MIRA Final Report: Registration of CT scans of the lung using traditional registration methods with Elastix

Àlex Chicano, Cansu Yalçin, Ricardo Montoya del Ángel
EMJMD in Medical Imaging and Applications (MAIA), Master's in Medical Image Computing (MIC)
Univeristy of Girona

***Abstract*—Image registration is one of the main pipelines for medical imaging applications. It consists of aligning two different medical images into the same space. In this project work, we have performed non-rigid image registration on the Lung CT images taken from the Chronic obstructive pulmonary disease (COPD) dataset. The dataset contains inhale, and exhale images, as well as their landmarks corresponding to the same patient. The aim of the project is register both inhale and exhale images, and to measure the performance using the landmark point positions after registration. Target registration error (TRE) is used as a metric to determine the accuracy of the calculated landmark positions. The pipeline used in this project led us to get a $1,4223$ mm mean TRE value in a given test set, ranking second place at the registration challenge event.**

***Index Terms*—Image registration, Elastix, B-spline, Lung, Segmentation**

I. Introduction

Image registration is a vital step for medical imaging application pipelines, whose objective is to minimize undesirable effects (movement or deformation), analyze changes in images throughout time, or fuse image modalities. In simple terms, this process consists of making two or more images look alike, and aligning the images such that they are in the same space.

In the special case of a pair of images, one image is called the fixed image, and the other the moving image. Image registration aims to apply a transformation (mapping) on the moving image such that it resembles the fixed image.

Image registration can be classified into two general groups: intensity-based and feature-based registration. The former relies on the intensity values of the images whereas the latter uses key points of the images to register. In this project, only intensity-based registration was performed. Other classification criteria for registration methods can be used depending on other factors such as the transformation implemented in the registration workflow, for instance. A proper discussion over the classification of these methods and their impact on the registration framework will be given in section II.

The objective of this project is to create a registration pipeline for chest CT scan images, when a pair of inhale and exhale CT scans are provided. Landmarks inside of the lung region are available to compute the performance metric between different registration methods.

II. Materials and methods

A. Dataset description

The training set consists of 4 cases (patients), taken from the Chronic obstructive pulmonary disease (COPD) dataset, where each case is composed of the following:

1) Inhale chest CT scan (raw)
2) Exhale chest CT scan (raw)
3) 300 Lung landmarks of the inhale CT scan
4) 300 Lung landmarks of the exhale CT scan

The landmarks between inhale and exhale images are matched and are provided in text files. Additionally, a table with information on the dataset is provided including image dimensions, voxel size, original landmarks displacement, and the target localization error (TLE).

The testing set consisted of 3 cases with the same characteristics described above for the training set, except for the 300 exhale landmarks, which are not provided.

B. Preprocessing

Given that all pairs of images were provided as raw data, a first preprocessing step was performed to:

- Change the orientation of the images to a visualization standard, according to the information given in the project presentation slides (pg. 5).
- Convert to NIFTI format.

This was achieved using ITK-Snap software. In a nutshell, to read raw binary data, the dimensions and the corresponding voxel spacing were filled up in the correct format. After that, reorientation was performed by selecting the *superior* to *inferior* setting in a tool embedded in ITK-Snap under the tool/reorient image section. Finally, the image is saved as a *nii.gz* format file.

Regarding the landmarks, the only preprocessing done was to create a copy of them in .pts format. This basically means that, inside the text document, in the first line, the number of points (in this case 300) is provided.

Additional preprocessing steps, namely lung segmentation and contrast enhancement, were also performed. Lung segmentation was achieved using two different methods: a manual segmentation using the *3D Slicer* [2] image visualization and processing tool, and an automatic segmentation using a pretrained UNet available as a library for Python [1].

1) Manual segmentation: In the 3D slicer, the segmentation editor tool was used, where a segmentation using the *seeds and grows* algorithm is implemented. For this, we manually initialized some seeds inside and outside of the lungs for all three main visualization planes (axial, sagittal, and coronal). Then, the algorithm is run and a segmentation is generated. We then visualize the segmentation and correct possible errors (mainly in low-contrast regions) using the tool brush and running the method again. This method took between 5-8 minutes per patient (two images), and the quality of the segmentation was highly dependent on the user's judgment.

2) Automatic segmentation: The *Automated lung segmentation in CT under the presence of severe pathologies* tool [1], available for Python, was used to automatize the segmentation process, speed up the segmentation pipeline, and make the output less dependant on the user experience or judgment. In a nutshell, the *lung mask* library was installed in our environment and executed using simple classes and methods in the Python API (See the Jupyter notebook *lung_segmentation.ipynb* in the source code for more information).

For our task of segmenting both lungs, two pre-trained models were available: *R231* and *R231CovidWeb*. The only difference between both models is that the latter uses additional positive COVID cases for the model training, making it more robust to changes of intensity inside of the lungs. We use this last model as it gave better results for our training set, as it will be seen in section III-E. The only preprocessing needed on the images is to transform their intensities to Hounsfield units (HU). Given that we did not possess the DICOM file of the images where the information on the transformation between the raw intensities and the HU units, we decided to use an approximation, where all pixel intensities are transformed following the function $f(x) = x - 1024$. This essentially means that all intensities were shifted by -1024.

With this model, the segmentation time was reduced to only 2 minutes per patient and was completely independent of the users' abilities to segment. Naturally, the automatic segmentation method was selected for the rest of the project. An example of this segmentation is presented in figure 1.

Finally, we also attempted to use different histogram-transforming functions from CV2, such as "normalize" or "equalizeHist", but we came across different issues regarding the SimpleITK Image metadata. Given that the main purpose of this preprocessing was to obtain a better segmentation, we decided that the results were already successful for this preprocessing.

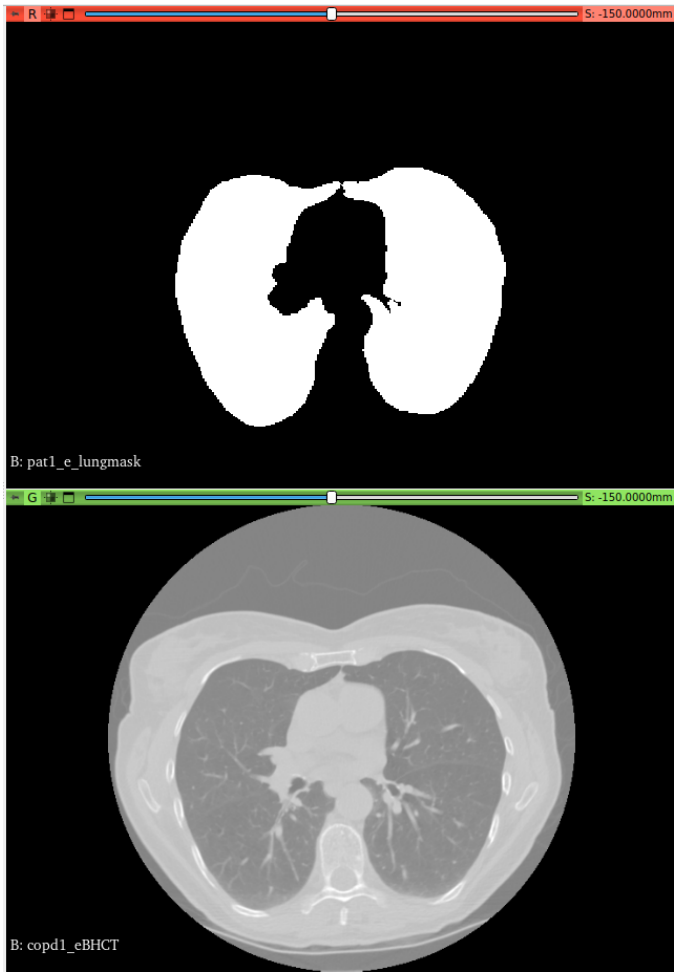

Fig. 1. Automatic segmentation example for patient 1.

C. Registration framework components

A basic image registration framework consists of 4 main components:

1) Transform: transformation applied on the moving image to warp it into the fixed image.
2) Metric: Used to measure the similarity between the moving and the fixed images.
3) Interpolator: Assigns intensity values to pixels outside of the pixel grid after the transformation is applied.
4) Optimizer: find the minimum (or maximum) for the metric using an optimization algorithm.

When building a registration method the task essentially consists of selecting the best parameters for the components listed above, given the task and the dataset provided. Among these main components, there exist additional parameters that will affect the registration pipeline (like the number of iterations, number of samples, etc.) that must be taken into consideration.

Given the central role played by this, the selection of parameters will be discussed with high detail in section III, where we will discuss the reasoning behind the selection of the parameters and the impact on the registration performance.

D. Performance metrics

Performance was measured in terms of the target registration error (TRE) using the inhale and exhale landmarks provided for each patient. The TRE is the distance after registration

between corresponding points, not used in calculating the registration transform. For each patient, we can compute the mean TRE and its standard deviation (STD). The smaller the TRE, the better the registration method, although the mean TRE is not expected to reach smaller values than the TLE, as this is considered the landmark placement error, and mean TRE values below this threshold cannot be trusted.

Considering that we will obtain a mean TRE value for each patient, in order to obtain a single number that we can use to compare several registration algorithms, the mean of all the three relevant metrics (TRE mean, TRE STD and execution time) across patients was computed per method. These values are the ones presented in the results section III.

III. Results and discussion: parameter selection

The registration parameters selection is the core part of this work, as the setting of these values defines the registration and, therefore, the final output and performance of the model. The tuning of these parameters can be considered as the training step of our pipeline to find the best model for our dataset.

For this reason, we decided to structure our report so that the parameter selection was fused with the results section so that the steps followed and the methodology behind them are more intuitive to the reader. Next, we present the experiment settings we performed, the motivation behind them, and the result they achieved. This will lead to a more comprehensive follow-up of the selection steps for the best registration parameters. Most of these parameters are reflected in a file called *the parameter file*, so the task can be summarized as finding the best settings for the parameter file. The final subsection III-F will focus more on the best registration pipeline, its results, and their implications.

Figure 2 shows the first pipeline proposed to find the best parameters. These steps will be discussed in the following subsections.

A. Registration tool

Given that during one of the last lectures of this course, we were introduced to *VoxelMorph* project for applying Deep Learning methods to a registration framework we decided to explore this field and also in parallel follow the procedures used in previous coursework. After finding some examples of brain registration we tried applying these methods to our case and obtained some very unsuccessful registrations. The dilemma at this point was whether to focus on this methodology and train our nets for better registration or proceed with traditional methods. Given that we were provided with enough data for training we decided to pursue more known methods and focus on tuning parameters in order to find the optimal registration framework.

We decided to use *Elastix* as registration software for this project given that we used it frequently during the current semester, which gave us more familiarity with its use and good practices. In addition, we decided to use the *Elastix* library for SITK called *SimpleElastix*. *SimpleElastix* has all the *Elastix* functions, incorporating them in the Python API as an easy-to-use library.

As *Elastix*, *SimpleElastix* registration is defined using a parameter file, a txt file containing the parameters to be used for the registration. In the next sub-sections, we will describe the selection method we followed and the importance of each parameter for the final registration.

B. Transform

We started by setting the registration transform, as we considered it to be the base of the registration algorithm. Even though the patient movement between the inhale and exhale images is clearly a non-rigid movement, we decided to explore also rigid transforms to compare speed, computational costs, and possible composition between transforms to improve the registration performance.

To start, we would like to make a distinction between the two main types of transformation:

1) Global: The same transform is applied to all pixels in the image.
2) Local: Each pixel may have a different transform applied to it (different deformation maps depending on the region).

Taking this in consideration, the combinations we explored were the following:

1) Affine (global)
2) B-spline (local)
3) Composition (Affine $\rightarrow$ B-spline)

For the global transform, we used the affine transform, as it is the second most general transform (after the projective one) among the global transforms (the projective transform is highly computationally expensive and is out of the scope of this course). It contains translation, rotation, scaling, shearing, and tearing.

As for the local transform, B-spline registration was chosen. It is a method of deformable registration that uses B-spline curves to define a continuous deformation field that maps each and every voxel in a moving image to a corresponding voxel within a fixed or reference image [3]. It is also a non-rigid transformation and, different from the affine transform, it is not a global transform.

Finally, a composition of transformations was also considered, consisting of using the registration algorithms in a Markov chain (one after the other). In this case, we registered the images first using the affine transform and then the BSpline registration was performed on the already warped image.

The registration algorithm was run using the default *Elastix* settings for affine and b-spline registration, respectively, with and without lung segmentation. Table I shows the performance of these transformations among all patients, where each column stands for the mean of the means, the mean of the standard deviations, and the mean of the time (in seconds).

Further steps were decided based on these results. First, we can notice that better results are obtained using the lung segmentation masks in all cases. Secondly, we can observe that

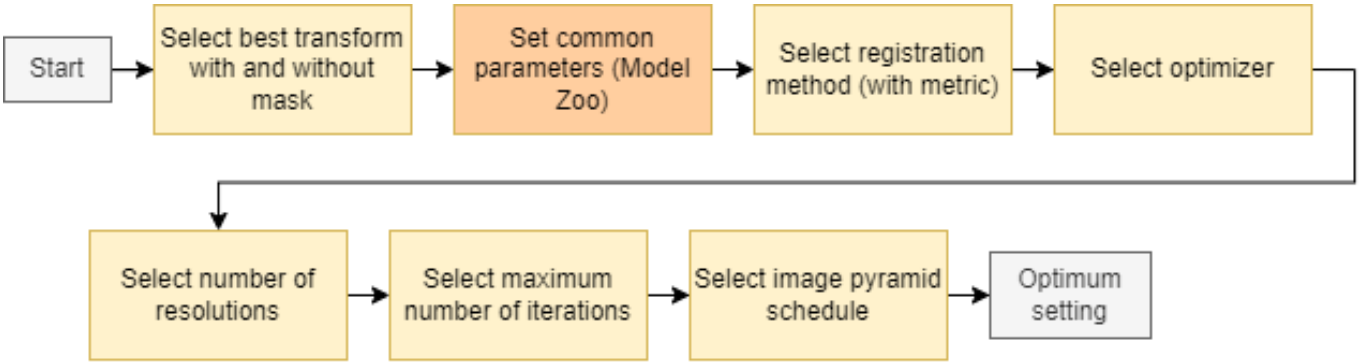

Fig. 2. First parameter searching pipeline.

the B-spline alone and affine + B-spline settings performed the best results giving very similar TRE values. We decided to focus on these two settings and investigate ways of improving their performance.

TABLE I
DEAFULT SETTING TRANSFORMATION PERFORMANCES. THE COLUMN RESULTS ARE MEANS OF MEAN, STD, AND TIME ACROSS ALL 4 PATIENTS.

Method	***Mean***	***STD***	***Time (s)***
Affine + Bspline with mask	6.75	6.94	228.86
B-spline with mask	6.80	6.99	215.83
Affine with mask	12.58	6.33	52.64
Affine + Bspline without mask	15.30	6.98	337.00
B-spline without mask	16.82	8.00	261.44
Affine without mask	20.34	7.64	55.04

C. Common parameter settings

In order to further explore the parameters and find the best settings for our dataset, we have got inspired by Elastix's model Zoo for different parameter settings. The studies of Par0004, Par0008, and Par0011 were performed on intra-patient 3d lung CT scan datasets. Because of the similarity between our projects, we compared and combined some settings from these files. The settings from these parameter files helped us to determine which parameter settings are common and different in 3D lung CT imaging studies.

The common settings for the given three parameter files are given in Table II below. Our parameter files were updated based on these settings.

TABLE II
COMMONLY USED PARAMETER SETTINGS FOR LUNG CT SCAN REGISTRATION.

Parameter	***Setting***
FixedImagePyramid	"FixedRecursiveImagePyramid"
MovingImagePyramid	"MovingRecursiveImagePyramid"
Interpolator	"BSplineInterpolator"
ResampleInterpolator	"FinalBSplineInterpolator"
Resampler	"DefaultResampler"

The common setting changes clearly showed an improvement in our results, as it can be observed in table III. Based on the updated results, the difference between using Affine + B-spline and only B-spline transformations became more acute. Therefore, from now on, we decided to keep the transformation with Affine + B-spline combination settings, although we kept in mind that the B-spline transform alone could still be used in advanced steps.

TABLE III
REGISTRATION PERFORMANCES WITH COMMON SETTINGS INCLUDED.

Method	***Mean***	***Mean of STD***	***Time***
Affine + Bspline with mask	3.80	4.55	210.26
B-spline with mask	6.57	6.79	195.74

D. Different parameter settings – 1st round of experimentations

In this section, we explored the parameter settings that were different among the Zoo examples and, therefore, could be tuned to our specific dataset. From them, the best parameter setting for our registration was established. Given a large number of possible combinations, we decided to use a decision tree-like approach and explore the parameters in a sequential as shown in figure 2. In this manner, we are giving more importance to some parameters than others, as the more important parameters will be explored first.

In table IV, all the different parameter settings are listed according to their type (registration, metric, etc.), sorted in order of importance according to our method. As such, we started with the registration method and the metric at the same time, as they are linked to each other.

1) Metric and registration method: In this section, we have performed 2 main experiments related to the registration method and the metric parameter:

- Setting 1: Registration "MultiMetricMultiResolutionRegistration" and Metric "AdvancedNormalizedCorrelation" "TransformBendingEnergyPenalty" setting from Par0011.

TABLE IV
DIFFERENTLY USED PARAMETER SETTINGS

Parameter type	***Setting***
Registration	"MultiResolutionRegistration" "MultiMetricMultiResolutionRegistration"
Metric	"AdvancedNormalizedCorrelation" "AdvancedMattesMutualInformation" "TransformBendingEnergyPenalty"
Optimizer	"StandardGradientDescent" "AdaptiveStochasticGradientDescent"
NumberOfResolutions	3, 4, 5
MaximumNumberOfIterations	256, 600, 2000
ImagePyramidSchedule	444333222111, 84421

- Setting 2 (Default): Registration "MultiResolutionRegistration" and Metric "AdvancedMattesMutualInformation" from the Par0008 file.

On one hand, *MultiMetricMultiResolutionRegistration* gives a framework for registration with a multi-resolution approach, using multiple metrics. On the other hand, the *MultiResolutionRegistration* uses the traditional multi-resolution technique that we explored in the course. As seen in table V the traditional multi-resolution setting gave the best performance.

TABLE V
METRIC BASED PERFORMANCES

Method	***Mean***	***Mean of Std's***	***Time***
Setting 2	3.80	4.55	210.26
Setting 1	4.46	3.98	268.96

2) Optimizer: Another key registration component is the optimizer. We selected between two options:

- Setting 1 (Default): Optimizer "AdaptiveStochasticGradientDescent"
- Setting 2: Optimizer "StandardGradientDescent"

The only difference between the standard and adaptive SGD is that the latter uses an adaptive gain. This optimizer is recommended by *Elastix* to be used in combination with the Random image sampler, or with the RandomCoordinate image sampler.

As table VI shows, the default setting, i.e. the Adaptive SGD, was selected given its superior performance.

TABLE VI
OPTIMIZER-BASED PERFORMANCES.

Method	***Mean***	***Mean of Std's***	***Time***
Setting 1	3.80	4.55	210.26
Setting 2	7.81	4.81	202.60

3) Number of Resolutions: The number of resolutions used for the multi-resolution technique may affect significantly the registration performance. Nevertheless, as seen in the labs, the number of registration is subjective and must be defined based on experimentation. Given the default value of 3, we decided to increase the number of registration by the unit, exploring 4 and 5 as a number of resolutions:

- Setting 1: 3
- Setting 2 (Default): 4
- Setting 3: 5

TABLE VII
NUMBER OF RESOLUTIONS-BASED PERFORMANCES

Method	***Mean***	***Mean of Std's***	***Time***
Setting 3	3.58	4.13	218.27
Setting 2	3.80	4.55	210.26
Setting 1	15.28	8.00	198.70

As shown in table VII, we can observe that an increased number of resolutions, in this case, 5, performed better. A natural next step would be to increase the number of resolutions, as it seems to be that the larger the number, the better the performance. Nevertheless, given that the improvement between 4 and 5 resolutions is small, and that the running time increased, we decided to keep this number of resolutions as maximum.

4) Maximum number of iterations: As in any iterative method, the number of iterations plays a crucial role in the performance and running time of the method. In this case, we decided to check the direct performance changes based on the selective increase in the number of iterations.

- Setting 1 (Default): 256
- Setting 2: 600
- Setting 3: 2000

TABLE VIII
MAXIMUM NUMBER OF ITERATIONS-BASED PERFORMANCES

Method	***Mean***	***Mean of Std's***	***Time***
Setting 3	2.809	3.517	574.960
Setting 2	3.724	4.891	283.637
Setting 1	3.80	4.55	210.26

In table VIII, we can observe that, as expected, the larger the number of iterations, the better the performance. In our selected settings, setting 3 meaning 2000 iterations performed the best with the downside of taking a lot of time (approximately 40 minutes). Moreover, the *Elastix* documentation establishes that the optimal number has been observed to be 2000 for most of the registration pipelines explored by them. However, it is suggested to use 1000 iterations for experiments and leave 2000 only for the final registration attempt. So for the other experiments, 1000 iterations will be used, but the final registration will be performed using 2000 iterations (or more if the improvement is observed to be increasing).

5) Image pyramid schedule: Finally, for the multi-resolution settings, the image pyramid schedule defines the downsampling factors for fixed and moving image pyramids. With that in mind, we explored using the different settings used in the zoo, giving us two main settings:

- Setting 1: 4 4 4 3 3 3 2 2 2 1 1 1
- Setting 2: 8 4 4 1

As stated in table IX we can observe that both schedules gave the same result, just varying in the execution time. This

TABLE IX
IMAGE PYRAMID SCHEDULE-BASED PERFORMANCES

Method	***Mean***	***Mean of Std's***	***Time***
Setting 1	2.73	3.16	348.16
Setting 2	2.73	3.16	507.46

TABLE X
FIRST OPTIMAL PARAMETER SETTINGS

Parameter	***Setting***
FixedImagePyramid	"FixedRecursiveImagePyramid"
MovingImagePyramid	"MovingRecursiveImagePyramid"
Interpolator	"BSplineInterpolator"
ResampleInterpolator	"FinalBSplineInterpolator"
Resampler	"DefaultResampler"
Registration	"MultiResolutionRegistration"
Metric	"AdvancedMattesMutualInformation"
Optimizer	"AdaptiveStochasticGradientDescent"
NumberOfResolutions	5
MaximumNumberOfIterations	1000
ImagePyramidSchedule	444333222111

result made us think that the settings given by the zoo may not be optimal given the changes we made across the search for the best parameters. As such, we decided to make a second parameter search, more personalised for specific parameters that needed to be re-checked given the overall changes in the parameter file, based on the parameters we found so far (table X).

E. Different parameter settings – 2nd round of experimentations

Going deeper into *Elastix* documentation allowed us to spot some parameters that were not changed previously and that needed to be tuned according to the changes we made in the first round of experimentation. Also, there was a pair of parameters that were never explored by us and that could boost the final performance. Table XI summarizes the new parameters we explored.

TABLE XI
PARAMETERS FOR THE SECOND ROUND OF EXPERIMENTATIONS.

Parameter type	***Setting***
Transform	Affine Composed (Affine + Bspline)
Pyramid Schedule	4 4 4 3 3 3 2 2 2 1 1 1 16 16 4 8 8 3 4 4 2 2 2 1 1 1 1
Number of spatial samples	2048 5000 10000
Sampling attempts	8 50
SP_A	50 80
Final grid spacing	16 16 16 16 16 4

First, we decided to check if, after all the changes made in the registration pipeline, the composed transform was still the one with the best performance. We decided to check this because considering that the patient movement was basically purely non-local, starting the registration process with an affine (non-local) transform should not improve the registration performance a lot, yet taking a considerable extra amount of time. As it will be seen in the results figure 3, both transformation cases perform similarly, as expected.

As for the pyramid schedule, we should remember that this parameter was already changed and checked in the first round of experimentations. Nevertheless, we missed taking into account key considerations. First, this setting is important for the given dataset, as the images do not have equal pixel sizes and the number of slices. Although the exact number varies per image, the pixel size of the images was around 0.6 mm for the x and y coordinates, and 2.5 mm for the z coordinate. If we compute the ratio between these two sizes, we obtain 4.166 units, meaning circa a 1:4 correspondence between coordinates. This means that the pyramid schedule should be changed according to this ratio to keep similar information between coordinate directions. With this new consideration, we proposed a tailored schedule that considers the pixel size differences, starting with the (16 16 4) factor and gradually reducing these factors until arriving at the original image resolution.

The number of spatial samples defines the number of image voxels used for computing the metric value and its derivative in each iteration. This parameter changes considerably the performance as a larger number of samples makes the resolution more robust but, at the same time, more time-consuming. We decided to increase the number of samples from the one we had before, to the *Elastix* default (5000) to a considerably large number (10000).

Sampling attempts define the maximum number of sampling attempts, as sometimes not enough corresponding samples are used. We can anticipate that in our case this parameter does not change our results at all.

The SP_A parameter is a tricky one. It is a parameter of the Adaptive SGD and, according to the documentation, its definition is completely dependent of the application and the dataset. For this, its value must be tuned around the default value. We decided only to check the impact of this number increased from the default 50 to 80.

Finally, the final grid spacing voxel parameter was changed. It is a parameter of the b-spline transform and controls the grid spacing of the b-spline transform for each dimension. Following a similar to the one presented for the pyramid schedule, the grid spacing was changed from (16 16 16) to (16 16 4), to take into account the pixel size difference between dimensions.

To summarize all of these experiments, we decided to present them in a box plot, as shown in figure 3. Among all the results shown, we highlight the following findings. First, just by changing the pyramid schedule all the models outperformed our best finding from the first round (first line, green). Second, all the best six methods (at the bottom) that can be seen in figure 3 use almost the same parameter file and are very close to each other in performance. The difference between them

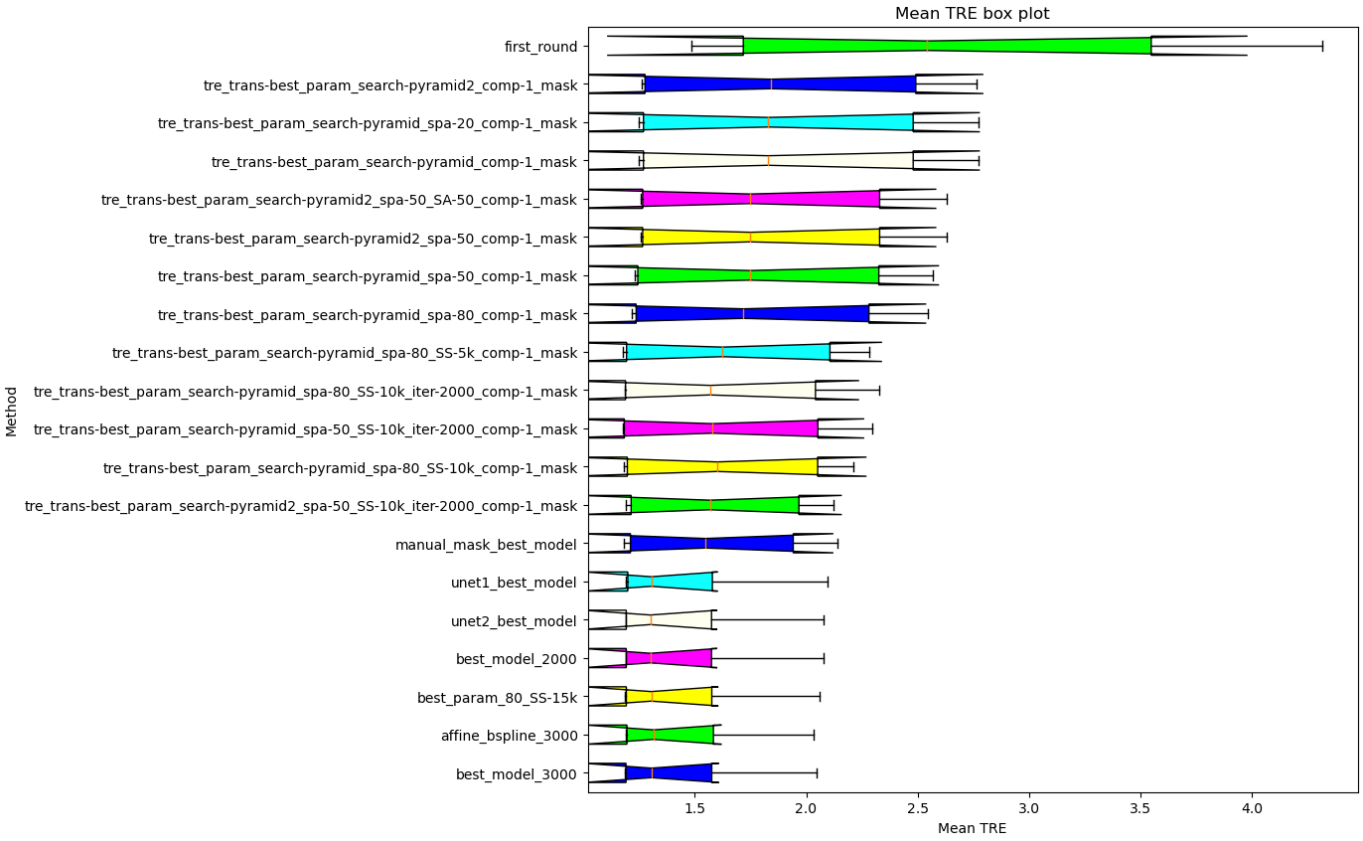

Fig. 3. Second round of parameter searching results. The metric shown is the mean of the mean TRE across all 4 patients.

TABLE XII
TRE VALUES ACROSS PATIENTS FOR EACH MODEL IN THE SECOND ROUND OF EXPERIMENTATIONS, FROM BEST TRE TO WORST.

Method	pat mean	pat std	time
best_model_3000	1.4613	1.1745	288.2448
affine_bspline_3000	1.4635	1.1829	627.9734
best_param_80_SS-15k	1.4642	1.1887	269.1705
best_model_2000	1.4667	1.1953	204.7159
unet2_best_model	1.4667	1.1953	206.2431
unet1_best_model	1.4729	1.2107	200.3660
manual_mask_best_model	1.6044	1.4144	195.6547
tre_trans-best_param_search-pyramid2_spa-50_SS-10k_iter-2000_comp-1_mask	1.6124	1.4318	200.3104
tre_trans-best_param_search-pyramid_spa-80_SS-10k_comp-1_mask	1.6484	1.5287	127.8192
tre_trans-best_param_search-pyramid_spa-50_SS-10k_iter-2000_comp-1_mask	1.6572	1.5645	212.8079
tre_trans-best_param_search-pyramid_spa-80_SS-10k_iter-2000_comp-1_mask	1.6622	1.5601	212.9988
tre_trans-best_param_search-pyramid_spa-80_SS-5k_comp-1_mask	1.6763	1.5572	95.3161
tre_trans-best_param_search-pyramid_spa-80_comp-1_mask	1.7996	1.7706	75.2703
tre_trans-best_param_search-pyramid_spa-50_comp-1_mask	1.8227	1.7995	76.9016
tre_trans-best_param_search-pyramid2_spa-50_comp-1_mask	1.8462	1.8932	61.8877
tre_trans-best_param_search-pyramid2_spa-50_SA-50_comp-1_mask	1.8462	1.8932	63.2591
tre_trans-best_param_search-pyramid_comp-1_mask	1.9206	1.9245	87.4694
tre_trans-best_param_search-pyramid_spa-20_comp-1_mask	1.9206	1.9245	76.6288
tre_trans-best_param_search-pyramid2_comp-1_mask	1.9272	1.9706	61.9890
first round	2.7225	3.1193	70.1546

are final decisions in key parameters:

1) The segmentation mask used (UNet1 or Unet2 model as seen in section II).
2) The transformation (composed affine + b-spline or pure b-spline)
3) The number of spatial samples
4) The number of iterations (going from 2000 to 3000).

Given that all models present very close results, it is

worth analyzing the execution time and the standard deviation. Table XII shows the mean values for mean, STD, and time across patients for the main models of the second round. This table shows how the best 6 models differ in execution time considerably. For instance, the best two models, the pure bspline and composed affine + bspline do not differ a lot in mean value but the execution time for the latter is 2 times slower.

Another interesting thing to consider is the presence of an important outlier for all the best 6 cases. As seen by the shape of the box plots, there was always one patient mean TRE above 2.0, meaning that no registration method could reduce its error below this threshold for that particular patient. As this will be seen in the next final subsection, this result is coming from the second patient.

F. Final result discussion

TABLE XIII
TRE VALUES FOR THE BEST REGISTRATION PIPELINE.

Patient	mean TRE	std TRE	Time
1	1.1869	0.7739	284.9377
2	2.0471	2.4054	286.6929
3	1.1923	0.7494	295.5550
4	1.4189	0.7693	285.7934

As a final note, we present the results and the parameters for the final best parameter file in tables XIII and XIV, respectively.

As mentioned before, we achieved low TRE mean values, below 1.5 mm, for 3 of the patients. Also, their STDs are all around 0.7 mm, making the distribution of the TRE distances narrow and close to the mean. On the other hand, patient 2 can be considered an outlier as its mean was above 2.0 mm and its STD was even larger with 2.4 mm, giving the sense of a wide distribution for the distances of the points.

One of the reasons for this result could be that the images for patient 2 have fewer slices in the z direction. Also, a detailed observation of the exhale slices makes us notice a decrease in the intensity contrast between the lung and the exterior, making it possible for the registration method to fail to determine the border between background and foreground.

Therefore, an improvement to this pipeline could be the implementation of a workable intensity normalization and contrast enhancement, to standardize inhomogeneous intensities coming from different vendor machines and to enhance intensity changes in outlier cases, like patient 2.

Still, one of the strengths of this pipeline is the speed and high performance it can achieve. Considering the overall time per patient presented in table XIII and the segmentation time, a complete registration takes around 7-8 minutes.

G. Challenge results

During the challenge day, 3 new patients were provided, with the same characteristics as the training set, except only for the absence of the exhale landmark text file, used only by the referees to compute the performances.

TABLE XIV
FINAL OPTIMAL PARAMETER SETTINGS

Parameter	***Setting***
FixedImagePyramid	"FixedRecursiveImagePyramid"
MovingImagePyramid	"MovingRecursiveImagePyramid"
Interpolator	"BSplineInterpolator"
ResampleInterpolator	"FinalBSplineInterpolator"
Resampler	"DefaultResampler"
Registration	"MultiResolutionRegistration"
Metric	"AdvancedMattesMutualInformation"
Optimizer	"AdaptiveStochasticGradientDescent"
NumberOfResolutions	5
MaximumNumberOfIterations	3000
ImagePyramidSchedule	16 16 4 8 8 3 4 4 2 2 2 1 1 1 1
Number of spatial samples	10000
Sampling attempts	8
SP_A	80
Final grid spacing	16 16 4

Regarding execution time, the segmentation pipeline took 4 minutes and 42 seconds to segment all 3 patients and output the corresponding *nii.gz* files. The registration pipeline took 13 minutes and 21 seconds for all 3 patients, with the inhale landmarks transformed into the exhale space as output.

Our results ranked 2nd place during the challenge day, standing for its good performance among other teams. Additionally, we were the second group to upload the results, proving that our pipeline was fast and reliable. We obtained a mean TRE value across patients of $1,4223$ mm, where the best team obtained $1,3278$ mm, making the difference between models minimal.

IV. CONCLUSION

We built a CT scan lung registration pipeline capable of registering inhale and exhale images of the same patient with high reliability and speed. Our method reached a mean TRE of 1.46 mm among patients of the validation set, a mean TRE of $1,4223$ mm among the challenge patients (test set), and an average time of 8 minutes for registration per patient.

We achieved this by using parameter searching and tuning experimenting with settings tried by other members of the community for lung CT scan registration as starting point. By using this established literature, we were able to create a baseline for our performance, which later was outperformed by more deep and specialized experiments that eventually led us to an optimized parameter file.

Image preprocessing steps were crucial for obtaining high performances, giving special care to the lung segmentation method. For this, we used an automatic segmentation tool available for the Python API, which uses a pre-trained UNet.

REFERENCES

[1] Johannes Hofmanninger, Forian Prayer, Jeanny Pan, Sebastian Röhrich, Helmut Prosch, and Georg Langs. Automatic lung segmentation in routine imaging is primarily a data diversity problem, not a methodology problem. *Eur Radiol Exp*, 4(1):50, December 2020.

[2] Steve Pieper, Jean Christophe Fillion-Robin, and Ron Kikinis. 3D Slicer, 2015. Pages: 3-7 Issue: 3.

[3] James Shackleford, Nagarajan Kandasamy, and Gregory Sharp. *High Performance Deformable Image Registration Algorithms for Manycore Processors*. Morgan Kaufmann, 2014.