Lab4 Report - Image Registration

Mohammad Rami Koujan
M.Sc. VIBOT
University of Girona

1 Introduction

Image registration is the process of overlaying two or more images of the same scene taken at different times, from different viewpoints, and/or by different sensors. It geometrically aligns two images, the reference (fixed) and sensed (moving) images. The present differences between images are introduced due to different imaging conditions, e.g. patient movement. Image registration is, in fact, widely used in many fields. For instance, remote sensing, medical imaging, computer vision, etc. Therefore, the objective in this lab assignment is to get familiar with the concept of image registration by starting from a given registration framewrok, understanding the different steps implemented inside it, and modifying some of the used parameters.

2 Registration Framework

A typical registration framework can always be decomposed into four major parts:

- Transformation part: The purpose of this part is to geometrically transform the moving image depending on some parameters that are passed to this part. In other words, it answers the following question: what type of transformations are allowed and what constraints should the model satisfy?
- Metric part: It is mainly used to measure the similarity between the fixed and moving images. That is to say, it deals with how to find the transformation which maximises the similarity subject to the desired constraints.
- interpolator: Determines the pixel intensity of the registered image based on the position and neighbour pixels.
- Optimizer: Finds the best metric value with respect to the transformation parameters. Thus, the question that imposes itself in this part is: how similar are the images/features after registration, in other words how good is the registration. The following figure shows a block diagram for such a registration framework.

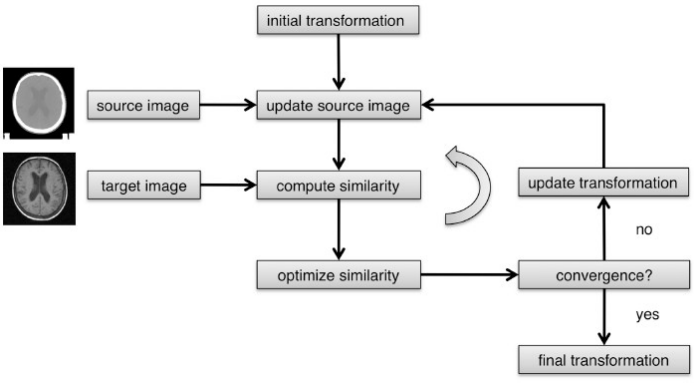

Figure 1: Image registration framework

3 Implementation Aspects of the Registration framework

This section explains in details the provided registration framework alongside the achieved modifications requested in different parts of this assignment. The provided framework contains files that implement the blocks explained in the previous section. The main function is called "$affine\_function$". It takes the required geometric transformation parameters, scaling vector for these parameters, fixed and moving images, and the type of the metric (Sum of Squared Differences or Mutual Information) and transformation (Rigid or Affine) as inputs to this function and gives the metric value as an output. Inside this function, there are three main blocks of code according to the task carried out by those groups of lines:

- First the transformation parameters vector is multiplied by the scaling vector, the purpose of this step is explained later. The type of the transformation passed to this function by the "ttype" parameter is checked then to calculate the transformation matrix based on the value of this parameter ('a' for Affine & 'r' for Rigid). The followings are the equations used to compute this matrix for both types.

$$M\_Rigid = \begin{bmatrix} cos(\theta) & sin(\theta) & translateX \\ -sin(\theta) & cos(\theta) & translatey \\ 0 & 0 & 1 \end{bmatrix}$$

$$scaling = \begin{bmatrix} resizeX & 0 & 0 \\ 0 & resizeY & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Rotation = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$shearing = \begin{bmatrix} 1 & shearXY & 0 \\ shearYX & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Translation = \begin{bmatrix} 1 & 0 & translateX \\ 0 & 1 & translateY \\ 0 & 0 & 1 \end{bmatrix}$$

  $M\_Affine = Translation * shearing * scaling * Rotation$

- Secondly, another function, "$affine\_transform\_2d\_double$", is called and given the moving image, the transformation matrix, and the mode of interpolation as input parameters. This function has two main tasks.

  1. Computing the geometric transformation starting from generating the grid of pixel locations of the output image, making the center of rotation of the transformation as the image center (this is done by subtracting the center point coordinates from each point in the grid), multiplying the matrix M by the generated grid and then shifting the image back to it original center which is the top left corner (lines 52,53 of the original script file). It is noteworthy that M here is considered as the transformation matrix that transforms the registered image back to to the moving one (the inverse transformation matrix) which is done to avoid the problem of empty pixels in the registered image when doing the transformation from the moving image to the registered one. Therefore, for the purpose of applying some specific geometric transformations to the moving image and visualizing the results, the inverse of M should be computed in the "$affine\_function$" by adding an inversion step. However, for the sake of images registration, an optimizer is used to repeatedly tune the transformation parameters that minimize the defined metric, which makes applying M or inverse of M the same since from the point of view of the optimizer it is a matrix with 7 parameters to be tuned to get the minimum of the metric.

2. Calling another function "*image_interpolation*" and passing the moving image, the transformed grid, type of interpolation, and image size to the input of this function which generates the registered image based on the transformed grid of pixels and the type of interpolation determined by the input parameters (nearest, bilinear, or bicubic with replicated or zero boundaries).

- In the third principal part of the "*affine_function*", the value of the 'mtype' parameter is checked to determine the type of the desired metric to be calculated ('sd' for Sum of Squared Differences and 'm' for mutual information). Then, either SSD or the mutual information metric is computed. SSD metric is, in fact, worked out by the normalized summation of the squared differences between the resultant (registered) image returned by "*affine_transform_2d_double*" function and the fixed (reference)image while mutual information metric is computed by finding the individual and joint normalized histograms of both images and then using them in the following formula to get the final value of this metric. Therefore, this part is responsible for the "similarity measure" step in the

$$C(A,B) = \sum_{i=0}^{I} \sum_{j=0}^{J} p_{AB}(i,j) \log \frac{p_{AB}(i,j)}{p_A(i) p_B(j)}$$

  shown framework.

Furthermore, as the optimizer, which is explained later, tries to find the transformation parameters that minimize the similarity measure metric and for the case of the mutual information the maximum value the better, the mutual information is multiplied by -1.

Another function ,"affineReg2D", is used as a run-file to do the following steps:

1. Reading the fixed and moving images from the directory inside which those images are stored.
2. Filtering those read images with a Gaussian filter in order to smooth the images intensity levels which helps to minimize the differences between them and consequently reaching the minimum value by the optimizer in a faster way.
3. Determining the type of the similarity measure metric and the transformation among the ones supported by the "affine˙function" function.
4. defining the scale vector which should be of the same size of the transformation parameters vector that depends on the type of transformation itself.
5. Dividing the transformation parameters vector by the scaling vector. The aim of this step is to normalize those parameters by some factor which will help the optimizer to scan wider range of changes in the similarity measure metric by only small changes in those parameters, depending on the value of scale vector, and consequently obtaining the minimum value of this metric in less time by doing less number of evaluations in each iteration. However, this advantage is at the cost of the accuracy because, as mentioned before, small changes in those parameters give larger changes in the similarity measure metric and as a result getting more accurate numbers close to the minimum one becomes more difficult. Indeed, the scale vector provides a tool for doing a trade-off between accuracy of the obtained similarity metric value and computational cost. It is remarkable that dividing the transformation parameters by the scaling vector will not affect the transformation results since this effect is countered by multiplying by the same scale vector as a first step in the "*affine_function*" function, as explained above.
6. Calling the optimizer and providing it with the "*affine_function*" as the function to be minimized, the transformation parameters vector as the argument of this function, and some other factors that determine the number of iterations to be taken by the optimizer, the conditions of ending the optimization procedure, and so on. This is done in line 28 of the original "affineReg2D" function.

7. Finally since the result of the optimization step is the transformation parameters, those parameters are used to register the moving image, by calling "$affine\_transform\_2d\_double$" function, and then displaying the results alongside the image of difference between the fixed and moving images.

For multi-resolution case, the "affineReg2D" function has been modified as follows:

- First a new parameter has been defined to determine the required resolution (number of levels).
- A loop is used to carry out the registration for different resolutions. This is done by scaling the transformations parameters vector, as before, by the scaling vector and multiplying the translation parameters by two since for each resolution the fixed and moving images are resized by a factor of $\frac{1}{2^{num-i}}$, where "i" is the counter of the loop and "num" is the number of desired levels defined in step one. Thus, the loop starts from the lowest resolution and then goes up by a factor of two each time which explains the need to only multiply the translation by 2 when moving from one resolution to the other while passing the rest parameters without any changes.

Actually, the benefits of using multi-resolution registration is that a good initial estimation of the transformations parameters can be obtained more quickly (less computations) when starting from a lower resolution and then this estimation will be passed from one level to the other taking into account the change in translation depending on the image resizing factor between each resolution level and the next one.

4 Registration Results of the Single Resolution Case

This section presents the results of testing the previously explained registration framework on some images. The tests were carried out on two pairs of fixed-moving images. The following figure shows the results of registering image 'lenag3.png' to 'lenag2.png' for two types of transformations and metrics. The following notation is used:

- sd-r: means Sum of squared differences is the similarity measure metric and the type of transformation is rigid
- m-r: means mutual information is the similarity measure metric and the type of transformation is rigid
- sd-a: means Sum of squared differences is the similarity measure metric and the type of transformation is affine
- m-a: means mutual information is the similarity measure metric and the type of transformation is affine

Figure 2: Registration results for 'lenag3.png' to 'lenag2.png'. The first row shows two groups for sd-r & and m-r respectively while the second row for sd-a & m-a respectively

Table 1 presents the time response, the maximum number of iterations, and the best metric achieved for each case of the previously shown groups of images.

Measured parameters	Elapsed time (sec)	Maximum number of iterations	Best metric achieved
sd-r	9.088038	147	0.0340039
m-r	6.403297	54	0.996589
sd-a	9.532844	146	0.00456383
m-a	18.894655	221	1.97489

Table 1: Results of registering 'lenag3.png' to 'lenag2.png'

In addition, a scale vector and initial parameters with the following values scale=[50 50 50], x=[0 0 0] were used for the rigid transformation cases and another ones with the following values scale=[1 1 50 50 50 1 1], x=[0 0 pi/16 1 1 0 0] were used for the affine transformation cases. Figures 3 and 4 show metric values plotted as a function of the iteration number for the previously shown groups of images.

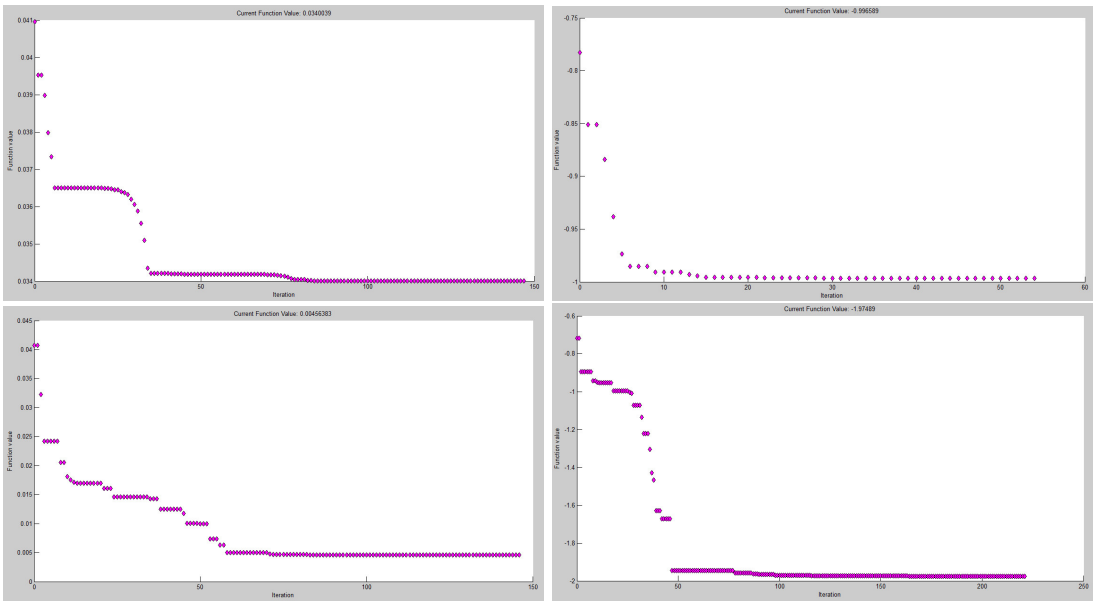

Figure 3: Registration results for 'lenag3.png' to 'lenag2.png'. The first row shows metric value against iteration number for sd-r & and m-r respectively while the second row for sd-a & m-a respectively

The same tests were applied to register image 'lenag1.png' to 'lenag4.png' and the following results were noted.

Figure 4: Registration results for 'lenag1.png' to 'lenag4.png'. The first row shows two groups for sd-r & and m-r respectively while the second row for sd-a & m-a respectively

Table 1 presents the time response, the maximum number of iterations, and the best metric achieved for each case of the previously shown groups of images.

Measured parameters	Elapsed time (sec)	Maximum number of iterations	Best metric achieved
sd-r	11.103161	121	0.155822
m-r	8.192329	65	2.01254
sd-a	14.970913	189	0.103113
m-a	14.299014	97	1.7501

Table 2: Results of registering 'lenag1.png' to 'lenag4.png'

In addition, a scale vector and initial parameters with the following values scale=[50 50 50], x=[-4 4 0] were used for the rigid transformation cases and another ones with the following values scale=[1 1 50 50 50 1 1], x=[0 0 0 0.9 0.9 0 0] were used for the affine transformation cases. Figures 3 and 4 show metric values plotted as a function of the iteration number for the previously shown groups of images.

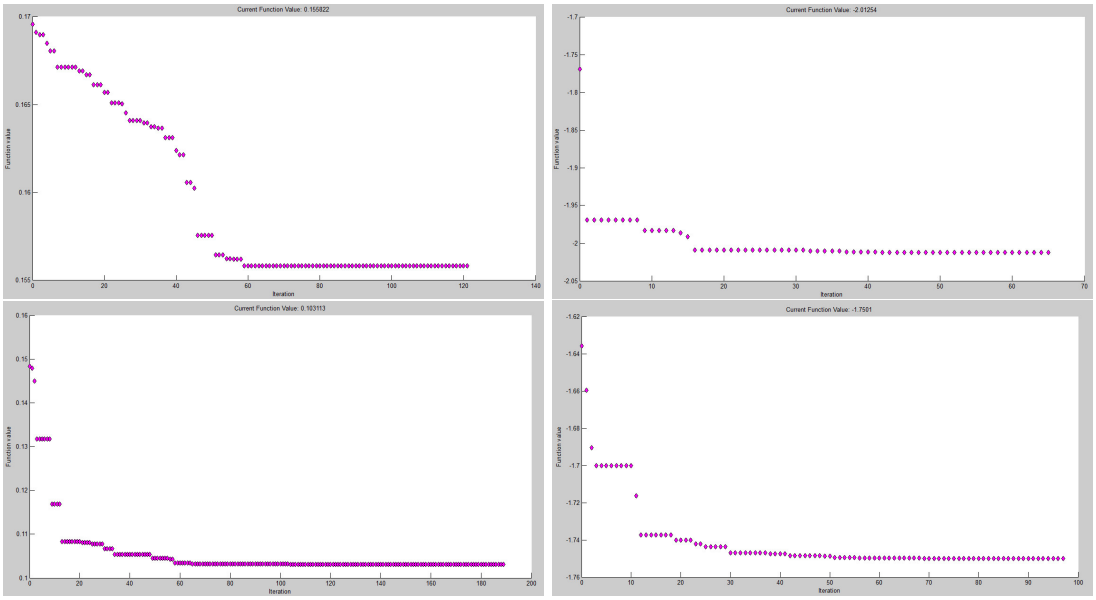

Figure 5: Registration results for 'lenag1.png' to 'lenag4.png'. The first row shows metric value against iteration number for sd-r & and m-r respectively while the second row for sd-a & m-a respectively

Based on the preceding results of registration of the two groups of images, the following conclusions can be drawn:

- As stated previously, tuning the scale vector is a trade-off between the time response and accuracy of the chosen metric. Therefore, the scale vector values for the presented tests were selected to achieve a quite good balance between both of them .
- The initialization of the transformation parameters vector is a critical task on which the final result is quite dependent since the optimizer starts from this set of values and tries to obtain a local minimum value.
- For the same metric, choosing the affine transformation to register the images took longer time than for the rigid transformation case because the degrees of freedom (number of tuned parameters) is larger (7 vs 3) and needs more iterations.
- The results of registering image 'lenag3.png' to 'lenag2.png' are better than registering image 'lenag1.png' to 'lenag4.png' which is expected as in the second case image 'lenag4.png' is modified in terms of image contrast. This casues the results of registration to this image worse because the used metrics are dependent on the intensity levels of the fixed and registered images.

- A good quantitative error measure that can be proposed in this case is "Target Registration Error" which is based on Fiducial markers, which are not provided with the tested images. However, the mean squared error could provide an acceptable and easy to compute measure.

5 Registration Results of the Multi-Resolution Case

This section presents the results of testing some selected representative cases for multi-resolution case. The following table demonstrates the time required and metric value for registering image 'lenag3.png' to 'lenag2.png' for different resolution levels.

Measured parameters	Elapsed time(sec), resolution=2,3,4			Best metric achieved, resolution=2,3,4		
sd-r	8.656205	7.800818	7.407416	0.0348526	0.0348526	0.0348526
m-r	6.374741	5.961622	5.8654	0.984935	0.984935	0.984935
sd-a	9.3265	8.85897	8.7456	0.00456081	0.00456081	0.00456081
m-a	17.279887	17.0589	16.9855	2.03446	2.03446	2.03446

Table 3: Results of registering 'lenag3.png' to 'lenag2.png'

As the table shows, increasing the number of resolutions has the effect of achieving lower time response with the same final result of the metric which is the main advantage of multi-resolution registration. This is, in fact, an expected result, as explained before, since in multi-resolution registration the lower resolution provide a faster estimation of the transformation parameters which are used as an input to the next higher level after adapting the translation.

6 Conclusion

In this lab assignment, a registration framework is analysed, modified and tested against a set of moving and fixed images. The presented results have demonstrated that there are a number of crucial things that should be taken into account when using a registration framework. e.g. the initial guess of the transformation parameters, the used similarity metric measure, optimization algorithm, scaling vector, and finally the error measure in order to compare the different registration results.