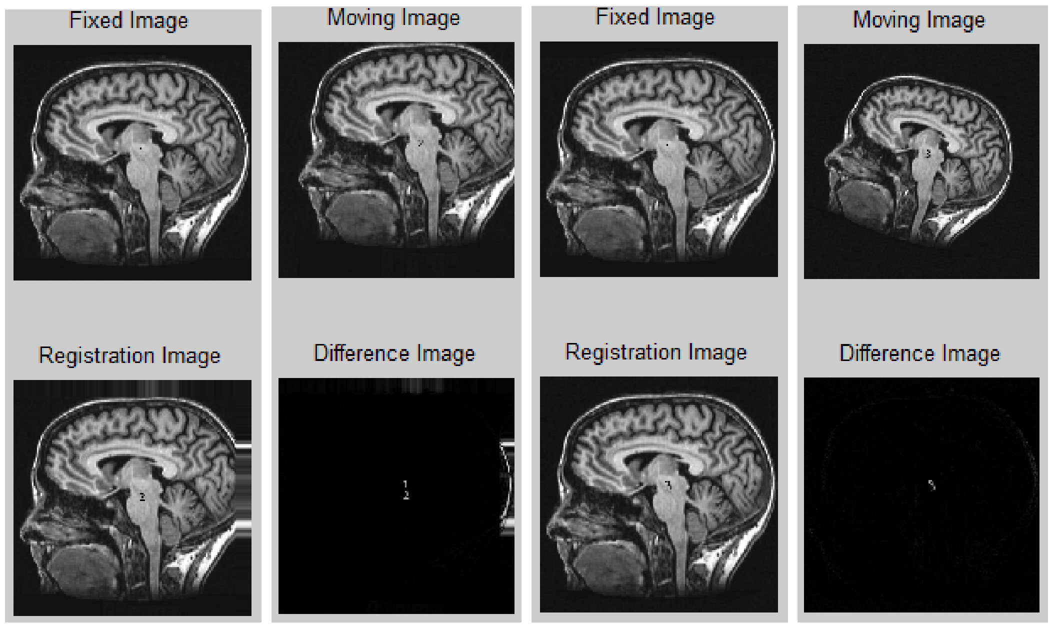

Figure 3: Image registration using SSD and an affine transformation. Fixed (Brain 1) and moving images (Brain 2 and Brain 3) (top row) and registered moving and its difference with fixed (bottom row)

5 Problems faced

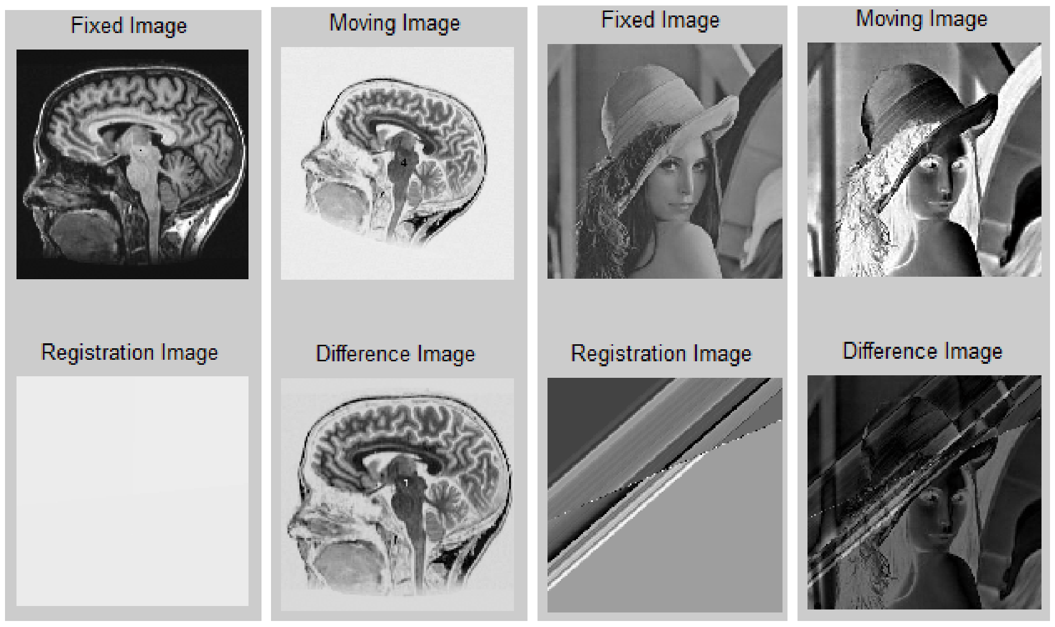

Figure 4: Image registration using SSD and an affine transformation. Fixed (Brain 1 and Lena 2) and moving images (Brain 4 and Lena 4) (top row) and registered moving and its difference with fixed (bottom row)

From the above registration results we observe that, our similarity metric i.e SSD gave proper registration results as expected. However, SSD fails for some images. Fig.4 shows the cases where SSD doesn't give expected registration results. The reason is SSD metric looks for minimized squared difference. With inverted color, the squared differences would be maximized if properly registered i.e with correct registration. This is why the registration result is not as expected. Hence we need another similarity metric better than SSD which would flexibly handle cases of inverted color images.

6 Multi-resolution Registration

The algorithm converges to a completely different extrema, if the initial estimate is far from the initial transformation. This issue can be resolved by using multi-resolution registration framework. The size of the input image is decreased. The registration is implemented on these small images. The output of this registration is then used as an initial estimate for registration on a larger image. Smaller the image, closer will be its estimate to the minima, hence faster is the convergence of the algorithm.

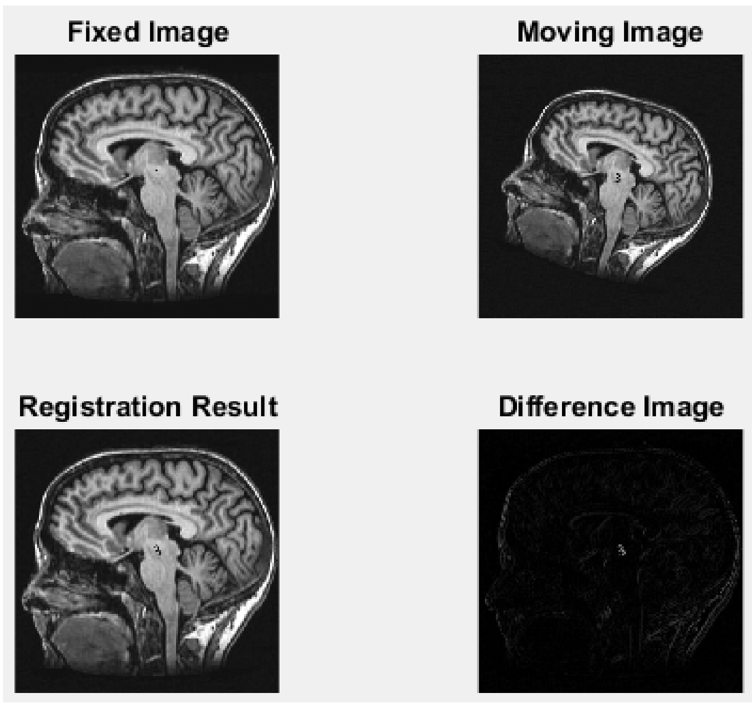

Figure 5: Output using multi-resolution framework. It uses affine transformation, sum of squared difference and 3-stage multi-resolution registration framework

As the image increases in size, the translation also increases with the same amount. So the smaller image output is first scaled appropriately and then fed into the larger image registration framework. Hence, the output of the smaller image registration cannot be directly given as input to the larger image registration. Here we construct a mulit-resolution image pyramid of maximum 3 stages, where the size of the image decreases as we go down the multi-resolution image pyramid. A decrease in the run time of multi-resolution framework can be observed, as the initial registration provides good estimate for further registration.

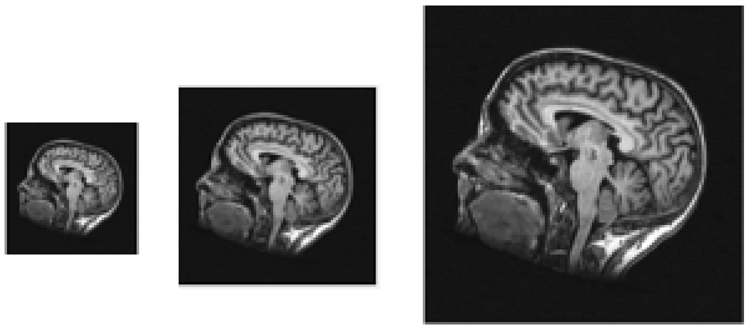

Figure 6: Three stage multi-resolution image pyramid

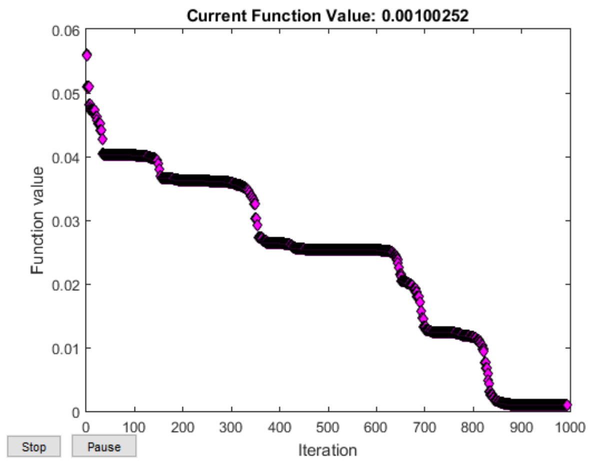

Figure 7: Similarity metric vs number of iterations for single stage registration

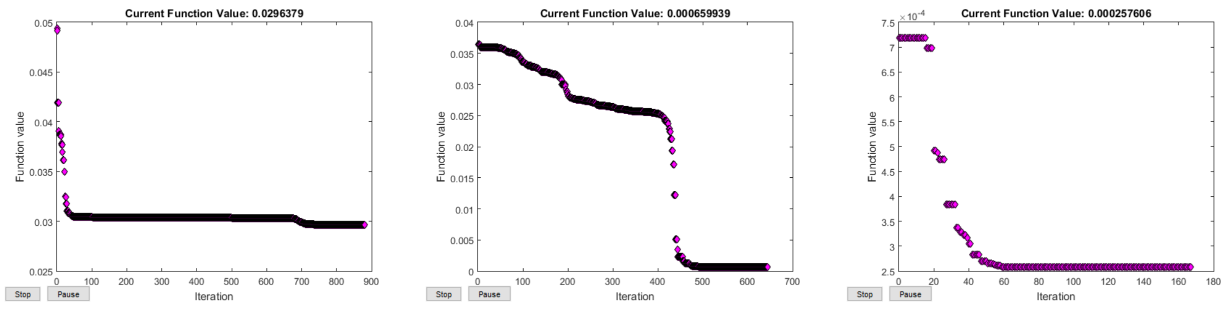

Figure 8: Similarity metric vs number of iterations for three stage registration

Fig.7 and Fig.8 represent the number of iterations used to reach the goal transformation parameters. Although the total number of iterations are more in multi-resolution (combined number of iterations or all stages), the run time is smaller because most of these iterations are on a smaller image, reducing the run time. Likewise, here a significant decrease in run time was observed despite similar output. The time with single stage registration was found to be 40 seconds whereas for a two stage registration was 28.64 seconds and for a three stage resolution, the run time was found to be 18 seconds. the mean squared error for all the different staged registrations was found to be 0.02954. We analyzed the results both qualitatively and quantitatively, and observed that they were same for all image combinations with a slight improvement in some cases despite reduced computation time.

7 Conclusion

In this lab, we implemented an image registration system which can be further used for medical image analysis. We also compare the performance between the single stage registration and multi-resolution registration framework. We also mention the problems in our similarity metric (SSD), fails in case of inverted color images. Moreover, we also tweak the parameters like scaling factor and the initial input for proper registration of images. This lab has solidified our knowledge in image registration through hands on experience.