UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Web Application for Image Registration

Using elastix

Dissertation presented by
Jacob ELIAT-ELIAT , Jonathan LEGAT

for obtaining the Master's degree in
Computer Science

Supervisor
Benoît MACQ

Reader(s)
Peter VAN ROY, Ives DEVILLE

Academic year 2017-2018

Acknowledgements

We would like to thank Vincent Nicolas for his feedback and insight with the development of the web application,
Benoît Macq for his support during the creation of this thesis,

Celine Chapeaux, Olivier Legat for their feedback on the thesis,

Anthony Dechamps, Aurian De Potter for their insight on Django

and our friends and family for their moral support.

Contents

Chapter 1

Introduction

Medical Image analysis and in particular image registration is a field which has experienced a massive amount of efforts and research. We were tasked with improving the current solutions for registration, which is a hard task considering the high quality of the tools that already exist.

During the creation of this thesis, the objective changed many times because we wanted to create something useful and enjoyed creative freedom. In order to find our definitive objective, we inspired ourselves from the "design thinking processes". This process tries to foster innovation and creative thinking.

We hope that our result can be used in image registration or that it will motivate further development into user-friendliness for registration tools.

In the chapter **"Registration"**, we will explain what image registration consists of, its uses, sub-fields and quirks.

We explored the **"State of the Art"** and realized that many solutions already exist for image registration, but while some of them try to be user-friendly, most of their intended audience is users that are very much familiar with the fields of medical image registration and computer science.

In the chapter **"Our Solution"**, we then explain the direction we took for improving the current tools, which is to make a web application with many features intended to improve the users' experience, we will also describe the software we developed.

Then, in the chapter on **"Process"**, we will explain how we got there, from the initial idea of making a web application to the development process itself. We pivoted a lot from our initial thesis subject which was almost unrelated except for being in the same field, we used design thinking and interacted with people in the field including the CEO of *Intuitim* to find a problem that can be solved.

We explain the different technologies we used, from web development tools to registration toolboxes in the chapter **"Technologies"**.

We will then discuss the details of implementation in the chapters **"Architecture"** and **"Tests"**.

Our project, while a complete solution, could use a few additional features. The improvements that could be done are described in the chapter **"Possible direction"**.

Chapter 2

Registration

"Registration is the determination of a geometrical transformation that aligns points in one view of an object with corresponding points in another view of that object or another object."[11]

In the medical imaging world specifically, it mostly refers to finding a transformation that aligns several 3D scans (referred as views in the above definition) of some anatomical regions so that points in one image correspond to points in the second one.

The transformation is applied to an image called the **Moving Image**$I_M(x)$, in order to align it to another image called the **Fixed Image** $I_F(x)$.

More formally, the goal of registration is to find a coordinate transformation $T(x)$ that makes $I_M(T(x))$ spatially aligned with $I_F(x)$.

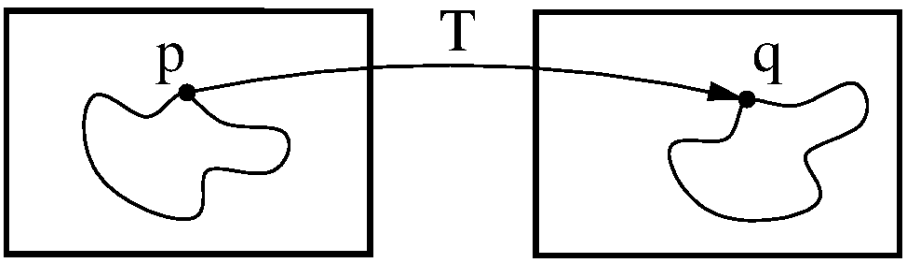

Figure 2.1: An illustration of the coordinate transformation function in image registration. source: *the Simple Elastix documentation* `http://simpleelastix.readthedocs.io/_images/ImageRegistrationConcept.png`

2.1 Applications Of Image Registration

Registration is useful in many fields and in many cases.
The three major research areas are[13]:

- Computer Vision and Pattern Recognition
- Medical Image Analysis
- Remotely Sensed Data Processing (eg. satellite images)

But there are many more fields where medical image registration can be used such as:

- Robotics and automatic inspection
- Computer-Aided Design and manufacturing
- Astronomy
- etc..

Building a generic solution that can be used in every field is tempting. However, the problems and solutions differ a lot and it makes solving all of them with a single software/algorithm very impractical. The algorithms used, the image type and format (DICOM images are almost only used in medical settings for example), the "dimensionality" and the goal of registration vary greatly between these fields.

Due to these large variations in needs and techniques between the fields, our thesis will focus mainly on medical image analysis. This is because we were in contact with someone in the industry who could inform us about the needs that should be addressed.

Medical image analysis is already a subset of what is possible with registration, but within it, there exist many applications where registration is useful, such as[14]:

- fusion of anatomical images
- computer-aided diagnosis and disease following-up
- evaluation of surgical and radiotherapeutic procedures
- surgery simulation
- atlas building and comparison
- radiation therapy
- assisted/guided surgery
- anatomy segmentation
- and many more.

Some of these applications require specific algorithms, but there exist tools built to respond to the needs of many of these sub-fields.

2.2 Different types of registration

Registration varies depending on the method, and the types of images that are being registered. As said above, our thesis is focused on medical image registration. Therefore, this classification will be geared towards medical image registration and not registration in general which should be categorized differently.

There are many ways of classifying registration algorithms and not a single standard one[14]. To describe them, we will use the nine fundamental criterion presented in *A survey of medical image registration* by Maintz and Viergever.[15]

2.2.1 Dimensionality

"Image dimensionality" refers to the number of dimensions in space that the image has.

For medical image registration, it can be anywhere between 2D and 4D, where the fourth "dimension" is time. It is entirely possible to do registration on 2D images or 4D images (3D+temporal) and some tools we use in this thesis can be used for it. In medical image registration, 2D images are sometimes used, (x-ray for example), and some registration algorithms have been specifically created for use with 2D, for example by using perspective projection.[15]

Medical images can also be "4D", which refers to 3D+time. An example is a series of scans of someone breathing. The motion is very important to the registration and depends on time, therefore it might be useful to do registration on scans of two people breathing, and try to align them in time as well as space. Additionally, images can be registered to a different dimension. A 3D image can be registered with a 2D image, not all points in the 3D image will have a corresponding point in the 2D image, but all points in the 2D image will be registered.

In our thesis, we focused mainly on 3D image registration, which is the main type of image that needs to be registered. 3D images are usually scans and they can be created with MRI, CT, and other machines usually involving radio waves.

2.2.2 Nature of registration basis

The registration basis can be intrinsic or extrinsic.

Extrinsic means that registration is based on a foreign "artificial" object introduced in the images that are designed to be "well visible and accurately detectable."[15]

On the other hand, intrinsic registration methods are based on the image itself, be it landmarks, segments (for this method, the images are segmented beforehand), or the properties of the voxels/pixels themselves, for example, the voxels' intensities.

2.2.3 Nature of the transformation

The type of transformation has a large impact on the end result, and different types of transformations are used for different purposes. Let's present some of the commonly used transformations:

Linear transformations

Linear transformations preserve importantly co-linearity and parallelism.[15] Co-linearity means that if three points are on a line before the transformation they are still on a line afterward. Linear transformations are defined by the degrees of freedom the transformation has. On a 3D object, it goes up to 12 : three for translation, three for scaling, three for rotation and three for shearing. A notable advantage of linear transformations is that they can easily be represented as a matrix equation.

Rigid transformations only use translations and rotations.

Rigid transformations have therefore up to 6 degrees of freedom (three for translation and three for rotations) in a 3D image.

They are better for some purpose because they don't introduce deformations in the image and therefore the volumes are preserved. Furthermore, there is less room for error since there are fewer degrees of freedom.

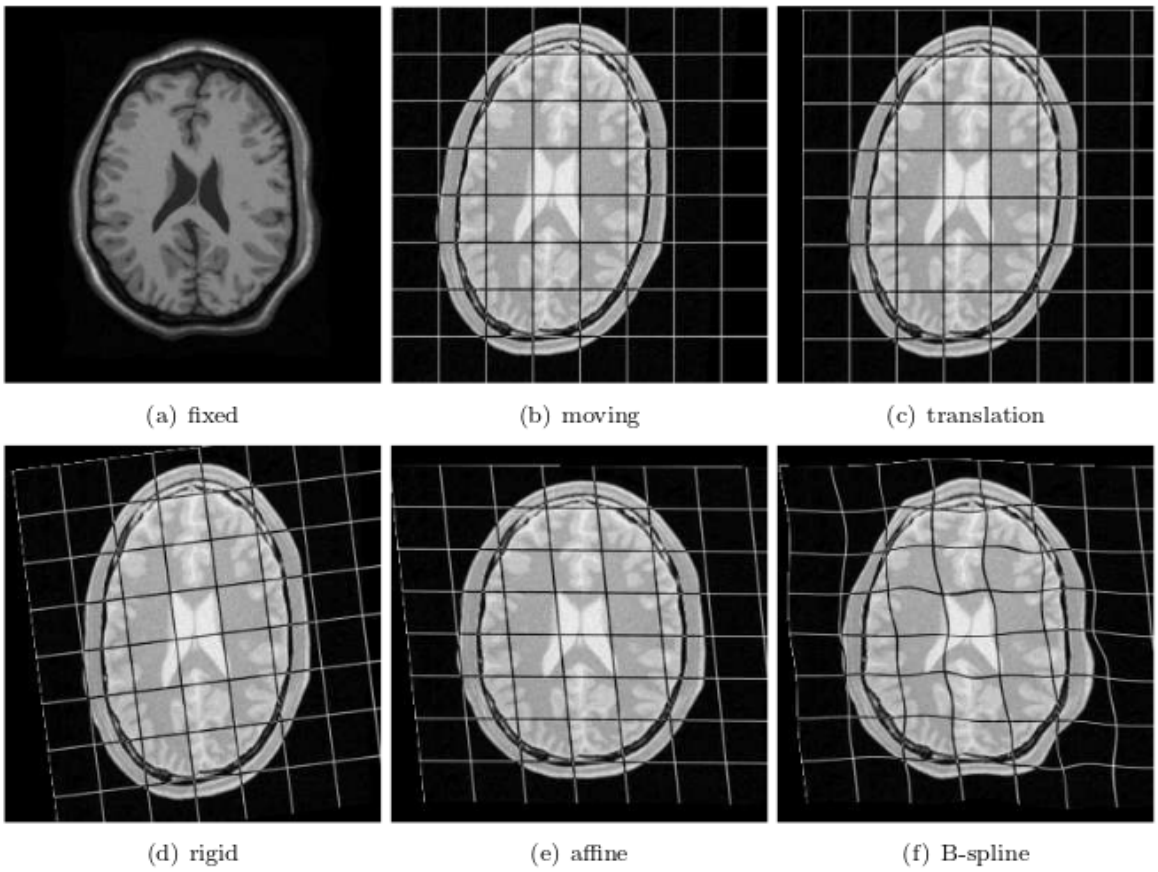

Figure 2.2: Different types of transformations that can be applied in the Elastix toolbox, source: *the elastix manual*[16]

Due to the aforementioned smaller room of error. Rigid registrations are also often used as an initialization for more complex transformations. This allows to get a coarse global alignment of the entire anatomy. without introducing too many errors.[51]

Affine registration is another type of linear transformation.

Affine registration has 12 degrees of freedom on a 3D image, it can do translations, scaling, rotation and shearing.[16]

It's mainly used when the internal structure of the image doesn't show distortion or deformations.

Non-linear transformations

There are many non-linear transformations. Since unlike for linear transformations, there is no restriction in the type of transformation to be applied, there is no simple mathematical representation that works for every non-linear transformation like there was for linear transformations using a matrix equation.

Some can be represented as a polynomial transformation in term of the old coordinates, while other can only be represented by local vector displacement, sometimes one per voxel.[15]

B-spline transformations are a commonly used type of polynomial transformation.

They are based on B-splines functions which are linear combinations of basis functions, which are piecewise polynomials (meaning they are composed by several polynomial functions one after

the other.) These functions can be used to define a vector field that is used as a transformation.

2.2.4 Domain of the transformation

"A transformation is called global if it applies to the entire image and local if subsections of the image each have their own transformations defined"[15]

2.2.5 Interaction

There are three levels of interactions between the software and the user:

- **interactive** where the user does the registration himself with some assistance from tools.
- **semi-automatic**, where the user provides pieces of information to the tool such as an initialization or where the user *steers* the algorithm in the right direction of optimization
- **automatic** in which the only thing the users chooses is the algorithms used and some meta-parameters.

2.2.6 Optimization procedure

The parameters can either be **computed** from all the available data or **searched** in their domain.[15]

The latter is the most frequent because it is rare that the parameters can be directly computed, but since the parameters are searched for, it is possible that the solution will not be the global optimum depending on the optimization algorithm.

2.2.7 Modalities involved in the registration

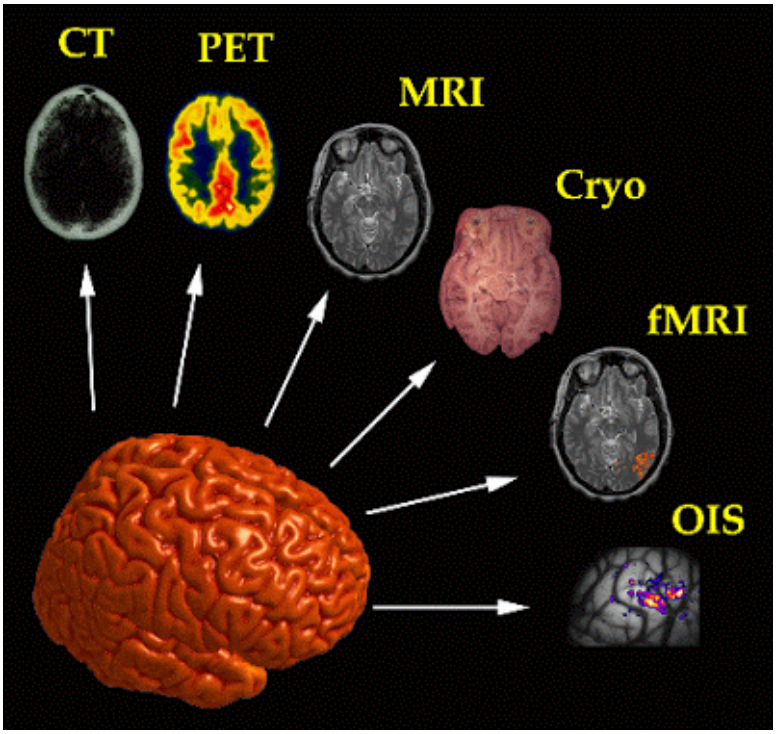

Figure 2.3: Different modalities of a brain image, taken from `socialfiction.org`

"Modality" refers to the way (modality) in which the image was acquired.

This distinction is mostly exclusive to medical image registration, in which images are acquired using various processes (ex: CT scans, PET scans, X-ray, MRI, etc...). These modalities lead to differences in the images even if the subjects are exactly the same. The resolution might be different, the range of detail captured might differ, the intensity might not be the same (for example bones could appear white using some techniques and grey using another).

The biggest impact of the different modalities is on the range of algorithm, metrics, etc. that can be used depending on if the registration is "monomodal" (both images have been captured using the same modality) or "multimodal".

Some assumptions that can be used in monomodal registration cannot be used in multimodal registration. For example, you could assume in monomodal registration that parts of the fixed image that have the same intensity of grey than part of the moving image are likely to be the same. However, such an assumption is not valid in multimodal registration.

2.2.8 Subject

The subject can either be the same for both the fixed image and the moving image in which case the registration is said to be **intrasubject**, or a different person in which case the registration is **intersubject**.[15]

2.2.9 Object

In medical image registration, the object refers to the body part that is being registered.[15]

There are algorithms that are optimized depending on the body parts because the deformations experienced are different depending on the body part. For example, the boundary of the brain is unlikely to vary a lot from a scan to another, while the boundary of the lung is likely to experience large variations.

2.2.10 Other categorizations of registration

Beyond the 9 categories explained above there are other categorizations that can be made:

Intensity-based registration use pixel intensities as a basis for registration.[16]

Feature-based registration use features (ex: the boundary of the object or a set of easily identifiable points).[16]

Parametric transformations can be defined using a parametric transformation function while non-parametric cannot, and instead define a vector for each pixel/voxel.[16]

The type of metric used in the optimization process is also very important. The three main types are **Mean Square Error** based which uses the mean square error as a basis, **Correlation-based** and finally, **Mutual Information** based. The latest category is particularly useful when it comes to multimodal images which might have a different intensity for the same type of voxels. It's a measure of how well the signal (here the intensity) of the second image can be predicted by the one from the first image.[52]

It should be mentioned that some algorithms seek an additional property of the transformation function beyond registration e.g. diffeomorphism.[47]

2.3 Example of registration process: Elastix

Elastix is a registration toolbox. It is the toolbox we use in this thesis. Therefore, we will present it from a more practical point of view in the chapter where we clarify the tools we used 6, but in this chapter, we will introduce the types of registration it is capable of doing and how the registration process is implemented.

This section is paraphrased from *The elastix manual*[16], the paper accompanying elastix [9] and other online resources by elastix such as the generated parameter documentation.

Elastix has a modular design which makes it easy to differentiate between the different parts of the registration algorithm. Elastix performs Intensity-based parametric registration (it can also use features to improve accuracy) in a fully automated manner.

The elastix's registration algorithm consists in trying to optimize a metric by modifying the parameters of the transformation. This is repeated several times using "image pyramids" and/or using different types of registrations.

Using "Image Pyramids" essentially means using different version of the same image, starting with a down-sampled and/or blurred version and increasing the quality, in order to first "hide" the details and reveal them little by little as the algorithm has already registered the "big picture" and can now focus on the details.

2.3.1 The algorithm step by step

When starting registration, Elastix will[9]:

- Step 1: The i-th image pyramid is initialized: moving and fixed image are down-sampled and/or smoothed.
- Step 2: Sample the fixed image. This will result in an image, possibly of lower resolution. Masks can be used here to restrict the regions that impact the registration.
- Step 3: Interpolate the moving image. This allows the image to be continuous: every point in the 3D image now has a value.
- Step 4: Calculate the value of the metric with an initial registration.
- Step 5: Find what to change to improve the metric using an optimization algorithm. (since elastix only does parametric registration, "what to change" means which parameters to increase or decrease and how much.)
- Step 6: Repeat steps 4 and 5 until the maximum number of iterations has been reached.
- Step 7: Change the level in the image pyramid and repeat everything from step 1, until the lowest level of the image pyramid is reached (the level with the highest resolution and lowest amount of smoothing).
- Step 8: The registration is now complete, apply it to the moving image, for this, the moving image needs to be interpolated first using the "resampleInterpolator".
- Step 9: Use the "Resampler" to generate the final image.
- The registration is now complete.

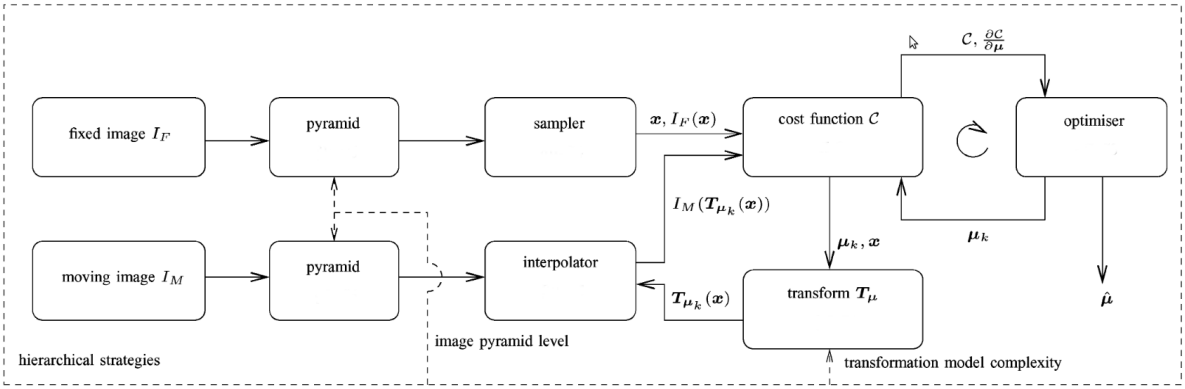

Figure 2.4: The elastix algorithm. Source: *elastix: A Toolbox for Intensity-Based Medical Image Registration*[9]

2.3.2 Components

This algorithm requires many modules, which each perform a specific part of the registration process.

The elastix modules are separated into several types of components, though some are very similar (fixedImagePyramid and movingImagePyramid are often the same algorithms, and ResampleInterpolator and Interpolator both do the same thing, just not during the same phase of the registration process.)

In this section, we will describe the main components and explain their role in the process. In addition, we will present a few examples of modules that can fill the role of these components.

ImageSamplers/Resampler

The purpose of this module is to diminish the size of the fixed image. If an image is too big, registration might take a long time, using an image sampler allows elastix to reduce the time the process takes.

The main options are "Full", which doesn't do sampling at all, "Grid" and "Random", which select the subset of voxels on which to do the sampling as their name describe. [9]

Resampler works the same way as the sampler and the options are the same, but its use is different. It is to generate the final image.

Interpolators/ResampleInterpolators

Certain phases of registration are done on a continuous image (optimization), yet image formats generally store voxels (example: CT scans stored in DICOM), which aren't continuous.

Interpolators are what allows elastix to go from voxels to a continuous image. The best option according to the authors of the manual[16](Stefan Klein & Marius Staring) is 1st order B-spline, which is the same as a linear interpolation (a linear interpolation calculates the weighted average of surrounding voxels, where the weight is the distance between the voxels and the point). The reason for the 1st order is to gain speed and memory, if speed is not an issue, a higher order is better.

ResampleInterpolators are used to get the final image once the optimization is finished and the transformation is calculated, here it is better to use a higher order B-spline since it's only done once, speed and memory are much less of an issue. The author recommends 3 as the B-spline order.[16]

Transforms

Transforms define the function that is used to "transform" an image into the other. There are many options and the choice depends mostly on the type of problem encountered.

For example, brains don't tend to have much deformation, therefore, translation, scaling and rotation are the only transformations needed. Depending on the mode of acquisition, scaling might not even be needed.

On the contrary for organs such as lungs, the deformations within the subject are much more important, and a more complex transformation is needed. (B-spline or thin-plate B-spline).

Metrics And Optimizers

During the registration process, the goal is to obtain a transformation which aligns the moving image with the fixed image. To do that, the algorithm tries to optimize on a metric which should represent how good the transformation was. For this part of the process, two things are needed: A metric and an optimizer.[9]

The optimizers are strategies to try to find the global minima of the metrics. For example, "Fullsearch" searches all the possibilities and chooses the best, while "GradientDescent" Calculates the direction of the local minima and searches for it. Optimizers should be chosen mostly depending on requirements in memory/processing power.

Metrics are what the optimizer tries to optimize. Ideally, a metric is at 1 when both images are perfectly registered. It gets lower the bigger the "distance" between the perfect registration and the current result is. Some metrics use "mutual information" which means they also use information from the moving image.

Some metrics also include a penalty for deformation that are too unlikely. If the deformation is too extreme any image can end up similar to any other image, but it won't be properly registered. For example, a very big deformation could probably morph a heart into a lung. However, what we want is to have the lung registered with the lung and the heart with the heart. Where each part of the lung/heart of the moving image registered with the corresponding part of the fixed image.

Registrations

The "Registration" component defines the framework the user wants to work on, it can either be multi-resolution or not, and multi-metric or not.[16]

All registrations that are proposed by elastix by default use a multi-resolution strategy: first, they do registration on lower resolution (or smoothed) images, then move to higher and higher resolution. The goal is to first align more general shapes before trying to align the details. It is possible, but not advisable, to put the number of resolutions to 1.

Some use multi-metric registration with different metrics depending on the pyramid level. It's also possible to use several metrics at each resolution level and weight them. Some metrics perform better at higher resolution, or lower smoothing and some at lower resolution/higher

smoothing, therefore, using weighted metrics can improve the performance of registration.

Obviously one has to understand the metric and their effects to fully use a more complex strategy such as multi-metric multi-resolution registration.

ImagePyramids

The multi-resolution strategies described earlier are defined with the ImagePyramids. There are ImagePyramids both for the fixed and moving image and they can be different.

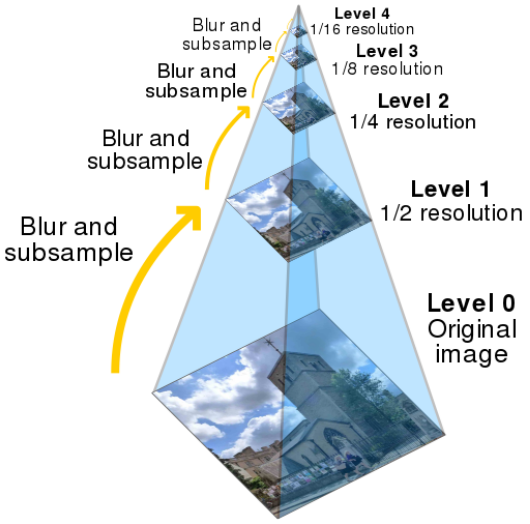

Figure 2.5: illustration of image pyramids by Wikipedia user Cmglee

There are two main types of ImagePyramids: downsampling and smoothing.[16] Downsampled image pyramids simply have fewer data at first, while Smoothed image pyramids start smoothed and progressively diminish the amount of smoothing. These can be combined.

The advantage of downsampling is that it creates a lower resolution image which will be faster to register and uses less memory. While smoothing is better if the goal is quality.

2.4 Validation for registration

Validating a registration model is a hard task. To validate a registration process, one needs to evaluate its accuracy.

For this, several approaches exist.

2.4.1 Using the similarity measure

Using the similarity measure that was used in the registration algorithm is a poor solution for several reasons.

"Similarity measure frequently used have no geometrical/physical significance"[14] This means that you can have an algorithm that performs very well in term of similarity measure but does not solve the registration problem at all.

It is trivially possible to get a moving image that is exactly the same as the fixed image by applying a strong and convoluted deformation. However, it would not be aligned. A bone

image might be deformed immensely to look like a heart image, but that doesn't mean it is aligned.

That does not mean the similarity measure is completely useless. Of course, if the deformation is properly constrained, then the alignment would have to be proper for a constrained deformation to get a high similarity measure.

Furthermore, combining several similarity measures can compensate for the aforementioned shortcomings of similarity measures because they can provide a different information each.

2.4.2 Generating Images

One approach that does work is to transform an image and then use the registration algorithm to realign it.[53]

The deformation is done using an algorithm that keeps track of the voxels's initial position. That information can be used to see if the registration has properly aligned the voxels with their initial position.

One problem with this is that if the transformations do not reflect reality properly. Then, the algorithm we are assessing might perform very well with some transformations, but very poorly in reality.

To compensate for that, some researchers have been generating images by simulating the acquisition process as well as the physics of the body.

2.4.3 Manually

The most reliable method used in registration competitions[28] is to manually assess the algorithm performance.

For that, there are several solutions, one of them is to evaluate registration accuracy based on point correspondence.

A human has to manually find corresponding points in both input images (the moving and fixed image) and then the accuracy is assessed by calculating the distances between the points.

Another possibility is to define regions manually (segmentation) and then compute the overlap between the regions.

In competition, the registration is assessed in many different ways, often including both point correspondences and region correspondences.

2.4.4 Other Tricks

These tricks are paraphrased from the elastix manual.[16]

Despite the similarity measure being useless to assess the quality of registration by itself. If using an optimization algorithm, it might be very useful to check the convergence of the metric to see if there is anything weird going on.

Another trick is to observe the Jacobian determinant of the deformation field. If it is negative, then the deformation wraps the image into itself which isn't good, if it's too high then the deformation is too high, it should be close to one.

For monomodal images, a trick is to create an image using the differences between the images' voxel intensities. If the generated image is uniformly noisy, then the hypothesis of the registration being successful cannot be rejected. Naturally, it can be uniformly noisy and still fail, another method should be combined with this "trick" to better determine if it is a success. If there are many edges and bright spots, then there might be an issue.

2.5 Summary

In summary, registration has for goal to find a transformation that aligns two images. It has many applications in various fields, and even within the medical field, it has a wide variety of uses.

This variety in fields of use introduced a variety of needs, and many different algorithms and tools have been developed to answer each specific need.

In our case, we use elastix, which is a registration toolbox, but it cannot perform every single category of registration: It does intensity-based, automatic and parametric registration.

Validation is important for registration, but it is a hard task with automatic validation having many shortcomings. Therefore, it makes manual validation the preferred option.

Chapter 3

State of the Art, Technologies

Before we present our solution, let's present the current tools that are commonly used for registration.

Many of our choices were decided because we had to differ from what already exists, so exploring the solutions offered by the alternatives explains the direction of this master thesis. Let's begin by presenting some state of the art registration tools.

3.1 Commonly used registration tools

3.1.1 ITK

ITK or the insight segmentation and registration toolkit is an "open source, cross-platform system that provides developers with an extensive suite of software tools for image analysis."[2]

ITK provides more than simply registration and segmentation, the software guide accompanying it has more than 700 pages.[4] ITK is considered by some as the "standard-bearer" for image processing algorithms and in particular for some registration methods.[5]

Multiple toolkits extend ITK registration methods in unique ways.

In general, ITK is one of the most complete toolkits for image registration, it is considered as a reference implementation.

One major issue is that it is considered complicated to use, even simple registration takes a lot of effort to code using ITK (It requires expertise in some relatively advanced programming concepts such as templated C++ which is something some users don't have).

Some efforts have been made to make it more accessible, including simpleITK[3], which is a python interface for ITK. This interface tries to hide the internal structure of ITK and simplify the complexities frequently encountered by ITK users. However, some functionalities of ITK are not available for simpleITK users. Furthermore, while the use of ITK is made simpler, it still isn't easy.

The user stills need to write code for simpleITK and understand the ins and out of registration to make proper use of it.

3.1.2 ANTs

ANTs (Advanced Normalization Tools) is built from ITK, it provides a high degree of customization possibilities through command line interface and scripting.[54]

ANTs is a great toolbox in term of performance, it reached first place in a number of competitions including the initial EMPIRE10[28], some of the SATA2013, BRATS 2013 and so on.

An advantage of ANTS is that solutions can be used in many similar contexts. Therefore, creating state-of-the-art registration solutions for specific use-cases (though in registration it is not currently possible to construct truly generic solutions that would work for all cases).

A notable disadvantage is that ants can *only* be called from within the terminal interface, and not included in code like other toolboxes such as ITK can, though there exist solutions which try to fix that problem such as Nipype (neuroimaging in python pipelines and interfaces)[49].

In general, ANTS is a great toolbox for registration, it provides a simple terminal interface, but still performs extremely well.

3.1.3 niftyReg

NiftyReg package is a "command line tools to perform rigid, affine and non-linear registration of [images in the format] nifti or analyze images as well as utilities".[6]

NiftyReg has performed very well in some registration contests, including the "ongoing" part of the EMPIRE10 challenge where teams can still put out new registration solutions, and a solution using niftyReg holds the second position as of this writing (June 30th, 2018).

NiftyReg is created to use the nifty image format specifically. Of course, it still is possible to transform an image from another format to nifty beforehand, but the registration toolbox only takes nifty images as input or output.

A major issue of this registration toolbox is the lack of proper documentation. The documentation provided is a wiki where the options are merely displayed with no explanation whatsoever into what they do.[8] Some examples are provided, but they don't seem very helpful if a user tries to perform a more personalized registration.

Furthermore, NiftyReg's goal is to work with the nifty image format, therefore, images will have to be transformed into nifty images before using the toolbox.

3.1.4 elastix

elastix is a registration toolbox that is mainly built around ITK, providing most options of ITK as well as some algorithms developed specifically for elastix.

An advantage of elastix is its modular design, which allows users to pick algorithms, metrics, and much more on a case by case basis very easily. This modular design allows for very simple parameter files (also called parameter maps) that can be shared among users easily.

In the EMPIRE10 competition, the team using elastix didn't perform as well as teams using niftyreg and ANTS, but it was still competitive.

The documentation of elastix is quite extensive, with some generated automatically as well as various publications about the framework[9][10].

Among the other open source tools, this was our favorite toolbox for the following reasons:

- Extensive documentation
- Access to ITK classes and filters

- Wide variety of algorithms
- Many examples of use, easily accessible from the wiki
- Easy to use compared to the alternatives

We will go talk more in depth about elastix in 6 because we used it in the project as a registration toolbox.

3.2 User-friendly tools for the uninitiated

3.2.1 Matlab

Matlab offers a registration toolbox[55] with a modular design similar to elastix, but it has less available options, being restricted to linear transformations.

In terms of optimizers, metrics and so on it is also much more restricted than the other toolboxes. The main advantage is that it's in Matlab, which is a large software suite containing options for everything from statistical analysis to deep learning. Matlab is also frequently used by students and researchers, which might make it easier for them to install it.

The syntax is also relatively simple, with an automatic picker of metric and optimizer depending on the mode (monomodal or multimodal).

3.2.2 ITK-Snap

ITK-Snap is an application which does image segmentation and contouring on medical images.[56] It boasts to be a user-friendly application that gives users ease of use, making it more accessible to doctors and non-professionals. ITK-Snap is compatible with nifti and Dicom. In term of user-friendliness for an application in the field of medical image analysis, it corresponds to what we think is ideal for beginners.

This software has tutorials and documentation to allow users to get the hang of it.

ITK-Snap does not address registration. It focuses on segmentation and visualization but it is worth mentioning, being a very user-friendly tool in a field where they are sparse.

Besides ITK-Snap there a few web applications that simplify the handling of DICOMs and sometimes offer some additional features such as Osimis and Intuitim[66][67]

3.3 Common trends

Registration Toolboxes have many elements in common:

First, all of these tools actually perform very well and contain a high variety of algorithms, metrics, optimization methods, etc... It was very clear that we could not compete in term of performance. The quality of these tools is extremely high and hard to match by a two-person team doing their master thesis.

Secondly, while these tools aren't particularly difficult to use, they all require a decent understanding of the registration process to be used effectively, though some have a variety of pre-made parameter maps to help with that.

Generally, the user interface is a console-based one, though it is possible to use some of these in conjunction with a viewer such as a 3D slicer. Some tools have been directly integrated to the viewers.

Our conclusion was that we needed to find a different way to differentiate than based on performance or number of algorithms available. There exist also some attempts at a grouping of these tools, which also requires a massive effort.

These tools tend to be based on Insight Toolkit, though niftyReg is an exception. Insight Toolkit is too hard to use for a normal user. Therefore, efforts have been made to create wrappers that give easy access to the algorithms from ITK while also adding new options.

3.4 Possible improvements to user's experience

As mentioned above the algorithms are still an area of research into which a significant amount of effort is being put.

On the other hands, less research has been focused on accessibility, interoperability and extensibility.[31]

In this thesis, we decided to focus on accessibility and general ease of use.

Efforts have been made to make the algorithms simpler to use[31] but there are several key aspects of usability that aren't being really addressed.

- Long build time, complicated installation process.

  The toolboxes in use for registration sometimes don't offer an executable file, are often in compiled language, and therefore have to be recompiled to be used on different platforms which can take a large amount of time.[29][38]

  This implies that a teacher trying to teach registration cannot really expect his students to easily install a good registration toolbox.

- Registration takes a lot of processing power.

  That can't really be changed without diminishing the quality of the registration, but it has consequences: A user's computer might be busy computing a registration solution for hours making it hard to the user to conduct other tasks.

- Validation is not readily available.

  Proper validation is most often at least partially manual, but offering proper tools can go a long way to simplify it.

  It's up to the user to compute the relevant images and open the solution in a viewer. There are also other helpful metrics that can be used for validations and they often have to be computed by hand.

- The parameters are hard to understand and to come by.

  Elastix, for example, has a lot of possible algorithms and parameters, but while they can be found online, it's hard to know all the options (e.g. metrics), and even harder to choose a proper one.

- The interaction-style, often in the console can be intimidating.

  While this isn't a big hurdle to overcome for someone who wants to experience with registration, surely having a more user-friendly interface would help users get into registration.

These are our conclusions, but further studies need to be carried out to properly assess the needs and the problems encountered in image registration. For now, we have already realized an exploratory observation of the field in order to have enough information for the realization of this thesis. Indeed, we relied on our own experiences with image registration as well as that of our promoter Pr. Macq and people in the domain we know of such as Mr. V. Nicolas CEO of *Intuitim*, a firm focused on medical images.

An assessment of these needs could be carried out with interviews, online quizzes or reviews of other commercial tools and their plus-value (to assess what is missing on open-source ones).

Chapter 4

Our solution

The solution we decided to create is a web application that allows users to use elastix to perform registration remotely.

Screenshots of our solution are available in the appendix.10

Besides performing registration remotely, we added features to simplify the registration process such as handling of files, viewing the images as well as the automatic creation of visualizations and so on. for registration which we'll present in this chapter.

But first, let's discuss why we choose to do a web app and the options we explored before settling on this solution.

4.1 Options explored

During the creation of this thesis, we explored many options. Through reading documentation and design thinking, we changed our direction in order to develop work with value. The criteria to take into account was mainly be adapted to answer a market demand as well as adapted to the workload of two individuals.

4.1.1 Improve toolboxes algorithms

At the beginning of this thesis, we were looking for possibly improving the current toolboxes. The first option we considered was to extend the toolboxes with some new algorithms.

This is hard to achieve as known algorithms have already been improved, optimized and reviewed many times.

The field of medical image registration is "one of the most active in the field of medical image analysis."[14] and while it is probably not impossible to come up with new solutions, it is hard to be competitive with the professionals in the field.

4.1.2 Improve toolboxes usability

Another approach to improve toolboxes is to improve their usability. Since it isn't practical for us to improve the available tools, we decided to improve their usability.

Efforts have been made for it such as simpleElastix, but we thought it was possible to do even better.

The first option we considered was doing an application that gathered some available tools in one place such as viewers, toolboxes and so on.

We were told and confirmed with some research that some of our potential users might be reluctant to install an application on their computer. For example, people working in hospitals would not install the app due to security. As security is taken seriously in hospital settings with intrusive investigations into the hospital's material[37].

Another problem with this is how hard these tools can be to install. If we were to make an application relying on an existing registration tool, our users would have to install all these tools, which could prove very hard. Solutions such as docker and virtual machines can mitigate this problem, but they come with their own set of drawbacks.

The solution we found was to make a web application. An open-source web application for image registration does not exist currently so our solution would be useful for at least a subset of image registration users.

4.2 Option chosen: Web application

A web application is an application that is placed on a server and can be accessed remotely by users using a web browser.

The advantages are numerous.

4.2.1 Advantages of a web application

The most obvious advantage is the fact that client-side installation is no longer required. This helps by negating the long and tedious installation of dependencies. It also means that it can run on any platform. The data is saved separately from the user's computer.

Since the data is hosted remotely if a user's computer crashes nothing is lost for him, and his data can be accessed from anywhere.

Registration can be a very long process, therefore it's an obvious advantage to be able to perform registration remotely on a server, which frees the user's own computer. This was possible before but it needed more efforts from the client, such as setting up an ssh connection

Thanks to the application being centralized, only the server needs updating and not the client. We use many separate tools in our solution and these tools each receive updates. It might be very bothersome to keep every user up to date. Fortunately, if everything is run on a server, only the server has to be updated.

Current registration tools most often only really offer a terminal interface, we created a graphical interface to go with the web application.

We put an emphasis on usability once the registration backend was completed. As a result, while it might not have the best usability for a web application. A more in-depth analysis and assessment would be needed to verify each of our design choices, it is a net improvement on the existing tools.

4.2.2 Disadvantages of a web application

There are also some disadvantages of it being a web application: The connection might be blocked by a firewall. A common example is in schools, or workplaces blocking a website with a firewall to avoid users to be distracted or access sensitive content. If it were to happen on our solution, there isn't much that can be done to avoid it.

A web application has to be hosted somewhere and the server costs are proportional to the processor, ram, HDD, and location. This is a major problem because the processing power required to run a registration is high as well as the Dicom files being quite large. These two main factors would make the costs unfeasible for us students to pay. What we want to do instead is provide the source code so that anyone who wants can run it on their own server.

Making it open source also has the advantage of allowing the web application server to be installed on an intranet with no access to the internet. We paid attention to not use online resources such as CDNs, which might be cut from the internet if the server is run from an intranet. Instead, we saved everything locally to ensure our website would still work even if it was isolated from the rest of the internet.

A side-effect of using a web application is that the data has to be on the server, which means that if the data (here often scans) is private, the user has to trust whoever runs the server to keep it private and secure. It also means that the data has to be uploaded to the server using the internet, which can take a while depending on the size of the image.

4.3 Features

Now that we looked into our reasoning for making a web application we will talk about the content of said web application. There is more to it than simply registration, some features are of course required such as uploading of images, but some other features offer an added value.

4.3.1 User Accounts

We manage user accounts, which ensures that users can keep their images, studies and everything else private.

In fact, most of our website is restricted to logged-in users only. We have a nice introductory page for non-logged users, but as soon as they want to upload an image, perform a registration, or do any kind of task that requires uploaded files, they are required to log-in.

To deal with users account, we used Django's default login application. An advantage of not creating our own user account application is that it gives more guarantees of security: if there was an exploit, other users of the login application might have noticed.

There is also a mechanism to recover a password using an e-mail, but it requires the server to own a mail address. Which we don't so we didn't test it properly. However, since we use the default tools from the Django framework the effort needed to put this in place once an address is acquired are minimal (a line to change in the settings).

4.3.2 Registration

The main feature we looked for is of course image registration.

Image registration is performed using elastix and outputs a registered image, as well as an image representing the difference between the two images for validation.

We put some limit on the possibilities of registration, though some of them might be trivial to add later. to implement every possible option from elastix:

- No point-set registration.
- No masks. Masks are used in registration to hide some areas that are not relevant and might introduce noise. They can be pretty useful, therefore us not being able to deal with masks is quite noticeable and we will talk about it in "possible improvements".
- No registration using multiple images.

Elastix offers a lot of registration options. Our project currently does not utilize all the functionalities of elastix. This is due to the need to tailor the project to these very specific functionalities. We wished to have a broad solution which doesn't overload the user.

To perform registration, the user simply has to select the fixed image, the moving image and the parameter map which he created.

4.3.3 Image visualization

We use cornerstone-demo that we included in our project for visualization.

The advantages of cornerstone are explained in chapter 6, but cornerstone demo specifically offers many features that people in the field use.

We first tried to create our own viewer using cornerstone but discovered we don't know enough about the need of people in the field to create one that would satisfy them. Knowing this, we decided to rely on cornerstone demo which already has many features and has been created with its target users (which are the same as ours) in mind.

Let's list these feature and the way they can be useful for a user of registration:

- A sidebar with the images of the study.

  This allows the user to choose which image to view by simply dragging and dropping the image into the screen.

- a "WW/WC" button. WW/WWC respectively refer to Window Width and Window Center

  To understand what this does, one needs to understand what the window is.

  A Dicom image (e.g. a CT scan) might have more intensities that can be shown on a typical screen using intensities. This window of grey allows the user to select which range of intensities he wants to visualize, furthermore, it also allows the user to focus on a specific range of intensities which might, for example, give him the bones in white and the rest in black.

- "Invert", "Zoom" and "Pan".

  As expected from their names, the invert tool inverts the color, the zoom tools zooms, and the pan tool moves the image.

  Inverting can be practical to see some hard to distinguish details. Zooming and panning make it possible to focus on a particular area.

- "Stack scroll", "play" and "stop" are buttons for navigating between the slices.

- The length measurement and angle measurement tools give the size of objects in mm and degrees.

- Pixel probe, rectangular and elliptical ROI allow users to get information on particular places in the image.

  Pixel probe gives the position of the pixel, the stored pixel value, and an additional value depending on the modality.

  Elliptical and Rectangular ROI gives geometrical values (size of each side, etc..) as well as the mean and standard deviation of the pixels values.

- Layout is a tool for users to display several images simultaneously, in a grid. Images can be dragged and dropped to each square of the grid, and the same image can be displayed twice. The stack scroll tool can be used on each image separately to see different slices.

  Used in conjunction the layout and stack scroll tools can allow a user to see different slices of the same image at the same time.

  This tool is particularly useful for registration because it can allow users to see the fixed image, the moving image, the moving image with the transformation applied and the differences in intensities between the fixed image and the moving image with the transformation applied.

Cornerstone-demo has been very useful to make a proper viewer, however, it required adaptations to make it respect the Django architecture properly. Despite our efforts, cornerstone-demo still doesn't fully respect the Django's MVT architecture (though it is pretty close) and instead uses some templates as if they were static files. This does not pose any practical problems but it isn't perfect architecturally, but making it entirely compliant to the MVT architecture would require a complete overhaul of the cornerstone-demo codebase.

We also modified cornerstone-demo to make our own listing of studies instead of using the one included. This allows it to have a consistent appearance with the other listing of our web application and remove redundancy. One downside is that cornerstone-demo relied on the use of tabs for much of its codebase, and such tabs were no longer needed since we used our own study management. We decided to hide the top navbar and remove useless tabs, instead of trying to change the entire cornerstone-demo codebase.

4.3.4 Parameter Map Creation

Parameter maps are used by elastix to store all the user's choices to define the different algorithm used for registration by the user.

As we will see they are quite complex and give a lot of freedom to the user, therefore we needed to make additional efforts to accommodate the parameter map's quirks.

Problem definition

Elastix's parameter maps are complex. There are many interactions between the different parameters, but in general, parameters can refer to Modules, Parameters of the modules, Parameters unrelated to the algorithms used which can give information about the image pixel type, the seed used for random numbers and other optional information.

Modules are specific algorithms which take the role of a **Component** of registration. For example "Transform" is a component, and many modules representing a specific transformation algorithm exist: one for rigid transforms, one for B-spline, and so on. A more in-depth explanation of components is provided in the chapter on registration 2, where every component of elastix is detailed. Some modules require additional parameters which need to be defined separately. If they aren't, the default value will be used instead.

There are 10 components which each require a module but it's possible to have more than 10 because registrations can be done consecutively. Often rigid registration is performed first to give a better initialization to a non-rigid one (such as B-spline). The order of components is not enforced, and it's possible to use more than 10 components even in a single registration (for example multi-metric registration requires two "metric" components)

The takeaway from looking into elastix's parameter maps is that they are complex and can be confusing for beginners, therefore any interface designed to create them need to be highly configurable.

A simple form with 10 entries, one for each component will not be able to deal with such complexity, and therefore we had to come up with a more modulable way to allow the user to create his own parameter file.

Solution

Our solution is to use Jquery and AJAX to create dependent selects that are automatically updated depending on the user's choice. For example, if the user selects "metric" as his com-

ponent, a dependent select will have its options updated to contain all the possible metric modules.

This translate to many different interactions which we had to implement:

First, there is an "add" button which adds rows in a table containing all the current parameters. This table has several columns, one for the component, one for the corresponding module, and one for the module's parameters when they exist plus another with a delete button for users to delete the rows.

The newly created row contains a select in which a component can be chosen. Once the component is selected, a corresponding module can be selected among those that can take the role of the selected component. The list of available modules for this specific component is fetched from the server using AJAX.

Once the module is selected, the parameters that the module are also fetched from the server using AJAX and appear in the corresponding column.

Each row can also be deleted using the row's delete button which is represented as a red button with a trashcan icon.

There is an additional button on each row which allows the user to switch between using the default value or custom values. If they choose to use the default value, the module's parameters will be hidden and will not be gathered when the final parameter map is submitted.

One might think the ideal solution would be to show the default parameters instead of hiding the parameter's input, but this is made impossible by the fact that the default values are computed at runtime and therefore impossible to get before the registration is launched because some of these default value depending on the image or on other parameters.

To submit the parameter map once it is done, there is a submit button which will send all the information, excluding the parameters where the user decided to use the default value, to the backend.

The parameter map can still be edited afterward, in which case the saved parameters are all loaded as if the user selected them himself using jQuery.

Our solution is quite powerful but not exhaustive: there are still some edge cases where a specific behavior needs to be enabled, but it works for the majority of components and metrics.

4.3.5 File management

To perform registration files are needed: images, studies and parameters maps.

To generate, edit them, and delete them, we created a file management interface.

It consists, for every page, on a page where the list of available entries is listed with all the information the user provided about it, a page to create a new entry or edit the entries.

Images upload

The user can upload Dicom files compressed as a zip to the server. He can also add some metadata, such as the date, a description, the modality, as well as the patient's name and ID.

On the backend, there is validation to make sure the image is really composed of `.dcm` files in case the user messes up and uploads the wrong file.

Instead of using user-defined names, we use uuid4, which is a function provided by a python module, giving us a unique ID: the uniqueness is not guaranteed but statistically extremely unlikely, which we use to store the files.

Studies

Studies are images meant to be visualized together.

We created an interface to generate them and add some metadata.

The metadata on studies are name of the study, name of the patient, modality, description, and date. All of these except the name of the study are completely optional because sometimes it would be complicated for example to decide on the modality if the images are multimodal (though something like "CT_MRI" would still be informative) or to decide on a patient if the study is intrapatient.

Users can then add images to the study using a select who has an option for each image, and when they do, all of the meta-information on the images (number of slices, date, modality, patient name and id and description) are immediately displayed in a table using jquery. It is also possible to remove the current images from the study.

When adding an image, the option from the select for the image disappear so that users don't try to add twice the same image without realizing it's already there.

4.4 Summary

In summary, we created a solution with all the essential features for registration.

We focused on first getting registration to work and then adding features around it to improve usability, including a viewer and management for files and parameters.

Our solution can still be massively improved by adding more tools for usability and enabling more types of registration but our current solution is still useful in many use-cases.

Chapter 5

Process

The creation of a web application takes a lot of time and requires to choose an approach. It is important to create something that is desired. We need to identify the "Customers" and the pain or needs of the "Customers". We want to have an added value to ease their pain or help with their needs. To do this, we wanted to use what we had learned through our curriculum.

We will explain the design thinking process which is the pioneer method for innovation. This process helps us to define the *"why is it done?"*. Then, we will detail the agile development method and more precisely Scrum, which explains the *"how is it done?"*. We had to make sure that the requirements were being respected all the way through for the customer. Mr. Vincent Nicolas from Intuitim acted as the customer.

We went through a lot of iterations of what we imagined the product should be before actually getting into doing the project. So let us first define the 2 core concepts we used to come up with our solution.

5.1 Design thinking process

Lets first discuss the design thinking process and how we were inspired by it. As seen in the figure above 5.1, it is not a linear process, but more of a circular one. The design thinking process is a method for creative thinking and problem-solving. It consists of 6 stages which don't necessarily need to be done in a linear order: we can always fall back to a certain stage to reevaluate. At its core is finding the people's needs and find the best way to fulfill those needs.

5.1.1 Empathize

"To create meaningful innovations, you need to know your users and care about their lives." [32]

The first stage is to empathize. Empathize is defined as "understand and share the feelings of another" (taken from Cambridge dictionary). In Design thinking, this translates to fully understanding our users in the context of our task. The point is to find a way to understand the feelings they get from the actions that are taken. We need to find what is meaningful to them, physically and emotionally, within the context of our task. We need to empathize with the users because we are trying to solve their problems and not our own. The 3 elements required to empathize are:

- **Observe:** This step consists of looking at the user and how he acts in his everyday life. This is actually important because it helps to learn things that he may not even realize. For example, via his composure and his reactions to certain scenarios that one could not get from just a simple interview.

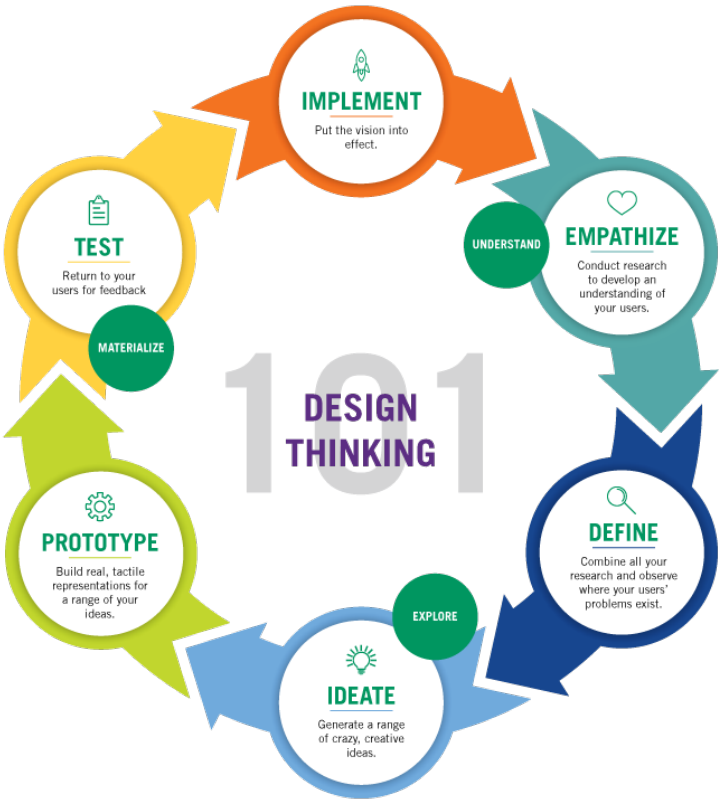

Figure 5.1: Design thinking process taken from `https://online.concordia.edu/business-news/design-thinking-ideation/` on 18/07/18

- **Engage:** This is basically interviewing, but in a more natural way and ask questions like 'why?' to get to the soul of their problem and get them to tell you what they are thinking.
- **Watch and listen:** This is a more active approach of observing. Insight is gained by telling the user to do a task (something related to the problem you wish to solve), and as they do it, having them say out loud everything they think. The environment you are in can help find better, more meaningful questions.

We face in this stage several difficulties which drove us to aggregate different approach that allow us to reach a suitable replacement of the methodology.

5.1.2 Define

"Framing the right problem is the only way to create the right solution." [32]

The define stage (also known as the *Define mode* [32]) is when we try to "craft a meaningful and actionable problem statement"[32]. We want to get a point-of-view. We want our statement or definition to be based on the insight we currently have. We can get insight from the previous step of Empathize or from future steps where we would revert back to this stage in order to redefine the process.

When defining a point of view you wish to narrow "the problem" to a few specific problems. From the previous step, the information acquired was dispersed and getting a specific idea from all this information allows insight. This insight is the key advantage against competitors that aren't using the design thinking process and therefore won't get this insight.

Getting this insight is achieved by finding patterns from the different groups of users. You need to understand who your **users** are, try to focus on a few **needs** of these users. This **insight** needs to be able to be expressed. These three points will give you a good point-of-view.

A good point-of-view is one that (taken from [32]):

- Provides focus and frames the problem
- Inspires your team
- Informs criteria for evaluating competing ideas
- Empowers your team to make decisions independently in parallel
- Captures the hearts and minds of people you meet
- Saves you from the impossible task of developing concepts that are all things to all people

You want to create a list of "How might we...?" questions that originate from your point-of-view to help with the next step.

5.1.3 Ideate

"It's not about coming up with the 'right' idea, it's about generating the broadest range of possibilities."[32]
Ideate is the stage where ideas are created. We want to get as many ideas on the table to prototype and see if the client can enjoy. We want ideas to solve the problems defined in the "define" stage. This is done through many approaches. Mind-mapping sketching, brainstorming or even prototyping immediately to get ideas while prototyping.

5.1.4 Prototype

"Build to think and test to learn."[32]

Prototyping serves to help reflect on the possible ideas, fail as quickly and cheaply as possible, test out possibilities for cheap, and just get started.

In this stage we aren't necessarily talking about a functional and large prototype that costs millions to make, instead, we are referring to a cheap prototype that can give a proper understanding of the user experience. For example, making a prototype on paper with post its and scissors.

5.1.5 Test

"Testing is an opportunity to learn about your solution and your user."[32]

Test is the stage where the prototype is brought to the user and they test it out. This helps refine many of the previous stages which can be jumped back to. One can empathize with the user some more as one discusses with them and observe the use of the prototype. It also helps define a different point-of-view via the newly acquired knowledge. It makes it possible to find out what works via the user experience. The prototype is most likely not perfect hence testing allows to refine the prototypes.

When the testing is being done, it is better to let users play around with it without explaining what it does. The goal is to test out the user experience that the prototype can give by trying to put users in the right setting for it.

You can bring in multiple prototypes to compare them from one another and see the pros and cons to help with the process of the tests. This can help discover needs that were not understood previously.

5.1.6 Implement

Some people don't include the implement phase (also known as deploy) in the design thinking process. This is because it is a given that you should implement your prototype when you found a proper solution and have done the necessary iterations. In our case, we implemented it using the agile methodology which will be defined below.

5.2 Agile

There are numerous project management methodologies, such as waterfall, a sequential method, and agile which, being the opposite of waterfall, is flexible and requires a lot of communication. Beyond these two, there exist many more other methodologies IPM, CCPM, CPM, PRINCE2, ...

We decided to use the agile methodology but more specifically Scrum.

Let's first define Agile.

Agile is a project management methodology. Agile takes a big project and separates it into smaller manageable chunks. These chunks are defined as iterations, and at the end of each iteration, the goal is to have something tangible and of value. Then, we reassess with the users and stakeholder and redefine the values. This is very different to the waterfall method which is sequential. In waterfall, as long as one part of the project isn't done we don't start another. For example, design is a step after research hence with waterfall you have no designers until the research phase is complete and we are in the design phase. While with agile, we have a mixed team communicating with each iteration working on the smaller chunks and assessing their work.

For Scrum, "Agile programming" is more of a philosophy than a prescriptive list of things to do and their main values are[33]:

- **Individuals and interactions** over process and tools.
- **Working Software** over comprehensive documentation.
- **Customer Collaboration** over contract negotiation (In this case Mr. Vincent Nicolas will act as the customer).
- **Responding to Change** over following a plan.

These values fit with our mindset as our software project was bound to see change through the course of the year.

The 12 principles of agile are [33]:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
3. Welcome changing requirements, even late in development. Agile's processes harness change for the customer's competitive advantage.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity — the art of maximizing the amount of work not done — is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile methods are more flexible and tend to yield a better result or to yield a result when things go wrong whereas waterfall might not give you anything working at all if the project requirements are subject to change.

There are many different project management methodologies based on the agile methods but most of them aren't adapted specifically for a thesis and tend to be tailored for the corporate world.

There are many options that incorporate the agile methodology such as Scrum, Kanban, XP, crystal, feature-driven development.

5.2.1 Scrum

We decided to use Scrum, therefore, we will explain it in this section. We are going to define the specific terms used in Scrum, that are at the base of its structure.

Scrum is one of the most popular agile methods. "Scrum is founded on empirical process control theory, or empiricism."[34] It consists of artifacts, events, and roles. That will be discussed further below. First, let's mention the three pillars of Scrum that are key. They need to be kept in mind throughout the Scrum process.

1. **Transparency**: is very important because without transparency we cannot respect Individuals and interactions or even customer collaboration. This transparency is defined as "Significant aspects of the process must be visible to those responsible for the outcome. " [34]
2. **Inspection**: people have to inspect the artifacts and the progress regularly. It is useful but if its done 24/7 then it could effect progress. It is suggested to be done by an expert.
3. **Adaption**: The key difference of agile vs waterfall is its adaptability. So it has obviously more adaption.

Team Roles

There are numerous roles in the Scrum process. There is the Product Owner, the Development Team and the Scrum Master.

The Product Owner
Is a single person that is accountable for the understanding of the backlogs, and expressing it to be well understood by the development team. They also order the importance of what has to be done in the backlog. Make everything clear and understood as to make sure the path is visible and clear.

The Development Team
The people who will develop the project so that it is considered "done" at the end of each sprint. The development team is self-organized. Everyone in the development team is considered an equal developer and the team is responsible as a whole to oversee the progress. Having too few developers like 2 or 3 can make it that the development team is missing skills and slow down the process, however, too many people (>9) makes coordination hard.
The Scrum Master
The Scrum Master is the professional that ensures that Scrum is being respected and understood by everyone. The Scrum Master helps coach the development team in being self-organized and cross-functional. He facilitates the Scrum meetings and helps the product owner with the backlogs and can even incorporate Scrum in an organization that does not have it yet. In our case we do not have a Scrum Master as we are 2 students and we won't be meeting as frequently as if we were in an organization, instead we made sure to both have a good understanding of Scrum. The Scrum Master also makes sure the Sprint Planning is done within their time frame and that everyone is there.

Scrum Events

Scrum events are all set in advance and they are timed. This gives consistency. A sprint cannot be shortened or lengthened once it starts. Everyone will look back on his or her past experiences and replan a sprint based off of them. This will allow them to adapt and learn.

Sprint

A sprint is a time-frame that last between 1-4 weeks which is set. At the end of the sprint, there will be something considered "done" and it is a sort of completed product that could be released. Sprints are sequential, that means a new sprint will start immediately after the previous sprint is finished. Having longer sprints would cause products to be too complex and we want simplicity. The scope and goals of a sprint shouldn't be changed unless re-negotiations are done with the Product Owner and the Development Team. Only the product owner can cancel a sprint if the goals are obsolete.

Sprints have "Sprint Planning, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective." [34].

Sprint Planning

The sprint planning is where everything that should be done and how it will be done gets decided. The whole team is there including the Scrum Master, the development team and the product owner. The planning is done at the start of a sprint and lasts 2-8 hours for a single, two, three

or four weeks depending on the length of the sprint. The 2 main topics are:

What can be done this sprint?[34]
Everyone has different goals. The Development Team tries to figure out what will need to be achieved and need to fully understand what will be achieved. The Product Owner expresses his objectives for the sprint and looks at the items of the Product Backlog that will be required to achieve this Sprint. It's the Development Team that will check if it can be done within this Sprint. These things are decided via the Product Backlog, result of the previous Sprints and the projected capabilities of the Development Team during this Sprint.

They decide on the sprint goal as well during the sprint planning. The Sprint Goal is a goal that should be achieved during the Sprint by implementing the Product Backlog.

How we choose the work to be done?
So once we decided the goal and the product backlog for this sprint, the development team decides how they will achieve this, to be considered "done". It is the development team who is self-organized but this decision is done at the start of the sprint to help have transparency and so that everyone is on the same page. Obviously, the development team can ask for help in the clarification of the product backlog and the goals for this sprint with the Product Owner. They can even reevaluate if they consider things to be impossible in order to adapt with their team. At the end of the sprint planning, the development team would be able to explain their approach and how they will achieve the sprint goals to the Scrum Master and Product Owner.

Why insist on the sprint goal? First of all the sprint goal is "a short expression of the purpose of a Sprint, often a business problem that is addressed. Functionality might be adjusted during the Sprint in order to achieve the Sprint Goal." [36] The reason for this is first to have the development team work on one goal. Even though the different team members work on different parts they maintain unity with this goal in mind. It also gives them flexibility and they can always change if things don't end up as expected with the product owner in accordance with the Sprint Backlogs.

Daily Scrum

Daily Scrums are "daily time-boxed event of 15 minutes, or less, for the Development Team to re-plan the next day of development work during a Sprint. Updates are reflected in the Sprint Backlog." [36]

The Development Team uses daily Scrums to track the progress of the sprint. This is done entirely by the Development Team and usually consists on stating what was done the day before, what will be done today, and if there are any problems that could slow down the progress of the goal. Attacking these three things allows to have consistency and visualization of progress. Since daily Scrums are done every day there is no real need for other meetings. However, people can obviously meet up and tackle problems or discuss into further details the plan and adapt.

It promotes self-organization, communication, and quick decision-making. The daily Scrums allow to identify problems and adapt. These meetings are for Inspection and Adaption in the development team. There is some transparency included in the daily Scrum but because it does not include the product owner it is not considered for transparency for the whole Scrum team but for the development team.

Sprint Review

Sprint Review is a "time-boxed event of 4 hours, or less, to conclude the development work of a Sprint. It serves for the Scrum Team and the stakeholders to inspect the Increment of product resulting from the Sprint, assess the impact of the work performed on overall progress and update the Product backlog in order to maximize the value of the next period." [36]

Sprint reviews are done at the end to look at the feedback and adapt the product backlog at the next sprint planning. Everyone gets to see what was done and what wasn't done. This allows them to learn what was done well during the previous sprint and what to improve upon in the next one. This is usually considered an informal meeting and hence when things are presented, it is better to have more collaboration for transparency. It is also here that we look at the marketplace to see what steps should be considered. There is also a review of the timeline, budget and other articles of interest that should be taken into account.

Sprint Retrospective

Sprint retrospective is a "time-boxed event of 3 hours, or less, to end a Sprint. It serves for the Scrum Team to inspect the past Sprint and plan for improvements to be enacted during the next Sprint." [36] The sprint retrospective should be done in a positive and productive mindset. The possible improvements for the next sprint are not just effectiveness and amount of work done, but also a more enjoyable experience. These improvements allow adaptation from sprint to sprint and allow an opportunity to inspect them as well.

Scrum Artifacts

Artifacts are all the things of value and work created or to be created. We define them nicely and clearly to help with transparency so that everyone can see them and understand them. But as well to inspect and adapt, if they are defined we can question them and figure out the problems and solutions given the artifacts already defined.

Product Backlog

The product backlog is "an ordered list of the work to be done in order to create, maintain and sustain a product. Managed by the Product Owner." [36] The product backlog is never considered complete. That means that it can adapt and change to new needs. The product backlog should be well defined and visible for all to see. The product backlog is a list of items. These items are "features, functions, requirements, enhancements, and fixes"[34]. These items are listed in an order of importance. The product backlog can increase in size even after the product is out, as there is new feedback on the product. This is a key concept in Scrum, with adaption. This allows easier Scrum planning, in which items are picked from the product backlog and placed into the new sprint backlog. The sprint backlog is the same concept as a product backlog however it lasts only for the duration of the sprint. Items are also placed in order of importance, and the development team members work on a certain item until its considered done during the sprint. The development team can modify the sprint backlog if it is required as Scrum allows for self-initiative from the members. Decisions are visible and should focus on the sprint goal.

Bad decisions usually stem from misinformation or miscommunication, that is why a key aspect of Scrum is **Artifact Transparency**. When we have all the information we can take the best decisions. This transparency is not achieved easily. Transparency is learned over time from past experiences and analysis. That is why the Scrum Master is in charge of enforcing this key aspect.

5.3 Our process

Now that we identified the 2 big concepts, that we utilized during the master thesis, we will define our trajectory and how we used these concepts.

At the start of the academic year 2018, we worked on an entirely different thematic which consisted of "Multi-Atlas Segmentation and Registration in brain tumors" and we were to do the gold standard using state of the art tools so that Eliott Brion, and Jean Leger to our gold standard.

This consisted of doing research and looking through the different technologies so as to do the gold standard vs Brion and Leger's Ph.D. solution of using machine learning to do better Segmentation and Registration.

During the start of this academic year, we had no previous experience in image registration and segmentation. The field is hard to get into, even if there are extensive toolboxes they aren't particularly intuitive, and the options are numerous. We had felt a pain while looking into the other solutions. Not only in the complexity of how registration works, but as well as the nonexistent user-friendliness of these powerful, but complex toolkits which were available.

Due to the discovery of this need, we pivoted into the creation of an application for image registration and segmentation.

We attempted to empathize with the potential customers. Unfortunately, we did not have any of our target clients at hand, no contact with radiologists or people intrigued.

So we took our own experience of the start of the year as a basis to understand what they would wish. We took this decision while observing our global situation towards the project.

So using the empathy that we had gotten from ourselves which as previously mentioned is not the best way but it was the best we had available. We defined the problems with a first version of the specifications. We did screen caps of what we imagined it should be as a prototype and started off coding it. We did a very simple prototype of our concept (as shown at 5.2). It was entirely done in python, the technologies we used were pyQt5 for the graphical interface, and pydicom in conjunction with matplotlib for the display of the DICOM files. We had in mind a concept for advanced options and a simple button which would run a generic image registration. The registration or segmentation was not actually implemented as we followed the design thinking process and wanted to have something where the user experience could be tested.

At the end of the first semester, we met with Vincent Nicolas, the Co-Founder and CEO of Intuitim 5.3. He is an expert in the field of Medical Imagery. This gave him a special insight that one could acquire from Empathy and the Design Thinking Process. With his insight, he had already acquired he guided us towards creating a web application instead of a standalone application. One reason is that our target audience was doctors and potential users who wish to learn about image registrations. If one day our application was to scale up and be used by hospitals, the hospitals would have a very difficult time to install on their computers a standalone application because of security measures. Thanks to this insight we pivoted from our prototype to a new goal which was a web application that focused on Image Registration and not segmentation.

At the start of the second semester, we redefined our goals, and we decided to use Scrum.

This is a master thesis and we were 2 students and not a group of 6-8 development team members with difficulty of meeting for very long periods of times. Due to that we did not follow precisely the Scrum's guidelines, but instead had to adapt it:

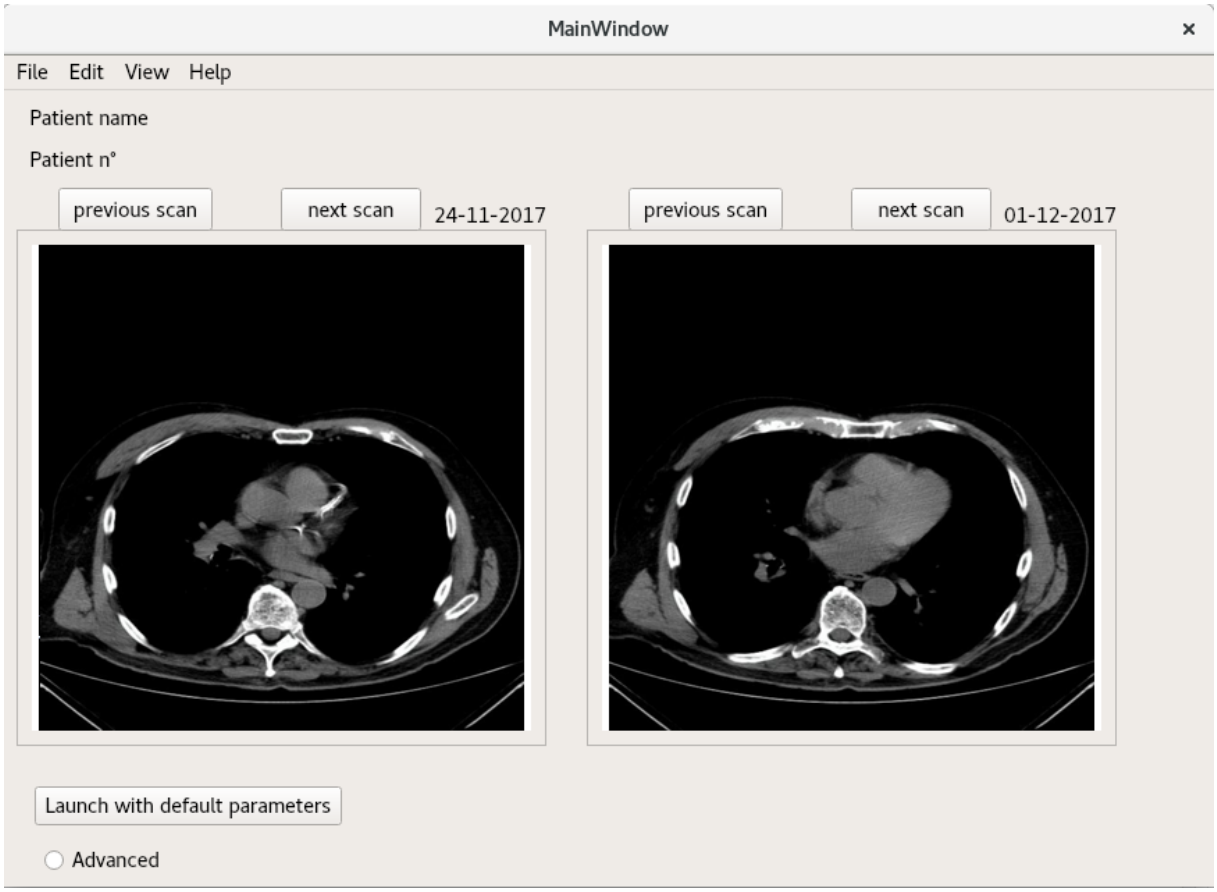

Figure 5.2: First Prototype

Figure 5.3: Intuitim Company

The guidelines of Scrum say that sprint planning should be done with everyone, the development team, Scrum Master and product owner involved for long periods of time. Furthermore, the Product Backlog should be defined as a team with the Product Owner.

These elements of Scrum made it necessary to adapt the Scrum method to the best of our ability, keeping the concepts of Product Backlogs, Sprints and everything else. However, all of this had to be done with few meetings and short sessions as the participants were tight on their schedules.

So, the Product Backlog was defined by us and confirmed by Vincent Nicolas and Benoît Macq. This allowed us to obtain a satisfactory result while respecting everyone's time constraints.

Our product backlog changed through the months as the product became clearer and as we iterated back and forth between the different stages of design thinking. But at the first iteration it consisted of:

Must have

- Registration based on simpleElastix on a webpage.

- Visualization on a webpage: We should be able to visualize both the basic DICOM image and the result of registration.
- Meta parameter optimization: Use some performance measure to test for all kind of registration with all sensible meta-parameters combination until we find the best one. Only works for large datasets.
- Big red button: a button that does the registration out of the box without additional interactions needed(can be poor quality).
- Possibility to play around with access to simpleElastix parameters.
- Looking simultaneously two images of the same patients at different times.
- Web server.

Should have

- Users having independent DICOM files.
- Security measures.
- Unit tests (selenium). Selenium is the minimum for unit tests, you do a series of operations and it records it, then selenium can do the same series of operations faster and tell you if it gives it the same results. It, therefore, speeds up the process.
- Collaborative machine learning; If someone goes to the trouble of finding the best meta-parameters for a dataset, we want to be able to re-use these meta-parameters for datasets that share enough similarities.
- Slider of gray nuances for DICOM files: The nuances of gray of a DICOM file can't all be shown on a screen.

Could have

- Graphically appealing design.
- Proper human-computer interface (tested on real users, in a serious manner).
- Describe parameters to define meta-parameters: Doctors don't use the same terms as registration specialists, so it would be nice to have some sort of translation from one to another.
- Scalability.
- Annotating.
- Masks (mask the part you don't want in your registration process).
- 3D image viewer (with VTK).

Won't Have

- Segmentation.

We used the MoSCoW method to order the priorities of our backlog. "MoSCoW is a fairly simple way to sort features (or user stories) into priority order. Its splitted in **Must Have**, **Should Have**, **Could Have**, **Wont Have**."[43] This is a fairly simple way to categorize the user stories. We placed them in our Trello and would place a few items from the product backlog to the Sprint backlog and during that sprint achieve each sprint's sprint goal.

However, after our first suggestion of the product backlog, it was suggested that it did not provide a global view of our solution. For this purpose, we wrote user stories to better define our product backlog. We also created a series of prototypes on papers which can be found on the appendix of this thesis in figures 19, 20, 21 and 22. In the listing below P denotes the priority level, and E denotes the difficulty estimate (both on a scale from 1 to 10).

5.4 User stories

- (P:8, E: 2) Doctor wants to load his Dicom files
- (P:7, E: 9) Student can test out parameters
- (P:7, E: 2) Users can log-in/ register
- (P:6, E: 1) Users can keep files private
- (P:5, E: 1) Users can make some files public
- (P:10, E: 4) Users can perform registration using default parameters with a single button click (once the images are chosen)
- (P:2, E: 1) User receive instructions on how to use the website
- (P:10, E: 4) Users can visualize its images
- (P:8, E: 7) User can assess visually the result of registration
- (P:7, E: 5) User can compare visually multiple transformations
- (P:7, E: 6) User wants to visualize some statistics based on registration
- (P:8, E: 5) User can choose the range of grey they want to see.
- (P:6, E: 3) User can annotate the result or the images to be registered
- (P:6, E: 8) User can apply masks on the image
- (P:6, E: 10) User can benefit from registration made on another dataset to infer better parameters based on the similarity between their dataset and said other datasets.

The prototypes are available in the appendix. But what is important to keep in mind is we kept going back to the design thinking process in consortium with the Scrum methodology to try innovate and find out what is the best solution to make image registration available to more people.

We went on, and for our first Scrum, we wanted to have a product "done" as Scrum suggests. Sprint Backlog:

- Registration based on simpleElastix on a webpage
- Visualization on a webpage: We should be able to visualize both the basic DICOM and the registered image.

- Web server with Django

We decided to use different technologies to implement the features of each Sprint. Sprints lasted different amount of times defined beforehand depending on the amount of work required to be done. For code version control we used GitHub.

The first Sprint lasted 2 weeks and we had a first version where we had a basic local server running with Django. Our version was simple and unoptimized. The registration that was done on the first Scrum was equivalent to the Hello world of SimpleElastix. It was a simple translation function with no parameters.

The original viewer from this sprint is nowhere as complete as CornerStoneDemo that was implemented on one of our later Scrums, it relied on transforming the slices into png and sending that to the front-end. Discussions with Vincent Nicolas allowed us to revise that approach. This was done by meeting up together and working in close proximity to one another. We had a whiteboard with our sprint backlog and a calendar representing what needed to be done. We also had a Trello to see who was doing what. For difficult challenges, we did it in pair programming. This was very helpful as it allowed reflection and would provoke fewer errors in our code and it was also positive for motivation. Pair programming is an integrated part of agile methodology in the software development area, in particular, it is suggested by "extreme programming" methodologies.

2nd Sprint backlog:

- Student can test out parameters
- Doctor wants to load his Dicom files
- User can choose the range of grey they want to see.

For this sprint, we implemented the user stories instead from the previous style of our Product Backlog. It consisted of making a file management system for uploading DICOM. We were still not using cornerstone demo so we added tools from the cornerstone tools to handle the different ranges of gray that you can get from DICOM files 6. We also implemented the first version of the parameter picker. This was chaotic as there were many different approaches to handle this. We tried to implement it using just javascript then using a mix of AJAX and jquery we came with our definitive solution. Django's Forms aren't very practical to make dynamic forms. So working with JQuery and AJAX is best. For security, we made sure to include the CSRF token6.

3rd Sprint Backlog:

- People should be able to create an account and use it as there own.
- CornerStoneDemo has all the features desired in the display of DICOM files.

The 3rd Sprint lasted 2 weeks and consisted of implementing the users so they had access to private files and implementing Cornerstone Demo. Cornerstone Demo is explained in full detail in the chapter "Our solution"4. The objective of cornerstone demo is to be a representation of what can be achieved with cornerstone tools, by implementing all of the tools in conjunction in a fully-featured viewer.
To make cornerstone-demo work, it requires the generation of JSON files and it has a loadstudy.js which is in charge of loading the files. What was done to make it work was change all the gets and use wadouri instead of dicomweb loader which is another way to load the image to the frontend. CornerStone Demo did not use single images like our previous sprint's solution, but instead studies which each can contain 1 or more DICOM files.

At the end of the sprint, during the sprint review, we realized that our file-management system was insufficient as we did not count on using Studies but just Scans. Therefore we pivoted and changed our Product Backlog with our new insight. We also wanted to utilize the advantages of having an administrator on a server. So we would implement the admin interface on the next sprint.

4th Sprint Backlog:

- Users should be able to create a study.
- Users should be able to modify parameters
- An admin should be able to work on the DB

The 4th Sprint was done in the summer vacation and we worked in close proximity. We used the whiteboard. When things were hard to implement we didn't hesitate to ask one another for some help. We would meet up and talk about 10-15 minutes on what would be done today and who worked on what. These Daily Scrums allowed us to work at a better pace and keep one another accountable of what needed to be done. During the first 2 sprints, it was more difficult to truly respect Scrum. But with all the feedback from one another and looking at how we addressed the problem and see what was done well and what was badly done we got to adapt. For example, we realized that Scrum insists on listing all the items and hence allow a more focused approach to the tasks at hand. That is why towards the end we would list everything needed to be done and whenever something new would come to mind we would write it down, and then discuss it on our next daily Scrum and possibly add it to the backlog.

To implement studies we used jquery to create a new form. We once again made sure to use the CSRF token, and to keep everything clean with the sidebar we used the template inheritance. Hence we refactored our previous version of uploading DICOMs to conform with our new implementation. Additionally, we gave it a much sleeker look. We also added the NavBar to have the User info because previously it was only accessible via exact URL link.

5th sprint backlog:

- Make it open source

During the final sprint, we changed our GitHub repository from private to public that way people can implement and distribute our web application and could even make it accessible only via intranet fairly easily.

5.5 Summary

What this thesis taught us was that there is not necessarily only one solution in Software Project Management. For example, we used both design thinking process and Scrum.

Unfortunately, Scrum was not a 100% perfect fit as we had time constraints with our Product Owner and the development team consisted of only the 2 of us. Therefore, we had to do our best to adapt the process to our specific needs. The best way of advancing is to learn throughout the process and reflect back on "the good and the bad". This gives us greater freedom and a more manageable process phase.

The agile principles are good to keep in mind when working on a project, and design thinking allowed our team to find a niche where we could create a useful product.

Chapter 6

Technologies

6.1 Introduction

As seen in the chapter "our solution"4, we have chosen to create a web application.

To implement our solution we decided to use a large amount of external open-source resources. First, because it is useful to save time and secondly because the open source code has been reviewed and is expected to carry a minimal amount of bugs.

Dealing with DICOM file means encountering a lot of edge cases, and it is almost impossible to deal with all of them, fortunately, the open source tools we use are designed to work with DICOM and therefore deal with a lot of these edge cases by themselves.

In our code, we had to use many specific programming languages, as well as HTML and JSON which aren't technically programming languages. Furthermore, we also used specific libraries, a web application framework, and a registration toolbox.

We will first start by presenting the languages we used and explain the reasoning behind their choices.

6.2 Languages

The programming languages we worked on are Python, JavaScript and Jquery. We also use HTML and JSON.

6.2.1 Python

Python is an interpreted high-level language. [57]

The main advantage of using this language is how easy it is, most things that users have to care about in languages such as C++ are easily dealt with, using features such as:

- Dynamic memory allocation. Python uses a garbage collector that deals with memory management.
- Dynamic typing, variable type don't have to be declared and are enforced. Type conversions still have to be explicit.

The syntax is "Mainstream", the python programming language with a few exceptions has very intuitive keywords and is frequently compared to pseudocode.

It is Object Oriented. This is very practical when building a web application. It should be noted that python does not advise for private attributes like java does. It has the concept of property which can create access functions for attributes which effectively makes them private.

There exist an impressive amounts of "modules" which can already do pretty much everything. We use them extensively in this projects because of their high quality and ease of use. They offer solutions to many common problems in an often efficient, optimized and secure way.

Python typically needs less code to perform the same task. The study *A Comparative Study of Programming Languages in Rosetta Code* comes to the conclusion that the code needed for the same task in other mainstream language is longer than what is needed in python. This isn't the best metric to display the fact that less effort is needed for many tasks but in our experience, it mostly holds true that the amount of code needed is lowered by the fact that many things are dealt with implicitly.

The community is also huge. This is reflected by the presence of great tools created for it such as Django, "simpleElastix" and "simpleITK".

Another reason is that we have used it previously for our AI courses and are therefore familiar with its syntax and some of its concepts.

The main disadvantage of python is that it is slower than most other mainstream languages, even high level ones.[58] In our case, it's not that much of a problem because the parts of our code which take the most time are in elastix which is coded in C++ with a wrapper (simpleElastix) giving us access to the classes in python.

Therefore despite us using python, the most computationally intensive tasks are still done in compiled languages such as C++ but accessed in a transparent way in python.

6.2.2 JavaScript

JavaScript is a client side scripting language.[59]

What this means is that it is executed on the client's machine in the browser. Thanks to that, the user does not have to wait for the backend response for JavaScript to be executed, the entire code is sent to the user and interpreted by the browser.

We use it in many parts of the application: Cornerstone, though not coded by us, includes a large amount of JavaScript and we had to modify some of it. The parameter picker is in JavaScript, the DICOM uploading is partly in JavaScript and many more small parts of the application also use JavaScript. Anything that is animated client side is coded in javascript.

It is pretty hard to do anything without JavaScript on the web currently (flash for example isn't used as much and will retire in 2020).

The choice of JavaScript was a quick one, the reality is that there just isn't any real alternative. Everyone uses it to the community is huge and there are many libraries coded specifically for JavaScript.

There still exists some alternative such as:

- CoffeeScript: A language that is transcompiled into JavaScript, it increases readability for the developer but the user receives JavaScript.
- Dart: A Google alternative, which has better performances but the community is smaller and it has fewer capabilities than JavaScript.
- Typescript: A Microsoft alternative that compiles into JavaScript. Its goal is to enhance JavaScript capabilities.

- Many more exist but the basic idea often stays somewhat the same.

A common trend in these languages is that they can be compiled into JavaScript or are somehow compatible with JavaScript and have a smaller community.

These languages have to be interpreted by the browser, so if the browser doesn't offer support for it then the language is unusable. This is the reason many of these languages are transcompiled into JavaScript.

By using an alternative we could have gained some client-side speed, which we don't care about as much as other applications might, since what slows the application down is intensive back-end computations for registration. We could also have used some specific functionality that these languages might have, but these advantages would have to compete with the huge community support of JavaScript, we deemed it not worth it and kept JavaScript.

What we did use instead of alternatives were JavaScript libraries which simplifies our use of JavaScript or extend javascript such as jQuery in conjunction with AJAX.

Separately, for integrating variables coming from Django we had to use the Django template language, which is translated to the template which in this case is HTML, but it can also be used to generate JavaScript or even XML.

jQuery

JQuery is a JavaScript library designed to simplify some frequently used functionalities of JavaScript.

Though it is technically a library we include it in this section because it does not really add functionalities, but it fundamentally changes how the language is used.

Paraphrasing the *jQuery Cookbook*[19]:
The jQuery philosophy is **"Write less, do more".**
This translates to three concepts that are easier to write in jQuery:

- Selecting elements and applying jQuery methods on them.
- Chaining multiple jQuery methods on elements.
- Using the jQuery wrapper and implicit iteration. This means that if the jQuery selector selects multiple elements, a method can implicitly iterate over them.

```
function deleteButton(index){
    $('div#componentDiv'+index).remove()
}

$('button#deleteButton'+currentIndex).click(function(){
    deleteButton(currentIndex)
});
```

Listing 6.1: example of jQuery use in our code. We tried to do this in javascript previously and it uses at least three times as many lines of code

To illustrate how jQuery can save time and efforts, the code in 6.1 is something we originally tried in JavaScript but found much simpler in jQuery (in the end we use a different implementation so this jQuery code is not in the web app).

What it does is:

- Create a function deleteButton. This function will, for a parameter index:
  - Iterate over every div which is named "componentDiv[index]"
  - Remove them all
- Find a button named "deleteButton[currentIndex]
- Associate to it's click the function deleteButton with currentIndex as a parameter

In JavaScript this is doable, it just takes many more lines of code for the same result.

Everything we do in jQuery we could have done in JavaScript as well, it doesn't offer any new capabilities, it just makes the code much easier to read and much shorter.

Another advantage of jQuery is that it offers functions for AJAX functionality.

AJAX

AJAX stands for 'Asynchronous JavaScript and XML'.[60]

The goal is to make the web page update asynchronously without having to refresh the whole page.

It achieves this by communicating with the server "behind the scenes".

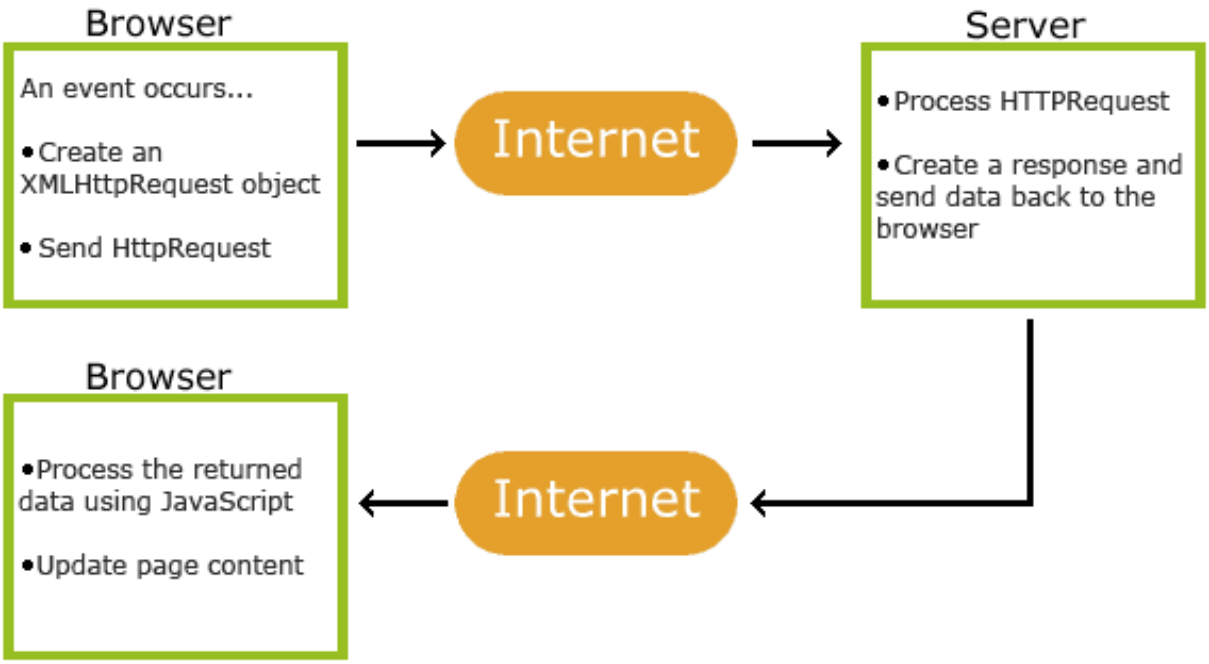

Figure 6.1: illustration of AJAX taken on https://www.w3schools.com/xml/ajax.gif on July 12, 2018

In figure 6.1, we can see how the browser and server communicate "behind the scene".

We use it to dynamically change the parameter maps in the parameter picker which was detailed in the chapter "our solution"4.

More specifically: When the user goes into the parameter picker, he does not download the entirety of the possible parameters, however, when he picks a particular top level parameter (such as "transform") an AJAX call is made to the server, which requests all the possible "transform" options and puts them in a selector.

If we couldn't use AJAX, we would have had to either refresh the whole page every time a parameter is selected or download the entirety of the possible parameters when the page is loaded. Both of these solutions are not practical.

Another use is to immediately answer when a registration or an upload of DICOM is started. For example, when the user wants to upload a DICOM image, the upload might take several minutes. Instead of redirecting the user, we grant him with a message saying that the upload worked (or failed).

6.2.3 Data Formatting and Markup languages

HTML

HTML is used in almost all web applications. It is a markup language that defines how the content of a webpage is displayed.[61]

We use specifically HTML5, but in reality, we don't use any specific HTML5 feature, we use HTML5 to keep up with time because it is the latest version.

Additionally, some parts of our HTML code are generated in JavaScript. For example, the amount of pre-existing HTML on the parameter picker page is dwarfed by the amount generated by the JavaScript.

Some part of our HTML code is also created by the Django Template Language.

Django Template Language

The Django template language allows us to introduce into HTML pages (in Django these are stored as templates) some variable, filters, loops, conditions etc... which will be evaluated server-side before being sent to the user.[41]

The syntax is quite simple, it's between curly braces, for example, to include a python variable in the HTML:

{{variable}}

The capabilities of this language are limited because it is evaluated server side, so once the page is sent to the client, the expression in Django Template Language have to have been already evaluated.

It can still be useful to include the value of some variable in the code in a simple way. It's also possible to perform many common operations, such as loops, conditions, etc.

JSON

JSON is a file format used to transfer JavaScript objects, though it can be used for other purposes.

The advantage for us is that it is very easy to generate JSON files in python using the json python module.

Due to this, we use JSON extensively:

- Cornerstone requires a JSON file that we generate in python and then transfer to cornerstone.
- The parameter picker uses a JSON file to transfer objects. These are also generated in python.

JSON is a very simple and lightweight file format, it contains only either objects comprised of a collection of name:value pairs, or an array of values.[62] It is also extremely easy to parse in javascript since it's essentially a javascript object saved in a file.

The main alternative is using XML files. While we use XML implicitly by doing XMLHttpRequest. When given the choice, we prefer JSON since it's lighter and easier to parse.

6.2.4 Appearance

Bootstrap

To improve the appearance of the web application, we use bootstrap.

Bootstrap is a front-end framework that provides CSS, javascript and other functionalities to enhance the appearance of the application.[63]

A major advantage of bootstrap is that it can be made responsive, meaning that a website created with bootstrap will look good regardless of screen size (provided bootstrap is used properly). This is not always true, some attention needs to be paid to responsiveness to ensure bootstrap responsiveness is working properly.

Using bootstrap allowed us to avoid doing our own CSS, which is a task best left to graphic designers since it defines the appearance of the web application and that appearance has to respect many graphical design rules and conventions to look good. We still do use some CSS, but only for some slight modifying some bootstrap behaviors and appearance, we avoid aligning items, doing tables, etc... in CSS.

6.3 Back-end Framework

To develop our solution we used Django, a python framework for the creation of web applications.

"A framework is a reusable, "semi-complete" application that can be specialized to produce custom applications." [18]

"The benefits of object-oriented application frameworks stem from the modularity, reusability, extensibility, and inversion of control they provide to developers."[18]

Let's define and explain the advantage of these three concepts.

Modularity means that the software can be divided into a lot of small parts. It can easily be seen by the fact that Django separates the whole "web application" into smaller applications each dealing with one specific thing. Also, the modular design is pre-existing, there is a division between the database, the templates, the views, the forms etc... within Django.

Thanks to this, we don't have to come up with our own division into modules, the architecture is pre-determined to a certain extent.

reusability translate to the possibility to use Django components as well as we can also use external applications that have been created for Django. Furthermore, our own components, applications, template and views can be reused easily.

Extensibility means that it's possible to modify the behavior of some of the modules and extend their functionalities, though we don't really do that.

Inversion of control means that the framework rather than the application determine which methods to call when an event is happening.

In short, it saves us a lot of time by having pre-created code that deals with all the "generic" part of a web application.

In exchange, there might be a little less flexibility, but in our case, the architecture can be very generic and will still perform exactly what we need.

6.3.1 Django

Django is a free and open source framework for web development. [41]

Its basis is an MVC framework. That consists of a model view controller, it does not have a user-created controller, instead, the frameworks take care of the controller and the developer creates the URLs which are linked to the views by Django. Due to that distinction, it has been called an MVT framework, though it is still considered by many as MVC.

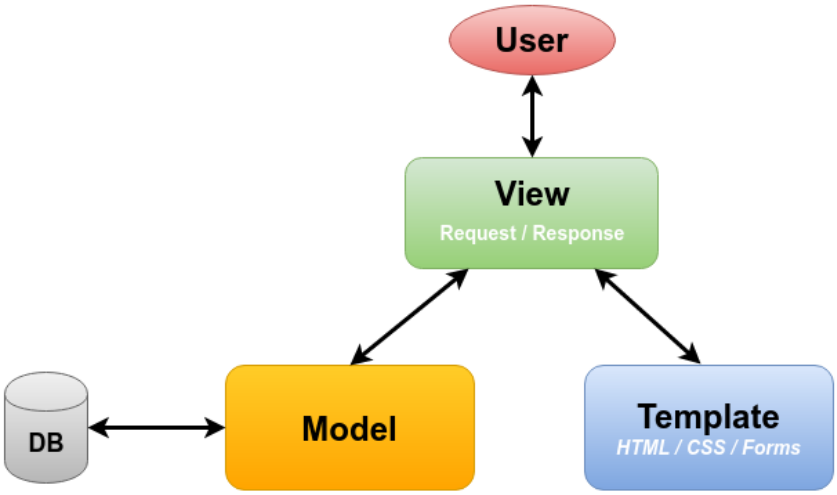

Figure 6.2: illustration of MVC taken from `https://sudoseed.wordpress.com/2017/06/30/web-framework-wars-django-vs-ruby-on-rails/` on July 31, 2018

MVC is an architectural pattern that divides a web application into three parts:

- The model represents the information saved in our database. It allows the access of the information, modifications and adding new entries. When adding new entries it needs to stay conform with some criterion. In our case, we work with multiple elements, a user can be added via the user creation web page, they can upload DICOM files which is then transformed, they can create parameter maps for the registrations which get added using the model. (The model can be scaled to add more).

  In Django data in the model is saved on an SQLite database, though media files including images (including DICOM images) are stored separately in a folder on the hard drive.

- The view. The view consists of what can be seen by the users. For example, a simple web page. However, it also consists of collecting all events and actions taken by the user. So a click on a link, submitting the parameter map and generally all interactions of the web pages. In our case, cornerstone (front-end viewer of DICOM) is also part of the view.

- The controller handles every event of the user. It reacts depending on the request of the user and eventually communicates with the model and the view. It is the controller which tells the view what needs to be shown.

MVT is slightly different: the controller is dealt with in the backend, instead, the view regroups data from the templates and the model together to give it to the users through URLs that are defined separately.

Template in Django can contain some code in the Django template language. Before sending the template, the framework will analyze it and execute it as if it were an executable code file to generate HTML or JavaScript. This allows the HTML file to have code and gives it access to variables, conditional statements (if/else), loops (while/for) and more...

A feature of Django worth mentioning is signals. This allows different applications to get informed when an action is taken from somewhere else. "In a nutshell, signals allow certain senders to notify a set of receivers that some action has taken place"[46]. In our case, we used it to delete files from the server hard drive and not just from database SQL. If we were to override the delete method from the model Scan to delete the file using os.remove(), it would not be properly called if it was a cascading delete or a grouped query delete. So in our case, we use Django.db.models.signal.post_delete to trigger a method that will go delete the file as shown in the code snippet.

Listing 6.2: Signal usage

```
def autoDeleteFileOnDelete(sender, instance, **kwargs):
    if instance.zipFile:
        if os.path.isfile(instance.zipFile.path):
            os.remove(instance.zipFile.path)

    if instance.id:
        if os.path.isdir(settings.MEDIA_ROOT + '/dicom/' + instance.id):
            rmtree(settings.MEDIA_ROOT + '/dicom/' + instance.id)

post_delete.connect(autoDeleteFileOnDelete, sender=Scan)
```

6.4 JavaScript medical image viewer: Cornerstone

As explained in the GitHub's readme: "Cornerstone is an open source project with a goal to deliver a complete web-based medical imaging platform."[22]

One of the objectives of cornerstone is to let the developer have as much freedom as possible for the image format and the back-end, while still providing a complete viewer.

Let's look at the main alternatives and their shortfalls:

- DWV[21] (DICOM Web Viewer) is "an open source zero footprint medical image viewer library."

  Like cornerstone, it doesn't make any assumptions about the backend.

  In reality, these are very similar, the authors have been in talks in a public google group to join efforts in some part of their code (namely the DICOM parser)[20], but during said discussion, the cornerstone author seems convinced his solution (at least for the DICOM parser) is faster, used in FDA approved products, and feature complete. The same can't be said of DWV.

I think both viewers would have been okay to use, however, cornerstone seems to have a slight edge.

- "Papaya is a pure JavaScript medical research image viewer, supporting DICOM and NIFTI formats, compatible across a range of web browsers." [23]

  It is a web application, it includes backend and frontend though the backend is also in javascript

  It handles viewing the images in 3D, which can be great.

  The first issue we had is with the fact that it's a complete application and not just a frontend, it makes integrating it with the rest of our work much harder because it is not a Django application.

  The second issue is that the license is custom. Both DWV and Cornerstone use well known licenses (MIT for cornerstone and GPL for DWV) however, Papaya include a license that seems to have been created in a custom way.

  It is also specified that the product is not for clinical use. While we aren't sure our application will ever be used in a clinical setting, it is best to not close doors on ourselves.

Because of these shortfalls, we decided it was best to use the cornerstone viewer.

Cornerstone is divided into several parts, three of which we use directly: "cornerstone" is the library in charge of displaying medical images, "cornerstone tools" are tools that can add functionalities to the viewer, and "cornerstone loaders" are several loaders which can load different types of images in different ways. We use wado-uri because we load DICOM instances over HTTP.

At first, we created a viewer ourselves, but we discovered there exist fully featured open-source solutions using cornerstone which would mean we don't have to develop each part individually.

Cornerstone is designed to be a "component that can be used as part of larger and more complex applications"[22] and not a standalone viewer. It is agnostic in term of image format, transport, and interaction paradigm. For these, you need to either develop a solution yourselves or use some of the tools developed by the same author.

However, there exist several study viewers that use various cornerstone libraries.

The Open Health Imaging Foundation[24] developed a viewer using cornerstone called the OHIF viewer. The OHIF viewer uses various cornerstone tools to be a fully featured viewer including a backend, a frontend, and many other tools to enhance the display of medical images.

However it contains a backend and a frontend and we wanted to integrate everything in our own backend, which would have required a decent amount of work.

Instead, we looked into the "cornerstone demo"[25]. Cornerstone Demo is a demonstration application that is designed to demonstrate the capabilities of cornerstone. It is done by the author of cornerstone.

"cornerstone demo" makes assumptions about the backend it isn't implemented. We had to adapt to these assumptions about the backend but it wasn't too restricting. For example, we needed to create and send JSON files containing information about the DICOM files.

The cornerstone features are further detailed in the chapter "Our solution" 4, but needless to say that developing all of them ourselves would have required a decent amount of time.

6.5 SimpleElastix

As stated previously, elastix is a "collection of algorithm that are commonly used to solve medical image registration problems" [9]. In practice, it is a software that can be used to perform registration using various algorithms.

Elastix only does intensity based parametric registration. Intensity based means that it registers on a complete image, not just a set of points (which would be feature based registration) , and parametric means the transformation can be represented with a formula that depends on a limited amount of parameters. While, non-parametric on the contrary, defines a displacement vector for every voxel.

6.5.1 Content of elastix

Elastix provides a library, a command-line interface, and separately simpleElastix provides wrappers for various languages.

There exist visualizers that include a module for elastix in them to perform registration and visualize in the same tool, but this paper will focus on the standalone elastix.

Elastix is mostly based on the Insight Segmentation and Registration Toolkit (ITK), although it offers some algorithms that are not included in the insight toolkit, and conversely doesn't offer all the possibilities that ITK offer. ITK is not focused solely on registration, but on all possible medical image processing, it is a much bigger tool, with more possibilities. Although most elastix registration algorithms come directly from ITK, some of them have been developed directly for elastix.

Elastix is capable of registration, segmentation, and generic usage of ITK filters; however, we will focus mainly on the registration part of elastix.

Let's examine how elastix fares against other tools for registrations, we will also explore how it works and its main features.

6.5.2 Features

Elastix has great features for registration, though it doesn't include a viewer or a graphical interface by itself, it should be used in conjunction with other tools such as 3Dslicer to compensate for the lack of these features.

List of main features

- Elastix performs intensity-based parametric image registration.
- It can open all the image formats supported by the ITK, which is all of the commonly used ones.
- It can be used in code (in C++ or in another language using a wrapper) or in command-line which can be great for scripting.
- Parameter maps are human-readable and can be shared easily. There exists a wiki where parameter maps used for various tasks are grouped.

- The design is modular, allowing for easy extension by adding a new module. This is also what allows users to pick algorithm/metric/optimizers/etc ... independently. A problem with this is that nothing prevents a user from picking a particularly bad combination, so users should be careful. There exist a few default parameter maps for beginners.
- All ITK tools for registration plus some elastix-specific modules are available.
- Elastix is open source. In comparison, some of the tools used in hospitals are costly. Elastix being open source means it can be acquired for free.

6.5.3 Comparison with competitors

The paper *A survey of medical image processing tools* [26] compares elastix with other medical image processing tools with a focus on the features that are available, andeElastix is among 3 out of 15 which support all tested types of images (MRI, Ultrasounds, X-Ray, fMRI, PET, CT-scan, EEG, Mammogram and 3D images). This paper also describes elastix as containing a and a viewer, which is factually wrong, as described in the elastix documentation: [16]. However, it is possible that the authors used a version of elastix that was integrated into a viewer such as 3Dslicer.

In general, this paper seems very poorly done, but it is the only comparison of registration tools based on features and not on the performance we could find.

Other advantages of elastix include: It is fully open source, it also includes libraries for other languages than its native language (C++) including python, java and R, it's also multiplatform and works with Linux, Windows and Mac OS X (some other registration tools are not available on all platforms).

The modular design of elastix is great to test and compare different registration method efficiently, it also allows integrating user modules, and the command-line interface allows users unfamiliar with programming to be able to use it without having to write code.
For example, it is very easy for an elastix user to compare two different metrics, these are two different modules and just changing the metric on the parameter map should do the trick. It requires very little effort.

While it is frequent to write code for elastix to make full usage of all the functions of elastix; the command line tool is very powerful and sufficient to use most elastix functionalities. It can parses parameter files and then it applies the registration algorithm described in these parameter files.

The parameter files are very simple in syntax, in it, the user should describe the different elastix module he wants to use, some parameters that these modules need as well as some information about the file used.

Most parameters have a default value (that varies depending on the value of other parameters). The default value will be used if the user doesn't specify a value.

An advantage of this parameter file structure is that they can be shared among users, which means that if someone finds a good set of parameters for a particular problem, e.g. interpatient monomodal CT scans of brains, he can share these parameters with ease.

There exist a wiki containing a collection of parameter files used in various papers[30]. This can be very practical for a relatively new user that want a better registration than what generic default parameters offer but doesn't want to learn all the possible module to find by himself the

best one.

The paper accompanying the initial release of the toolbox [9] offers some example of usage of elastix by using the command line options to script a handful of comparison between various parameters and their performances. It is also a recommended read for anyone wanting to try the toolbox.

```
(Registration "MultiResolutionRegistration")
(Interpolator "BSplineInterpolator")
(ResampleInterpolator "FinalBSplineInterpolator")
(Resampler "DefaultResampler")
(FixedImagePyramid "FixedRecursiveImagePyramid")
(MovingImagePyramid "MovingRecursiveImagePyramid")
(Optimizer "AdaptiveStochasticGradientDescent")
(Transform "TranslationTransform")
(Metric "AdvancedMattesMutualInformation")
```

Figure 6.3: Example of a parameter file. Its human-readable format is great for sharing and reusability. Parameter maps can get much more complex, this is simplified for the sake of the example

6.5.4 Usability

Despite the troubles we had during the installation, elastix is very easy to use. The ability to create human-readable parameter maps and to feed them to elastix is practical. Providing libraries for different language is also handy. Since we used python as a programming language, we decided to use it, and simpleElastix, a python wrapper allows us to call elastix and ITK function within python. But there exist many wrappers for other languages including Matlab, Lua, and many more.

The lack of viewer isn't detrimental given the possibility to just open a registered image with any viewer, developing a viewer themselves would use resources unnecessarily because there exist a variety of good viewers.

There lacks a truly machine-readable list of possible parameters. While there exist such a list generated by dioxygen, it isn't fully standardized, and it might be hard to parse all the information about every possible parameter using a script which can be very useful for automation purpose. (The documentation is generated from comments in the code). The documentation contains information about most modules, but there are some, where it is insufficient to understand how to use a module.

Another issue is that it lacks a central error handling mechanism. Errors are printed by various part of the toolbox, and it's hard to parse them automatically or to list all the possible errors.

Elastix gives off errors frequently. It prefers to exit with an error than performing a registration that gives a bad result. Trying to register two images with no similarities will create an error. For example, a scan of the brain and a scan of the lung. There are so many possible errors that it's hard to even list them, and many come from the underlying ITK library and are therefore not parsed at all by elastix (but can be output in a log file).

The positive effect of exiting when the result is insufficient is that it avoids using resources unnecessarily, but it can be quite frustrating when testing the toolbox.

The error messages are sometimes not very precise (some errors could be caused by several loosely related problems), and sometimes the error comes from ITK and there is no elastix explanation for the problem.

6.5.5 SimpleElastix differences with Elastix

SimpleElastix is an extension of simpleITK which adds a wrapper around elastix (and therefore access to elastix) for various languages including python.

Its goal is to increase the usability of simpleElastix, for this, it provides two things[31]:

- "A low level ITK-filter based interface for elastix [...]"

  These are a way to access elastix classes using an interface similar to the one used in ITK.

  This is done to later simplify interoperability between these two libraries.

- "A facade interface to these filters that enables SimpleITK to build elastix and Transformix bindings for Python [...](and other languages)."

  This refers to a "facade" design pattern which hides the internal mechanisms of the algorithms.

  There are actually two facades, one functional and one object-oriented. The object-oriented one has more customization options.

- "Preconfigured registration methods that serve as starting points for tuning registration algorithms to domain specific applications."

So more than just a wrapper, simpleElastix enables interoperability with simpleITK objects and also adds features that simplify the use of the toolbox, such as preconfigured registration methods and a simplified facade interface.

6.6 DICOM

DICOM is "a standard for handling, storing, printing, and transmitting information in medical imaging, it includes a file format definition and a network communications protocol."[45]

In our software, we use the file format part of it and ignore the network communication protocol part because it is pretty complex, might require accommodations on the user's computer which we want to avoid as a web app.

The problem that comes with it is that our web app might not be used in certain medical settings which set a restriction on image transfer.

The file format part of it is actually very unpractical to handle because it contains many fields that are optional and often not filled, therefore it's hard to know exactly what to rely on.

For example, to get the slice ordering from a DICOM file, it sounds practical to use "slice location" but sometimes it does not exist, "instance number" also sounds useful but isn't guaranteed to be unique and contains no information about slice number, the ones that are useful are "Image Patient Position" and "Image Orientation Patient".

DICOM is a very powerful and very complete specification. It has many uses: it can be used to store more than just images including videos, metadatas, and many more, it is everywhere in the medical field, sometimes it's the only format that is allowed in storage servers.

Despite these advantages, we felt like using DICOM might have been a mistake: The format is hard to respect properly because it's is engineered to be secure, and capable of handling a plethora of files created by many different devices with different capabilities.

To be able to use DICOM despite these shortcoming, we had to make compromises and not respect some parts of the specification such as the aforementioned communication protocol, another file format such as nifty, while possessing lower capabilities, might have been better suited for the project because we would have been able to trivially respect the specifications.

6.7 Summary

In this chapter, we reviewed the various programming languages and tools that we used.

We choose state of the art tools after carefully reviewing the alternatives. Some choices were intertwined, for example: choosing to use Django meant we had to use python and would favorize simpleElastix.

In general, we choose the best alternative, or at the very least one that is competitive.

Some choices were hard to make, for example, there are toolboxes that are competitive with elastix and viewers that are competitive with cornerstone, but we did review them and assess that our choices were viable if not the best.

There are more libraries that are sparsely used in our project but it would be hard to mention all of them, some had no alternatives, or the alternatives would have the same result and most of them are self-evident such as the "re" python module for regular expressions or "pydicom" for reading DICOM's file metadata.

Chapter 7

Architecture

In Django the architecture is MVC/MVT, this is explained in the chapter on technologies used 6 in the section about Django.

Besides the division in Models/Views/Controller or Templates, there is a division of applications that can be done. It is suggested that each application should be doing a single task and no other. It also contains its own models, views and templates.

There are also internal subdivisions in the models and in the templates. The models are in the form of tables. While, the templates are written, among other languages, in Django template language. They use template inheritance, which forms a template inheritance tree.

But before we talk about this internal division, let's first present the basic Django architecture in term of the files and the code.

7.1 The Django architecture

About the content, in term of files and folders of the project, we created them by respecting the Django architecture.

This section does not contain many choices we made, but rather presents the division of the files and folders required for Django.

The project is divided into independent applications, which each perform a specific task. It also has global files.[41]

Files of Applications

A Django web application is a called a project and is composed of many "applications" which are subdivisions within the project.

A Django application as said, is supposed to do one thing only. The goal is to provide additional modularity for the project.

Each Django application is structured into models, views and templates, as per the MVT architecture, and is stored in its own unique folder.

Concretely, this translates to these files:

- models.py which contains the models

- views.py which contains the views
- templates/ a folder which contains the HTML templates
- urls.py containing the link between urls and views. It contains a list of URLS and which function in views.py should be called if said url is requested.
- static/ an optional folder containing static files such as JavaScript files, stylesheets, or images. The files here are not uploaded by any user and hence static on the application.

These files are included in each application, except the static folder which is optional (some application might not need static resources).

Global files and settings

Although most of the content of the web application/project is inside the various application folders, there is still some global content which is stored in the root folder.

- media/ a folder containing the media files. If a file is stored into the database, the SQL database will contain a path towards the file which will be stored in media. These files are usually uploaded by users and are subject to change.
- A global static folder containing the static files that aren't specific to an application such as jquery and bootstrap.
- The database itself (db.sqlite3)

  Although each application has its own models.py which defines the tables and row of the database. The content is still saved into the database.
- manage.py which is the script used to start the application and for other management tasks such as changing the database's data, and initiate new apps.

There is also a "special" application, in our case called registrator, which contains special information but doesn't perform a specific task beyond that.

For us, this application is called registrator and it contains notably "settings.py" which has all the Django settings specific to our application. It also contains a "urls.py" files containing URLs referring to each application.

7.2 Division into applications

As we mentioned above, the project is divided into applications, each performing a specific task.

In our case, it translates into these applications:

- display

  This application contains cornerstone and is tasked with displaying the DICOM images. It contains the result of registration, too.

  It also has studies in its model. Studies regroup the scans/DICOM images to be viewed together with Cornerstone. As studies are required for Cornerstone to function. The studies management is in the display application.

  A user can manage studies. They can create, edit and delete studies.
- elastix

  This application contains the necessary functions to perform a registration using elastix. It also comprises an interface to view all the current registration and will show which ones failed and which ones succeeded.

- filemanagement

  This application is tasked with allowing a user to manage their DICOM images.

  It uses a subprocess to deal with uncompressing zip files because uncompressing can be fairly slow. Subprocesses can be fine-tuned by giving them less processing power or delaying them in order to minimize their impact on performance.

  We don't respect the DICOM file transfer specifications, as said, it would require to ask the clients to install extra software which negates the advantage of having a web application.

- home

  This contains the homepage. There exists two homepages, one for users that are logged in and one for unregistered users. Which homepage to show is detected automatically. The home also handles the login and registration of accounts.

- parameters

  This contains management of parameters. The users can create, edit and delete the parametermaps.

7.3 The database

Our database is an SQLite database represented in the applications using Django's ORM.

SQLite is a self-contained, highly-reliable, embedded, full-featured, public-domain, SQL database engine[48]. For development it is sufficient, but in production, it should be substituted with a more scalable database depending on the needs of the production. Our advice is either mySQL or PostgreSQL. However, using SQLite in the development build is the one we will make open source. It is more flexible and makes it much easier for other developers to set up the software as it requires no preinstallation.

A disadvantage of SQLite, in our case, is that it locks the database when an operation is being done. So, if a user tries to perform too many operations at the same time, it might output an error. This is the reason why it should be substituted for production. Changing the database is easy and pretty transparent with Django.

Django creates an ORM or an Object-Relational Mapping which means that we manipulate objects and not tables directly. The attributes of these objects are stored as columns in the database, while each object represents a row in a table.

These objects are defined in the different application's "models.py" files.

A diagram representing every table and their relationship can be found on figure 7.1

It is easy to see that the database is not in a normal form. Normal forms[50] ensure that the tables are free of a certain type of redundancy depending on the form. However, it is not a concern here because the vast majority of our space is taken by the DICOM files and not the information on the database.

In fact, if users upload or create a single image, the database could represent less than 1% of the data, so redundancy could not really make a difference, as long as we ensure we don't store the image several times.

We will explain in the chapter on possible improvements 9 that a different architecture should be used if the web application is scaled massively, including possibly a key-value store for the DICOM files.

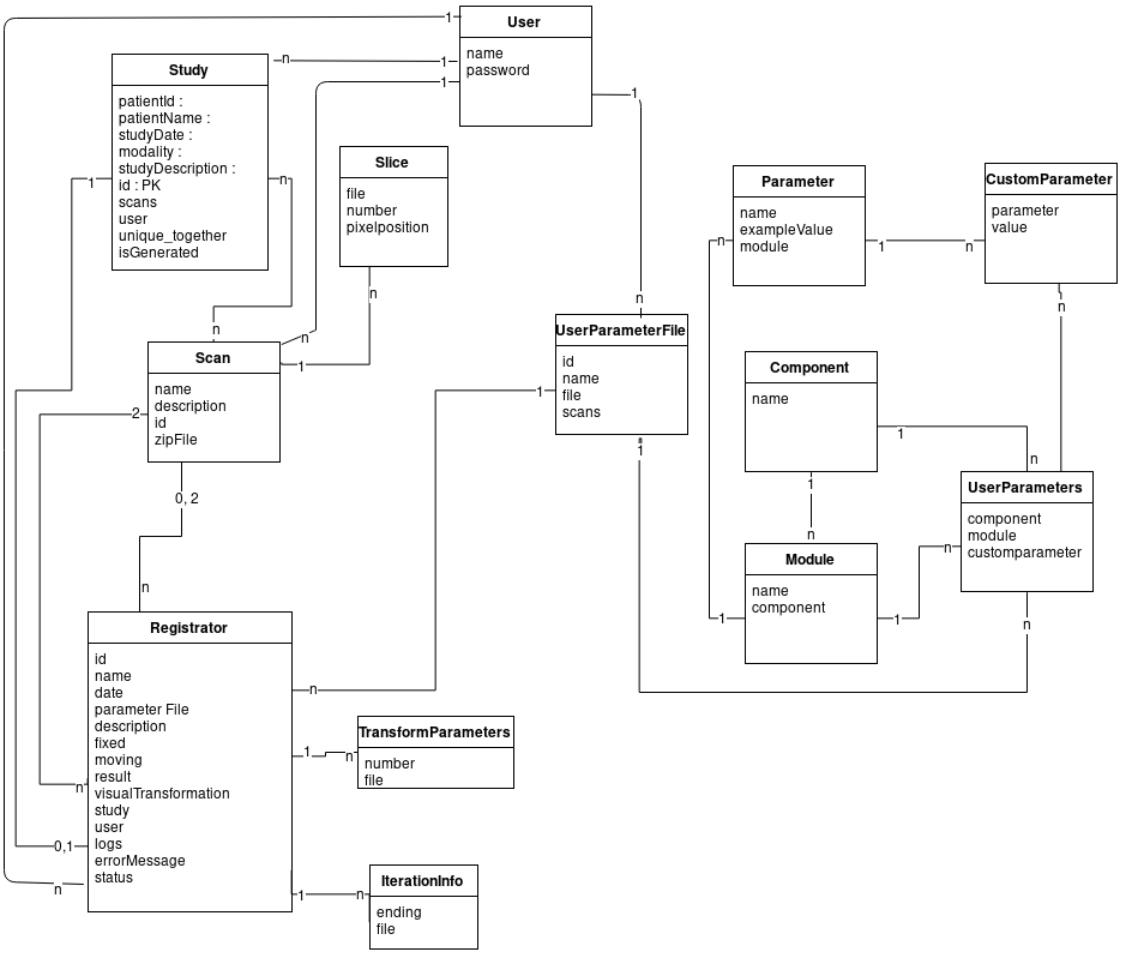

Figure 7.1: Database Entity-Relationship diagram

Since space is not an issue, there are some duplicates. For example, connecting the user to a registration even though it is possible to get that connection from the user column in the fixed image and the moving images. These duplications simplify the code, making the queries shorter and faster to execute.

Let's present the tables which, as said above, are represented as objects and their contents:

- **User** identifies the unique user by its name and password.
- **Study** contains the images that are meant to be displayed together.

  The study object representing the table is located in the "display" application
- **Registrator** contains the images and parameters used, as well as the result of registration.

  It is connected to two other tables named **IterationInfo** and **TransformParameters** which contain further files about successful registration.

  These tables are located in the "elastix" application
- **Scan** and **Slice** contain the image. Scan contains the entirety of the scan while slice contains each individual slice. This is represented using foreign keys.

Scan currently keeps a copy of the zip as it was uploaded.

This is located in the "filemanagement" application

- **Component**, **Module** and **Parameter** contains all the possible elastix parameters. "Component" contains the types of components used for registration, "Module" are each of the possible value for these modules, while "Parameter" contain the additional parameters that can be given to elastix.

  For example, "Transform" is a component of registration, "Affine" is a module (it's a type of transform), and there might be a supplementary parameter such as "NumberOfIteration" that is specific to this module.

  This is all in the "parameter" application. **UserParameterFile** contains the parameter map and the file is generated. The parameter map is contained as a many-to-many relationship with a **UserParameter** table. It contains a specific component and it's associated module. Additionally, it has a many-to-many relationship with **CustomParameter** which contains the module's parameters and their value.

  These are all in the parameter application.

7.4 The templates

7.2

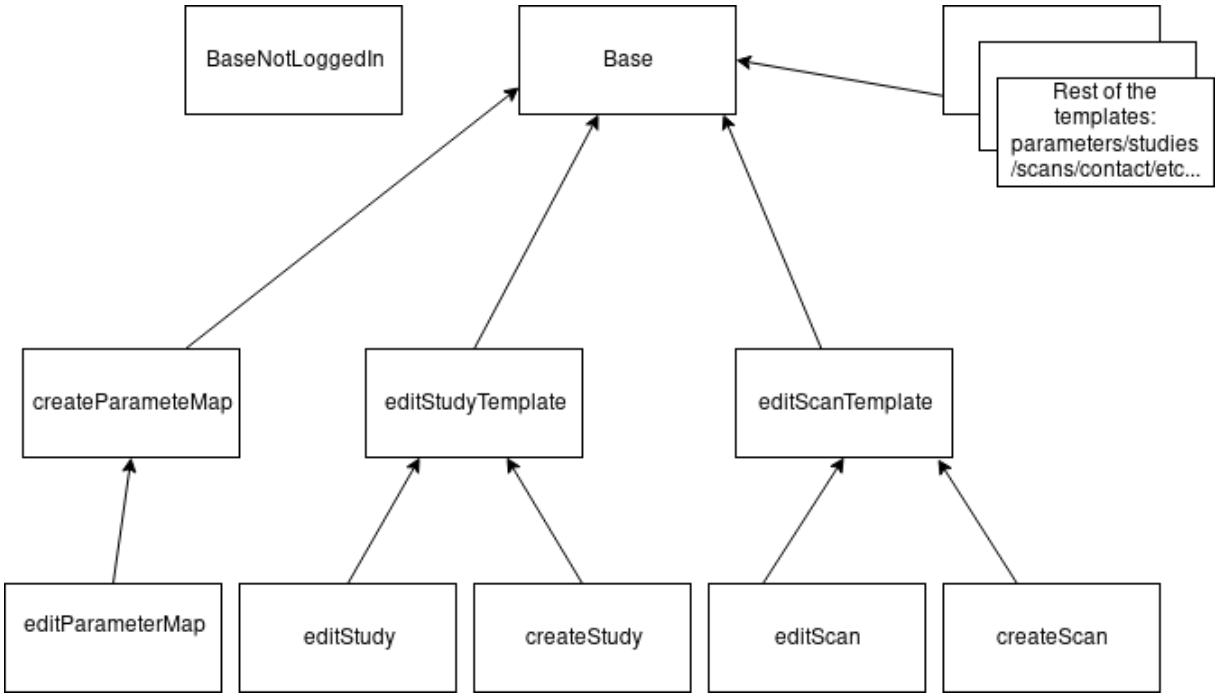

Figure 7.2: Our template inheritance tree

The templates in Django can use a "template inheritance tree" which means that some template will inherit content of others. [41]

In our case, we started with a "base" template. It contains essential functions, the top navbar and the sidebar that extended it most of our pages.

One level lower, there are also templates for editing and creating of files such as scans, studies and parameters. These inherit from a common template as to avoid code duplication.

7.5 Summary

In summary, for our architecture, we tried to mostly comply with the Django way of doing things. We kept our architecture simple and modular with divisions in the applications, models and templates.

For the database, we didn't hesitate to duplicate data, because of the relatively low percentage taken by everything that isn't the DICOM files.

For the templates, we re-used code when possible so that changes can be reflected on the whole website.

Chapter 8

Tests

Automated tests have many advantages over testing everything manually, therefore we decided to create some.

Many programming practices strongly recommend the use of automated testing, sometimes going as far as Test Driven Development. The alternative to automated testing is manual testing, which has many limitations, but for many parts of our software, automated testing is not practical or possible at all and therefore we still used manual testing.

Automated tests are appreciated because they help the programmers and maintainers in many ways including[44]:

Test coverage: It might be hard to cover all cases using manual testing, reduced testing time, reliability: The test will be performed the same way more reliably, less human efforts, increased fault detection, etc...

8.1 Different types of tests

There are several types of testing, many different categorizations exist, but for our purpose let's use a common categorization, which makes the distinction between automated Unit test, API tests and GUI tests.

Unit tests are code-level tests and are focused on testing individual functions. The objective is to be certain that the functions work for all kind of tests. While we would have loved to do unit testing on our code, in many cases our functions are either slow to execute (like importing DICOM or performing registration), hard to test using unit test because they rely on many environment variables (availability of scans and our JavaScript code often depends on user-generated content). In the end, we decided to focus on other types of testing such as GUI testing because of the technical limitations for unit testing.

API tests Are focused on the interaction between different software using an API. Our web application is meant to be accessed only by users using a GUI, therefore to use API tests, we would need to create an API that provides access to our backend functionalities without using the GUI. That would have required additional development time, therefore we decided to Avoid doing them.

We do use elastix calls and testing its interactions would make sense, however, elastix already has internal testing, therefore testing it ourselves would be redundant. Furthermore, registration takes a long time and it would take a month to test all edge case. It's also worth mentioning that elastix will intentionally return an error whenever the registration result is unlikely to be good and in many other cases, which means that the results of API testing would vary depending on

the image, and even on some randomness depending on the algorithm used.

GUI tests These tests are focused on the interaction between the user and the software at a GUI level. This is the most practical type of tests for us because our software is fundamentally a web-application relying on GUI for all interactions, and because it's possible to test many parts of the backend with only a few simple GUI tests.

In conclusion, we decided to focus on GUI test because Unit tests and API test would be unpractical, while still probably possible, and we thought we could cover our core functionalities with GUI test using a minimal amount of development time.

8.2 Selenium

Selenium Integrated Development Environment is a prototyping tool for writing selenium scripts.[64]

It consists of a browser extension which can record a user's action and replay them later as a test.

It is a very efficient way of doing automated testing since it (mostly) does not require coding, but it's perfectly fine since the end result is the same.

The recording might fail however in many circumstances, for example, if the web application requires uploading a file, or if selenium tries to differentiate a button using a CSS attribute that changes every execution.

In such cases, we had to modify the code to fix these issues.

8.3 Our tests

Since many of our pieces of software are interdependent, therefore, our tests have to be executed in a specific order:

- User registration
  Logging in and creating an account. It's necessary for the rest because an account is needed to access most of the web application's functionalities.
- Upload/editing of images
  The path used for the images, unfortunately, have to be absolute, therefore these tests need to be edited to work on another workstation.
- Studies
  This test creates studies, tests editing and deleting them.
- Parameters
  This is separated into two tests: one which tests the functionalities of the parameter picker, and the other which creates a parameter map that is usable for registration.
- Image registration
  As the name says it tests registration itself using the parameter map generated above. Registration takes a long time so this script is expected to run for about 30 minutes, and might not work if the images provided aren't good.

8.4 Summary

In conclusion, we made tests to check the core functionalities, further testing could be done to test edge cases and other minor functionalities. However, one has to be careful with GUI tests as the GUI is subject to changes and changes in the GUI will make the test fail and necessitate to remake the whole test.

The tests are very useful to check if anything went wrong after a change has been made in the backend. Before creating them we either had to manually test if nothing broke or if we didn't, we sometimes realized later that we broke something.

Our tests are however not as advantageous as they could be, because the backend is often slow (registration or DICOM image upload), the gain in speed are not that high compared to doing them manually, also our test coverage isn't that great currently but that can be improved.

Chapter 9

Possible improvements

The field of medical image registration is vast and there are an incredible amount of possibilities even just within image registration.

Even without extending the applications capabilities there are other possibilities for extending this work.

9.1 Improvement to the registration backend

Strictly in the field of registration, there are many directions we could take.

Firstly, elastix has many capabilities which are not fully able to utilize.

One of these capabilities is Masks. This is probably the most crucial one, good masks can improve massively the quality of registration.

The issue we had with integrating masks is that we had to either allow the user to upload a mask file which is complicated to create for a beginner or to find a way for users to create mask within our application which isn't easy either.

We decided to not try to allow upload of masks because our intended audience is partially beginners which would not understand what this is about, and we couldn't find a way to make it simple to understand.

Dealing with multiple images is another such capabilities that we cannot exploit.

It can be useful to have multiple images registered at once (groupwise registration), to "try to mitigate uncertainties associated with any one image by simultaneously registering all images in a population."[38]

The last capability we don't allow is point-set registration.

Point-set registration is a type of feature-based registration that uses a predetermined set of points as a basis for registration.

This is possible in simpleElastix and is also possible to do in our project, but all of the points coordinates have to be manually defined.

An extension of our thesis might be to make it simpler to do point-based registration by allowing the user to simply select the set of points they want directly from the viewer.

It's already possible in cornerstone (which we use for our viewer) to select a point and see its coordinates, but it doesn't get picked up by the backend.

Beyond just improving our backend using the possibilities offered by simpleElastix we could also use the possibilities offered by other registration tools.

For example, we already have simpleITK installed and it might be possible to exploit its other possibilities.

We could also allow the user to use other frameworks such as ANTSreg or niftyreg, and to compare the results visually.

9.2 Improvement to the backend beyond just registration

There are many fields in medical image analysis beyond registration. One of them which is close and has very similar tools is image segmentation.

"Image segmentation is the process of partitioning a digital image into multiple segments or sets of pixels, which are also known as super pixels. Basically, segmentation is used to simplify and/or analyze images. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images."[39]

Beyond segmentation, there are also many newer possibilities with development in machine learning and deep learning such as[40]

- Classification of the images
- Localization of anatomical features
- Detection (ex: of lesions)
- Segmentation using deep learning (it is possible using other techniques)
- Registration using deep learning (obviously it is possible using other techniques)

As you can see, deep learning has many possibilities, the problem is that it might require large amounts of data and processing power. Another issue is that the technologies are less old and established than the registration one, thus more subject to change.

9.3 Improvement to usability

Since we were handed DICOM images for testing and experimenting, we decided to make sure everything worked properly for DICOM, which proved to be a hard task.

However, there exist many formats that are not DICOM and can be used for registration, the most common one being the nifty format, which is actually the preferred format for simpleElastix, used in the example in the documentation[38].

There is also the possibility to extend our work beyond the field of strictly medical image registration and attempting to test it for other fields.

Nothing really restricts us from doing this already except for the fact that we currently only use the DICOM format which is rarely used outside of medical images and the lack of testing.

There are also some "Quality Of Life" that could be done:

- Allow users to connect using other ways such as google, twitters, etc.
- Simplify sharing parameter maps and results.

- Add a forum for help.
- Add a page for an organization which can share their images, results etc. between them.
- Make it possible to make the images public or private.
- etc.

There are infinite possibilities, but most of them require a decent amount of time to create and are a bit outside of our field of interest.

Another pretty important usability improvement would be the annotation of images. Currently, it's possible to annotate images using cornerstone, but these annotations are not saved to the backend.

9.4 Architecture Improvement

Currently, our web application is not extremely scalable, it has separated applications which are nice to keep a clean architecture, but to be able to scale massively it would be best to have separate microservice which communicate between them using, for example, the REST API.

Since we used Django to create our application, the simplest change would be the django REST framework which still uses django but is optimized to be used with the REST API.[65]

Obviously, if we want to create a scalable application, a different architecture would have to be thought of. Currently, our application is supposed to run on a single server and to be scalable it needs to be able to be separated into several independent microservices.

Another possible issue is how SQL database scale, which is pretty poor if we want to scale our database linearly, we would have to change and probably use an object store.

To illustrate a possible architecture, here is an example.
Our application would be divided between:

- A "light" microservice that serves files retrieved from the database, serves the templates and so on that can be duplicated.
- A separate microservice that deals with the registration process itself. It can be a big advantage for security to separate completely this microservice (on another VM or docker).

  The advantage of separating this microservice with the rest is that it can scale independently, and since it's likely that there would be a bigger need for computational power for this microservice than for the rest, it would be a waste to duplicate the entire application just to increase the processing allocated for registration.
- A SQL database containing the metadata of all the files as well as some user information.

  The idea is that metadata, even if scaled massively still does not use that much space compared to the actual data, so a SQL database would do the job.
- A noSQL key-value store containing the files and data.

  The key would be stored on the SQL database as a metadata on the files. This would make the database containing the values (DICOM files, parameter files, etc...) linearly scalable because noSQL database can be linearly scalable.

- Possibly a gateway capable of logging.

  Having a gateway that can gather statistics to improve the web application and that "hides" the internal structure from the outside (therefore increasing security) might be advisable.

An illustration of this proposed structure can be found on figure 9.1

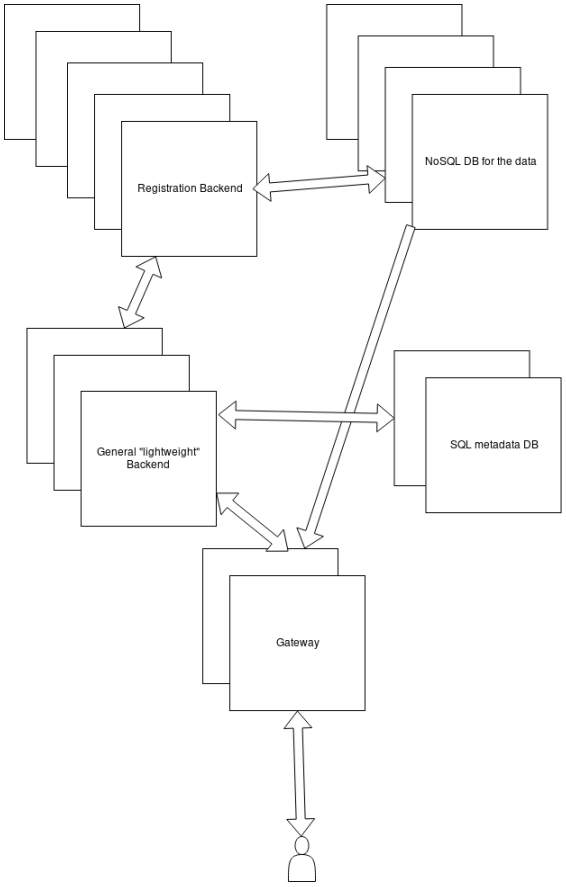

Figure 9.1: Example of possible scalable architecture

Chapter 10

Conclusion

We have explored the field of medical image registration and come up with a design thinking approach to find a software that could improve the experience of people working in the field.

Our main objective is clearly attained: provide a software of value for the field of medical image registration.

We used state of the art technologies relevant to registration including elastix and cornerstone, after reviewing alternatives. We also used many technologies that are competitive for the rest of the web application including jquery, ajax, Django and bootstrap.

We provide some new tools for registration, some of them by simple virtue of being a web application as opposed to a software executed via terminal command.

Our processes for coming up with this solution was design thinking and our process for creating the web application itself was agile programming with Scrum. In both cases, we had to adapt the process due to limitation, either in the size of our team or in our relatively low amount of contact with the industry which prevented a better emphasizing with the users.

Our architecture is kept simple and relevant by respecting the Django recommendations when possible and exploiting fully its possibilities.

We also created some automated tests to monitor changes and validate our application.

There are many options for improvement of our solution, including creating a better architecture for scalability, increasing the possibilities for registration and researching the architecture, but we believe our solution as it stands still provide a new advancement for usability over the previous ones.

We are conscientious that our thesis work can be improved by further research in order to understand, first, the exact demand of the market and then, to see which additional tools could be used to create a perfect software for making image registration more user-oriented.

Bibliography

[1] University of London, *List of open-source packages*, [Online]. Available: `http://www0.cs.ucl.ac.uk/opensource_mia_ws_2012/links.html`, [Accessed: 29-Jun-2018].

[2] *Website of the Insight Segmentation and Registration Toolkit* [Online]. Available: `itk.org` [Accessed: 1-Jul-2018].

[3] Bradley C. Lowekamp, David T. Chen, Luis Ibáñez and Daniel Blezek, *The Design of SimpleITK*, Front. Neuroinform., December 30, 2013

[4] Hans J. Johnson, Matthew M. McCormick, Luis Ibáñez and the Insight Software Consortium *The ITK Software Guide, Book 1 Introduction and Development Guidelines*

[5] Brian B. Avants, Nicholas J. Tustison, Michael Stauffer, Gang Song, Baohua Wu and James C. Gee, *The Insight ToolKit image registration framework*, Front. Neuroinform. April 28, 2014

[6] Marc Modat, Pankaj Daga, David Cash and Sébastien Ourselin. *KCL-BMEIS/niftyreg*, [Online]. Available: `https://github.com/KCL-BMEIS/niftyreg` [Accessed: 1-Jul-2018].

[7] Marc Modat and Jamie Robert McClelland, *Lung registration using the NiftyReg package*, Article · January 2010

[8] The documentation for niftyreg, [Online]. Available: `http://cmictig.cs.ucl.ac.uk/wiki/index.php/NiftyReg_documentation`, [Accessed: 30-Jun-2018].

[9] Stefan Klein, Marius Staring, Keelin Murphy, Max A. Viergever and Josien P.W. Pluim, *elastix: A Toolbox for Intensity-Based Medical Image Registration*, IEEE transactions on medical imaging, VOL. 29, NO. 1, January 2010

[10] Denis P. Shamonin, Esther E.Bron, Boudewijn P. F. Lelieveldt, Marion Smits, Stefan Klein and Marius Staring for the Alzheimer's Disease Neuroimaging Initiative, *Fast parallel image registration on CPU and GPU for diagnostic classification of Alzheimer's disease*, frontiers in neuroinformatics, January 16, 2014

[11] Milan Sonka, J. Michael Fitzpatrick, *Handbook of Medical Imaging, vol2. Medical Image Processing and Analysis*, SPIE Press

[12] Fatma El-Zahraa Ahmed El-Gamal, Mohammed Elmogy, Ahmed Atwan, *Current trends in medical image registration and fusion*, Egyptian Informatics Journal

[13] Lisa Gottesfeld Brown, *A Survey of Image Registration Techniques*, Department of Computer Science, Columbia University, New York, 1992

[14] Francisco Paulo Marques Oliveira & Joao Manuel R. S. Tavares, *Medical image registration: A review*, Computer Methods in Biomechanics and Biomedical Engineering · January 2014

[15] J. B. Antoine Maintz and Max A. Viergever, *A survey of medical image registration*, Medical Image Analysis(1998) volume 2, number 1, Oxford University Press

[16] Stefan Klein and Marius Staring, *Elastix the manual*, September 4, 2015

[17] Sebastian Nanz and Carlo A. Furia, *A Comparative Study of Programming Languages in Rosetta Code*, January 22, 2015

[18] Fayad, Mohamed, and Douglas C. Schmidt., *Object-oriented application frameworks.*, Communications of the ACM, 40.10 (1997): 32-38.

[19] Cody Lindley., *jQuery Cookbook*, "O'Reilly Media, Inc.", November 9, 2009

[20] Chris Hafey, Ives Martel, *Which js library dicom is more advanced? | google group*, [Online]. Available: `https://groups.google.com/forum/#!topic/comp.protocols.dicom/YWBEVYVgaDM`, [Accessed: 13-Jul-2018].

[21] ivmartel *dwv git repository* [Online]. Available: `https://github.com/ivmartel/dwv` [Accessed: 13-Jul-2018].

[22] chafey *cornerstone git repository* [Online]. Available: `https://github.com/cornerstonejs/cornerstone` [Accessed: 13-Jul-2018].

[23] martinezmj *papaya git repository* [Online]. Available: `https://github.com/rii-mango/Papaya` [Accessed: 13-Jul-2018].

[24] Open Health Imaging Fundation *OHIF website* [Online]. Available: `ohif.org` [Accessed: 14-Jul-2018].

[25] chafey *cornerstone demo github page* [Online]. Available: `https://github.com/chafey/cornerstoneDemo` [Accessed: 14-Jul-2018].

[26] Lee Lay Khoon, Liew Siau Chuin *A survey of medical image processing tools*. International Journal of Software Engineering & Computer Systems (IJSECS), ISSN: 2289-8522, Volume 2, February 2016.

[27] Gerald SMA Kerner,corresponding author Alexander Fischer, Michel JB Koole, Jan Pruim, and Harry JM Groen *Evaluation of elastix-based propagated align algorithm for VOI- and voxel-based analysis of longitudinal F-FDG PET/CT data from patients with non-small cell lung cancer (NSCLC).* `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4385310/#CR22` Published online March 21, 2015. doi: 10.1186/s13550-015-0089-z

[28] Murphy et al., *Evaluation of Registration Methods on Thoracic CT: The EMPIRE10 Challenge*. IEEE Trans Med Imaging. November 2011;30(11):1901-20.

[29] Ramón Casero, *notes on elastix*, [Online]. Available: `https://github.com/rcasero/doc/wiki/Notes-on-SimpleElastix` [Accessed: 26-May-2018].

[30] *wiki elastix containing a database of parameter files*, [Online]. Available: `http://elastix.bigr.nl/wiki/index.php?title=Parameter_file_database&oldid=1747`

[Accessed: 29-May-2018].

[31] Kasper Marstal, Floris Berendsen, Marius Staring and Stefan Klein, *SimpleElastix: A user-friendly, multi-lingual library for medical image registration*, 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops

[32] Plattner, H. *An Introduction to Design Thinking Process Guide.*, The Institute of Design at Stanford. (2010).

[33] Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001)., *Manifesto for Agile Software Development* , [Online]. Available: `http://www.agilemanifesto.org/` [Accessed: 21-Jul-2018].

[34] Sutherland, J., & Schwaber, K. (2013). *The Scrum guide. the definitive guide to Scrum: The rules of the game.* [Online]. Available: `ScrumGuides.org`.

[35] Rubin, K. (n.d.).
Artifact. [Online]. Available: `http://www.innolution.com/resources/glossary/artifact` [Accessed: 23-Jul-2018].

[36] *Scrum Glossary. (2018).* [Online]. Available: `https://www.scrum.org/scrum-glossary` [Accessed: 25-Jul-2018].

[37] American Hospital Association *Cybersecurity and Hospitals* [Online]. Available: `https://www.aha.org/system/files/2017-12/ahaprimer-cyberandhosp.pdf` [Accessed: 27-Jul-2018].

[38] simpleElastix team *simple Elastix documentation* [Online]. Available: `http://simpleelastix.readthedocs.io/` [Accessed: 18-Jul-2018].

[39] Ramandeep Kaur and Jaswinder Kaur *Current Methods in Medical Image Segmentation: A Review* ICCS-2014

[40] Justin Ker, Lipo Wang, Jai Rao and Tchoyoson Lim *Deep Learning Applications in Medical Image Analysis* IEEE Access, vol. 6, pp. 9375-9389, 2018.

[41] Adrian Holovaty, Jacob K. Moss *Django -The definitive guide to django: Web development done right*

[42] Matlab documentation on imregister [Online]. Available: `https://ch.mathworks.com/help/images/ref/imregister.html` [Accessed: 27-Jul-2018].

[43] Waters, K. (2009). *Prioritization using moscow.* Agile Planning, 12, 31.

[44] Kai Petersen and Mika Mäntylä *Benefits and limitations of automated software testing: Systematic literature review and practitioner survey* Conference Paper · June 2012

[45] DICOM Library *The DICOM library website* [Online]. Available: `https://www.dicomlibrary.com/dicom/` [Accessed: 31-Jul-2018].

[46] *"Documentation."* Security in Django | Django Documentation | Django, [Online]. Available: `docs.djangoproject.com/en/2.0/topics/signals/`, [Accessed: 1-Aug-2018].

[47] Janssens, Guillaume *Registration models for tracking organs and tumors in highly deformable anatomies : applications to radiotherapy* Prom. : Macq, Benoit

[48] Hipp R et. al.. *Sqlite* North Carolina: SQLite Development Team; 2015.

[49] Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, Ghosh SS. *Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in Python.* Front. Neuroimform. 5:13. (2011)

[50] E.F. Codd, *Relational Model of Data for Large Shared Data Banks*, IBM Research Laboratory, San Jose, California

[51] Marius Staring, Stefan Klein, Johan H.C. Reiber, Wiro J. Niessen, and Berend C. Stoel, *Pulmonary Image Registration with elastix using a Standard Intensity-Based Algorithm*, EMPIRE10 website, August 7, 2018

[52] Yunle Yang, Jing Jin, *Similarity metric in medical image registration*, Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)

[53] Ke Nie, Cynthia Chuang, Neil Kirby, Steve Braunstein, and Jean Pouliot, *Site-specific deformable imaging registration algorithm selection using patient-based simulated deformations*, Department of Radiation Oncology, University of California, San Francisco, California, published March 22, 2013

[54] Brian B. Avants, Nick Tustison and Hans Johnson, *Advanced Normalization Tools (ANTS)*, July 10, 2014

[55] The matlab website, [Online]. Available: `https://ch.mathworks.com/help/images/intensity-based-automatic-image-registration.html`, [Accessed: 8-Aug-2018].

[56] Paul Yushkevich, Joseph Piven, Heather Cody, Sean Ho, James C. Gee, and Guido Gerig, *User-Guided Level Set Segmentation of Anatomical Structures with ITK-SNAP*, 2005

[57] G. van Rossum, *Python tutorial*, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[58] Abdulkadir KARACI, *A Performance Comparison Of C# 2013, Delphi, Xe6, And Python 3.4 Languages*, International Journal of Programming Languages and Applications (IJPLA) Vol.5, No.3, July 2015

[59] Douglas Crockford, *JavaScript: The Good Parts*, O'Reilly Media, Inc. ©2008

[60] Steven Holzner, *Ajaw, A Beginner's Guide*, Copyright © 2009 by The McGraw-Hill Companies

[61] World Wide Web Consortium on html & css, [Online]. Available: `https://www.w3.org/standards/webdesign/htmlcss` [Accessed: 08-Aug-2018], W3C.

[62] The JSON official website, [Online]. Available: `https://www.json.org/`

[63] The bootstrap official website, [Online]. Available: `https://getbootstrap.com/`

[64] Selenium Project, *Selenium Documentation Realease 1.0*, February 24, 2010

[65] Joel Vainikka *Full-stack web development using Django REST framework and React*, Metropolia University of Applied Sciences Bachelor of Engineering Information and Communications Technology Thesis, May 16, 2018

[66] Osimis Official Website [Online] `http://www.osimis.io/` [Accessed: 13-Aug-2018]

[67] Intuitim Website [Online] `http://intuitim.com` [Accessed: 13-Aug-2018]

Appendices

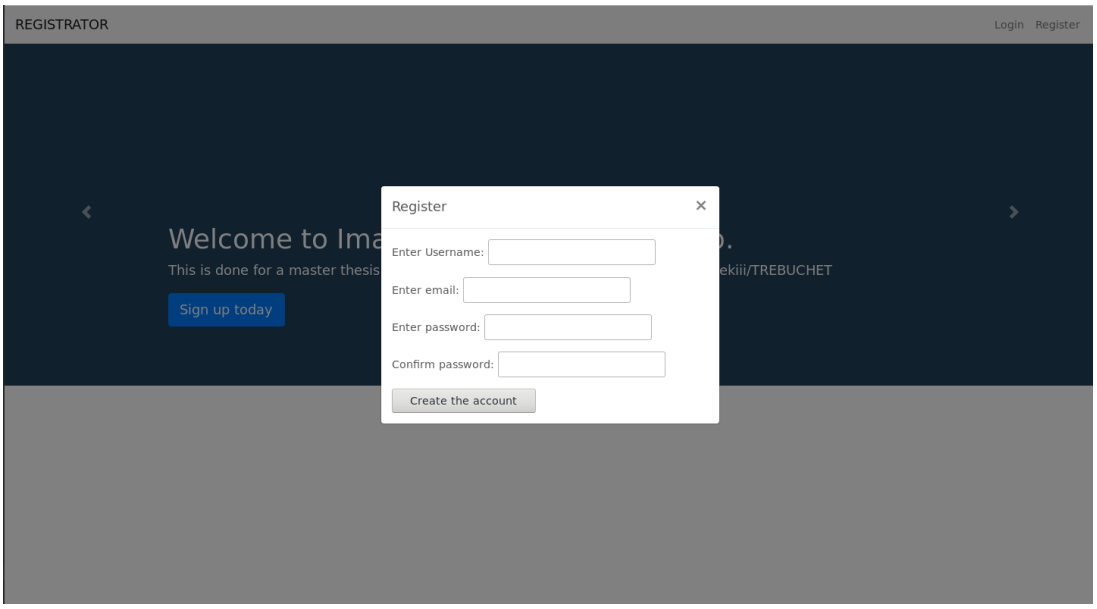

Figure 1: The user registration modal.

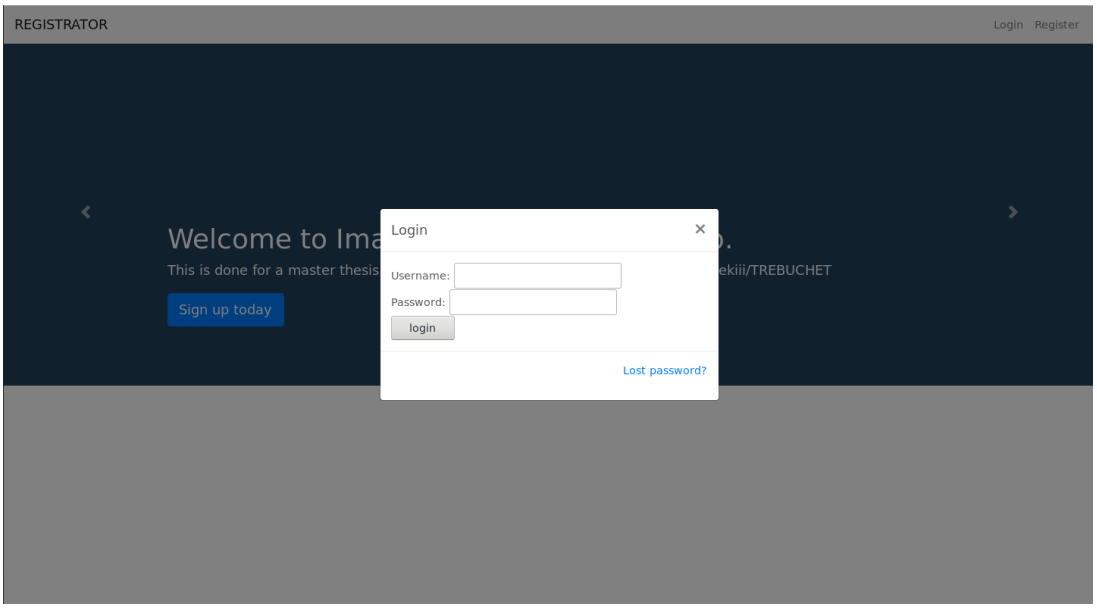

Figure 2: The user log in modal.

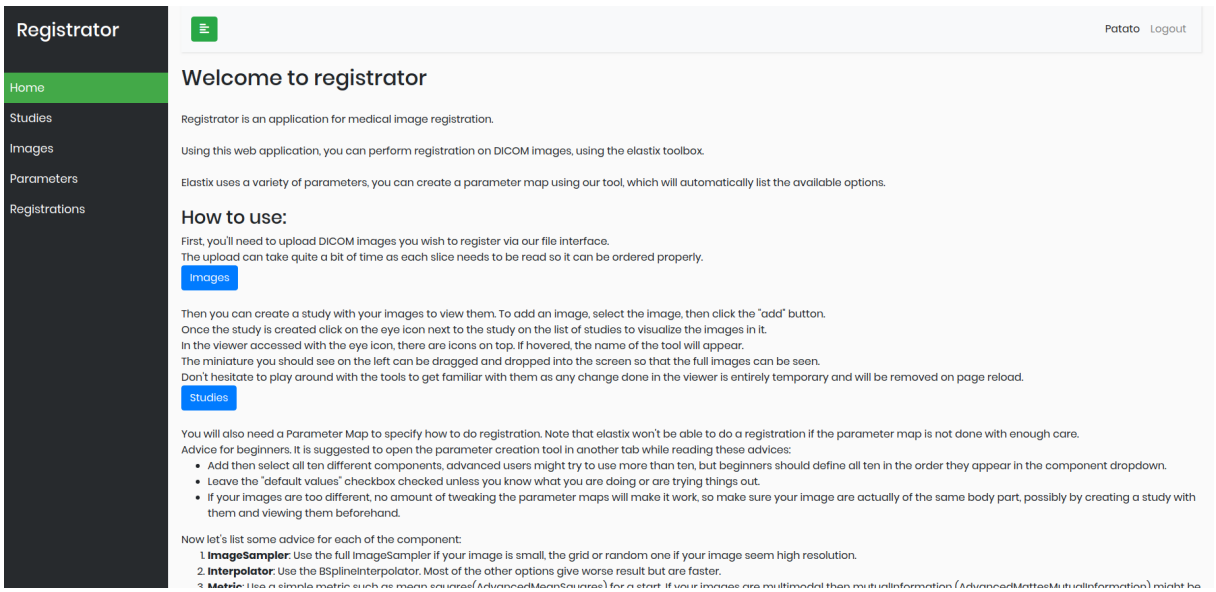

Figure 3: The home. It contains a tutorial for an easy first registration.

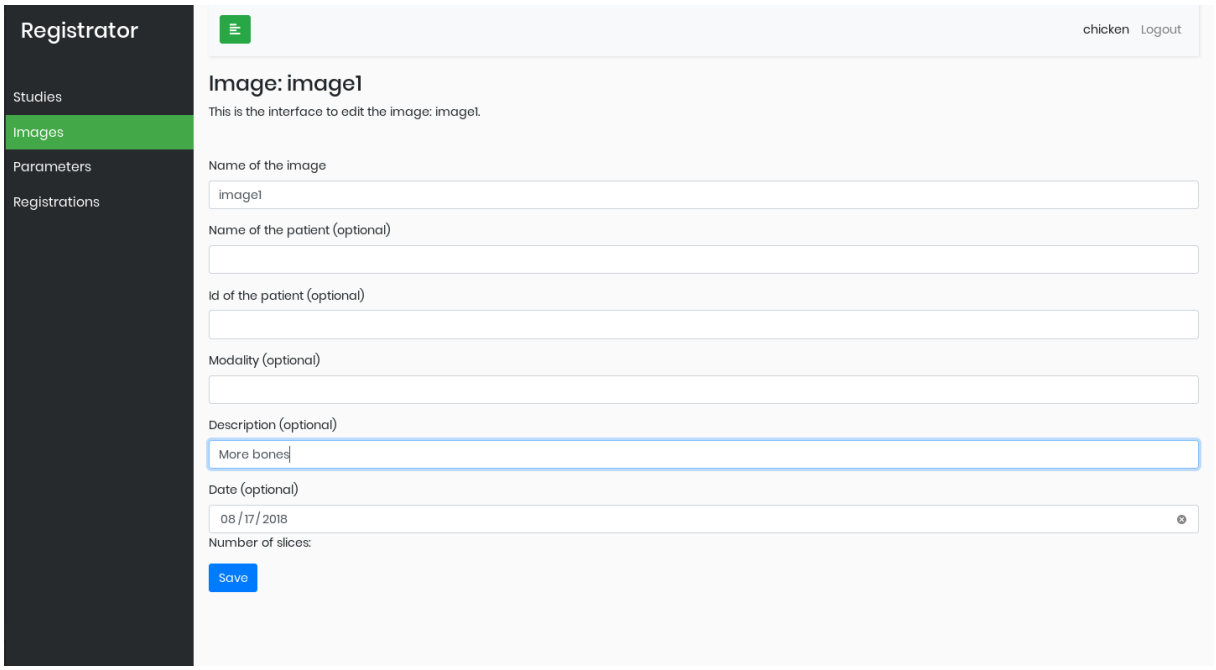

Figure 4: The interface to create an image.

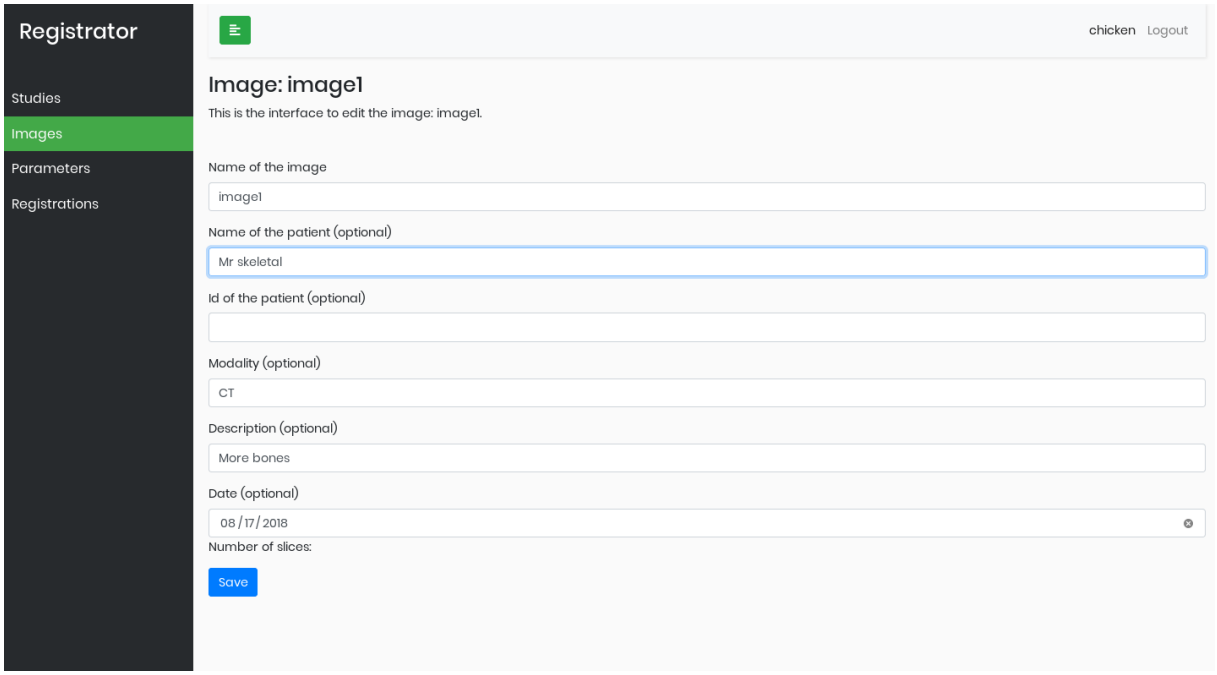

Figure 5: The interface to edit an image.

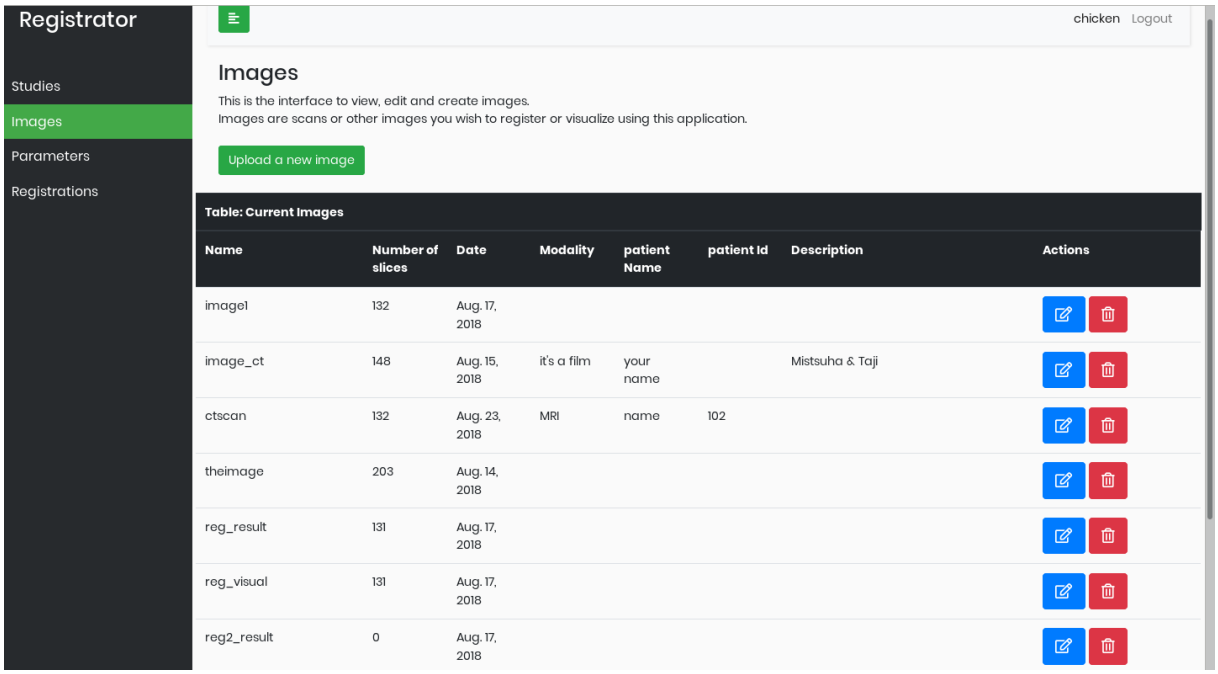

Figure 6: List of existing images.

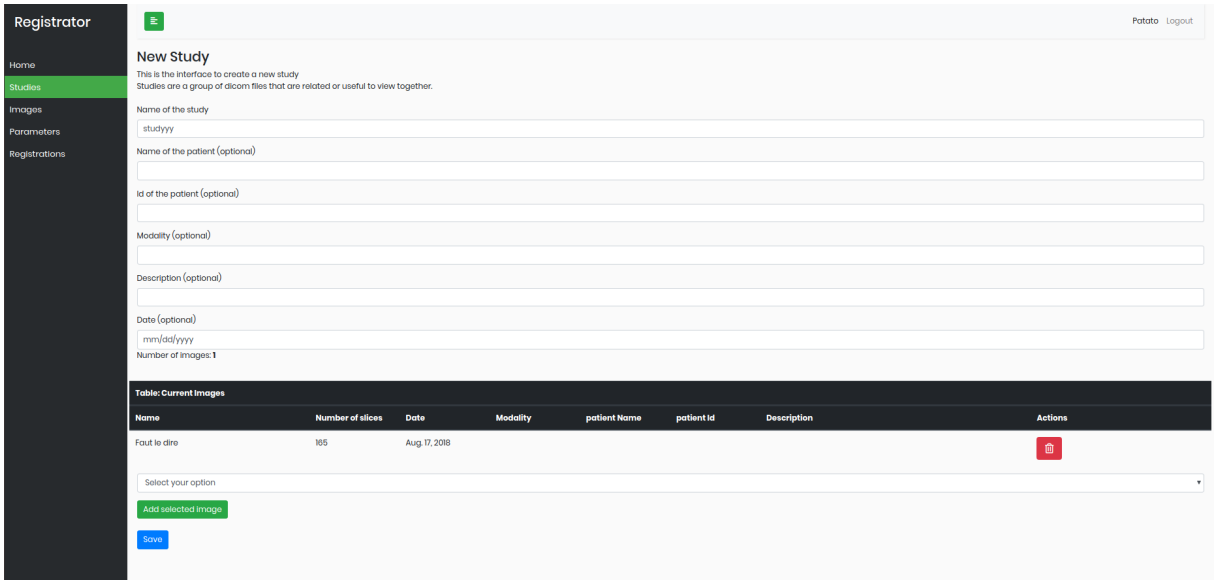

Figure 7: The interface to create a study.

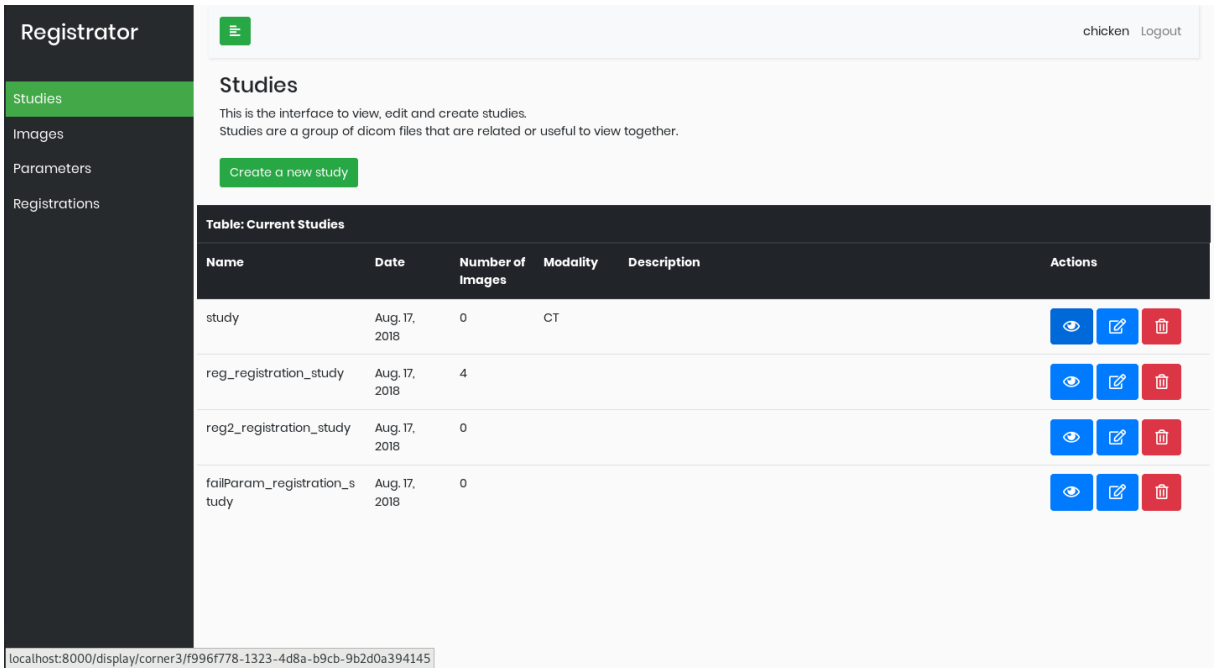

Figure 8: List of existing studies.

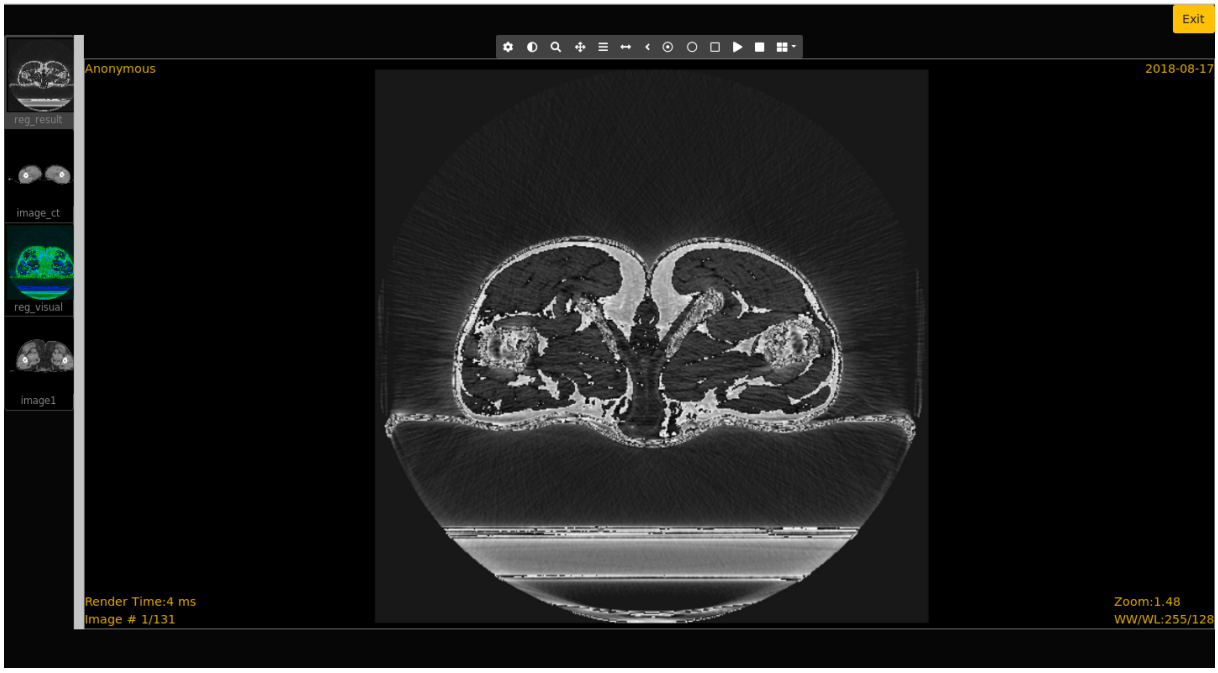

Figure 9: Cornerstone displaying a single image fullscreen.

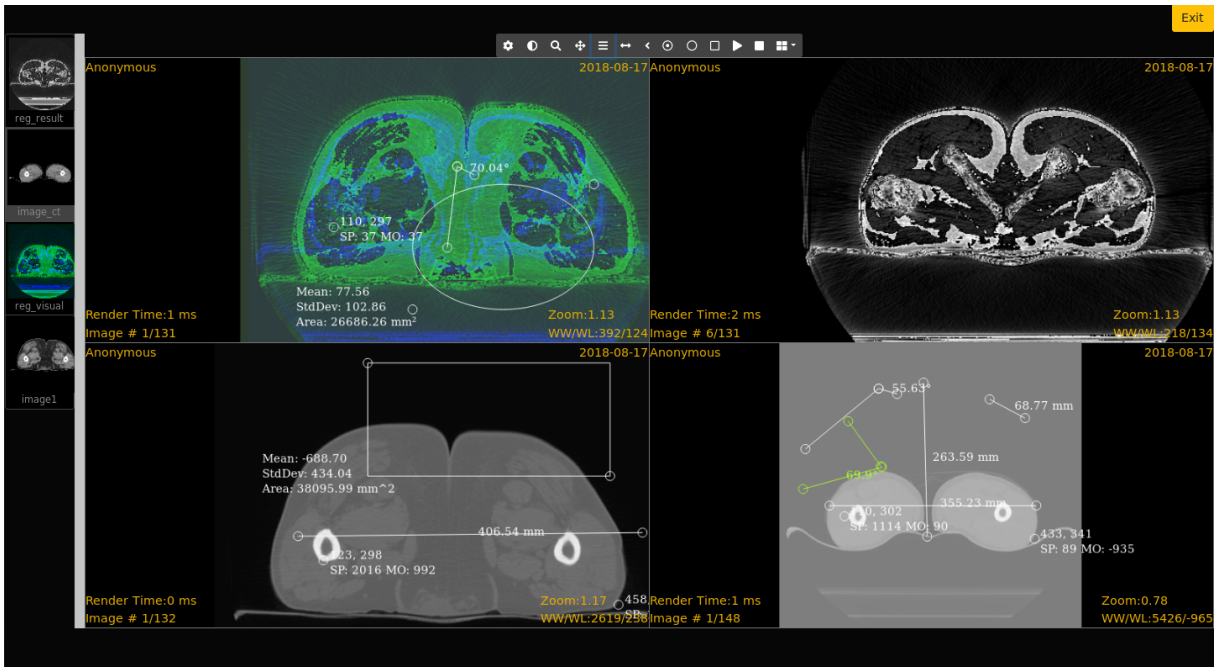

Figure 10: Cornerstone with many tools used to gather various metrics and change the zoom, window, etc. on 4 images simultaneaously.

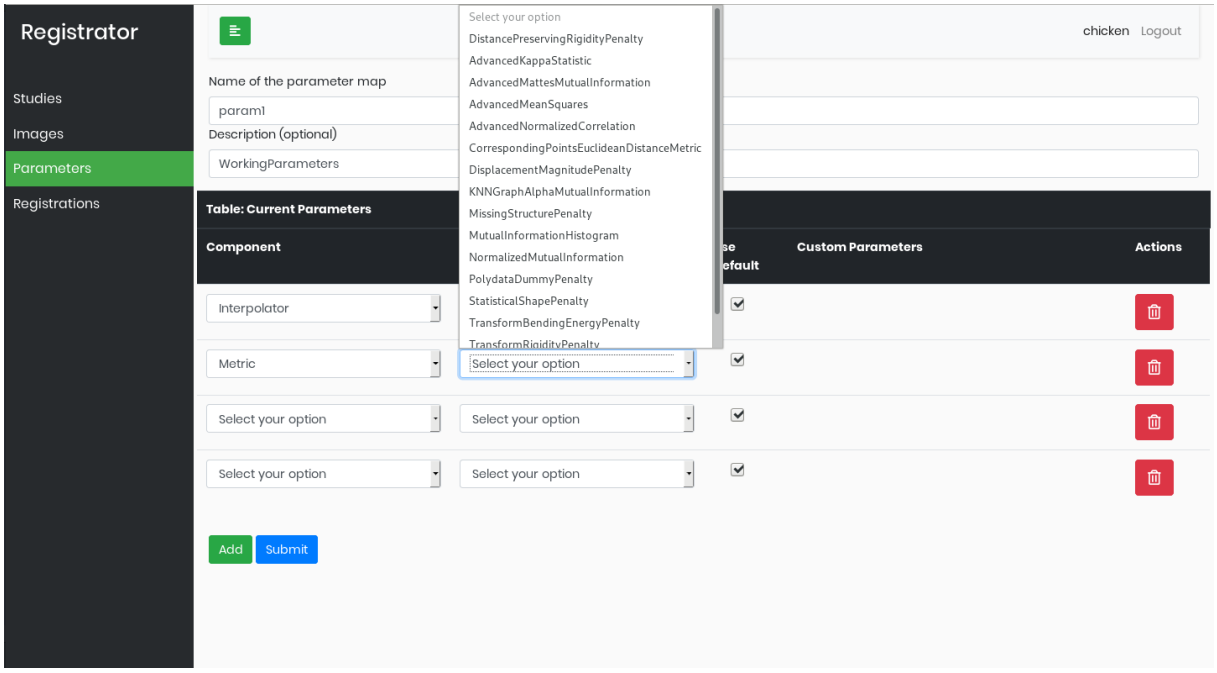

Figure 11: Parameter Map Creation: The list of available modules for the component "Metric".

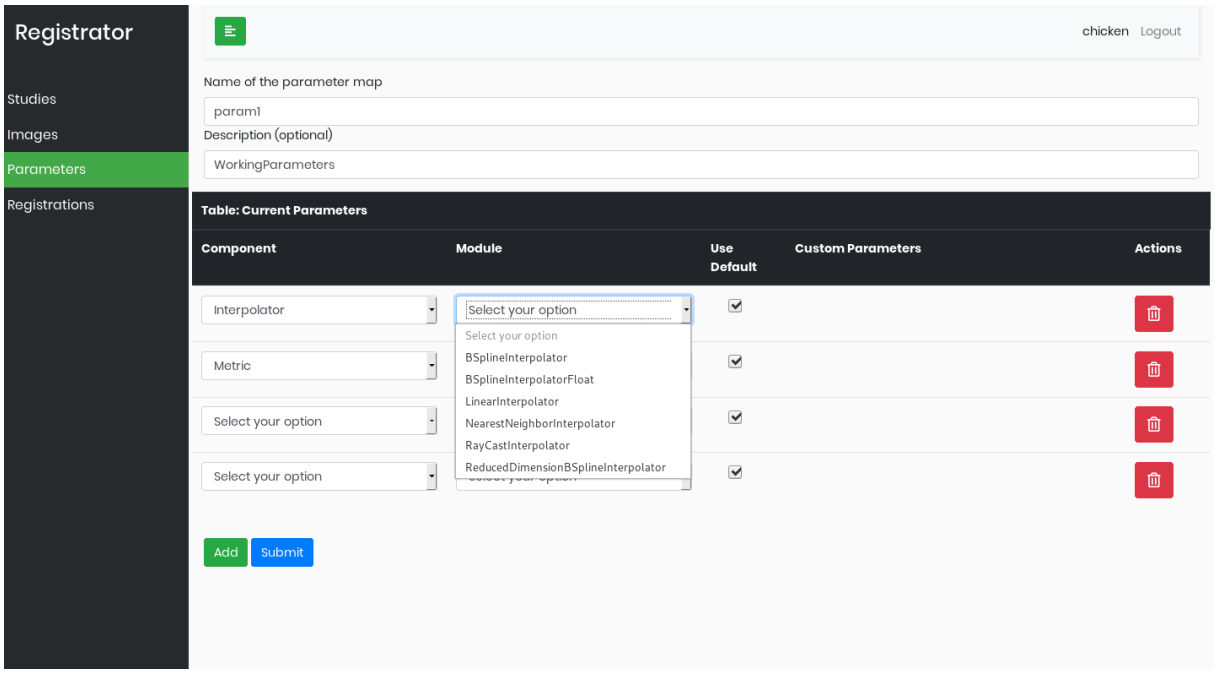

Figure 12: Parameter Map Creation: The list of available modules for the component "Interpolator".

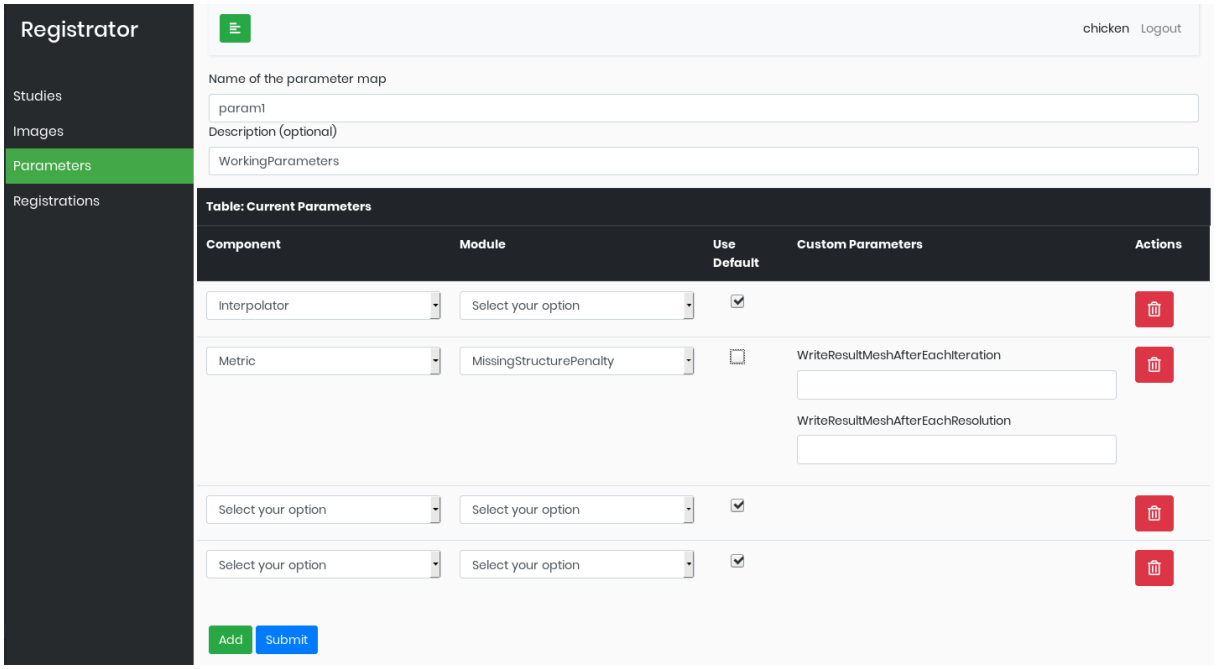

Figure 13: Parameter Map Creation: Unchecked "use default" tickbox allows user to define their custom parameters for the module.

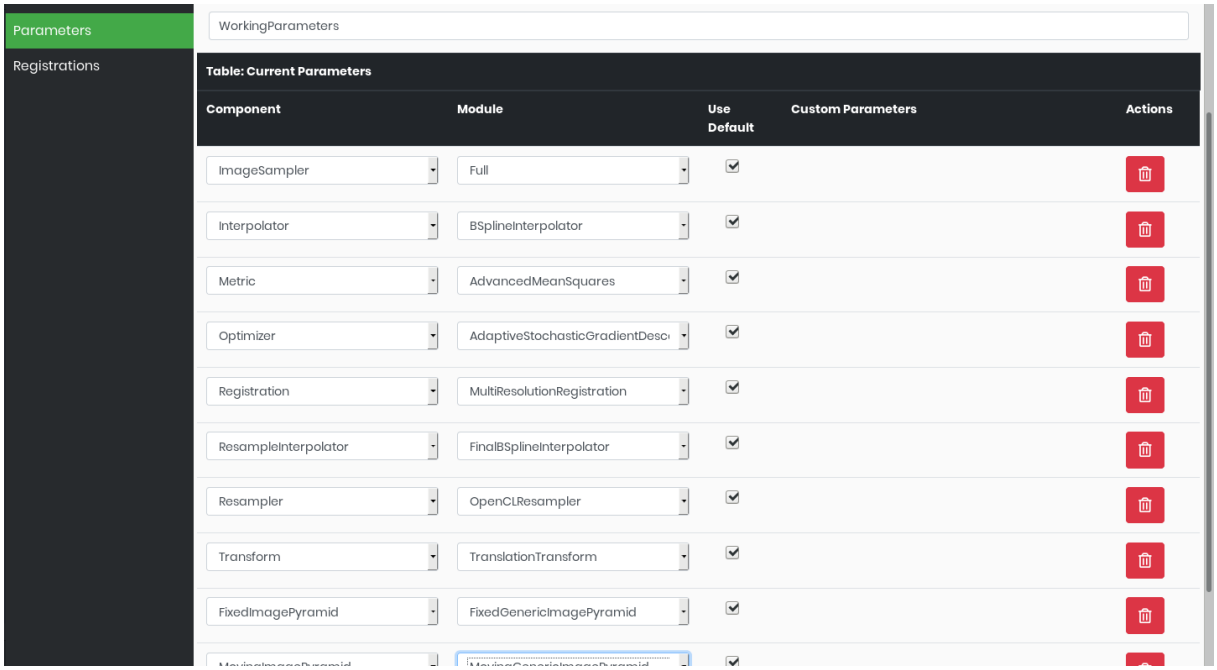

Figure 14: An example of a complete parameter map

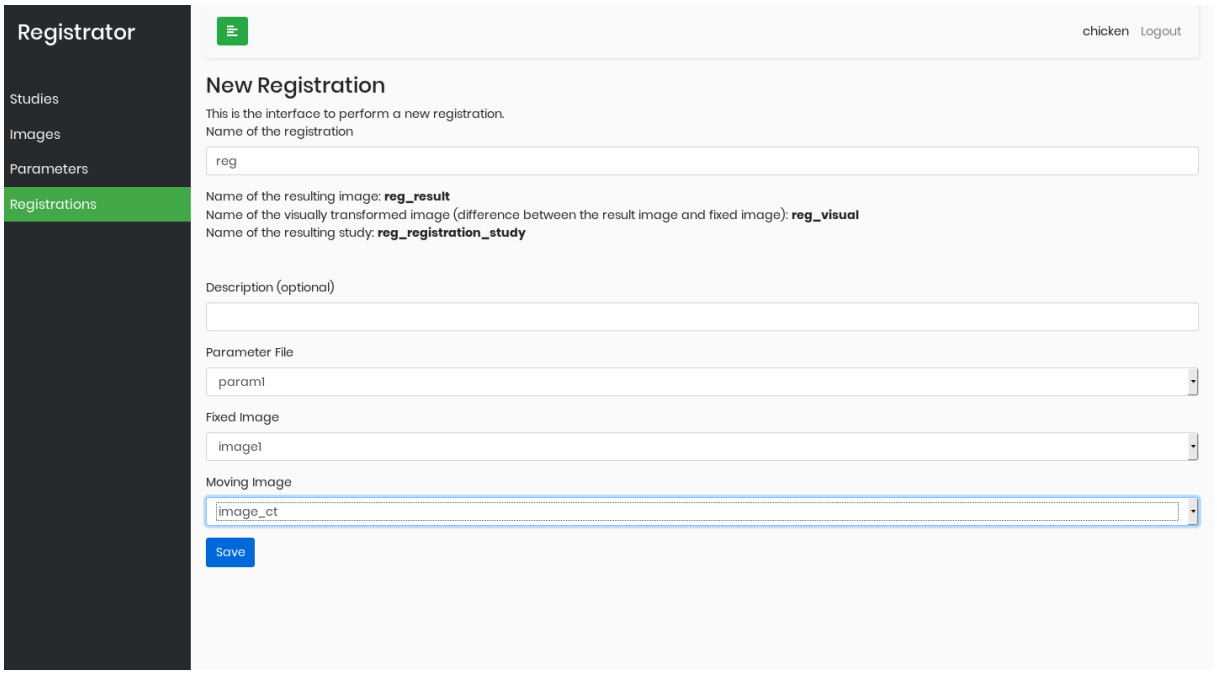

Figure 15: The POP home page TREBUCHET is an acronym for Test Registration Easily By Using Cool Helpful Elastix Toolbox.

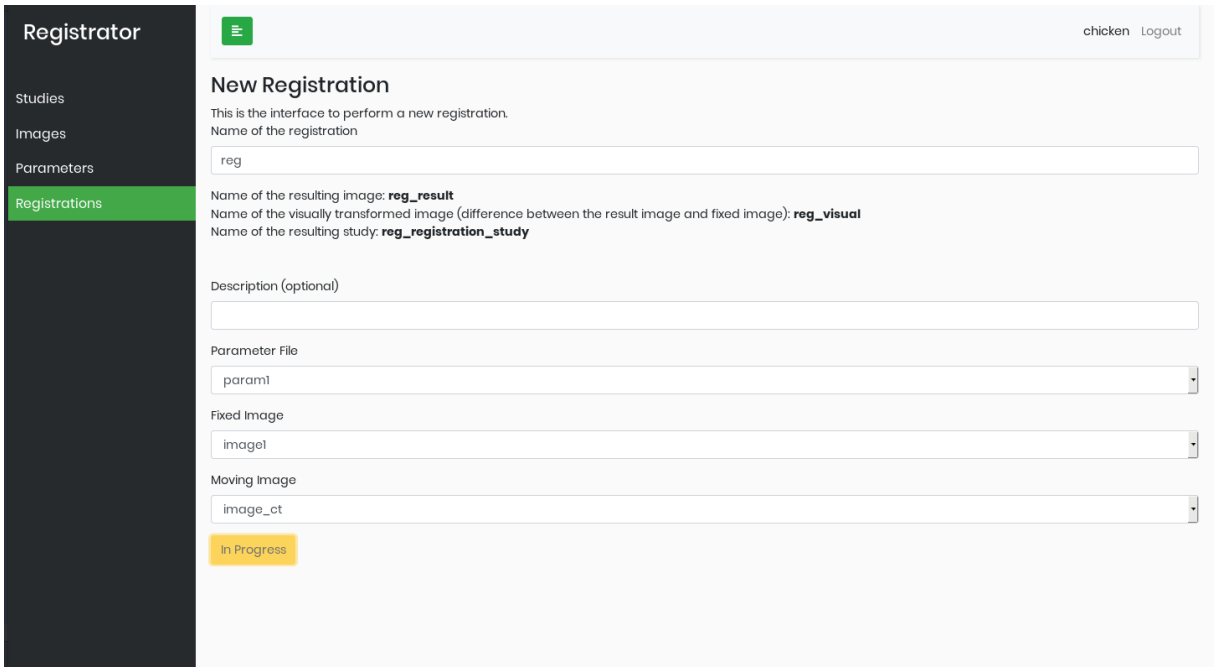

Figure 16: Registration right after being submitted. Notice the button became yellow and unclickable to avoid sending twice the same informations.

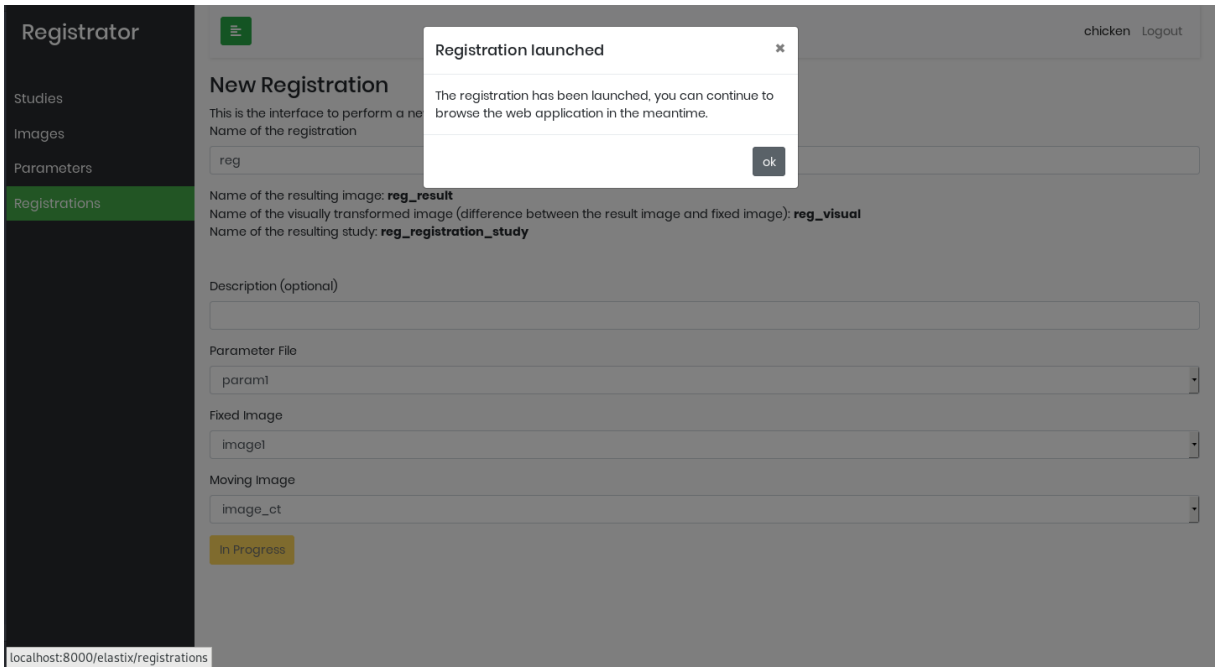

Figure 17: Modal that shows after a registration is launched. It redirects to the list of registrations.

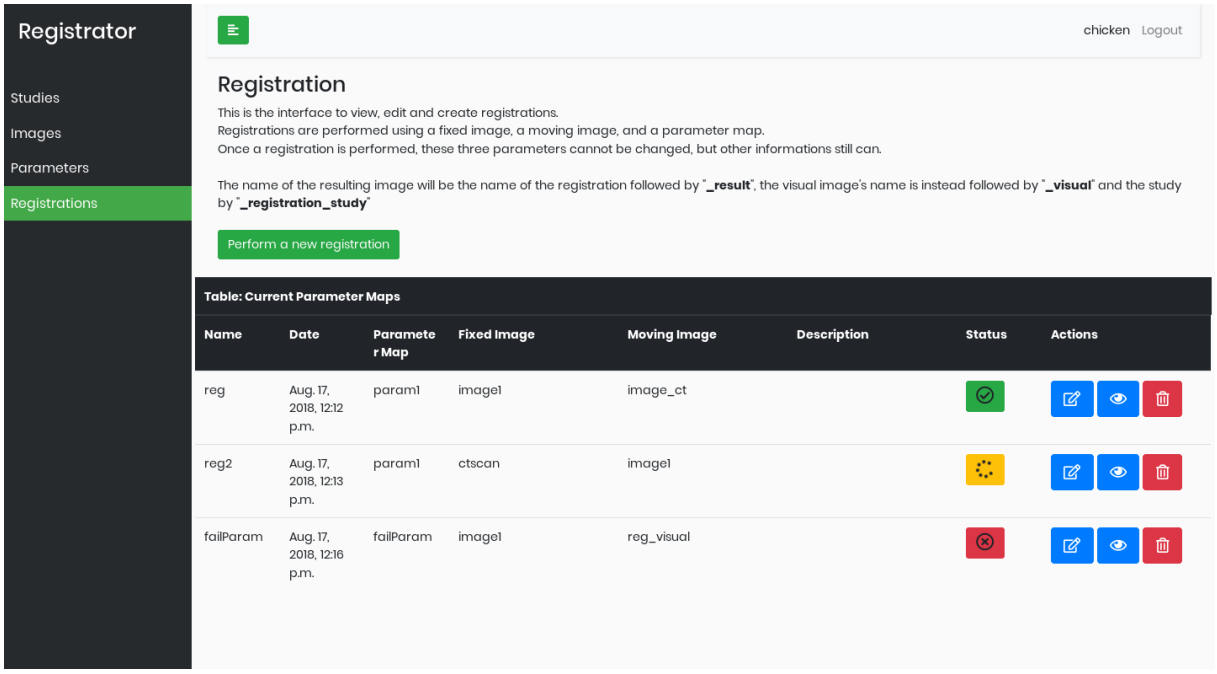

Figure 18: List of registration. A status displaying a green icon means the registration is finished, red means it failed and yellow means it is still in progress.

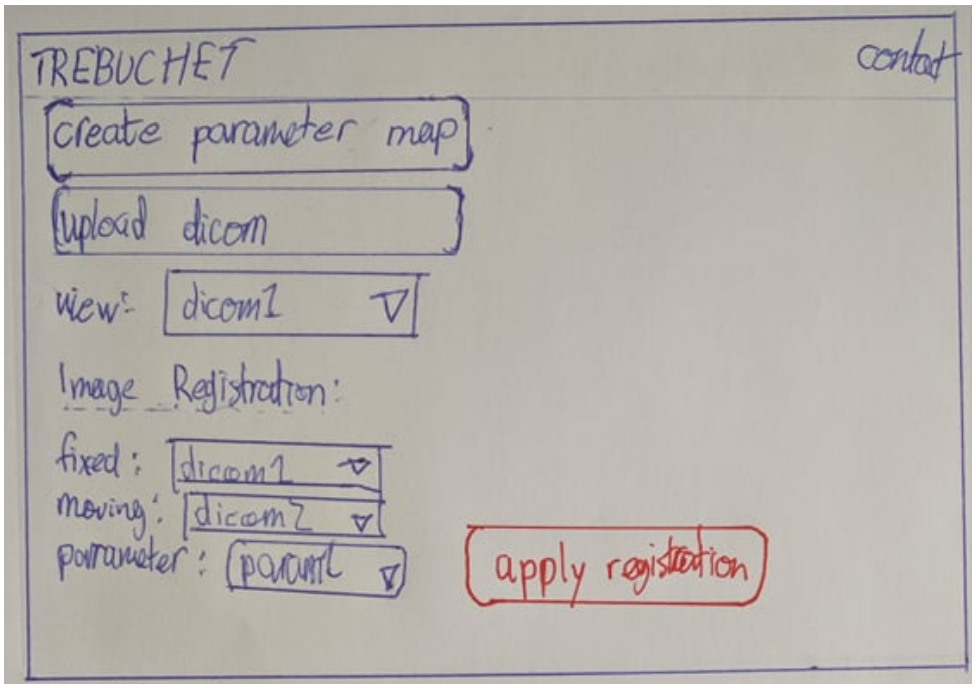

Figure 19: The POP home page TREBUCHET is an acronym for Test Registration Easily By Using Cool Helpful Elastix Toolbox.

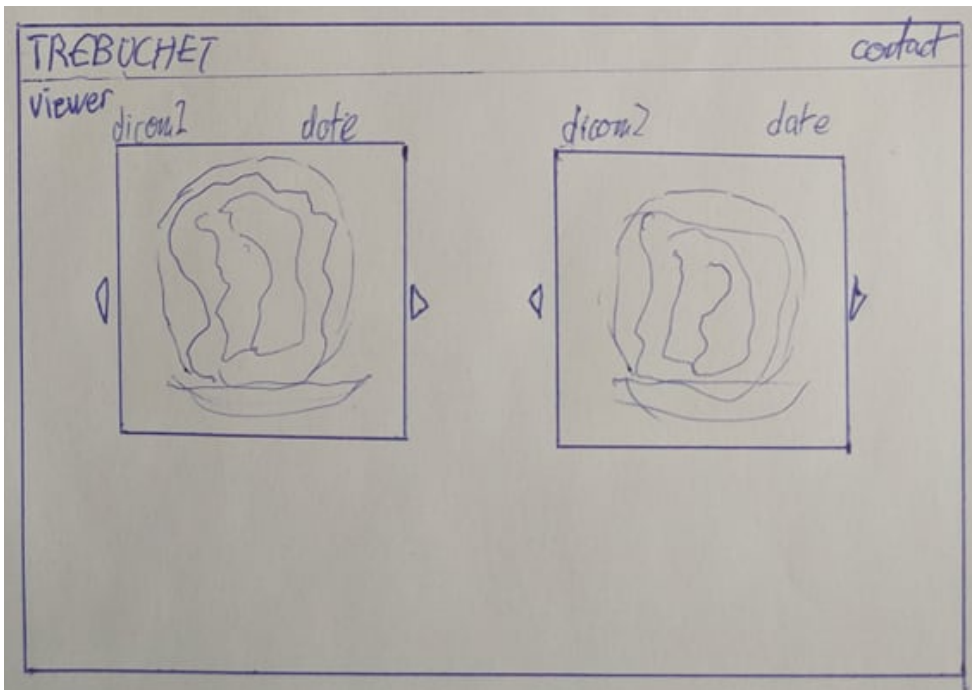

Figure 20: Prototype on paper of the viewer. Two image are displayed side by side for easy comparison.

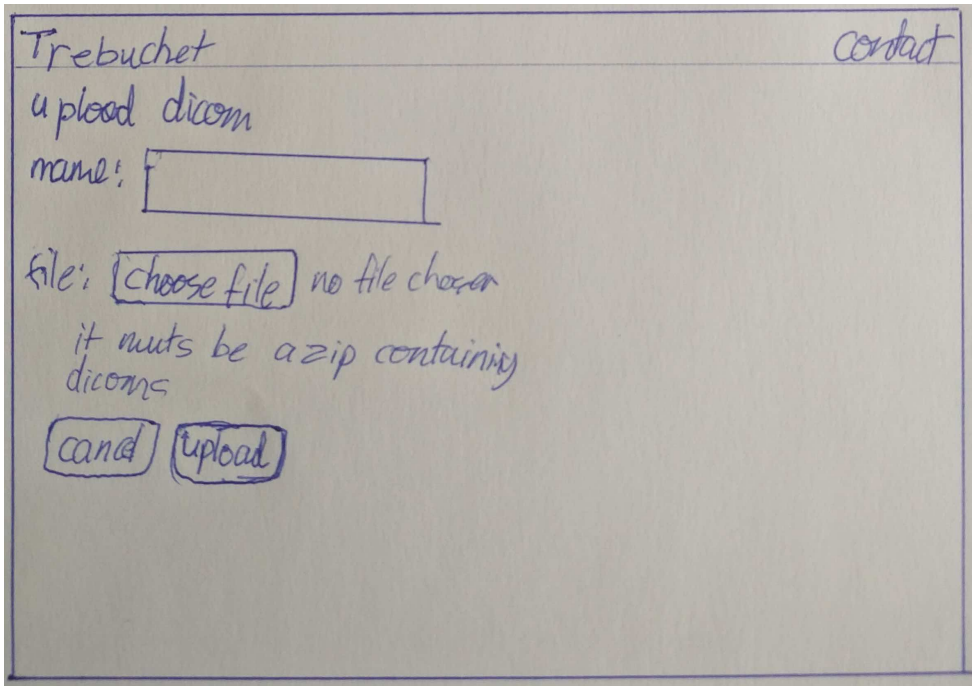

Figure 21: Prototype on paper of the upload of DICOMs.

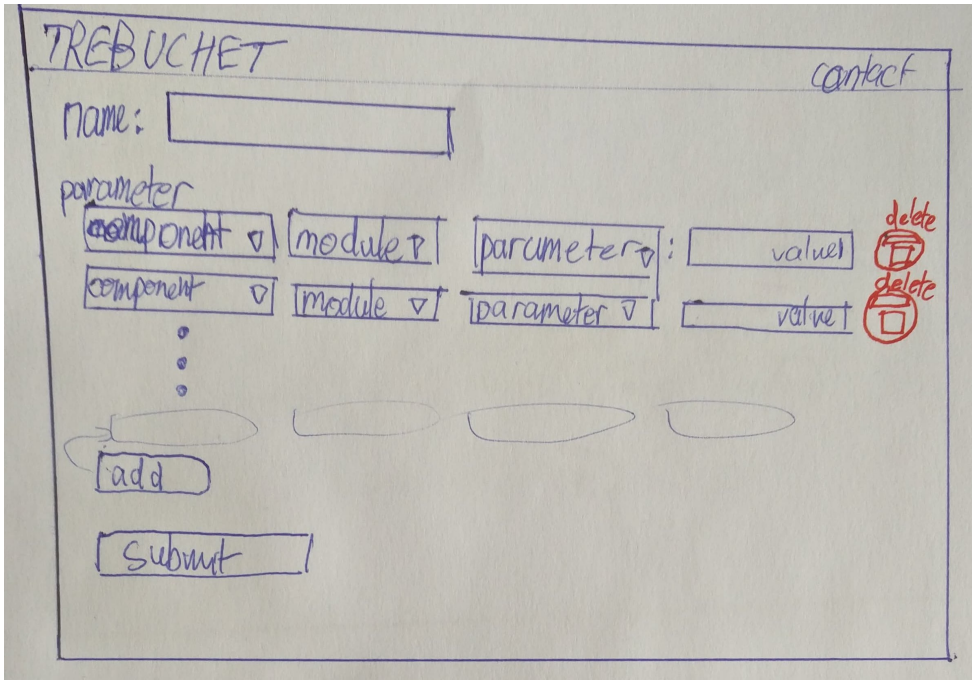

Figure 22: Prototype on paper of the parameter map creator.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**