Medical Imaging
Lab 4 : Registration

Anirudh Puligandla
Meng Di

May 21, 2017

1 Introduction

The main goal of this lab is to study the image registration framework and also to modify the framework to incorporate features, namely, one similarity metric, affine transformation implementation and multi-resolution framework. The whole course-work is divided into 4 sections for each of the above mentioned goals. the following sections in the detail about the framework and the and modifications applied.

2 Registration Framework

The framework is divided into 4 parts that are explained as follows:

- **Transform** : The transform function is first generated for the moving image by using some initial parameters for the transformation matrix by using rigid or affine transformation techniques. The provided code implements rigid body transformation, while, it was asked to implement affine transformation, that will be explained in later sections. The parameters for the matrix are initialized with $0s$ and an optimizer is applied to find the right parameters. The transformation is applied to the moving image in the beginning of the $affine\_transform\_2d\_double(Iin, M, mode, ImageSize)$ function, while, the transformation matrix is computed in the $AffineReg2D()$ function, in lines 36-50 .
- **Interpolation** : The next step after obtaining the transformation matrix is to apply to the moving image. under the $affine\_transform\_2d\_double()$ function, the transformation is first aplpied to the moving image and the obtained pixel values

are interpolated to fit the image dimensions. The provided code implements various interpolation methods such as, *bilinear*, *bicubic* and *nearest*, that can be selected accordingly.

- **SimilarityMeasure** : After obtaining the transformed and and interpolated image, it is then compared with the fixed image, by using a similarity measure. The provided implements $squareddifference$ metric to obtain the squared difference between each of the pixels of the 2 images, while, it was asked to implement $entropyofdifferenceimage$ metric. The code for this part can be found at the end of the $affine_function$ function, inside the $switch-case$ block. This function implicitly calls $image_interpolation()$ function to apply the interpolation.

- **Optimizer** : The last part of the framework is to optimize the transformation applied on the moving image. The above three steps are repeated and compared with the optimizer until the similarity metric follows a threshold criterion. The provided code is using the min search algorithm to optimize the parameters and it can be found in various places in the code such as, line no.39 in $affineReg2D$ function, for example.

The scale parameter is used to bring the translation, resize and rotation parameters to real values. Gaussian smoothing is used to achieve faster registration. Blurring spreads the intensity values that helps to better capture the difference between the two images. The center of rotation of the transformation is the orthogonal axis to the image, that is passing through the center of the image.

3 Similarity Metric

The $entropy\ of\ the\ difference\ image$ metric was implemented but it does not give positive results, as already discussed in the lab. it can be observed from the following image, that the process terminates in only one iterationwhile using the entropy metric.

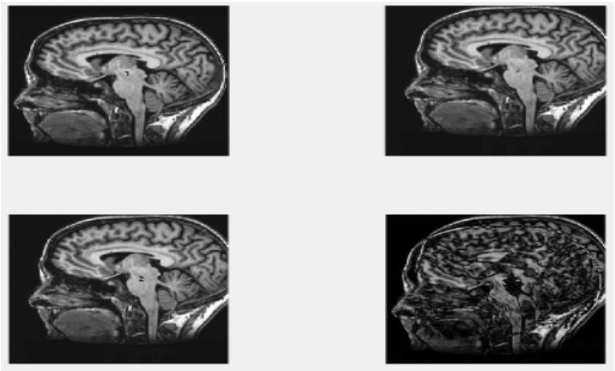

Figure 1: Showing clockwise from left-top: fixed image, moving image, difference image, and, transformed image.

4 Transformation

Under this section, affine transformation was to be implemented and applied on the given images. Unlike rigid transformation, this technique requires 6 parameters. The affine transformation matrix is defined as shown below.

$$A = \begin{bmatrix} x(1) & x(2) & x(3) \\ x(4) & x(5) & x(6) \\ 0 & 0 & 1 \end{bmatrix} ;\ where,\ x = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

We initialize the six parameters as shown above. There are 2 ones in the vector so that the optimizer can focus on optimizing these 2 values. Since, we have a limit on the number of iterations for the optimizer, all the parameters cannot be optimized correctly. The following figure shows the results on the *lena*3 image using rigid and affine transformations.

Figure 2: Left: rigid transformation, right: affine transformation. (note: the 4 subimages use the same convention as Figure 1)

5 Multi-Resolution

We were supposed to implement multi-resolution for the given parameters. Under this section, for n number of resolutions, the parameters are multiplied by 2, for each iteration. For the case of rigid transformation, the 1^{st} and 2^{nd} parameters are multiplied iteratively and the framework is executed until an optimal solution is found. While, the 3^{rd} and 6^{th} parameters are checked in multi-resolution. The algorithm was tested with 4 resolution levels, and the results were observed to be improved. The process is repeated for 4 different scaled pyramids of images.

It can be noted that using multi resolution can provide better results, as well as, improve computational cost. Multi-resolution also makes the optimizer find the global minimum, thus, avoid getting stuck at local minima. It can be observed here that the computational time greatly reduced by using greatly scaled images.

The following figure shows the results for *lena*3 image for rigid and affine transformations. Figure 4, below, shows the 4 pyramids with different resolutions for the same image.

Figure 3: Left: rigid transformation, right: affine transformation. (note: the 4 subimages use the same convention as Figure 1)

Figure 4: pyramids with 4 scales at multiples of 2

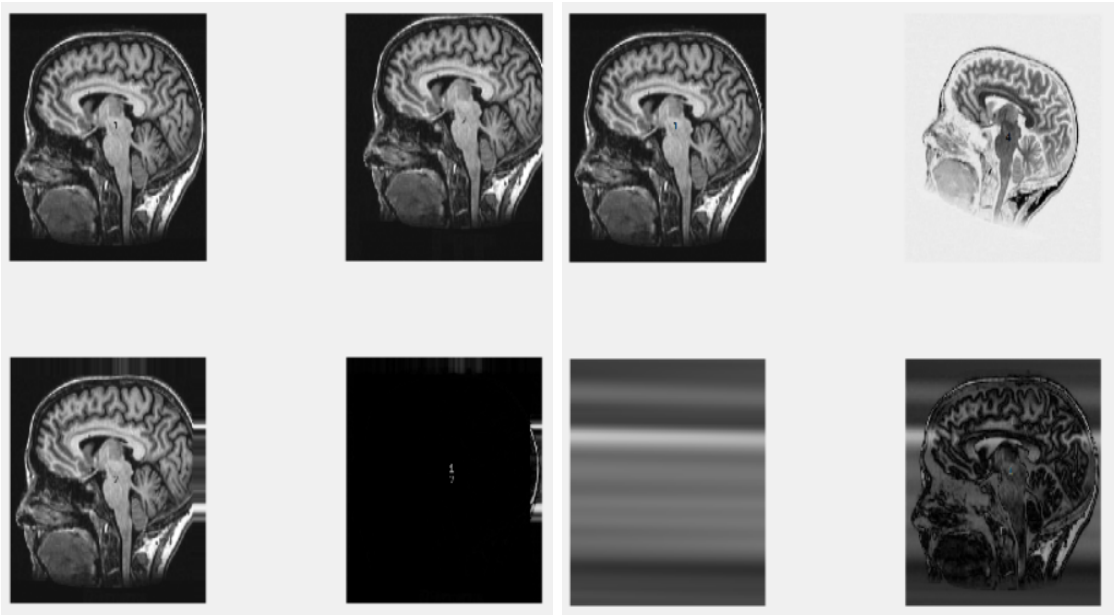

Figure 5: Left: rigid transformation, right: affine transformation.

6 Conclusion

The registration framework was studied and entropy of difference image metric, affine transformation and multi-resolution were implemented. Also, the advantage of multi-resolution framework over single resolution, was discussed.