UNIVERSIDAD REY JUAN CARLOS

Escuela de Ingeniería de Fuenlabrada

Biomedical Engineering

Course 2023-2024

Lab III: Medical Image Registration

MEDICAL IMAGE ANALYSIS

Group 11:

Enrique Almazán Sánchez

Guillermo Ots

Javier Alfonso Villoldo Fernández

OUTLINE

1. Objectives

The goal of this Laboratory is to become familiar with registration pipelines that have been covered in class and to gain a deeper understanding of the fundamental underlying procedures and their applicability to medical image tasks. In particular, geometric transformations, interpolation and rigid registration methods will be implemented with the help of 3D slicer.

Additionally, we will work on the implementation of our own transformation, interpolation, and mutual information functions in python. It is believed that this is a good approach as not only the processes used will be learnt but also knowledge will be acquired from the research activity required for the application of such processes.

These algorithms will be applied to a real CT slice evaluating the results and assessing how well the algorithm achieves its intended goal, which basically is searching for the geometric transformation that puts two images into spatial concordance.

2. Contributions

- Enrique Almazán Sánchez: Writing of the report regarding both parts (3D slicer and algorithm implementation through python) and revision of the code.
- Guillermo Ots Rodríguez: Python code for the required functions.
- Javier Alfonso Villoldo Fernández: Python code for the required functions and writing of the report regarding that part.

3. Use and analysis of image registration algorithms with 3D Slicer.

3.1. Intramodal Registration

First, the transparency of the foreground image, mr1, is changed in order to see how well aligned both images are.

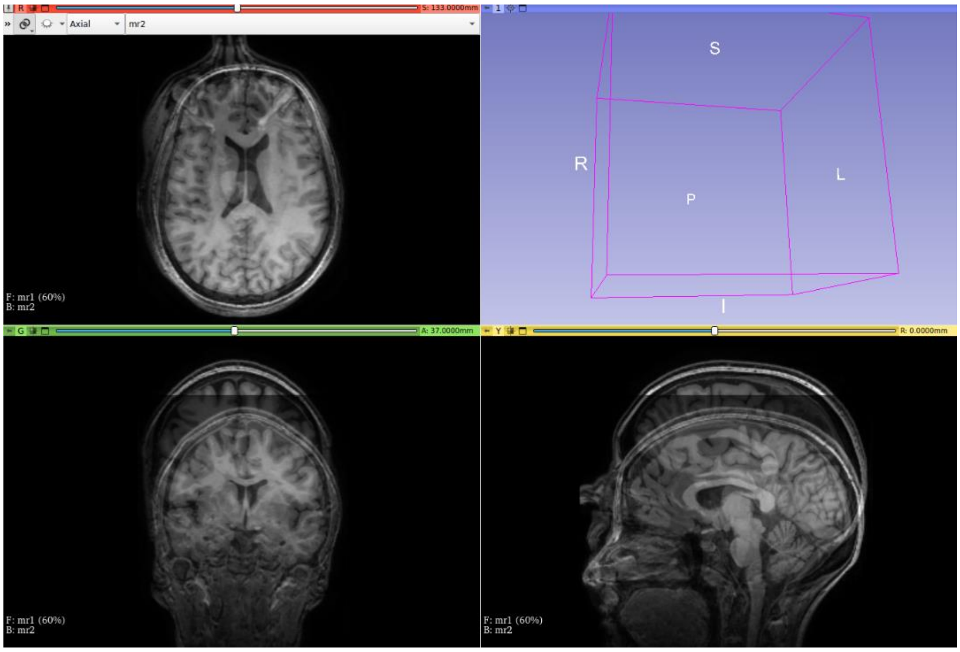

Figure 1: Superposition of the mr1 with changed transparency over the mr2.

In the figure above it can be seen that mr1 and mr2 are not well aligned, having the latter shifted upwards and slightly to the left, with respect to the former. Using the Checkerboard Filter, the following is shown.

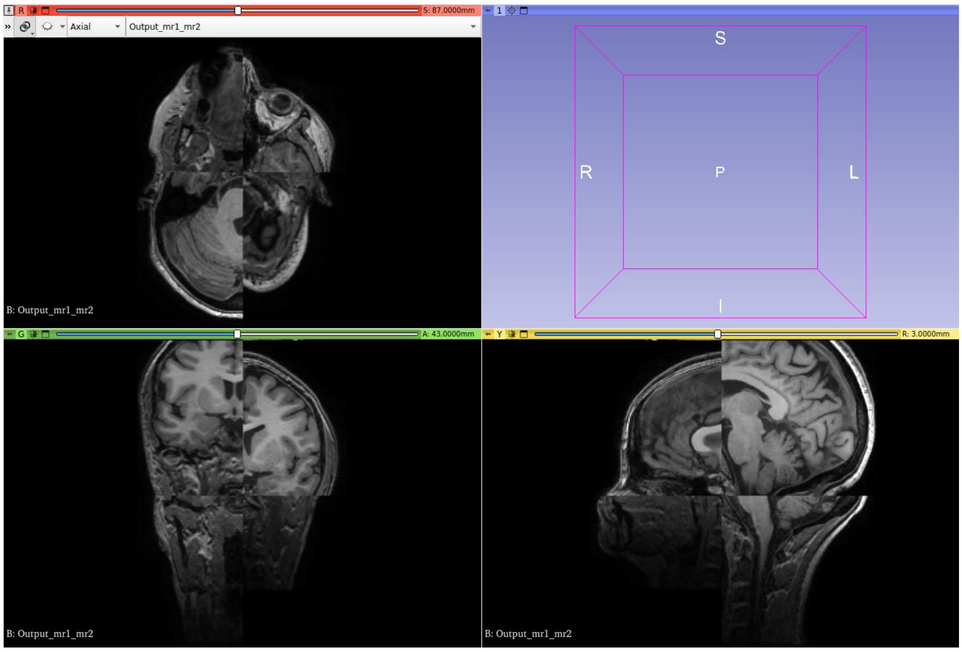

Figure 2: Output of the Checkerboard Filter of mr1 and mr2.

The checkerboard filter helped to compare and visualize both images (or volumes) side by side. It overlayed a checkerboard pattern on both, mr1 and mr2, allowing to easily identify the differences between them.

Thus, as they are not aligned a rigid and scaling registration algorithm with 7 Degrees of Freedom (DOF) is applied.

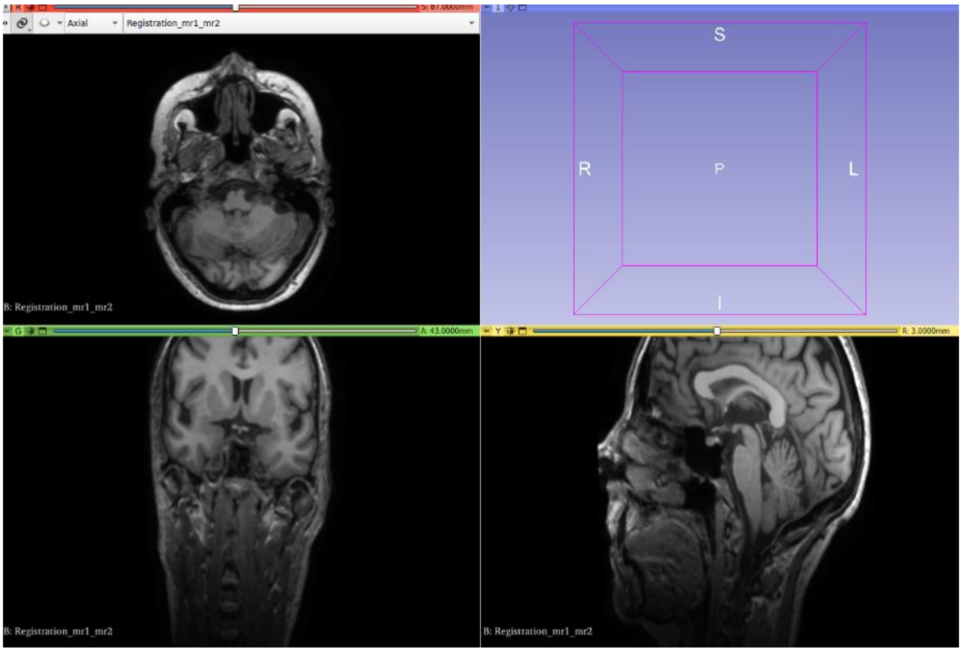

Figure 3: Applying rigid and scaling registration algorithm with 7 DOF.

As it can be seen in *Figure 3* above, applying this registration algorithm between the two MRI images, which refers to a spatial transformation that includes rotation, translation and scaling, results in a considerably stretched image.

The stretching caused in the registered image might partially be because of anatomical differences between the two images. When performing rigid registration, the algorithm attempts to align corresponding features in the images based on similarity metrics, such as mutual information. If the anatomical differences are significantly high, the algorithm may struggle to find a correct alignment.

Another cause, and probably the main one of the stretching, is the extra DOF that introduces scaling along a specified axis. Traditional rigid registration only includes rotations and translations, and not deformations, but the stretching that can be appreciated could be attributed to the seventh DOF.

Consequently, the rigid and scaling transformation may not be sufficient to align the images, in these cases a more advanced model like Affine, seen in *Figure 4,* shows more appropriate results. Affine transformations provide a better model because it can correct for differences not only in orientation but also in scale, and shape between two images thanks to the additional DOF (shearing along the xy, xz, yz planes) which enable the correction of more complex spatial variations that may arise due to anatomical differences or distortions.

However, if the DOF are incremented too much, such as in B-Spline registration algorithm (>27 DOF), the results worsen, having a distorted image as output, shown in *Figure 5*. This can be attributed to the fact that elastic registration, like B-Spline, which cause deformations to provide the best alignment, are not necessary in this case since our images only need to be shifted. Therefore, if applying an elastic registration, the algorithm will try to equalize the shape of both images, mr1 and mr2, leading to the deformation shown in *Figure 5*.

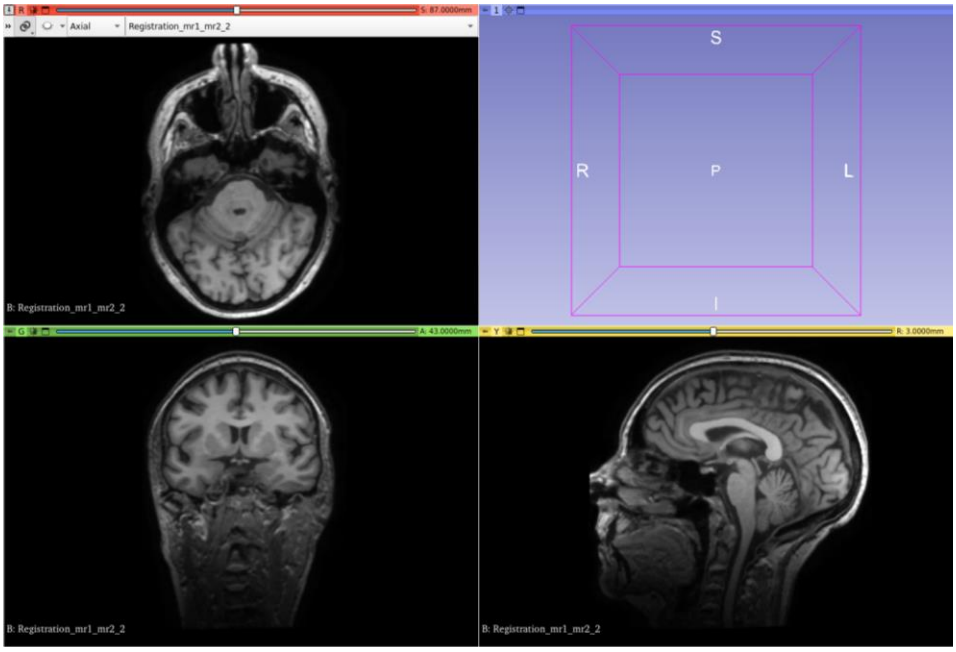

Figure 4: Applying affine registration algorithm with 12 DOF.

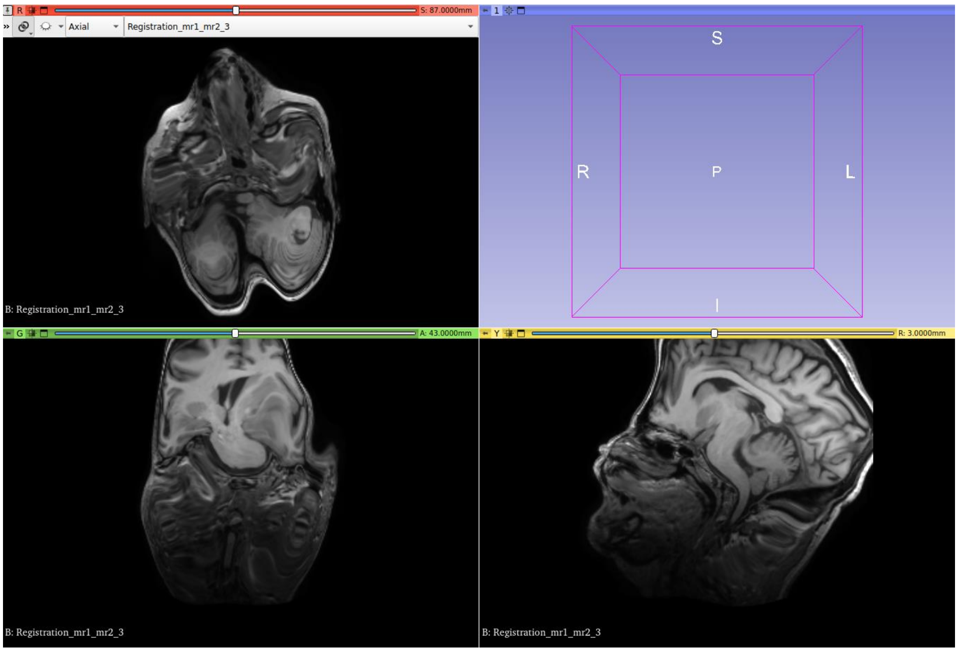

Figure 5: Applying B-Spline registration algorithm with more than 27 DOF.

3.2. Multimodal Registration

Now, a CT will be the new moving image, while the mr1 remains as the fixed image. The same procedure as in the previous section is followed, showing the alignment between both images by reducing the transparency of the foreground (mr1) in *Figure 6*. Likewise, the checkerboard filter is used to underline the differences between both images, shown in *Figure 7*.

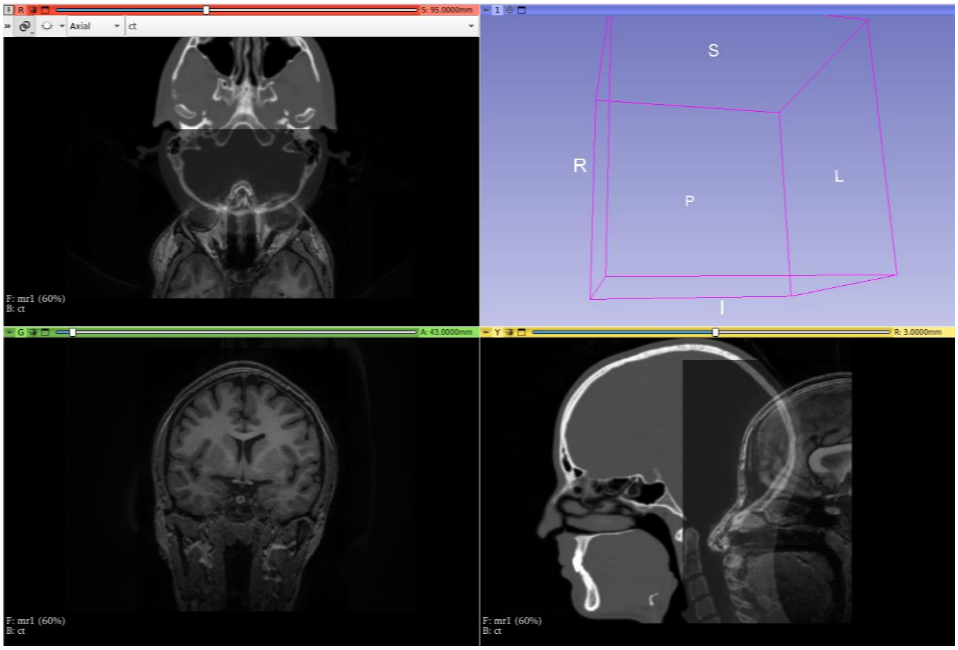

Figure 6: Superposition of the mr1 with changed transparency over the ct.

It is seen, that both images are not centered, having a displacement to the left of the ct (moving image) with respect to the mr1 (fixed image). Difference in their positions also stand out in Figure 7.

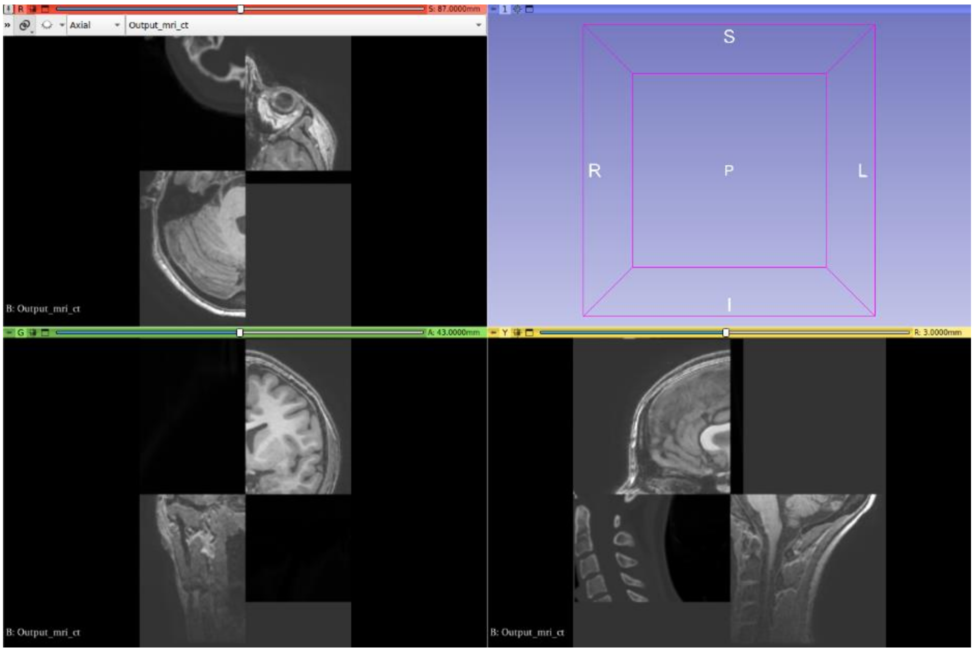

Figure 7: Output of the Checkerboard Filter of mr1 and ct.

Therefore, given the results presented above, they will need to be registered in order to align the ct with respect the mr1. First, rigid registration (6 DOF) is applied, leading to unfavourable outcomes shown in the figure below. This results, are obtained because both images to be registered are obtained from different data bases, meaning that the images can have different reference system. Hence, the rigid registration will place both images, mr1 and ct, in different places.

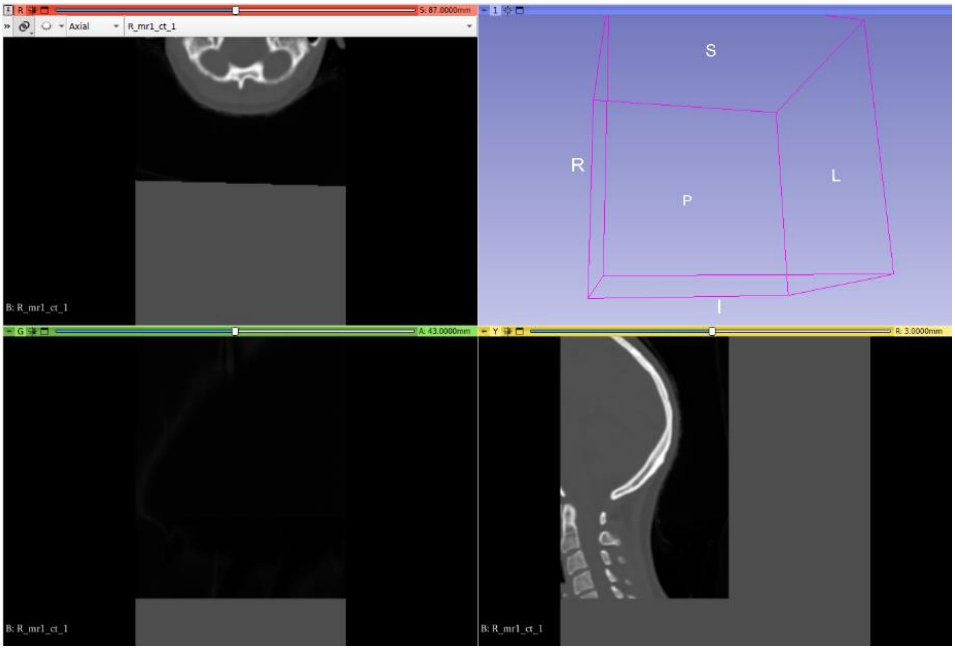

Figure 8: Output of rigid registration on ct with respect mr1.

To overcome these mediocre results, a first step needs to be taken before applying the registration algorithm. The aligning of the centroids of both heads by means of a coarse segmentation, having as a result the images shown below.

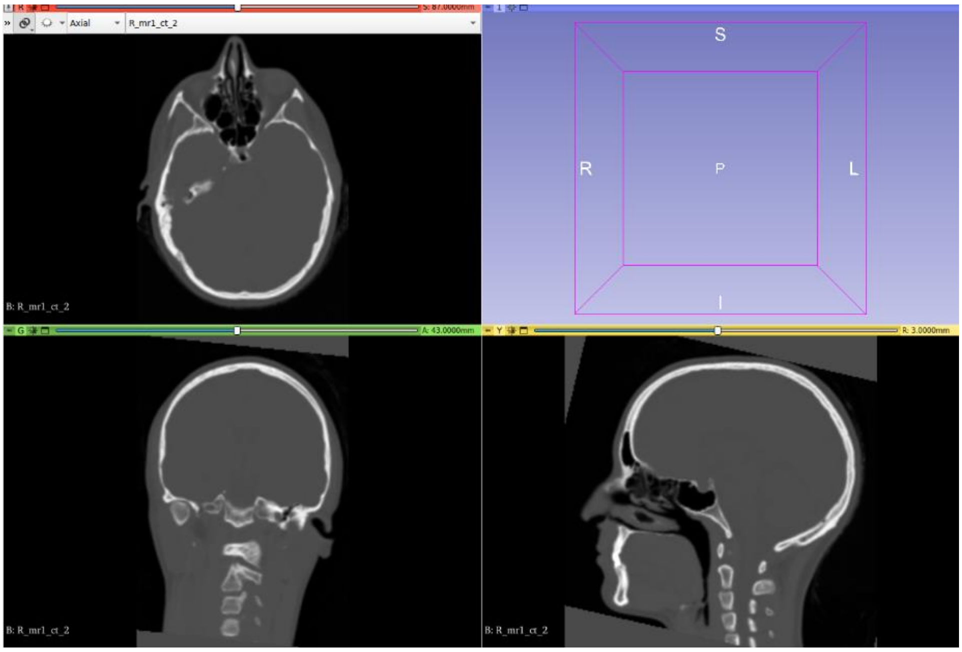

Figure 9: Output of rigid registration on ct with respect mr1 after aligning of the centroids of both heads.

Annex 1, which shows the superposition of mr1 (lowering its transparency) over the outcome shown above, helped to see that the first step of aligning the centroids was helpful, obtaining much better results.

Another possibility of overcoming the problems ecountered before is to apply an elastic registratoin of both images, after (again) aligning the centroids.

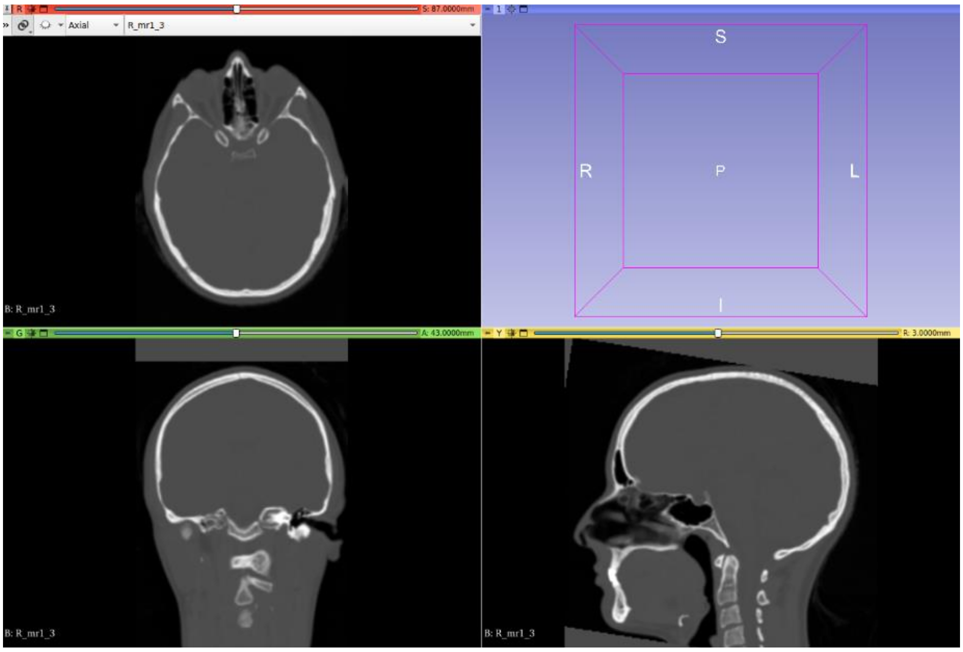

Figure 10: Output of elastic registration on ct with respect mr1 after aligning of the centroids of both heads.

The results obtained from an elastic registration, also in *Annex 2*, where the superposition of the mr1 over the outcome seen above is shown, are similar to those obtained after rigid registration, (both procedures after the aligning of the centroids).

However, comparing the images on the different projections, shown in *Annex 3*, the following can be concluded. The elastic registration appears to align better the images in the transversal plane, while the rigid registration in the coronal and sagital planes. This happens as when working in 3-dimensions, when taking a single slice of different planes, errors can appear in some of them.

Finally, the last approach consist on using the generic registration method of the Elastix library, leading to the results shown in *Figure 11*.

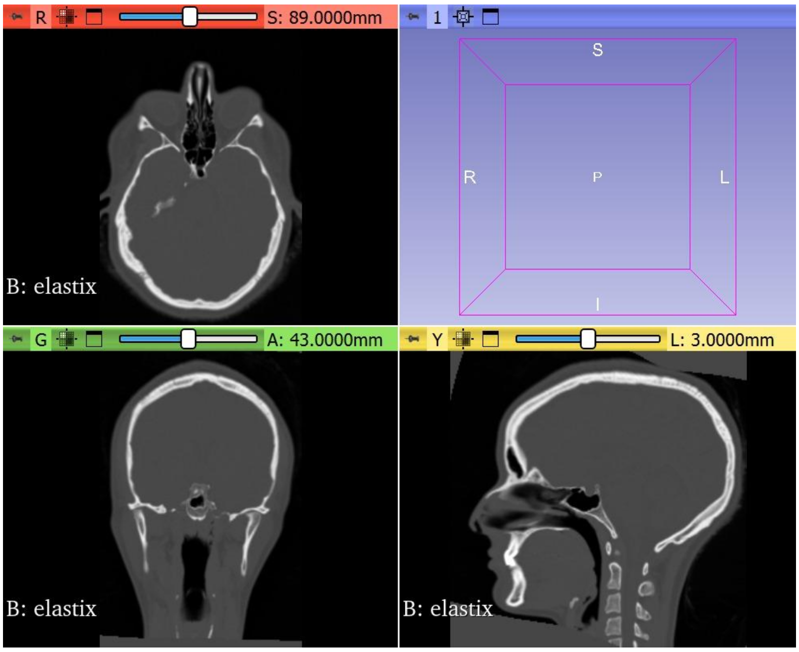

Figure 11: Output of Elastix registration on ct with respect mr1.

With the superposition of the mr1 over the results obtained with this powerful library, shown in Annex 4, it can be seen that they are similar, but slightly better than those obtained when applying rigid or elastic registration after the aligning of the centroids. It because Elastix optimizes better parameters of the registrations.

4. Implementation of a Registration Algorithm.

The steps provided in the laboratory guide will be followed, each explained in the following sections.

4.1. Transformation Function

This is a function that applies a rigid transformation to the 2D CT slice, which consists of a translation plus a rotation (4 DOF since it is 2D) applied to an image, with the parameters of the angle of rotation and the shift in the x-axis and y-axis passed to it as input. The transformations are achieved by multiplying the image array by the matrix corresponding to the transformation.

$$T_{AB} = \begin{bmatrix} cos\theta & -sin\theta & x \\ cos\theta & cos\theta & y \\ 0 & 0 & 1 \end{bmatrix}$$

A predefined function is used for the implementation of the translation and rotation:

<table>
<tr><td colspan="6">transformation (image, angle, translation):</td></tr>
<tr><td rowspan="3">transformation ()</td><td>image</td><td>Input image array on which to perform the transformation</td><td rowspan="3">affine_transform ()

[Apply geometric transformation to the image array]</td><td>image</td></tr>
<tr><td>angle</td><td>Angle, in radians, that defines how much the image will be rotated</td><td rowspan="2">transformation matrix</td></tr>
<tr><td>translation</td><td>Vector, that defines how much the image will be translated in the x-axis and y-axis</td></tr>
<tr><td colspan="6">Output: translated and rotated image</td></tr>
</table>

The output image with a rotation of 15 degrees and a translation of 55 pixels to the left in the x-axis and 20 pixels upwards in the y-axis.

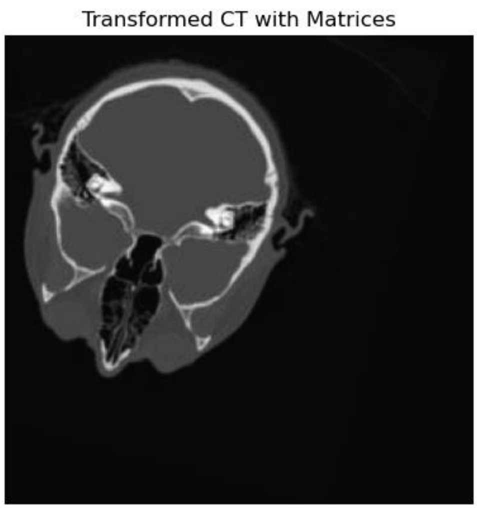

Figure 12: Translated and Rotated CT slice

4.2. Interpolation Function

This function interpolates the transformed image. Interpolation means the estimation of the values of the points from the transformed image grid such that they fall back into the grid of the original image.

Bilinear interpolation is performed using linear interpolation first in one direction, and then again in the other direction. It uses values of only the 4 nearest pixels, located in diagonal directions from a given pixel, to find the appropriate intensity values of that pixel.

A predefined function is provided for achieving this goal:

<table>
<tr><th colspan="4">bilinear_interpolate(image, points):</th></tr>
<tr><td rowspan="2">transformation ()</td><td rowspan="2">RegularGridInterpolator ()
[Create a 2D interpolator for the input image using regular grid interpolation]</td><td>image</td><td>Input image array on which to perform the interpolation.</td></tr>
<tr><td>points</td><td>An array of 2D coordinates where interpolation should be performed.</td></tr>
<tr><td colspan="4">Output: Interpolated values corresponding to the input points.</td></tr>
</table>

Although a manual approach of the interpolation function can also be implemented. The manual interpolation function estimates values at new grid points by computing a weighted sum of pixel values from the four nearest neighbors. This is based on their distances from the input/original coordinates, hence adjusting the displaced coordinates.

The fractional parts, dx and dy, indicate the distance of the input coordinates from their floor values, influencing the interpolation weights.

- **floor value**: The largest integer less than or equal to a given number.
- **ceil value**: The smallest integer greater than or equal to a given number.

To prevent index out-of-bounds errors, coordinates are clipped to the image boundaries. This ensures that the calculated coordinates fall within the valid range of image indices. The result is a bilinear interpolation, providing smoother transitions between pixel values for transformed image points.

<table>
<tr><th colspan="4">bilinear_interpolate_manual(image, points):</th></tr>
<tr><td>image</td><td colspan="3">Input image array on which to perform the interpolation.</td></tr>
<tr><td>points</td><td colspan="3">An array of 2D coordinates where interpolation should be performed. These ‘points’ are obtained from the personalized function ‘create_grid’ explained in Annex 4.</td></tr>
<tr><td rowspan="2">Bilinear interpolation through a linear combination.</td><td rowspan="2">np.clip()</td><td>np.floor()</td><td>First column of the points variable.</td></tr>
<tr><td>np.ceil()</td><td>Second column of the points variable.</td></tr>
<tr><td colspan="4">Output: Interpolated values corresponding to the input points.</td></tr>
</table>

For proper visualization of how the pixel grid shifts and rotated from the original image to the transformed image, and hence the importance of interpolating in the registration pipeline, the following plot, shown in *Figure 19*, is performed:

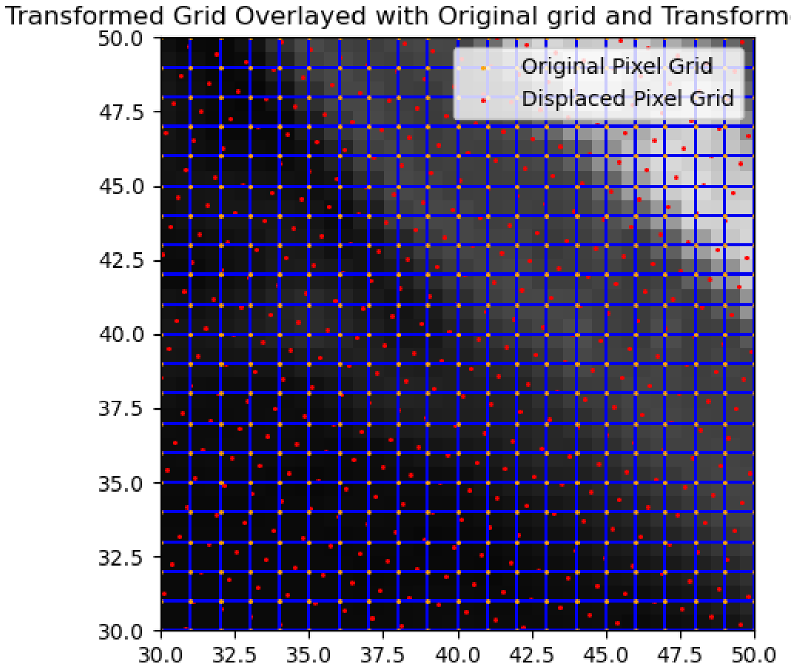

Figure 13: Transformed grid overlayed with the original image grid and the transformed image for visualization of necessary implementation of interpolation.

4.3. Mutual Information Function

This function will be implemented to calculate the mutual information between the two images. Mutual Information is an intensity-based measure that does not rely on specific image features and is used as a similarity metric to quantify the amount of information shared between two images. The optimization process aims to find the appropriate transformation parameters that minimize this measure, which would indicate a good alignment between the images.

$$MI(I,J) = H(I) + H(J) - H(I,J)$$

For two images (I) and (J) their intensity values are treated as random variables. The joint probability density function (P (I, J)) represents the probability of observing a specific pair of intensity values (I, J) in the images. The marginal probability density functions (P(I)) and (P(J)) represent the probabilities of observing specific intensity values in each image independently. We find these joint and marginal probabilities from the joint and marginal histograms respectively.

The entropy, which is a measure of the uncertainty of a random variable, is calculated, for the random variables (I), (J) and (I, J). Then with the equation provided above the mutual information between images is calculated, measuring the reduction in uncertainty about one image when the other image is known.

<table>
<tr><td colspan="4">mutual_information(image1, image2, bins=271):</td></tr>
<tr><td>image1, image2</td><td colspan="3">Input images, arrays on which to perform the computations.</td></tr>
<tr><td>bins</td><td colspan="3">Number of bins for histogram calculation</td></tr>
<tr><td rowspan="2">Mutual Information</td><td>joint_entropy → entropy</td><td>joint_prob → np.histogram2d()</td><td rowspan="2">image.flatten()</td></tr>
<tr><td>entropy_image → entropy</td><td>prob_img → np.histogram()</td></tr>
<tr><td colspan="4">Output: Mutual information value.</td></tr>
</table>

4.4. Optimization of Mutual Information

The optimization of MI function consists of an iterative process used to find the rotation angle and translation vector that returns the transformed image to the same state of the original image, by minimization of the mutual information metric. Actually, what is being minimized is the negative of the mutual information, since the MI quantifies the amount of information shared between two images, the aim, then, is to increase the MI metric, which is equivalent to minimizing the negative of the MI.

In each iteration of the optimization process, a transformation is applied to the initially transformed image and the negative of the mutual information is calculated between the newly transformed image and the original image. After that, these steps are repeated until a stopping criterion, defined a priori, is reached.

A personalized function is created that will be the one to be minimized by the optimizer:

<table>
<tr><td colspan="4">min_mutual_info (params, image1, image2):</td></tr>
<tr><td rowspan="7">min_mutual_info ()</td><td rowspan="3">params</td><td>angle</td><td rowspan="3">A list of the rigid transformation parameters provided by the optimizer that will be used to correct the transformed image (image2)</td></tr>
<tr><td>translation_x</td></tr>
<tr><td>translation_y</td></tr>
<tr><td>image1</td><td colspan="2">Reference image array used to calculate the mutual information with the transformed image</td></tr>
<tr><td>image2</td><td colspan="2">Initially transformed image</td></tr>
<tr><td colspan="2">transformation ()</td><td>mutual_information ()</td></tr>
<tr><td colspan="2">image2 will be transformed again with the geometric transformation calculated by the optimizer.</td><td>Calculates the mutual information between the reference image and the corrected one with the rotation and translation values obtained from the optimizer</td></tr>
<tr><td colspan="4">Output: the negative of the Mutual Information between the reference image and the corrected image at a certain iteration of the optimization process.</td></tr>
</table>

The *min_mutual_info()* function is optimized with the *minimize* predefined function from *scipy.optimize*:

<table>
<tr><th colspan="3">minimize (function, initial_params, args=(), method=None, bounds=None, tol=None):</th></tr>
<tr><td rowspan="6">minimize ()</td><td>function = mutual_informationI()</td><td>Objective function to be minimized</td></tr>
<tr><td>initial_params = [-1, -1, -1]</td><td>Array of the initial independent variables (rotation and translation in each axis) that optimizer will first use. Since a positive rotation and translation was initially used for the transformed image, negative values are given to guide the algorithm in the correct direction.</td></tr>
<tr><td>args = (interpolated_ct, ct)</td><td>Extra arguments passed to the objective function, in this case the images with which the mutual information will be calculated</td></tr>
<tr><td>method = ‘Powell’</td><td>Backbone optimization algorithm used. After trying several different methods, Powell provided the best results.</td></tr>
<tr><td>bounds = [(-20, 0), (-40, 0), (-40, 0)]</td><td>Range of parameter values in which the optimizer will search to find the one that minimizes MI</td></tr>
<tr><td>tol = 1e-7</td><td>Stopping criterion of the optimization</td></tr>
<tr><td colspan="3">Output: object with attributes such as the solution array x. A Boolean flag indicating if the optimizer exited successfully and message which describes the cause of the termination.</td></tr>
</table>

After several iterations, the resulting registered image is obtained, with a correction over the transformed/reference image of -12.5 degrees, a translation in x of -35.2 and a translation in y of -13.9:

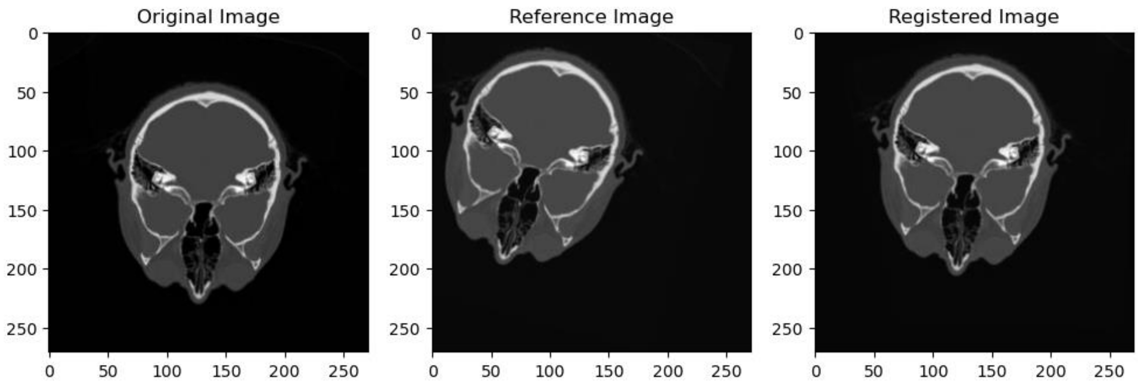

Figure 14: Comparison of the original, transformed, and registered images.

The image obtained in not in perfect concordance with the original ct slice but there is a noticeable improvement with respect to the reference image.

5. Conclusions

The outlined image registration process for 2D CT slices demonstrates a methodical approach integrating transformation, interpolation, and mutual information-based optimization. The use of a rigid transformation with four degrees of freedom, including rotation and translation, is complemented by bilinear interpolation for precise grid adjustments. Mutual information serves as a robust similarity metric, guiding the optimization process to align images effectively. The optimization method, employing the 'Powell' algorithm, refines the transformation parameters. The resulting registered image, exhibiting a correction of -12.5 degrees rotation and translations in both x and y axes, reflects a significant enhancement over the initial transformed image.

Additionally, it is worth noting that further exploration into alternative cost functions or enhancements to the existing one may contribute to refining the registration results. However, this will require a more dedicated study and investigation in order to find the optimal cost function for the proposed task.

6. Annexes

Annex 1: Superposition of mr1 with low transparency over the results obtained after the alignment of the centroids and rigid registration.

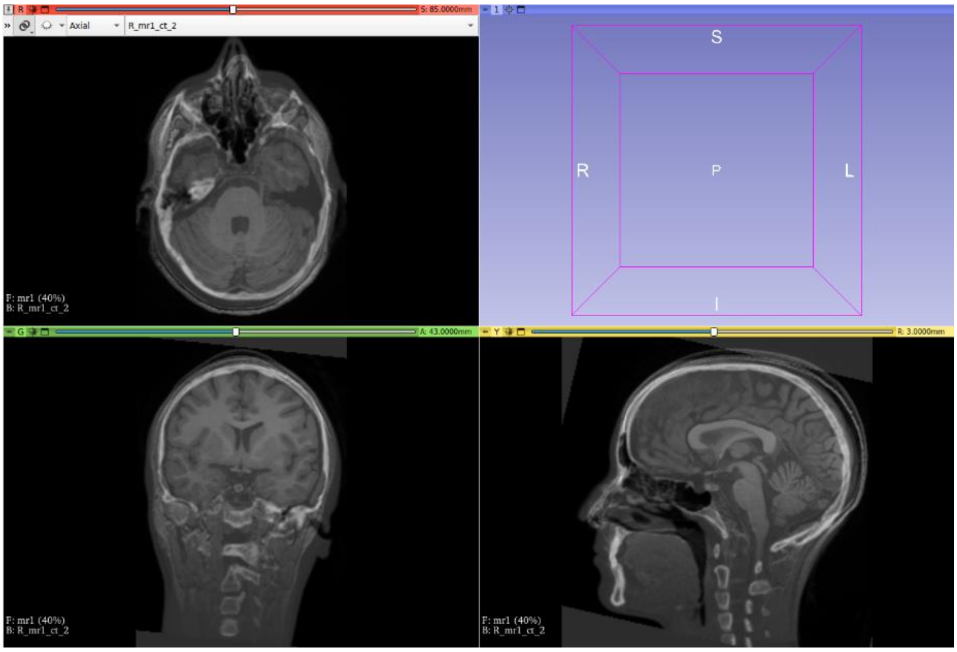

Figure 15: Superposition of mr1 with low transparency over the results obtained after the alignment of the centroids and rigid registration.

Annex 2: Superposition of mr1 with low transparency over the results obtained after the alignment of the centroids and elastic registration.

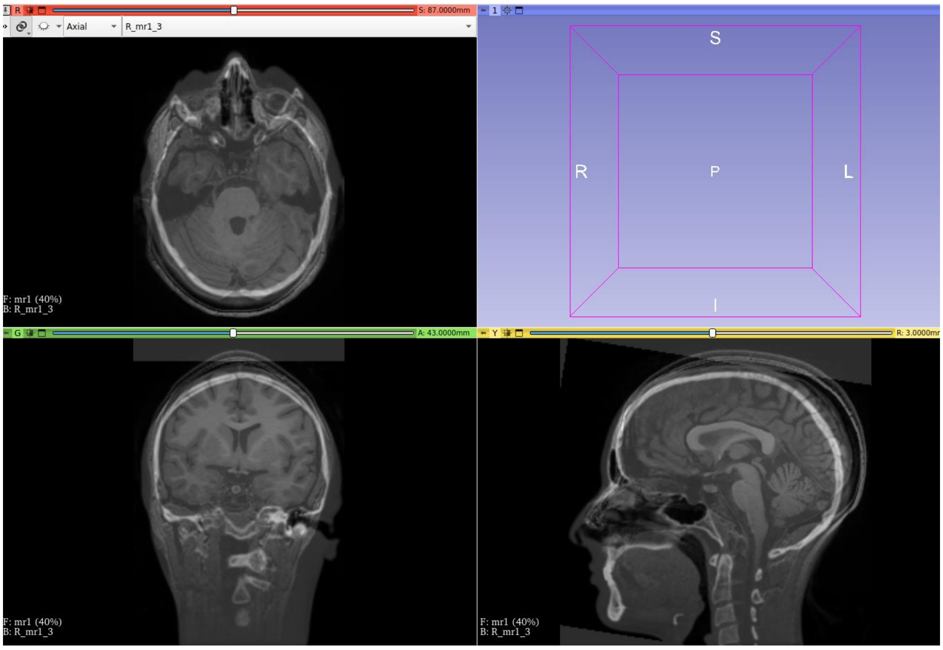

Figure 16: Superposition of mr1 with low transparency over the results obtained after the alignment of the centroids and elastic registration.

Annex 3: Comparisons between results obtained with rigid and elastic registration after the aligning of the centroids.

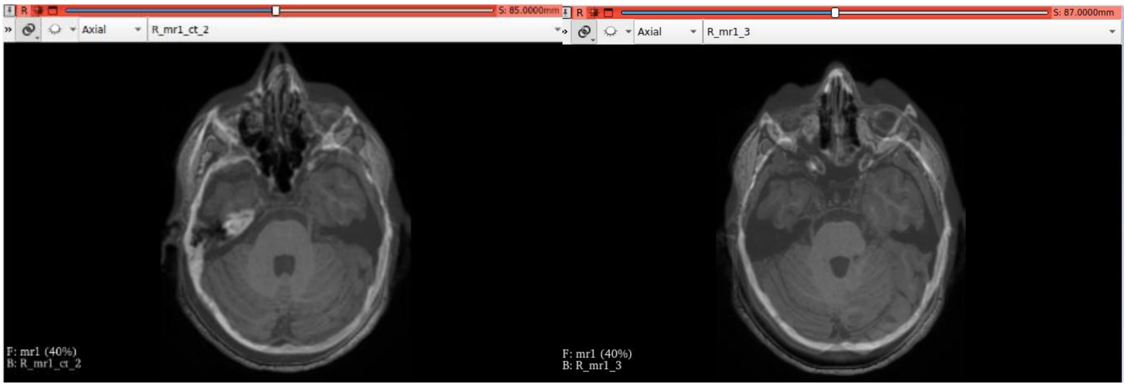

Figure 17: Comparison of the transversal plane, having rigid and elastic registration respectively.

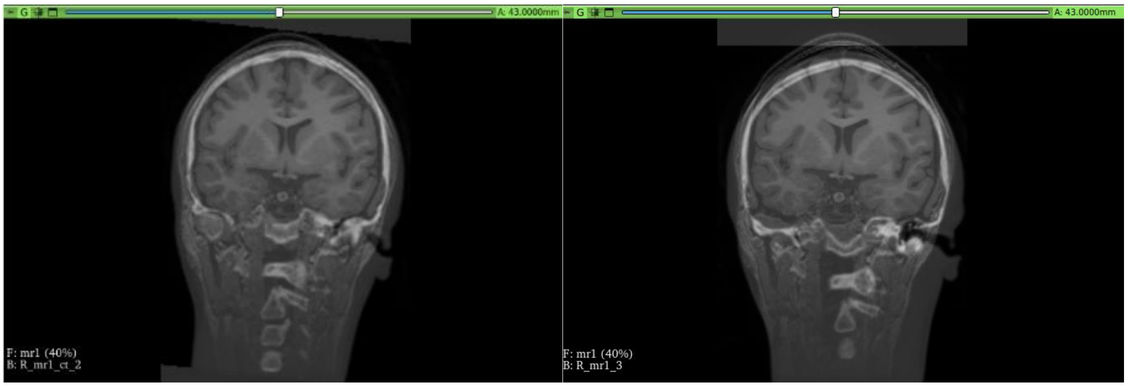

Figure 18: Comparison of the coronal plane, having rigid and elastic registration respectively.

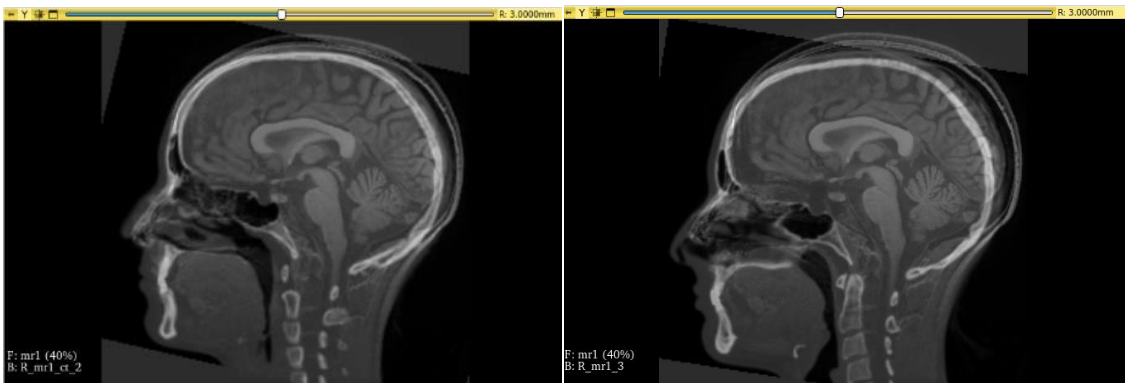

Figure 19: Comparison of the sagital plane, having rigid and elastic registration respectively.

Annex 4: Superposition of mr1 with low transparency over the results obtained after the Elastix registration.

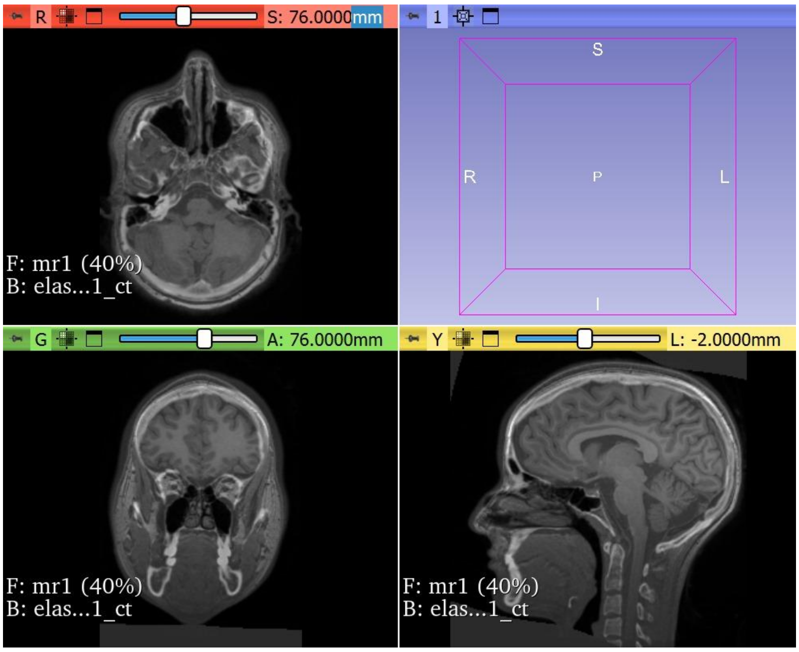

Figure 20: Superposition of mr1 with low transparency over the results obtained after Elastix registration.

Annex 4: Table explaining the personalized function ‘create_grid’.

<table>
<tr><td colspan="6">create_grid (image_shape, spacing):</td></tr>
<tr><td>Image_shape</td><td colspan="5">A tuple representing the shape of the image (rows, cols).</td></tr>
<tr><td>spacing</td><td colspan="5">The spacing between grid points.</td></tr>
<tr><td rowspan="2">Construction of the grid.</td><td rowspan="2">np. column_stack()</td><td rowspan="2">flatten()</td><td rowspan="2">np.mgrid</td><td>rows</td></tr>
<tr><td>cols</td></tr>
<tr><td colspan="6">Output: 2D array representing a regular grid of points.</td></tr>
</table>

7. Table of figures