Medical Image Segmentation and Application

Laboratory 3

Atlas Based segmentation (Integration to the EM algorithm)

Abdullah Thabit
Pierpaolo Vendittelli

Medical Imaging and Applications (MAIA)
Universitat de Girona
27-11-2019

Contents

1. Part A (MiRA)

1.1 Objective and Introduction

The primary goal of this first part is to build a probabilistic atlas from a set of brain volumes with the available labels of three classes (WM, GM and CSF). In order to perform this part, the team should first use the provided software (***elastix*** and ***transformix***) to register the images and obtaining the transformation matrix to apply to the labels.
Objectives of this first part:

- To understand how to perform a single registration of two 3D volumes using rigid and nonrigid registration with elastix.
- To develop an algorithm to build the probabilistic atlas.
- Generate the tissue models.

An Atlas is a template representing 'a-priori knowledge' useful to segment specific parts (or organs) inside an image. There are different types of atlases, the most simple is obtained averaging the intensity of a zone, while more complex ones are obtained adding statistics (variance or other models of distribution).

Elastix is a open source software widely used for medical image registration problems. Segmenting images using atlases turns the segmentation problem into a registration problem, as the quality of the segmentation depends on the quality of the registration itself. Elastix has different parameters to be tuned (the same parameters of a classical registration framework), such as: *Metric*, *Image sampler*, *Interpolator*, *Transform*, *Optimizer* which will be discussed in the apposite section.

1.2 Analysis and Implementation

1.2.1 Elastix

As mentioned in the introduction, thanks to the usage of atlases, the problem on image segmentation becomes a matter of image registration. The open source software *elastix* comes in our help giving us the possibility of register images. Elastix has different parameters to be tuned which are the following:

- **Images**: the images to be registered, one moving and one fixed. For all our experiments we set the fixed image the image *1000.nii.gz* while the moving images were all the rest of the training set
- **Metrics** are the similarity indexes used to assert the quality of the registration, Elastix gives access to different types of metrics such as MI, NCC, MSD, NMI (Normalized Mutual Information) and KS (Kappa Statistics)

- **Image Samplers** represent subset of the moving images, in case is not necessary to process the whole image.
- **Interpolators** are the different interpolators available during the optimization step. They vary from the less memory requiring (Nearest Neighborhood) to the most expensive N-th Order B-Spline
- **Transforms** are the set of transformations that can be applied to register the images. A transformation can be either rigid or non rigid. For our lab we used Affine transformation (rigid) and B-Spline (non rigid).

Elastix is a command line program, thus script can be executed to apply registration on multiple images. The command we used (inside a python script) to register the images is the following.

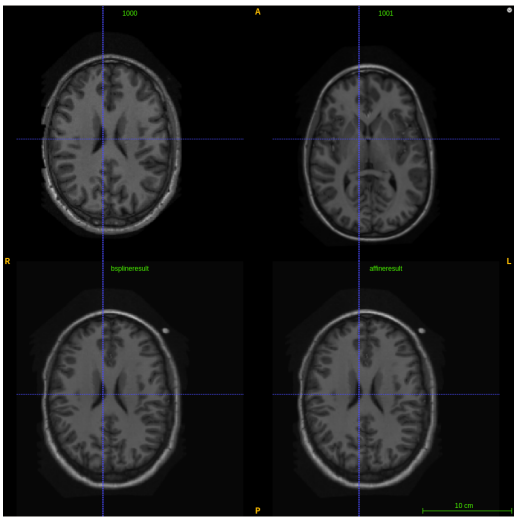
```
elastix -f {f\_img} -m {m\_img} -out {outPath} -p {paramFile1} -p {paramFile2}
```

Listing 1.1: Example of calling elastix

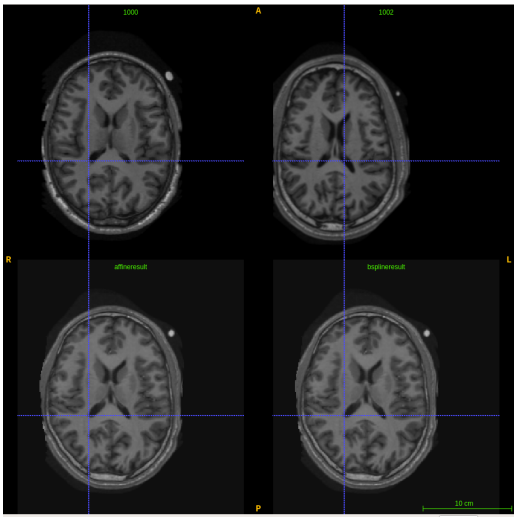
Where :

- **-f** = fixed image (image 1000.nii.gz)
- **-m** = moving image (every other image in the folder)
- **-out** = output directory where to save the results
- **-p** = parameter file with to the registration

The parameter file is a .txt file defining the components of the registration and their parameter values. For our lab we used both Affine and BSpline transformations. These two are contained in two different parameters file, together with other useful informations such as the Metric, the Optimizer, the Interpolator, the number of resolutions for the multi-resolution registration. The main differences between the parameter files of Affine and BSpline are in the optimizer (AdaptiveStochasticGradientDescent for Affine, StandardGradientDescent fo BSpline), the type of transform (AffineTransform in affine, BSplineTransform in BSpline), number of resolutions (4 in Affine, 5 in BSpline). As we can see from the above line of code, we can call elastix with a pipeline of parameters file.This is due to the fact that in order to tackle non rigid problems, it is better to first apply a rigid registration such as Affine, and then a non rigid such as BSpline. However these two parameters produce different outputs as we can see in the next images.



(a) Elastix performed on img training 1001. comparison affine vs bspline



(b) Elastix performed on img training 1002. comparison affine vs bspline

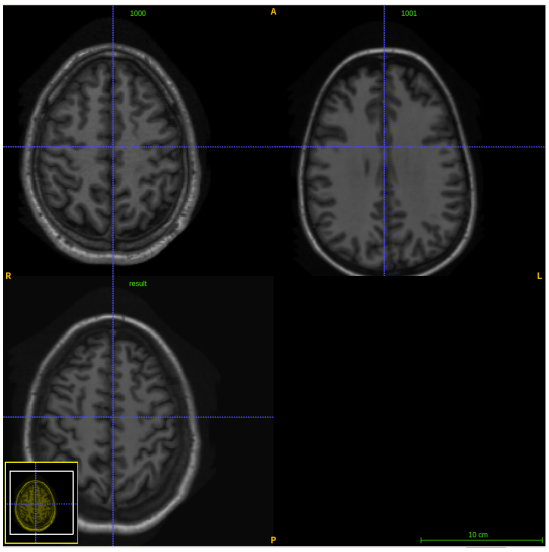
Figure 1.1: Result of Elastix registration, comparison of different parameters files

From the figure above we can appreciate that the registration looks visually correct both for the Affine transformation and for the BSpline transformation. In this two cases there are not visible differences between the Affine and BSpline transformations, this is probably due to the fact that for these two cases the registration problem was rigid. This means that with only the rigid parameters we can have a correct registration. In any case, the final strategy we adopted was to call elastix with a succession of parameters (Affine first, BSpline second) in order to tackle in a more precise way (if there are) cases of non rigid transformations. Thr following image is an example of the running Python script which applies this pipeline to the images in the training set.

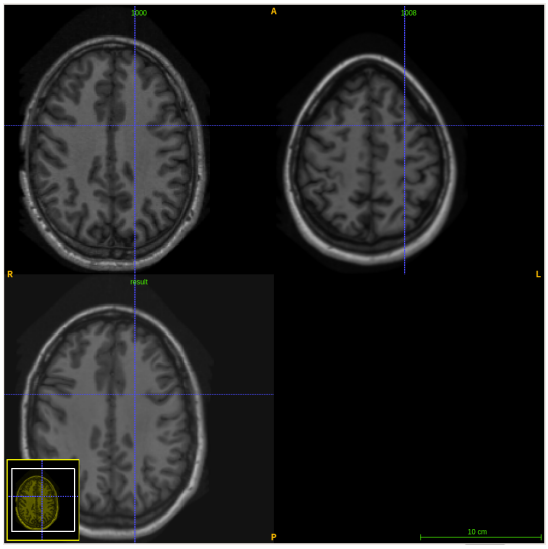
```
pierpaolo@pierpaolo-vendittelli:~/Desktop/MiSA/Lab3$ python register_training.py
[INFO] Running Elastix for image [1]...
('Needed: ', 121.18746089935303, ' seconds to apply elastix. Going to the next image')
[INFO] Running Elastix for image [2]...
('Needed: ', 127.2673909664154, ' seconds to apply elastix. Going to the next image')
[INFO] Running Elastix for image [3]...
('Needed: ', 127.92343306541443, ' seconds to apply elastix. Going to the next image')
[INFO] Running Elastix for image [4]...
('Needed: ', 128.59324097633362, ' seconds to apply elastix. Going to the next image')
[INFO] Running Elastix for image [5]...
('Needed: ', 129.59526205062866, ' seconds to apply elastix. Going to the next image')
[INFO] Running Elastix for image [6]...
('Needed: ', 127.5856020450592, ' seconds to apply elastix. Going to the next image')
[INFO] Running Elastix for image [7]...
('Needed: ', 127.83453392982483, ' seconds to apply elastix. Going to the next image')
[INFO] Running Elastix for image [8]...
```

Figure 1.2: Example of calling elastix for registration

The results of this pipeline can be appreciated in the following image. Again, we can say that the registration was performed correctly from a visual point of view.



(a) Elastix performed on img training 1001



(b) Elastix performed on img training 1008

Figure 1.3: Result of Elastix registration

1.2.2 Transformix

Elastix give access to another program, called ***transformix***. Transformix is useful to apply the same transfromation obtained from elastix to another image, which in our case is the label image of the registered (moving) image. As well as elastix, transformix is a command line program and can be called by the following command line .

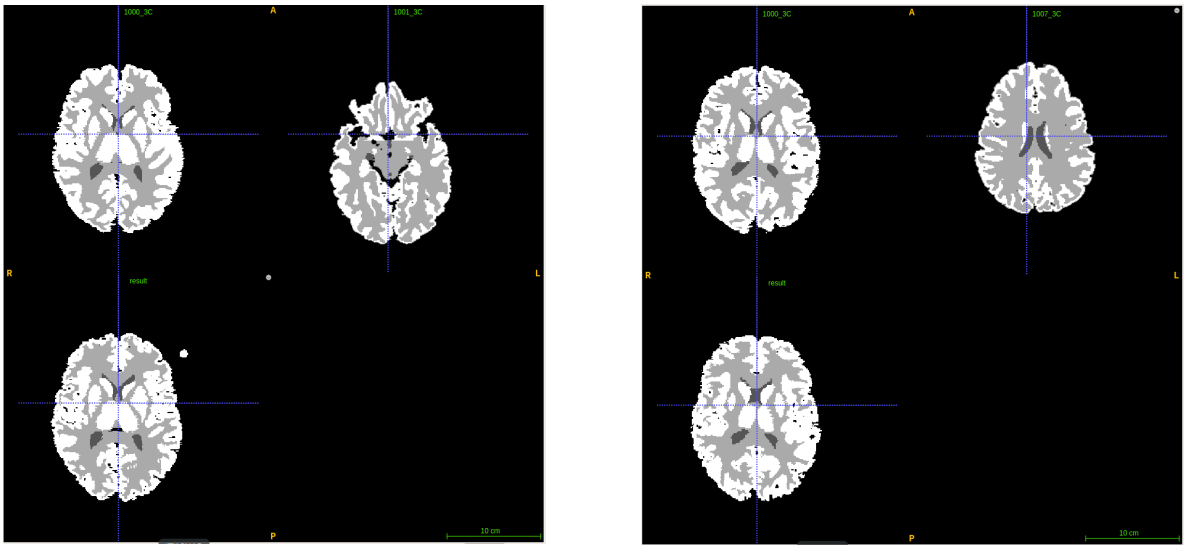
```
1 transformix -in {m\_mask} -out {outPath} -tp {paramFile}
```

Listing 1.2: Example of calling transformix

Where :

- **-in** = label to register
- **-out** = output directory where to save the results
- **-tp** = transformation parameter file generated by elastix

As we can see from the parameters that transformix takes as input, we are going to apply the transformation matrix generated previously from elastix to the labels of the moving image. It is important to point out that before applying transformix to the labels, a modification is needed to the transformation parameters. This modification is needed in order to have a correct interpolation of the pixel values. More precisely, The ***ResultImagePixelType*** has to be changed from 'short' to 'float' and ***FinalBSplInterpolationOrder*** has to be changed from '3' to '0'. The Following image shows the result of transformix on the labels of the images*1001.nii.gz* and *1007.nii.gz*.



(a) Transformix performed on label img 1001

(b) Transformix performed on label img 1007

Figure 1.4: Result of Transformix registration

1.2.3 Probabilistic Atlas

Once we registered all the training images, we can build the probabilistic atlas for the three zones of the brain, CSF, Gray matter an White matter. As mentioned before, the atlas is a template of 'a priori' knowledge useful for segmentation purposes. The atlas we built is a probabilistic atlas, therefore is bases on statistics. It gives, for each pixel of the image the a probability map (probability of belonging to one of the three classes) and thus, it is easy to integrate this algorithm with E.M. and Tissue Model and achieve so a better segmentation. The Probabilistic atlas was obtained by taking the mean of the vector containing the same class for each image as can be seen from the following pseudo-code.

Result: atlas_csf, atlas_gm, atlas_wm
initialization;
while *Read all the images* **do**
    Split according to the labels into three vector;
    Append to the vectors the next;
    Take the average;
end

Algorithm 1: Generation of Probabilistic Atlas

The results of this averaging are shown in the next Figure, where is possible to see the atlas of the image, csf, white matter and gray matter.

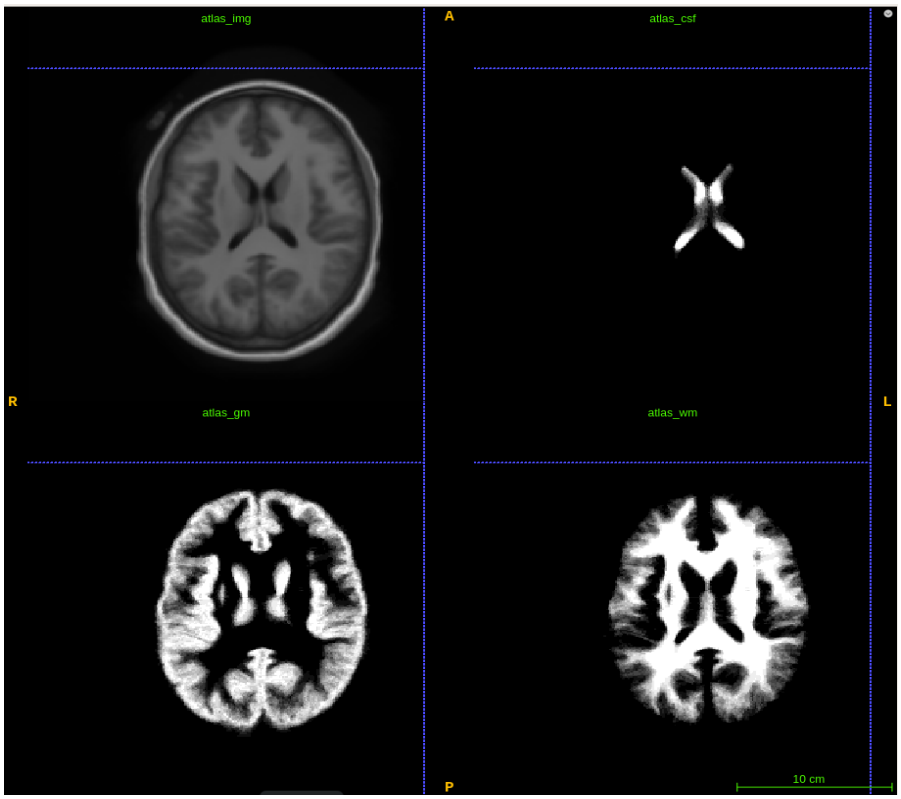


Figure 1.5: Atlas for the training images

1.2.4 Tissue Model

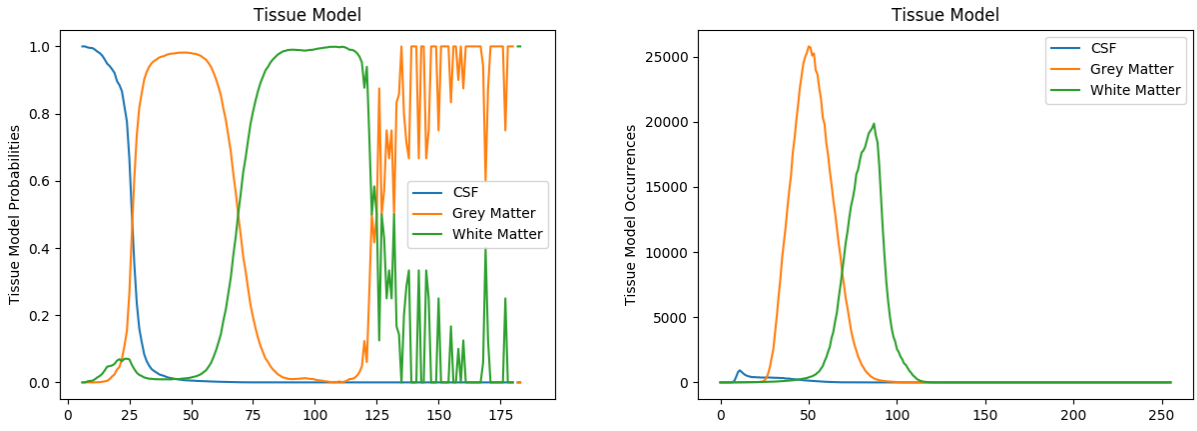
The last part of this section is about the generation of the tissue modes. The tissue model represent an intensity based Probability map, this means that with tissue models we are able to assign to each pixel a label only analyzing its intensity value. In order to generate the tissue models we calculate the histograms of the three classes of the training images and then normalize them to make them sum to one. More specifically, the following pseudocode summarized the steps necessary to the cration of the three tissue models.

Result: tm_csf, tm_gm, tm_wm
initialization;
while *Read all the images* **do**
    Apply min-max normalization;
    Remap into 8bit; Split according to the labels into three vector;
    Calculate the histogram of each vector; Take the average;
    Normalize the probabilities;
end

Algorithm 2: Generation of Tissue Models

The result of this calculation can be seen in the next Figure. From this figure we can notice that after the 125th bin, the curves of the gray matter and White matter behave in a not correct way. This is because the gray values of the images don't span the whole range of the 16 bit, and there where the curves are not correct, the probabilities for Gray matter, White matter and CSF are actually 0%, but since we are forcing the probabilities to map to a sum to 1, they appear like this. The Figure 1.6(b) shows the distribution of the tissues in the

image. The CSF, is the smallest curve because represents the smallest of the three classes.



(a) Tissue Models

(b) Not Normalized Tissue Models

Figure 1.6: Tissue Models

The tissue models are useful to segment the testing images. this is done by first normalizing (as the training set) all the test images, and then vectorizing each image and for each element of the image (pixel) assigning the label corresponding to the maximum probability for that level of intensity.

2. Part B (MiSA)

2.1 Objective and Introduction

In this second part of the lab we will be using the models and the data generated during the previous chapter (Part A) and during the previous lab (E.M implementation). In many medical image segmentation problems, the use of an anatomic or probabilistic Atlas is an essential part to guarantee spatially-consistent segmentation results. The goal of this coursework is to integrate the use of a probabilistic Atlas into the EM algorithm. The results should be provided by showing the Dice Score of each experiment effectuated.

Objectives:

- Check the effects of different clustering methods.
- Refine the first experiment by adding the previously generated EM.
- Improve the results by multiplying the intensities at the end.
- Improve the results by integrating the probabilities inside the EM.
- Compare the performances between our generated Atlas and the MNI template.

2.2 Implementation and Analysis of the Results

In this section we will discuss the different experiments as well as the results we obtained during this lab.

2.2.1 Initial Segmentation

The first experiment was to compare different segmentation methods and compare their performances. More in detail, we compared an unsupervised clustering algorithm (K-means), supervised segmentation based on intensity (Tissue Models), and supervised segmentation based on position (our probabilistic Atlas). For K-means segmentation, We used the scikit-learn K-means function to cluster the labels. Whereas, Tissue model segmentation was performed by looping through the bins of the tissue model (implemented in the MIRA part) and assigning their corresponding intensity probabilities for each tissue type, and finally taking the maximum probability as the label. The Atlas segmentation was based on propagation: The Atlas template is first registered to the testing images and then its probability maps are transformed to the testing image spatial space and the label is taken as the maximum probability for each voxel (same as for the tissue model).

Figure 2.1 below shows the box-plots with respect the the dice score for these simple segmentation methods (K-means, Atlas based, Tissue model based) for all the 20 images in the testing set. Table 2.1 shows the mean dice score for each tissue type using the aforementioned methods.

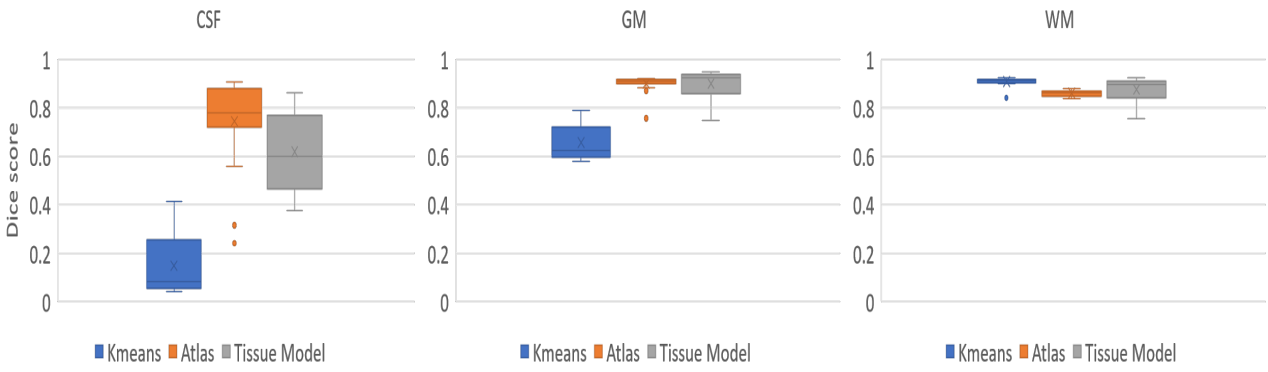


Figure 2.1: Initial segmentation methods

Table 2.1: Mean dice scores of initial segmentation methods

Method	CSF	GM	WM
K-means	0.14671	0.656929	**0.90889**
Atlas	**0.74436**	0.899704	0.860627
Tissue Model	0.619042717	**0.901253811**	0.87568146

From the table and figure above, we notice the Superior performance of the Atlas based segmentation for the minority class (CSF) with somewhat big margin since it's takes into account the spatial information for each class, whereas the K-means failed miserably for CSF, but performed better than the other for the WM segmentation, which is expected from an intensity unsupervised method that only takes into account the distance information between intensity values.

2.2.2 EM Refinement

To refine and improve the segmentation results obtained from the initial methods mentioned above (K-means, Atlas based and tissue model based), the output of these methods was fed to the Expectation Maximization (EM) algorithm, which is an unsupervised clustering algorithm that tries to model the under laying distribution of the data by maximizing the models parameters in an iterative approach.

Therefore, the initial parameters of the Gaussian models (mean, co-variance and class distribution) are first calculated from the segmentation results obtained before, then the EM algorithm developed in the previous lab was used to maximize the membership probabilities to each of the tissue types to get the final segmentation results.

The comparison of the different initialization for the EM algorithm with K-means, Atlas, and Tissue model is presented in the table 2.2 and figure 2.2 below.

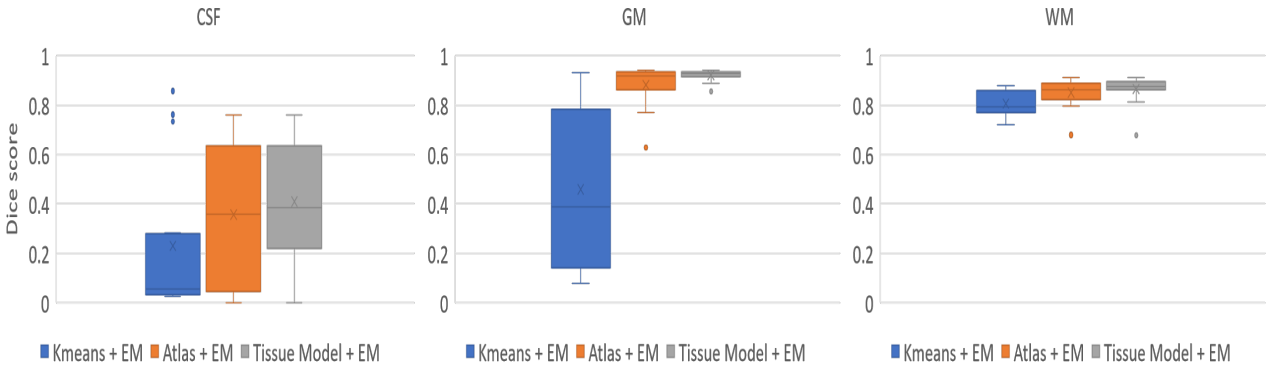


Figure 2.2: Initial segmentation methods + EM

Table 2.2: Mean dice scores of initial segmentation methods + EM

Method	CSF	GM	WM
K-means + EM	0.229595813	0.458678314	0.806300403
Atlas + EM	0.356704	0.881338216	0.850652937
Tissue Model + EM	**0.410125612**	**0.921448719**	**0.865221303**

From the pox-plots shown above we see a big difference in the EM convergence between the K-means initialization from one side and Atlas and tissue model from the other side. Tissue model initialization performed slightly better than that of Atlas initialization. The box-plots shows a slightly higher mean value for the dice score but the variance is lower for CSF segmentation when EM is initialized with the tissue model. This is confirmed in Table 2.2.

However, we can also notice that EM refinement did not help in improving the segmentation, in fact, we notice a huge drop in the CSF score for EM initialized Atlas compared to the original Atlas segmentation. Therefore, a better way of integrating the Atlas spatial information within the EM algorithm is needed.

2.2.3 Further improvements of EM segmentation

To further refine the segmentation results of the EM algorithm, the Atlas probabilities and tissue model probabilities were integrated with EM in different ways:

- multiply Atlas/tissue model probabilities with EM probabilities at the end to obtain the final segmentation result
- integrate Atlas inside EM by multiplying its probability maps with those of the EM at each iteration (Atlas inside EM [1])
- integrate Atlas inside EM by replacing the distribution probabilities with those of the Atlas, so spatial distribution are no longer needed to be estimated. (Atlas inside EM [2])

The figure 2.3 and table 2.3 below show the result of these updates to the EM algorithm.

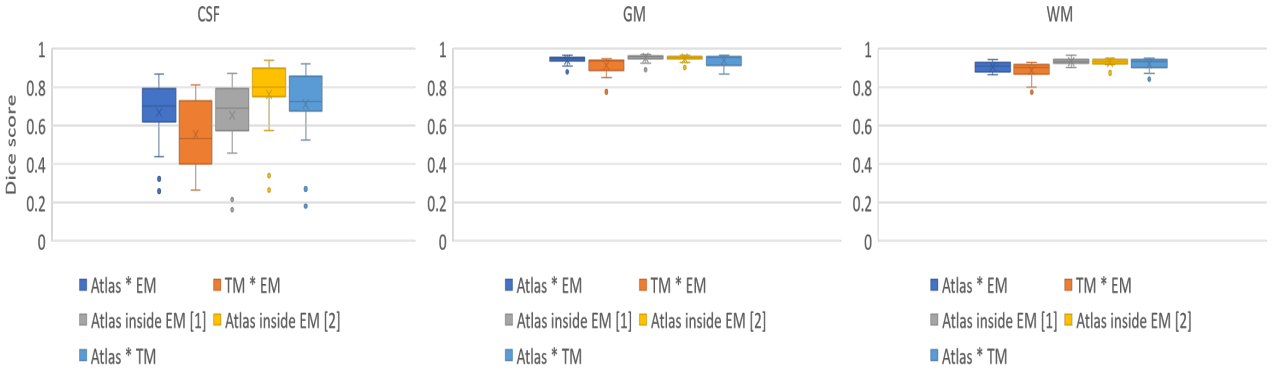


Figure 2.3: Further improvements

Table 2.3: Mean dice scores of further improvement to EM

Method	CSF	GM	WM
Atlas * EM	0.669044944	0.942705618	0.907182431
TM * EM	0.553361556	0.914150145	0.887011514
Atlas inside EM [1]	0.654242779	**0.953480648**	**0.933654514**
Atlas inside EM [2]	**0.76365715**	0.949334502	0.931286193
Atlas * TM	0.712649383	0.939602121	0.921944309

From the table above, we see how these different ways of utilizing spatial and intensity information from the probabilistic Atlas and tissue model respectively helped increasing the performance of the EM algorithm. Integrating the Atlas probabilities inside Em to update the membership weights performed the best, with a big margin for the minority class CSF, compared to the other EM refinement methods. However, it is also worth mentioning the good performance achieved by only multiplying the Atlas probability maps with those of the tissue model and therefore combine spatial and intensity information in a simple way.

From the different approaches of segmentation performed, the Atlas based segmentation showed the best performance at each step. To show the performance progress of integrating Atlas with EM, Figure 2.4 and table 2.4 below shows the box-plots and mean dice scores.

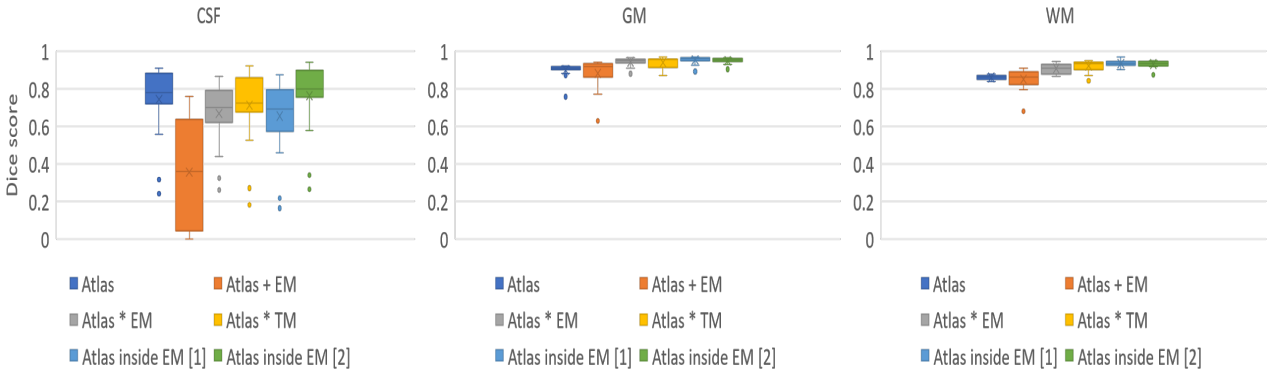


Figure 2.4: Atlas progress

Table 2.4: Mean dice scores of Atlas progress

Method	CSF	GM	WM
Atlas	0.74436012	0.899703958	0.860627251
Atlas + EM	0.356703818	0.881338216	0.850652937
Atlas * EM	0.669044944	0.942705618	0.907182431
Atlas * TM	0.712649383	0.939602121	0.921944309
Atlas inside EM [1]	0.654242779	**0.953480648**	**0.933654514**
Atlas inside EM [2]	**0.76365715**	0.949334502	0.931286193

From the box-plots in figure 2.4, we see how the EM initialization with Atlas segmentation dropped the accuracy for CSF but then started to improve as better integration methods were implemented. This is also confirmed by looking at the box-plots for GM and WM. The Atlas inside EM [2] showed the best results for CSF segmentation and about the same results with Atlas inside EM [1] for GM and WM.

2.2.4 Comparison with MNI

From the above experiments the advantage of using probabilistic Atlas for segmentation is evident, however, in these experiments we had the advantage of having the data set to build our own Atlas. What if there was not enough data to build an Atlas? To answer this question, we compared the performance of our Atlas to the publicly known MNI Atlas in different approaches as showing in Figure 2.5 below.

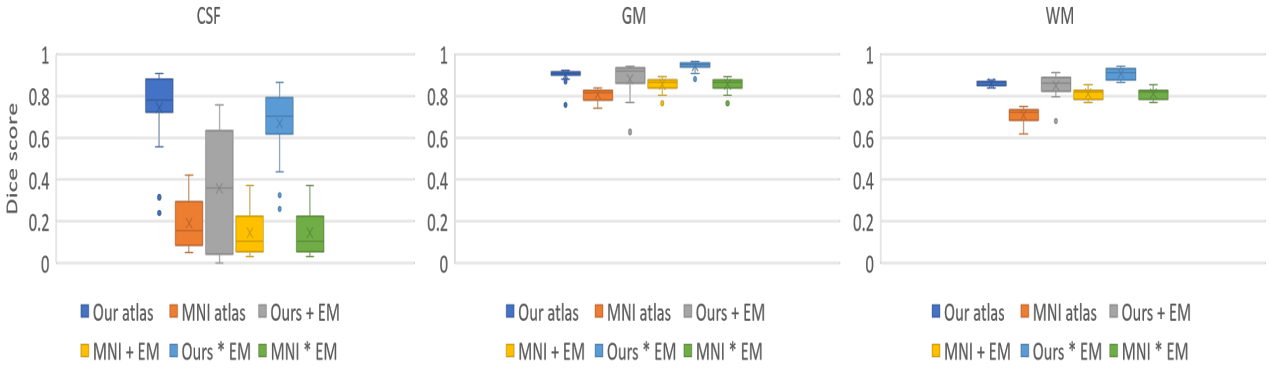


Figure 2.5: Our Atlas vs MNI

Table 2.5: Comparison between our Atlas vs MNI Atlas

Method	CSF	GM	WM
Our Atlas	**0.74436**	0.899704	0.860627
MNI Atlas	0.191734	0.807133	0.708954
Ours + EM	0.356703818	0.881338216	0.850652937
MNI + EM	0.145623055	0.8550033	0.8113151
Ours * EM	0.669044944	**0.942705618**	**0.907182431**
MNI * EM	0.145623055	0.8550033	0.8113151

From Figure 2.5 above, we see how our Atlas performed better than the MNI Atlas in both as stand alone label propagation segmentation and integrated with EM. This big difference is clearly shown for CSF segmentation and is confirmed with the GM and WM segmentation. This difference comes because our Atlas was built from the same data-set that was tested on therefore had the advantage over the MNI Atlas. Another advantage of building our own Atlas is the possibility to create the tissue model which is not available for the MNI Atlas.

From all experiments, almost all approaches performed better in segmenting the GM and WM and not very well with the CSF. This is mainly due to the bias ground truth for the CSF at the cortical sides of the brain. To conclude all our experiments, we can see that Atlas if integrated well inside EM, it can provide a good boost in the segmentation results, as discussed for our best model (Atlas inside EM [2]). Figure 2.6

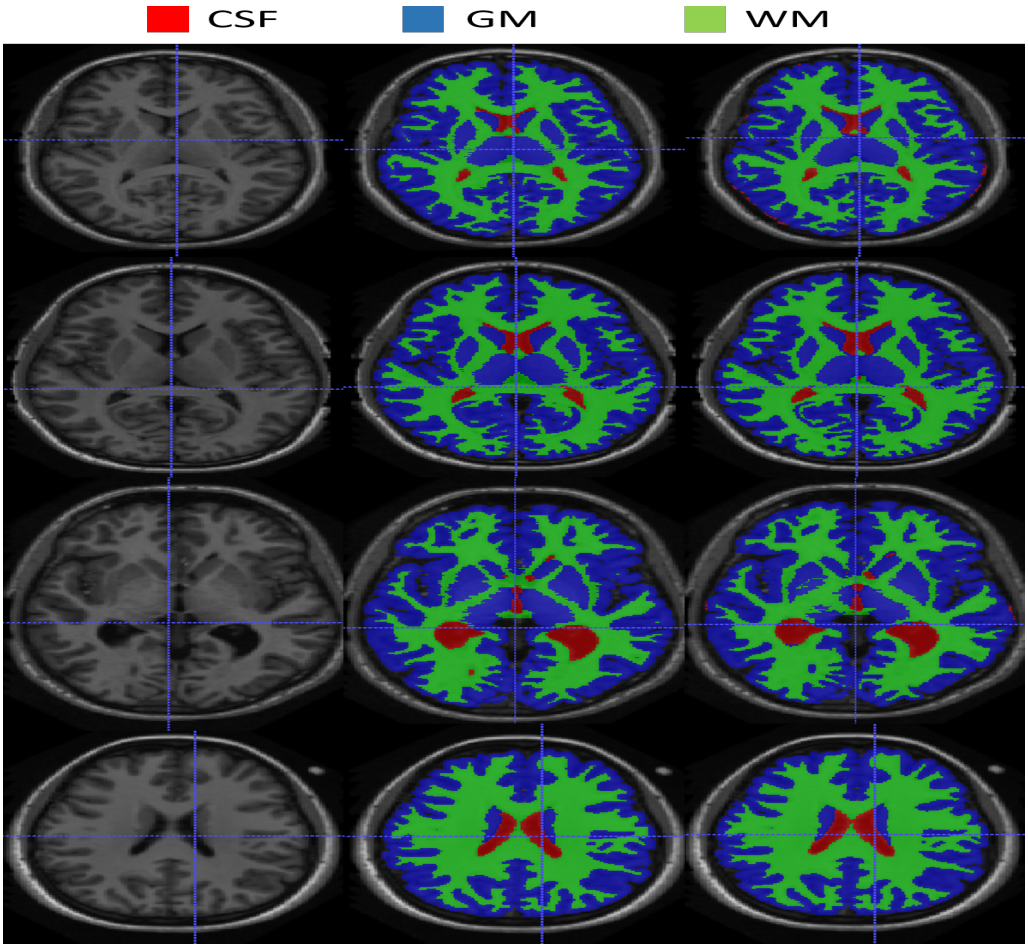


Figure 2.6: Some sample results of our best performing model (Atlas inside EM[2])

3. Conclusions and Project Management

As a general conclusion of the lab we can say that we learned how to efficiently use the elastix software to solve our registration problems. We learned the concepts behind atlases and tissue models, which are others powerful methods for segmentation. In particular we saw that atlas is a very powerful and simple concept which performs better than other clustering methods (such as kmeans). It is a completely different method from kmeans because it takes into account spatial information, while kmeans only consider intensity informations. Furthermore, the initialization of E.M. is really important to achieve a correct segmentation and when we combine spatial information coming from atlas the results are really good. The following tables are an approximation of the tasks division. We mostly worked during the labs, individually and had three additional meetings to finalize the work.

Table 3.1: MIRA MISA schedule

	Task	**Executed by**	**Details**
1	Apply Elastix and Transformix	All	Python script to execute registration and transformation
2	Creating Atlas	All	Registering the images to the reference image and creating Atlas
3	Tissue Model1	Pierpaolo	Computing the tissue model
4	Tissue Model2	Abdullah	Segmenting testing images
5	Initial segmentation	All	Initial segmentation kmeans, atlas, tissue models
6	EM + kmeans, Em+ atlas EM + tissuemodel	All	Adding EM to the previously obtained segmentations
7	EM * Atlas probabilities	Abdullah	Further improvement of integrating EM
8	MNI Comparison	Pierpaolo	Implementing MNI and comparing results
9	Code Documentation	Abdullah	Commenting the code
10	Report Writing	All	Writing the report, providing tables and figures