UCLouvain

École polytechnique de Louvain

Automatic MRI-CT Region Matching for Motion Tracking Using Computer Vision

Author: **Romain Pattyn**
Supervisors: **Benoît Macq, Damien Dasnoy**
Readers: **Benoît Macq, Damien Dasnoy, Christophe de Vleeschouwer**
Academic year 2019–2020
Master [120] in Computer Science and Engineering

Abstract

Radiotherapy is an effective cancer treatment. Despite the constant improvement of the technology, it still leads to undesired side-effects. Those are particularly exacerbated when radiotherapy is used to target mobile tumors because of the lack of precision on the tumor position during treatment. Since 2015 a new medical hardware, called MR-Linac, enables the acquisition of live Magnetic Resonance Images (**MRI**) of the regions targeted during treatment. The provision of live information opens the door to new adaptive radiotherapy treatments allowing, for instance, real-time estimation of the energy-deposition. Energy deposition estimation requires X-Ray-like images, such as Computerised Tomography (**CT**) scans, which have no direct correlation with the MRI provided by the MR-Linac. Methods have been developed that aim at reproducing the anatomical motion, captured live on MR images, onto images of CT modality. Those methods rely on movement comparison of regions that are similar on each modality. This master thesis focuses on the localisation of such regions by using **Computer Vision**. The method starts by identifying moving edges appearing on both image modalities and for which the movement has a specific direction. To identify such edges, the method works as a funnel. At each step, more and more points are discarded by including new constraints via tools such as **Sobel Filters** and **Dense Displacement Fields**. This localisation scheme is used to automate a step of an existing motion replication method that relies on interface tracking. As part of this thesis, a python module was developed that implements the method in that specific context. A validation protocol was developed with a doctor to evaluate the performances of the module. The results are promising.

Acknowledgments

I wish to express my gratitude to my supervisors, Benoît Macq and Damien Dasnoy, for their continuous support during the whole development of this master thesis, and for having let me explore paths on my own while redirecting me when needed. Also, sometimes, their pressure helped me focusing better.

I am also grateful to Geneviève Van Ooteghem, who kindly answered my requests, even the most urgent ones, and who participated intensively to the validation protocol.

Finally, I would like to thank my parents for the their support.

Contents

Introduction

This document is divided into four parts.

Part I - **Context** - sets the scene and progressively clarifies the purpose of this master thesis. It is divided into 3 chapters. Chapter 1 provides a brief overview of radiotherapy, and introduces the challenge of dose-deposition estimation on mobile tumors. Chapter 2 describes an algorithm that addresses the issue of estimating dose-deposition and to which this master thesis can contribute. Chapter 3 specifies the scope of this master thesis.

Part II - **Prerequisites** - provides the reader with some background information on the theoretical concepts that are applied in part III.

Part III - **Development** - is the core of this document. In chapter 5, the conceptual method, supporting the Python implementation, is detailed together with the relevant computer vision concepts. Results of a validation protocol, conducted with the support of a doctor, are provided in Chapter 6.

Part IV - **Appendix** - contains information for the implementation part of this master thesis. In particular, the first appendix contains a user manual of the Python module that was developed. The second appendix provides the list the list of variables used to parametrise the method presented in chapter 5.

The Python implementation of the method, which completes this master thesis, is published and available as open source code on GitHub at the following repository : `https://github.com/RomainPattyn/MasterThesis`.

Part I

Context

Chapter 1

Algorithms in Cancer Treatment

Cancer kills in Europe approximately 1.9 million people every year [1]. This makes cancer the second most frequent cause of death, the first cause being cardiovascular diseases, with 3.9 million deaths in Europe. Science has come with numerous different treatments against cancer, such as chemotherapy, radiotherapy, immunotherapy, hormone therapy. Each treatment has its pros and cons, and its effectiveness varies based on the cancer type, the cancer stage and the patient's condition.

Today, the fight against cancer is no longer left only in the hands of doctors. Modern technologies and computers have become an essential part of it. Engineers play now a key role in the improvement of medical science. Radiotherapy seems to be the cancer treatment that will benefit the most from the rise of computer power and new technologies.

This master thesis is made in the context of the collaboration between computer science and medicine, as it contributes to an algorithm that aims at improving the future of radiotherapy.

This first chapter provides the reader with some background information about radiotherapy. After a brief overview of the history of radiotherapy, some of the challenges it is facing today will be highlighted. They are relevant in the scope of this thesis. This chapter ends with an introduction to a revolutionary machine in radiotherapy, called MR-linac.

1.1 The Early Ages of Radiotherapy

Radiotherapy is a technique for cancer treatment where high-doses of radiations are sent to cancerous tumors. Radiations are emissions of energy in the form of waves or particles used to damage the DNA structures of the cancerous cells and, ultimately, to kill them.

The physics supporting radiotherapy started to emerge shortly after 1895, when **X-rays** were discovered by Wilhelm Röntgen, a German physician. In its early stages, the mechanism of radiotherapy was not well understood and the side effects were very high compared to the benefits in treating cancer. Therefore, until the 1920s, the medical application of X-rays was mostly limited to the treatment of skin cancers, which requires low penetration of radiation.

Radiotherapy took a new turn in 1958 with the use of **Linear Accelerators** in medicine. A linear accelerator is a machine that directly targets a tumor by using a beam. Linear accelerator only became common in the late 1970s, when computer assistance became possible. From then onwards, beams have helped treating a wide set of tumor types. [2]

The emergence of medical imaging techniques, such as **CT-Scanners** and **MRI-Scanners**, discovered in 1971 and 1980 respectively, opened the door to more sophisticated strategies in treatment planning and execution. In this context, computers have played a significant role in improving the image quality delivered by the scanners.

Hint 1 : MRI and CT Imaging

MRI (**Magnetic Resonance Imaging**) and CT (**Computerized Tomography**) are two modalities of non-invasive medical imaging procedures. The underlying technologies are different and so is the nature of the captured content.

CT-scanners, that use X-rays, depict the various electron densities of anatomical tissues. Images taken via MRI devices, which use a strong magnetic field and radio waves, depict magnetic properties of tissues. There is no direct correlation between those two properties. For example, edges, which are the manifestation of an interface between two tissues, could appear in one of the modalities but not in the other.

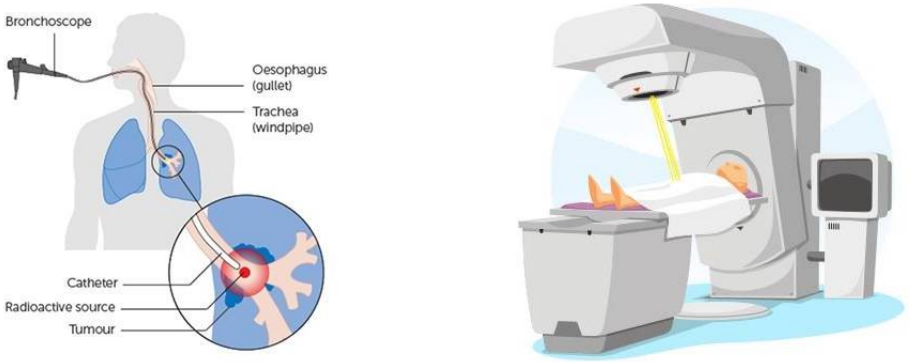
When working with both image modalities, this could become an issue. This is one of the challenges we were confronted with when developing our method. This topic is further developed in Chapter 3.

1.2 Radiotherapy Today

There are currently two methods for sending radiation doses to a tumor: internal radiation therapy and external beam radiation therapy. [3] [4]

Internal Radiation Therapy, also known as **Brachytheraphy**, puts a source of radiation inside the patient's body. The source either consists of a small radioactive capsule placed manually near the tumor, or of a radioactive liquid which is swallowed by the patient or injected.

External Beam Radiation Therapy places the patient in front of a **Linear Accelerator**. This machine, referred to as a **Linac**, creates a beam sending high-doses of energy directly on the cancerous tumor.



(a) Internal Radiotherapy [5] (b) External Beam Radiotherapy [6]

Figure 1.1: Two Methods to Irradiate a Tumor

The external beam radiotherapy is the one that is relevant for this thesis. Within the external beam radiotherapy family, two methods exist, based on the type of particles used :

- The first one, known as **Conventional Radiotherapy**, uses **Photons**. Photons are elementary particles having a mass of zero.
- The second one, known as **Particle Therapy**, uses particles that have a mass greater than zero. Various types of particles can be used in particle therapy but the most commonly used are **Protons**.

Proton therapy is said to better preserve the patient's healthy cells [1]. However, proton facilities

[1] When using particles with a mass greater than zero, the energy deposition follows a curve that falls drastically at a certain point, and that point can be predicted. As a result, one can organise that protons stop right after having irradiated the tumor, preserving a big part of the patient's body from radiations. This interesting phenomenon,

are very expensive and so, they are not widespread. The cost of a proton therapy facility ranges between 140 and 200 million US dollars [7]. In February 2019, less than 240 facilities in the world offered proton therapy [8]. The vast majority of radiotherapy devices today are based on photon therapy.

1.3 Challenges with External Radiotherapy

The ability of the energy to break the DNA and kill the tumorous cells unfortunately applies also to healthy cells. The destruction of healthy cells represents a collateral damage that can lead to severe unwanted side-effects. While ensuring that the prescribed dose effectively reaches the tumor, minimizing the radiation doses that are sent to healthy tissues is critical in radiotherapy treatments. Therefore, **targeting the tumor with accuracy is essential**.

However, during his treatment, a patient breathes. This induces anatomical motion in the entire chest area: organs, tissues and also cancerous tumors move. As per Geoffrey Ibbott, Ph.D., professor of Radiation Physics at MD Anderson : "*It's not unusual for a tumor to shift an inch or slightly more while a patient breathes in and out.*" [9].

Managing those constant movements is one of the challenges of external beam radiotherapy because every time the beam isn't pointing exactly towards the tumor, healthy tissues can be damaged. This also results in the effective dose sent to the tumor being reduced. As a consequence, when treating mobile tumors with external radiotherapy, managing the respiratory motion is a key success factor.

An obvious way to manage respiratory motion would be to ask the patient to hold his breath during the treatment. However, this is often not possible because of the duration of the treatment, which can last several minutes. Other approaches have been developed which we will mention in the following section.

Another challenge with external radiotherapy is to **determine the dose of radiation that is actually sent** to the tumor. There is a maximum dose of radiation that tissues can receive over a certain period of time and this needs to be precisely monitored.

known as Bragg Peak, is explained in Appendix D. This technology is promising. However, it requires precise localisation of the tumor, which currently puts limits to its global application. Proton Therapy is most often used in regions of small movement like the brain

Additional complexity comes from the fact that the amount of radiations delivered in a particular region is proportional to the density of electrons in that region. The higher the density, the more energy will be absorbed by the tissue.

1.4 Respiratory Motion Management

Mobile tumors are mostly situated in the upper body. They concern organs like liver, stomach, kidneys, lungs. For those tumors, managing respiratory motion and reducing its impact on treatment is key. Some measures are taken to reduce the amplitude of the movement itself. Others are taken to integrate the motion within the treatment, i.e. accept it and plan for it.

1.4.1 Motion Mitigation

The term motion mitigation covers various countermeasures that can be put in place to reduce the amplitude of the breathing-induced motion during the treatment. Some of those measures require the patient collaboration. **Audio coaching** [10] or **visual feedback** are techniques used to help the patient to regularize his breath. Other measures make use of external devices, such as **abdominal compression** [11], which restricts the motion of internal organs. Patients can also breathe through a mechanically-assisted ventilation [12].

The objective of these methods is either to regularize the breath, to make it more predictable and to reduce its amplitude, or to reduce its impact on the tumor movement.

1.4.2 Treatment Planning

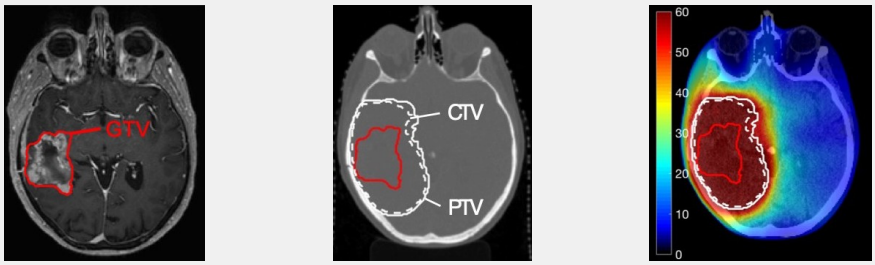
Thanks to the improvement of medical imaging over the last few decades, doctors are now able to create **treatment plans that anticipate and integrate movements**. They capture internal 3D images (usually X-ray-like images) of the tumorous regions before the treatment and use those images to provide margins to the expected motion of the target. This ensures that the tumor is entirely covered. However, it also means that some healthy tissues are irradiated, as shown in Figure 1.2c in Hint 2, on next page.

Unfortunately, even if motion mitigation methods are used to regularize the breath, nothing guarantees adequate correlation of anatomical movements between the planning and the treatment phases. This is a weakness of this kind of treatment planning, as it leads to potential errors and to a sub-optimal treatment.

The ultimate tumor tracking tool would consist of processing images that are taken during the treatment. The tool presented in next section represents a first step in that direction.

Hint 2 : General Procedure For Treatment Planning

Figure 1.2 shows a typical treatment planning for a brain tumor. It shows the steps taken by a doctor when defining the area to be treated on basis of medical images.



(a) GTV (b) CTV and PTV (c) Dose

Figure 1.2: Illustration of the concepts used in treatment planning

(a) As a first step, the doctor encircles the region that appears to be a tumor on the medical images. This is called the **GTV (Gross Tumor Volume)** and is the part that receives most of the irradiation doses.

(b) Unfortunately, not all tumorous cells show clearly on the images. This is why the doctor delimits another region, known as **CTV (Clinical Target Volume)**. It encloses the GTV plus other cells that the doctor identifies as potentially tumorous. Defining the CTV is subjective. It is based on the doctor's experience and on particular knowledge about the tumor's behavior. Finally, the **PTV (Planning Target Volume)** is determined. It includes the CTV and adds extra security margins to compensate variations and uncertainties that could occur. This is where the motion is taken into account.

(c) The beam will finally target the PTV in priority but will also irradiate several tissues at the border.

1.5 MR-linac

On 22 January 2015, Utrecht's University Medical Center (UMC) inaugurated a new medical hardware device known as the **MR-linac**. This machine combines a **Linear Accelerator** and a **Magnetic Resonance Imaging machine** into one single device. Currently, only photon based therapy is possible, but it's probably only a matter of time before proton versions are developed.

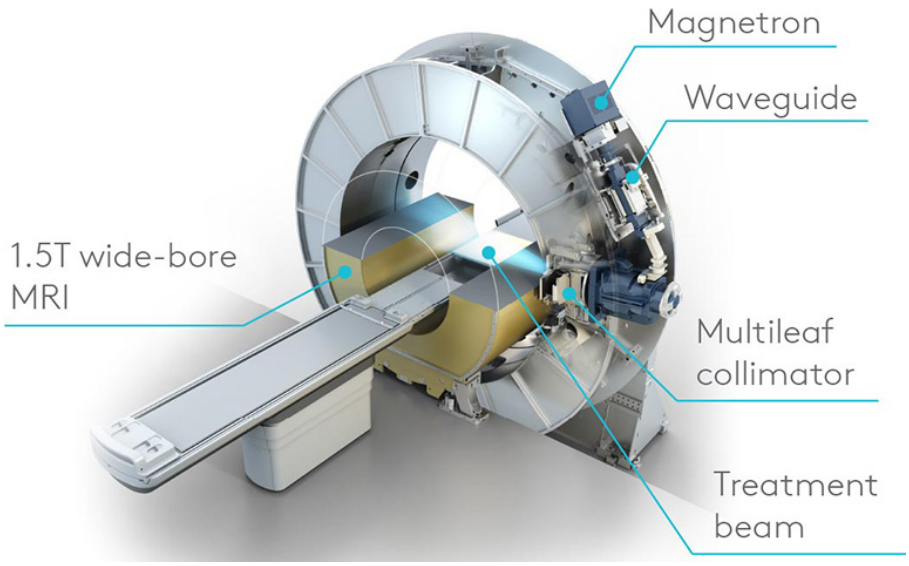


Figure 1.3: MR-Linac

The MR-linac is considered to be a game-changer in highly targeted radiotherapy. It opens the door to even more intelligent radiotherapy, where one can simultaneously control the beam and acquire real-time MR-images of the region that is targeted, allowing a better preservation of the healthy tissues. MR-images can be processed in real time to make the beam follow the tumor motion. This is called adaptive radiotherapy.

Algorithms of tumor tracking exploiting the advantages of an MR-linac are progressively appearing in the literature. One such algorithm was developed by Dasnoy et al. [13] and is at the center of this master thesis. Dasnoy's algorithm is detailed in chapter 2.

Today, there is only a very limited amount of facilities in the world that are equipped with an MR-linac device. This brand new technology is still at the early stage of its implementation. Having the opportunity to work on a project related to this technology is very exciting.

Chapter 2

Dasnoy's 3D Motion Reproduction Method

The possibilities offered by the new medical hardware MR-linac have triggered lots of researches around the world. Various algorithms and methods are being developed to optimize the use of that new technology and further increase the precision of the treatment.

This chapter introduces a method that was developed in that context. The method was developed by Damien Dasnoy, PhD student at Université Catholique de Louvain. It is important to review Dasnoy's method in order to clearly position the objective assigned to our project.

The method has not yet been used on an MR-linac but it is already very much documented. The explanations given in this chapter are summarized and simplified. For complete information, please refer to the original publication [13].

2.1 Introduction

As explained in Chapter 1, one of the challenges in radiotherapy is to make sure that the entire tumor is irradiated with minimum damages to healthy tissues. That means, targeting the right cells and sending the right irradiation doses.

Since 2015, the MR-linac eases the work of the doctors. Indeed, this new equipment sends radiations to the patient's body and, at the same time, captures real-time images of the region that is

targeted. The challenge related to the irradiation deposition dose remains because the MR-linac captures 2DMR[1] images while only x-Ray-like images (e.g. CT) could enable the estimation of the deposition dose.

This is where the method developed by D.Dasnoy intervenes. The overall idea is that once applied, this method could produce 3DCT[2] images during the MR-linac treatment, enabling the irradiation dose deposition to be estimated in real-time. The method could also be used as post-treatment verification.

The application of the method happens in two phases : **planning** and **treatment**.

The **planning phase** takes place before the treatment delivery with the MR-linac. The objective is to acquire images that will be needed during the treatment delivery. The patient's body is imaged in a X-ray device and a sequence of 3DCT images is acquired and processed.

During the **treatment phase**, Dasnoy's method creates in real-time, an artificial 3DCT sequence that replicates the anatomical movement of the live 2DMR images captured by the MR-linac.

2.2 Planning Phase

During a couple of minutes, the part of the patient's body that contains the tumor is imaged by using a CT scanner in order to create a sequence of CT scans. It is important that those images are acquired in the same conditions as the future treatment conditions, e.g. if motion mitigation measures are taken during the treatment, they must be taken during the planning phase too.

The sequence of CT scans, which covers multiple breathing periods, is then re-sampled and processed to produce a set of ten 3DCT images. The newly created set corresponds to a single breathing period and will be referred to as 4DCT[3].

As a final step, deformation fields, required during the treatment, are computed between the different images of the 4DCT. Deformation fields, sometimes called "displacement fields", are vector fields that map an image onto another one. More information about deformation fields is given in Section 4.3.

[1] 2DMR stands for "2 dimensional MR image"
[2] 3DCT stands for "3 dimensional CT image"
[3] in 4DCT, the additional dimension can be seen as the time

2.2.1 Some Details

To generate the 4DCT, the initial 3DCT images are sorted into 10 breathing moments within a breathing period. The frames which are too distant from one of the 10 chosen moments are discarded. All the images belonging to a same moment are then processed to produce a new image, which is the average of several images.

Figure 2.1 illustrates the creation of the 4DCT from the 3DCT scans. For illustration purpose, the data is simplified (only 3 breathing periods, covered by 10 images, and only 4 images in the 4DCT, instead of 10).

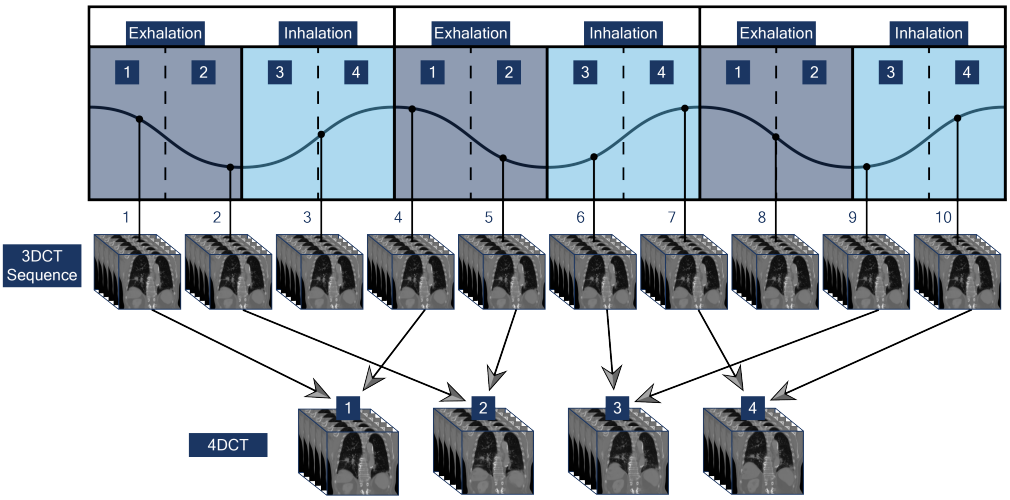


Figure 2.1: 4DCT Creation

Sorting and re-sampling is not always accurate. This means that inappropriate images might be taken during the process. This is due the fact that a single 4DCT element is covering a range of the breathing period, and not a an infinitesimal moment of it. In practice, images that do not correspond clearly to a particular breathing phase are not taken into account.

This is illustrated in Figure 2.1 by the fact that the third and the eighth images were not used. Unfortunately, outliers could still be taken into account. When that happens, they might degrade the result by creating blobs. Those blobs do not correspond to real anatomical parts of the patient's body, they are 4DCT reconstruction artifacts. Those blobs/artifacts are easily identified by the human eye but are a real challenge in computer vision. We will come back to that in Chapter 3.

Figure 2.2 shows what a blob would look like on CT images. On frames 1 and 2, the red circle highlights a blurry shape. It vanishes on frame 3, which means it is not actually part of the organ.

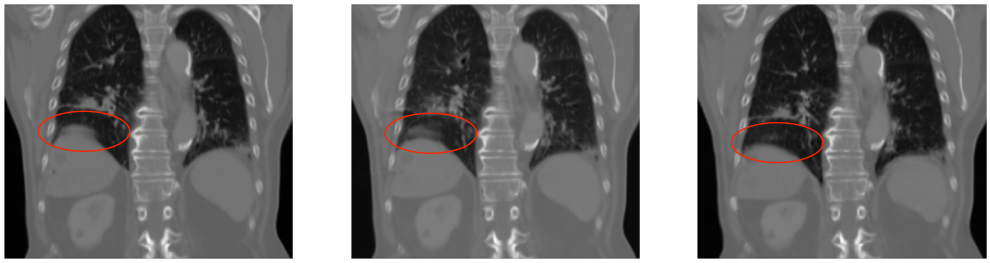


Figure 2.2: 4DCT Reconstruction Artifacts

2.3 Treatment Phase

During the treatment, the objective of Dasnoy's Method is to create real-time 3DCT images of the targeted region. To do so, two main sources of data are available in the form of images: the 2DMR images captured in real-time by the MR-linac and the 4DCT (together with its associated sequence of deformation fields) created during the planning phase.

Concretely, for every 2DMR image captured by the MR-linac, the algorithm produces an artificial 3DCT image. To do so, it selects a 3DCT image within the 4DCT (the one that is the closest in term of respiratory phase to the 2DMR), and deforms that image to match more precisely the respiratory phase of the 2DMR image. Deformation is needed because the 4DCT contains only 10 images for a breathing period while the 2DMR images are captured continuously.

An analogy will explain the general process. Imagine a man entering a tailor shop with the intention of buying a suit. The tailor explains to the customer that to cut the costs, he doesn't proceed like the other tailors. He doesn't take the measures of his clients to build a completely new suit based on those measures. Instead, he asks his client to try one of the pre-made suits, available in the sizes S, M, L or XL, and that he then shortens and re-models the one that fitted the best to match the dimensions of the customer.

Dasnoy's method works in the same way. The method first selects the pre-computed 3DCT image that fits the best the 2DMR image. In this context, "best fitting" means "closest in term of respiratory period". And the method then re-models the selected 3DCT to fit even better the 2DMR image. Here, "re-models" means "deforming the image using deformation fields".

From the algorithm's perspective, this process of 3DCT creation happens in 3 steps and is repeated for every 2DMR image received :

- Tracking the interface positions on the 2DMR images
- Selecting the pre-computed 3DCT image that has the closest interfaces
- Deforming the chosen 3DCT to match the 2DMR breathing phase

However, before launching this process, two preliminary steps are performed :

- Finding the position of the MRI plane[4] inside the 4DCT to extract a 2DCT sequence
- Positioning of trackers/navigators

These two steps must be done between the moment the patient is installed in the MR-linac and the start of the irradiation process.

2.3.1 Finding the position of the MRI plane inside the 4DCT

A critical step in Dasnoy's method is the selection of the 3DCT element of the 4DCT that is the closest in term of respiratory phase to a 2DMR image. But comparing a 2D and a 3D image is not trivial. What is done instead is that a sequence of 2D images is sliced out of the 4DCT. The 2DCT slices are situated at the same plane as the 2DMR images received by the MR-linac. The action of locating where a 2DMR image is in a 3DCT image is commonly referred to as "rigid registration" and is based on the bone structures appearing on both modalities.

2.3.2 Positioning of Trackers/Navigators

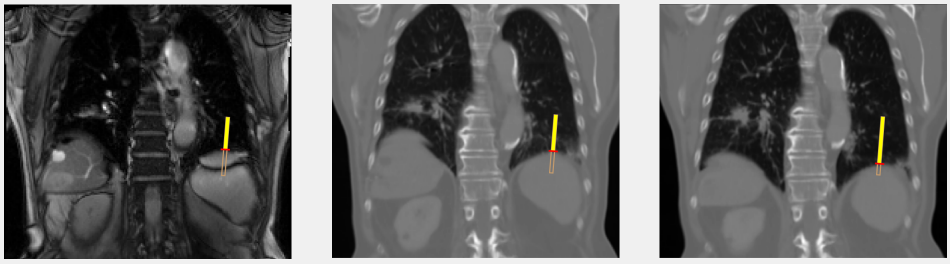
Selecting the right image and deforming it as needed requires complex image processing. It encompasses organ position and motion measurements. The tracking of the motion is done by placing navigators (See hint 3).

Initially, Dasnoy had chosen to use a manual positioning of the navigators by the doctor. The objective of this master thesis is to automate that part of the process. It is described in Part III.

[4] 2D plane that is captured by the MR image

Hint 3 : What is a navigator?

A navigator is a line segment that crosses an interface between two tissues. We call the intersection : *Crossing Point*. The interface that navigators are crossing, should appear on both the MR and CT modalities.



(a) Live MR Image (b) 2DCT Element (c) 2DCT Element

Figure 2.3: Navigators

2.3.3 3DCT Phase Creation

When the patient breathes, the interfaces between two organs may move, making the crossing points of the navigators oscillate between two boundaries. By measuring the distances from one of the end points of the navigators to the crossing point, one can estimate at which breathing period the patient currently is. The breathing period is then used to select the 2DCT image that is the closest. The 3DCT image, from which the selected 2DCT image was sliced out, is then deformed using the 3D Deformation Field computed at planning so that it fits better to the 2DMR image.

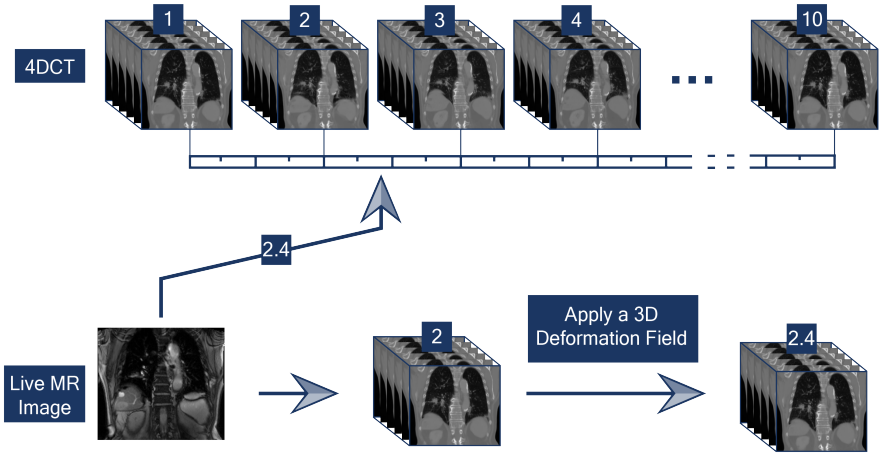


Figure 2.4: 3DCT Phase Creation

Chapter 3

Scope of this Master Thesis

Now that Dasnoy's method has been drafted, the scope and the objectives assigned to this master thesis can be clearly stated.

3.1 Problem Setting

The objective assigned to this master thesis is to automate the navigator positioning step within Dasnoy's method. This is why we have decided to call our method : **Auto-Navigator method**.

Currently, the doctor must place the navigators manually. Our objective is to prepare a few navigators as a proposal to the doctor. Based on that proposal, he/she will have the possibility to add other navigators and to remove the ones that are not relevant from his perspective. This logic of supporting the doctor and leave him with the final word, instead of automating fully the method, ensures there is a medical control on the output of the Auto-Navigator method.

The entire Auto-Navigator method must be fast in order to avoid idle time for the doctor. This was a key criteria for our development. We can already state that we have achieved an execution time ranging between 1 to 2 seconds for a MRI sequence containing 30 frames.

3.2 Scientific Field

The solution we came up with uses several **Computer Vision** tools.

Computer vision is a scientific field that focuses on the extraction of information by processing images. The problems solved using computer vision are of different natures. Well-known examples are face recognition systems and self-driving cars. In our case, computer vision will be used to process medical images and to identify specific edges.

At the beginning of this master thesis, we had the idea to use a neural network to determine the best navigators positions. Therefore, we started to use computer vision to extract features to feed the network. However, while investigating the problem, we noticed that it was possible to work without the use of a neural network. Hence, we decided to focus on that solution, leaving the field of deep learning aside.

Comparing two methods, one with deep learning, and the other one without, could be an interesting extension of this master thesis.

3.3 Re-Interpretation of the Problem

The method to be developed should deliver a proposal of navigators positions. The navigators should be positioned on edges that are present both on CT and MR images and that are moving the most. We favor a large movement because it makes it easier to differentiate the breathing periods in Dasnoy's algorithm.

3.3.1 Edges

Looking at a CT image, the human eye identifies an edge thanks to drastic change in gray intensity. Our method will do the same, and search for areas of high variations of pixel intensities. In mathematics, the high variation of a function is characterised by its derivative. In computer vision, the Sobel operator[1] does that. For more details, please refer to Chapter 4, Theory Background.

[1] Image processing tool used to estimate the derivative of an image. See section. 4.2.2 for more details.

3.3.2 Movement

To identify the movement of an object, one must compare its position at a time t_1 and at a time t_2. In computer vision, the movement is characterised in the same way : for each pixel, it computes a vector that represents the movement of a pixel between two pictures. This is done for every pixel of the image, and that leads to a vector field called displacement field. To identify pixels that are moving, it is necessary to look after pixels for which the associated vector in the displacement field has a high norm. For more details on displacement fields, please refer to Section 4.3.

3.4 Challenges related to Data

The re-interpretation of the problem in computer vision raised two challenges related to the data used as input to our method/module.

3.4.1 Challenge 1 : Discarding Artifacts

Artifacts are blobs appearing on images that do not correspond to real anatomical structures. The following two types of artifacts are present in the data used: the "4DCT reconstruction artifacts" and the "MRI banding artifacts". In a sequence of images, an artifact would typically appear on some images and then disappear. When trying to identify edges, artifacts must be discarded in order to allow the interface tracking to last during the entire sequence. Identifying artifacts is a challenge. Based on a single image, they cannot be differentiated from a real edge, since they also correspond to a local variation of pixel grey intensity. The solution we developed to solve that issue is explained in Section 5.3.

3.4.2 Challenge 2 : Edges must appear on both image modalities, CT and MRI

The procedure applied to acquire the two data-sets does not allow for a perfect alignment between a MR and a CT image. Since computer vision is based on the pixel coordinates, having no perfect match between the two modalities increases the difficulty in identifying edges that appear on both modalities. The solution we developed to face this challenge is detailed in Section 5.5.

3.5 Implementation and Data Description

In this master thesis, we have developed a method and we have implemented it in a Python module. To support the implementation of the module, concrete data sets were necessary.

The data that is processed by the Auto-Navigator method consists of two sequences of images related to the same patient: one sequence containing several 2DMR images and one containing 10 2DCT images. We recommend to have a minimum of 30 images within the MRI sequence so that the estimations computed during the method are more representative. This is essential for the success of this method.

The actual images that were used for the development of the module were captured from patients suffering from primary or secondary liver cancers. It is to be noted that both abdominal compression and audio-coaching were used as motion mitigation measures. Those images were acquired in accordance with The Code of Ethics of the World Medical Association within the framework of Dasnoy's researches.

3.6 Final Remark

The **Auto-Navigator method** presented in Chapter 5, is the preferred solution to a problem that we have analysed from different angles. There has been trial and error. The paths we have explored, which did not lead to anything promising, were discarded from this document.

The images that have been selected as illustrations to facilitate the understanding of Chapter 5 are all related to one single patient. The corresponding images related to two other patients can be found in the Appendix C.

Part II

Prerequisites

Chapter 4

Theory Background

This chapter provides background information on some concepts that will be referred to in Chapter 5, where the core of our method is explained. The reader can chose to read or not a section, depending on his knowledge of the subject. Concepts covered are :

1. Image Filter (Kernel)
2. Image Gradient (Sobel Filter)
3. Dense Displacement Field
4. Entropy
5. Mutual Information

4.1 Image Filter (Kernel)

In computer vision, an image filter, also called a Kernel, is a local operation that is repeated on every pixel of the image. This is the most basic tool used in image processing. This section focuses on linear kernels, as they are the ones used in this master thesis.

A linear kernel is represented by a square matrix of an odd size. The result of applying a linear kernel of size $2k+1$ on a pixel p is a weighted sum of the values of the pixels that are in a square of size $2k+1$ centered on p. The weights associated to each pixels are the elements of the matrix.

An illustration of a linear kernel of size (3, 3) is shown in Figure 4.1.

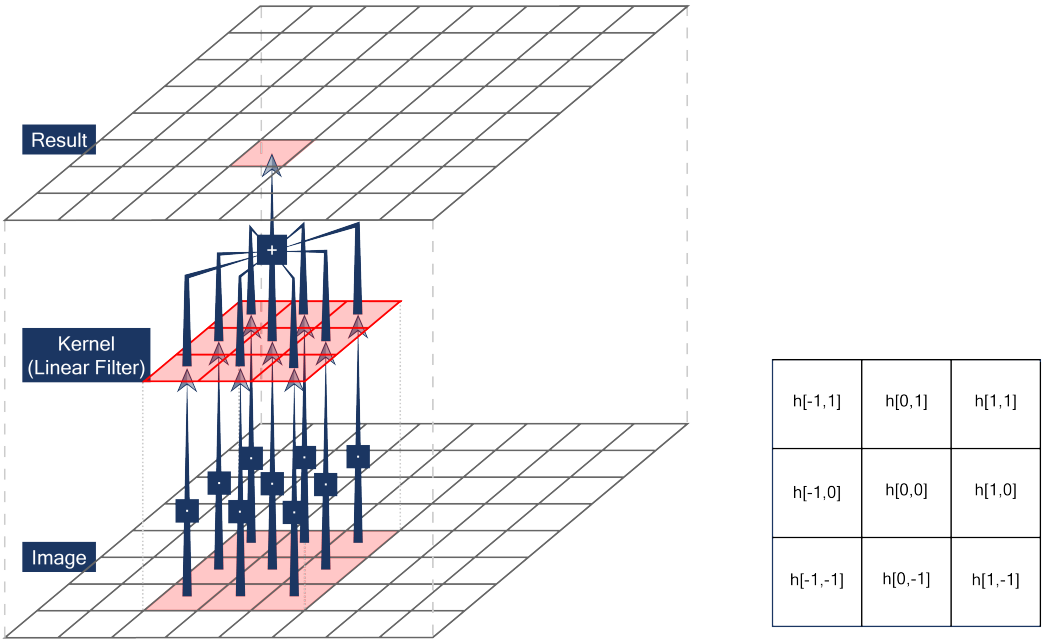


Figure 4.1: Linear Kernel of size (3, 3)

The Math

Suppose f is an image, h is a matrix of size $(2k+1) \times (2k+1)$ and g is the result of applying the filter h on the image f. The mathematical definition of g is :

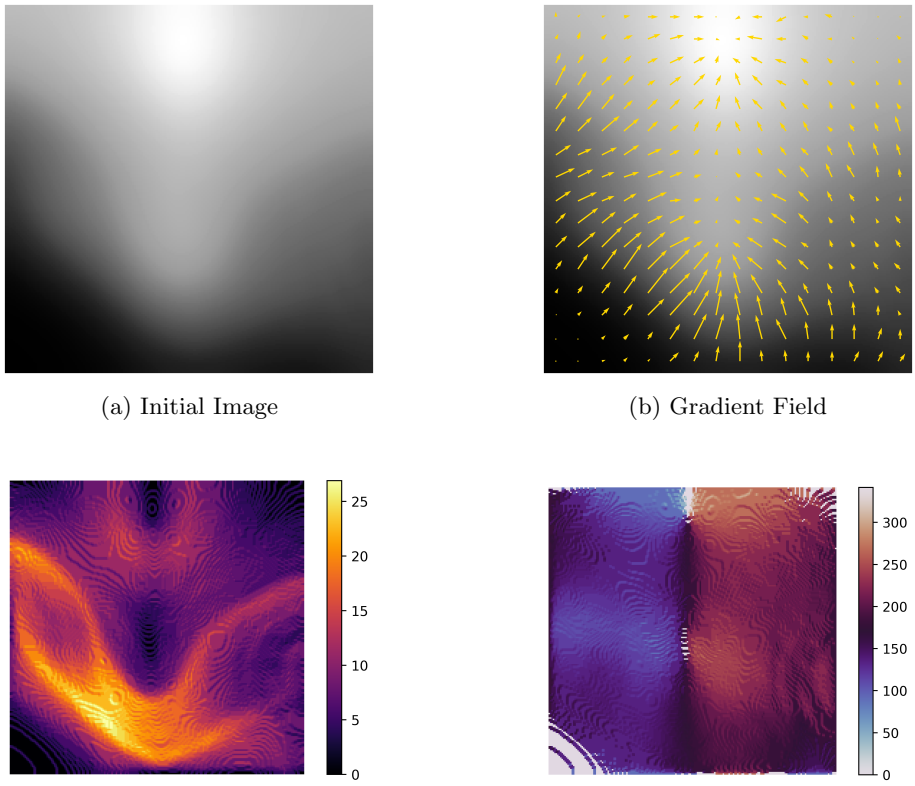
$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] * f[i-u, j-v] \tag{4.1}$$

The way h is indexed is shown in Figure 4.1.

4.2 Image Gradient

The gradient of a n-D image is a vector field that represents the variation of intensity in the image. For each pixel, the vector field contains a n-D vector that points towards the direction of the biggest intensity change. The norm of the vector, also called magnitude, reflects how big the change in intensity is.

Figure 4.2b illustrates the image gradient of a 2D image. Figures 4.2c and 4.2d show the magnitude and the angle matrices of that gradient field.



(a) Initial Image

(b) Gradient Field

(c) Magnitude Heatmap

(d) Angle Heatmap (in degrees)

Figure 4.2: Gradient Field Characteristics

Mathematically, the image gradient ∇f is computed by estimating the partial derivatives of the image in the x and y directions (respectively g_x and g_y).

$$\nabla f = \underbrace{g_x \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\text{gradient in the x direction}} + \underbrace{g_y \begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\text{gradient in the y direction}} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{4.2}$$

The direction and the magnitude of the gradient vector can be computed using simple geometric formulas.

$$\begin{aligned} ||\nabla f|| &= \sqrt{g_x^2 + g_y^2} \\ \theta_{\nabla f} &= \arctan \frac{g_y}{g_x} \end{aligned} \tag{4.3}$$

4.2.1 Central Difference Approximation

The central difference approximation provides a good first order estimation of the local value of the derivative of a 1D function.

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{4.4}$$

Considering that the row and the column of a pixel are discrete functions, one can use the central difference approximation to estimate the two partial derivatives, and thus the gradient.

For the image f, the intensity of pixel (i, j) is $f[i, j]$, and the two gradients can be approximated as follows:

$$\begin{aligned} g_x = \frac{\partial f}{\partial x} &\approx f\left[i + \frac{1}{2}, j\right] - f\left[i - \frac{1}{2}, j\right] \\ g_y = \frac{\partial f}{\partial y} &\approx f\left[i, j + \frac{1}{2}\right] - f\left[i, j - \frac{1}{2}\right] \end{aligned} \tag{4.5}$$

4.2.2 Sobel Estimation of the Gradient

The Sobel operator is a more sophisticated approximation of the gradient than the central difference. It combines a smoothing filter in one direction and a derivative filter in the other direction. The smoothing filter is applied first in order to reduce the noise amplification induced by the derivative filter.

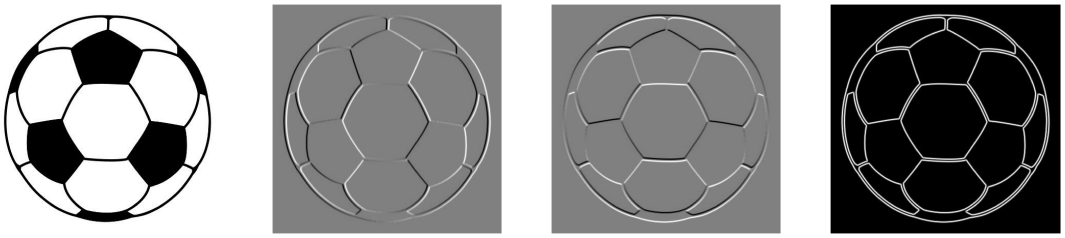
Let's develop the computations for a filter of size 3x3. The results of the smoothing filter are written as s_y and s_x. The Sobel filter estimations are written as S_y and S_x. Analytical development is only provided for S_x, the development for S_y can be obtained following the same logic.

$$\begin{aligned} s_y[x, y] &= f[x, y-1] + 2f[x, y] + f[x, y+1] && \text{(Smoothing)} \\ S_x[x, y] &= s_y[x+1, y] - s_y[x-1, y] && \text{(Central Difference)} \\ &= (f[x+1, y-1] + 2f[x+1, y] + f[x+1, y+1]) \\ &\quad - (f[x-1, y-1] + 2f[x-1, y] + f[x-1, y+1]) \end{aligned}$$

The result of applying a Sobel filter of size 3 is a weighted sum of the pixels which are in a square of size 3. The filter can thus be expressed as a linear kernel of size 3x3. The two kernels are :

$$K_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad K_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Figure 4.4 shows the result of applying Sobel filters on an image. Applying K_x (resp. K_y) highlights the change of contrast in the width (resp. height) direction of the image. Figure 4.4d shows the magnitude of the gradient field (computed with equation 4.3).



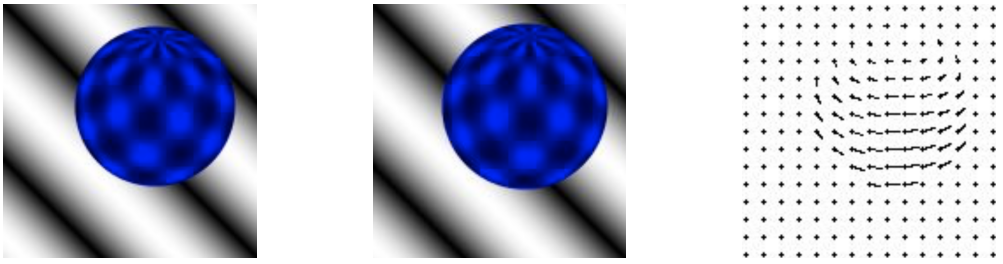
(a) Initial Image (b) Result after K_x (c) Result after K_y (d) Magnitudes

Figure 4.4: Application of a Sobel Filter

4.3 Dense Displacement Field

A dense displacement field is a vector field where each vector represents the movement of a particular pixel. This implies that if the image has n dimensions, the movement will be in n dimensions too. We will however restrain ourselves to 2D pictures.

Displacement fields are computed by using two slightly different pictures of a same scene. The differences in the pictures can occur for various reasons, such as a movement of the subject or a movement of the camera itself. Figure 4.5 shows an example of displacement field computed between two images where a sphere turns slightly on itself to the left. Figure 4.5c shows that the vectors corresponding to sphere are pointing to the left[1], while the vectors corresponding to the background, which is static, are zero vectors.



(a) Frame 1 (b) Frame 2 (c) Displacement Field

Figure 4.5: Displacement Field [14]

[1] To be complete, one can notice that the vectors associated to parts of the sphere that are on the other side of the rotation axis, are pointing to the right.

If a displacement field F_A^B is computed using two images A and B, for all pixels p in A, F_A^B contains a vector that represents the displacement of p from A to B.

The Math

Suppose the pixel $A\,[x_1,\, y_1]$ moves to $B\,[x_2,\, y_2]$, F_A^B would be such that :

$$F_A^B\,[x_1,\, y_1] = \,(x_2 - x_1,\, y_2 - y_1) \tag{4.6}$$

For simplicity in the notation, equation 4.6 can be reformulated as follows :

$$(x_2,\, y_2) = (x_1,\, y_1) + F_A^B\,[x_1,\, y_1] \tag{4.7}$$

4.4 Entropy

The entropy of a random variable measures the extend to which the variable is unpredictable. Suppose we have two dices, the first one ($\mathcal{X}$) is a classical dice with 6 sides and the second one ($\mathcal{Y}$) has 12 sides. X and Y are the two random variables representing the outcome of rolling the dices. The entropy of X will be lower than the entropy of Y. That is because Y can take more values than X and is thus less predictable.

Entropy is not only influenced by the size of the domain of a random variable. It is also sensitive to the frequency of occurrence of the elements of the domain. Suppose that the dice $\mathcal{Y}$ is loaded on one side (cheated) and that its outcome takes 9 times over 10 the value 12. The entropy of Y will then be lower than the entropy of X. This is because Y is now more predictable than X.

The mathematical definition of the entropy of a random variable X, having for domain $\mathcal{X}$ and for probability density function $P\,(X)$, is :

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) log\,(P(x)) \tag{4.8}$$

4.5 Mutual Information

The mutual information, $I(X;Y)$, measures the relationship between two random variables, X and Y. Informally, one could say that it measures how much a random variable is telling us about another random variable.

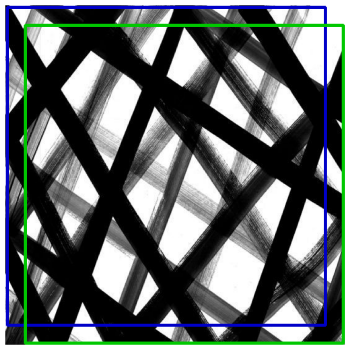
Suppose that X and Y are two random variables that correspond to the outcome of rolling a 6-sided dice. Knowing a concrete realisation of X will not provide any information about a concrete realisation of Y, because those are two completely independent events. This implies that there is no mutual information between X and Y, and thus $I(X;Y) = 0$.

Suppose now that a third random variable, Z, is defined as the sum of X and Y. Now imagine that for a rolling, Z takes the value 3. Knowing this, the possible outcomes of X and Y can be narrowed down to $(X,Y) \in \{(1,2),(2,1)\}$. Therefore, knowledge of Z reduces the unpredictability of the two random variables X and Y. Therefore, $I(X;Z) > 0$ and $I(Y;Z) > 0$.

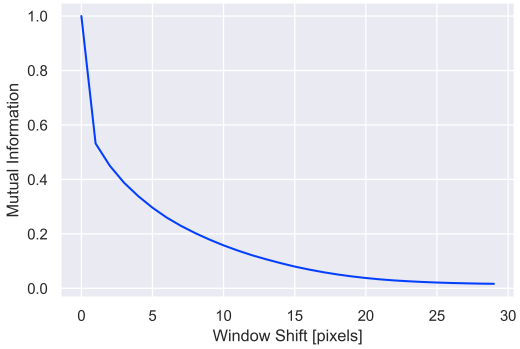
Using the concept of entropy, the mutual information can be defined as :

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned} \tag{4.9}$$

Figure 4.6 shows the evolution of the mutual information between two images when targeted area progressively moves off. The point at axis $x = 0$ corresponds to the mutual information between the blue squared image in Figure 4.6a and itself, and the point at $x = 30$ corresponds to the mutual information between the blue squared image and the green squared image in Figure 4.6a. The points in between are computed by progressively sliding the green squared image.



(a) Initial image where the window slides from the blue to the green image pixel by pixel



(b) Evolution of the MI

Figure 4.6: Mutual Information

Part III

Development

Chapter 5

Automation of Navigator Positioning

The method developed to automate the navigator positioning can be depicted as a **five step process**.

- **The first four steps** aim at identifying regions where it is relevant to position navigators. Those steps correspond to the first phase in the algorithm which is called **Region-Selection Phase**.
- **The fifth step** aims at positioning one navigator for every selected region. This last step corresponds to the second phase in the algorithm called **Navigator-Positioning Phase**.

It should be noted that the Region-Selection phase can be used to select body regions in other contexts than navigator positioning. Methods that rely on displacement fields of similar regions could for instance be implemented too. Therefore, navigator positioning can be seen as a specific application of a more generic method that aims at selecting matching MRI-CT regions.

In this chapter, an overview of the five step method is provided. Details for each step are then provided. However, before proceeding, some of the working assumptions must be clarified.

5.1 Working Assumptions

Before developing our method, a few working assumptions had to be made to answer the following questions :

- What are the characteristics of relevant regions?
- How to deal with two independent input data sets?
- How to deal with moving anatomical elements?

5.1.1 What are the characteristics of relevant regions?

The objective of the Region-Selection phase is to identify regions that are relevant to place navigators. In order to decide whether a region is relevant or not, the following criteria were chosen :

- A relevant region should **be located on an edge**, and that edge should be moving. This criteria comes from Dasnoy's tumor tracking method. (See section 2.1)

  This criteria is applied in step 1.

- The direction of the movement of an edge should be as close as possible to its **gradient direction**. This direction aims at maximising the amplitude of the crossing point[1] on the navigator. The logic behind this criteria is illustrated below. (See Hint 4)

  This criteria is applied in step 2.

- The edge should appear **both in the CT and the MRI sequences**, as per Dasnoy's tumor tracking method. (See section 2.1)

  This criteria is applied in step 3.

- An relevant region should be **equal or larger than a minimal size**. The rationale for this assumption is explained later. (See Hint 9)

  This criteria is applied in step 4.

[1] Crossing point : Intersection between the edge and the navigator.

Hint 4 : Why focusing on the gradient direction?

Figure 5.1 below depicts an edge that moves vertically between time t_1 and t_2. The positions t_1 and t_2 of the edge are represented by the green and the red circles respectively. Three navigators are placed: n_1, n_2 and n_3.

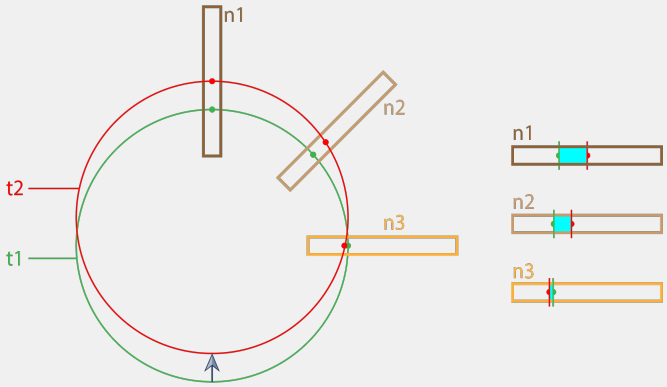


Figure 5.1: Gradient Direction Impact

One can see that the navigator n_1 records a much larger amplitude than the navigators n_2 and n_3. This is because navigator n_1 is crossing the edge at a point where the direction of the movement is close to the gradient direction. The gradients here are vectors that are perpendicular to the circle.

5.1.2 How to deal with two independent input data sets?

As already mentioned in the data description section (Section 3.5), the data that is received as input consists of two sequences of images from two different modalities: the MRI sequence and the CT sequence. The selected edges should appear on both modalities. To ensure this condition, it was decided to start by identifying edges on one of the two modalities and to remove afterwards the edges that do not appear on the other modality.

We have decided to start by processing the MRI sequence, the reason being that MRI images contain more details than CT images. Hence, they provide more reliable estimations of dense displacement fields, which is an important aspect at the start of the method.

5.1.3 How to deal with moving anatomical elements?

In order to identify relevant regions across the steps, various types of information/characteristics of the anatomical elements appearing on the image sequences will be looked at. A characteristic could for instance be the gradient magnitudes or the successive displacement distances. However, since anatomical objects are moving during image capturing, they do not appear at the same position (pixel coordinates) on the various images. This has raised two challenges :

(i) How do we point to an anatomical element that moves from frame to frame ?

(ii) How do we compute an average characteristic (e.g. gradient magnitudes) of an anatomical element that moves from frame to frame ?

(i) The concept of "Point" to refer to moving anatomical element

To refer to an anatomical element that moves from frame to frame, a list that contains all the coordinates it occupies through the sequence could have been used. However, as displacement fields tell how elements move from frame to frame, only the coordinates on one of the frames is needed to compute all the other positions. Later on, anatomical elements will be uniquely identified by their pixel coordinates in the first frame. This will be our definition of a **"Point"**. For instance, "*Point (100, 150)* " will refer to "*the anatomical element that appears at the pixel of coordinates (100, 150) in the first frame of the MRI sequence*".

Given the above definition, a **"Region"** can be defined as a **collection of points**. Therefore, a "Region" does not refer to a geographical area on the image, but refers to a part of an anatomical object of the body.

(ii) The "Follow-Point-Process" to track moving anatomical elements

Our method requires that various characteristics of the points (e.g. gradient magnitude, successive displacement distance,..) are computed. For instance, to compute the mean gradient magnitude of a point, the related anatomical element must be followed from frame to frame, the various gradient magnitudes must be collected, and the average must be calculated. In order to automate such logic, the **"Follow-Point-Process"**, "**FFP**", was developed.

This process takes as input a sequence of matrices, and outputs their average relatively to each point.

5.1.3.1 Example of FFP Execution

The Follow-Point-Process is an important part of the method. Figure 5.2 illustrates the process by means of an example.

Suppose that a sequence F contains only 4 MR images and that a sequence G contains the gradient magnitudes of the 4 MR images. F^i and G^i denote the i-th MR image and its corresponding gradient magnitude matrix. M_G is the result obtained by passing G as an input to the Follow-Point-Process.

Point p (red square in Figure 5.2) occupies the pixels $(1,0)$ in the first frame, $(2,2)$ in the second, $(4,1)$ in the third, and $(5,1)$ in the fourth. To compute the mean of the gradient magnitude of point p over the sequence, the elements $(1,0)$ of G^1, $(2,2)$ of G^2, $(4,1)$ of G^3 and $(5,1)$ of G^4 must be averaged.

Formally:

$$M_G^p = M_G[1,0] = \frac{1}{4}\left(G^1[1,0] + G^2[2,2] + G^3[4,1] + G^4[5,1]\right) \tag{5.1}$$

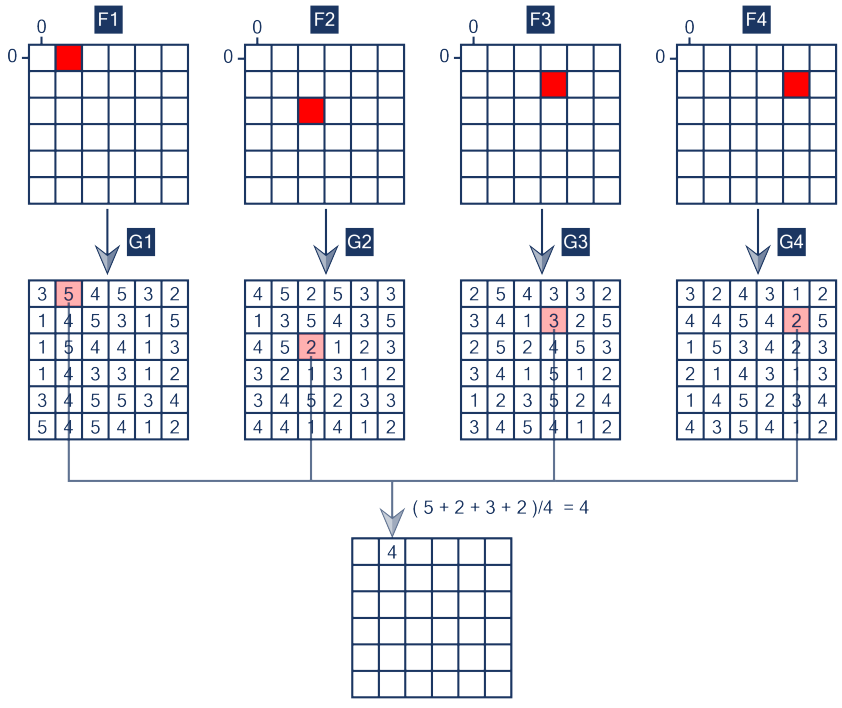


Figure 5.2: Follow-Point-Process example

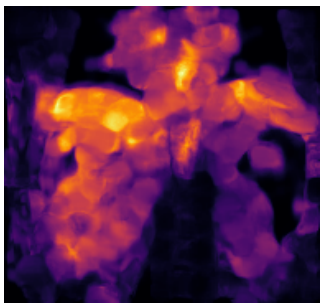
5.1.3.2 Implementation of the FFP

To follow the points through the sequence, displacement fields are used. By computing displacement fields between the first frame and all the others, the path followed by each pixel of the first frame can be traced back throughout the sequence. To sum the right elements of the matrices for each point, the matrices can be re-organised by using permutations computed based on the displacement fields. In the implementation part of the FFP, two options are available for computing the displacement fields.

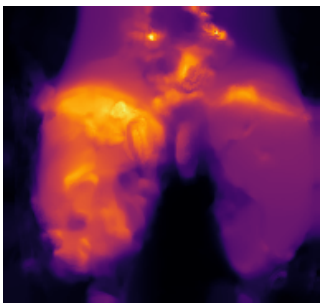
The first option is an implementation of Farneback's algorithm [15]. This algorithm is based on polynomial expansion. The advantage of the Farneback algorithm is that it is very efficient. Also, an implementation is available in the OpenCV library [16].

The second option is an implementation of Brox's algorithm [17]. This algorithm is an extension of Farneback's method. Studies have shown that Brox's algorithm is better suited for medical image optical flow estimation [18]. However, a drawback is that it can be very slow, and that there is no generally accepted implementation in the scientific community. For this master thesis, an open source implementation of Brox's algorithm found on the following GitHub repository was used : `https://github.com/pathak22/pyflow`. Even though the result obtained with that implementation seems very good, nothing can guarantee the accuracy of the code as no extensive evaluation was done so far.

The two figures below show examples of gradient magnitude matrices obtained with the two methods. One can see that Brox's algorithm provides a much smoother estimation of the dense displacement fields.



(a) Gradient magnitudes - Farneback

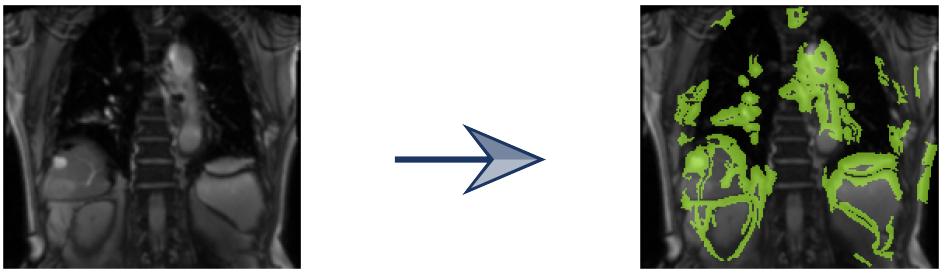


(b) Gradient magnitudes - Brox

5.2 Overview of the five step process

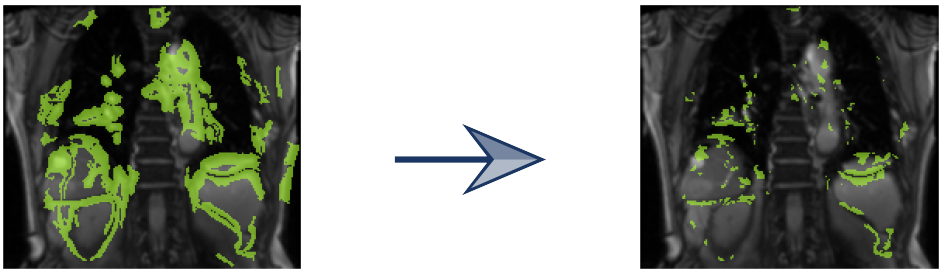
Our method is a five step process. It works as a funnel starting with a large collection of potential relevant points. At every step of the Region-Selection phase, the number of relevant points is reduced. Every step can be seen as a function that takes as input a binary mask containing potential relevant points, and filters those points according to one of the criteria listed in section 5.1.1. The output of every step is therefore also a binary mask. By discarding points that are of no interest, the number of further computations is progressively reduced. This is also the reason why the steps that remove the most points are put first.

The first step identifies in the MRI sequence, the points that are situated on moving edges.



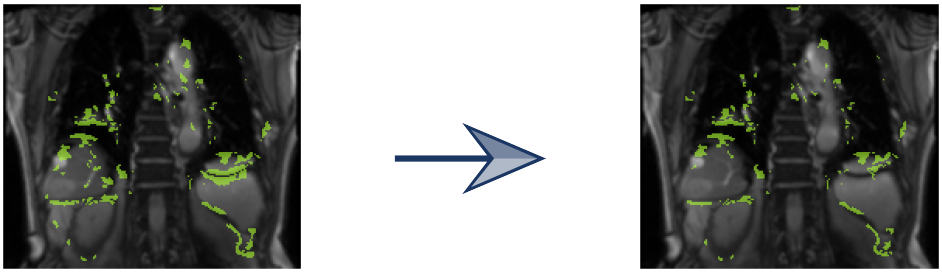
(a) Before step 1 (b) After step 1

The second step discards points that do not move in a specific direction.



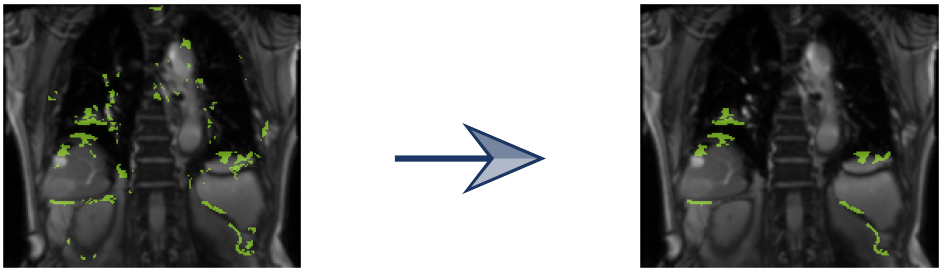
(a) Before step 2 (b) After step 2

In the third step, the CT knowledge is taken into account and the points situated on edges that are not appearing in the CT sequence are removed.



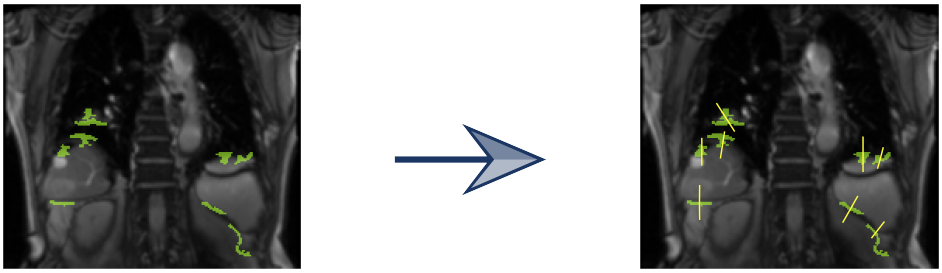
(a) Before step 3 (b) After step 3

The fourth step filters the remaining points to discard those that belong to too small groups of points.



(a) Before step 4 (b) After step 4

In the fifth step, one navigator is placed for every selected region. The coordinates of those positioned navigators correspond to the actual output of the method.



(a) Before step 5 (b) After step 5

5.3 Step 1 - Moving Edges Identification

The first step of the method, aims at selecting the points that are on moving edges. To do so, the algorithm computes an Edge-Matrix in which each element contains a scalar that captures the inclination of a point to be on an edge. Thereafter, it computes a Movement-Matrix in which each element contains the average distance by which each point moves from one frame to the next. Finally, both results are combined to select the points that are on edges and that are moving.

5.3.1 Edge-Matrix

The starting point consists of computing an approximation of the gradients magnitudes for every frame of the MRI sequence. This is done by using a Sobel filter. The size of the Sobel filter is one of the parameters of the method, and is referred to as k in the table of parameters in Appendix B. In our implementation, k was set to 5. The functions used are the two pre-implemented OpenCV [16] functions: `cv2.Sobel` and `cv2.cartToPolar`.

The output is a sequence of matrices that can be passed as argument to the Follow-Point-Process. Given that sequence of matrices, the FPP will deliver a matrix containing the **mean-gradient-magnitude** for every point. Since every element of that matrix is associated to a particular pixel, the matrix can be visualised by printing it as an image. The result is shown in Figure 5.10a.

The Math

To put this process into equations, the matrix containing the mean gradient magnitude per point of all the points is defined as μ_G. Based on the formal definition of the displacement field given in section 4.3, μ_G can be written as follows :

$$\mu_G = \frac{1}{D} \sum_{i=0}^{D-1} P_0^i \left(G_i\right) \tag{5.2}$$

where :

- D is the number of frames in the MRI sequence.
- G_i is the gradient magnitude matrix of the i-th frame of the MRI sequence.
- P_0^i is a function that re-arranges the input matrix such that :

$$\left[P_0^i \left(G_i\right)\right]_{(h,w)} = \left[G_i\right]_{(h,w)+V_0^i(h,w)} \forall h, w : \tag{5.3}$$

Hint 5 : Advantage of the FFP Regarding Artifacts

The objective of step one is to identify the points that are located on an edge. Therefore, an option could have been to look at the gradient magnitudes of the points in the reference-frame. However, by looking at the mean instead, the points that are on MRI artifacts appearing in the reference-frame can be discarded. As a reminder, artifacts are blobs that do not represent a real anatomical part of the patient's body. They are thus points that should not be considered as located on an edge. Because artifacts are only appearing at particular moments of the sequence, the Follow-Point-Process will output a low mean-gradient-magnitude value for those points. This indicates that they must be discarded. Figure 5.9 shows an MRI artifact. A white blob appears on the left image but not on the right one. When looking at the left frame only, the artifact would have been selected.

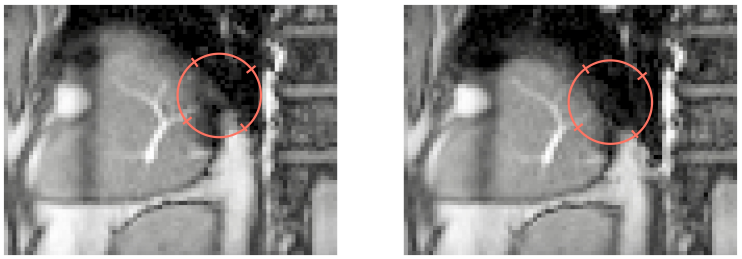


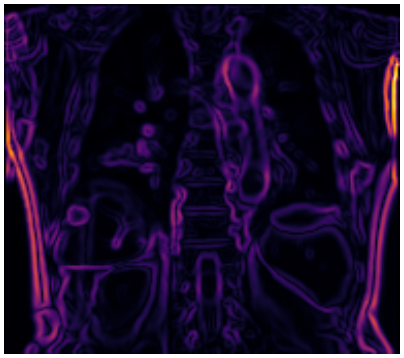
Figure 5.9: MRI Artifact

5.3.2 How to Compute the Movement-Matrix

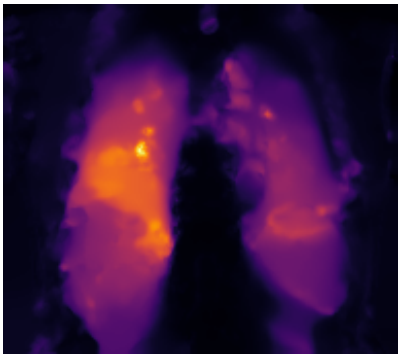
First, a sequence of matrices is computed such that each one contains the distance by which pixels move from a frame to the next one. To generate those matrices, a dense displacement field between all couples of consecutive frames is computed. The euclidean norm of those displacement vectors is taken.

That sequence of matrices is then passed as argument to the Follow-Point-Process to compute the **mean-distance-per-point**. That matrix can be visualised in Figure 5.10b.

Once the mean matrix is computed, a binary mask highlighting whether a particular point moves in average more or less than a given value can be generated. Figures 5.10c and 5.10d are two masks obtained using two different thresholds. White pixels correspond to points that moved in average less than the threshold. Conversely, black pixels moved more than the threshold. That threshold is one of the parameters of the method, and is referred to as τ_{dist} in the table of parameters of Appendix B. In our implementation, τ_{dist} was set to 0.5 pixels.



(a) μ_G



(b) μ_f

(c) $\tau = 0.6$

(d) $\tau = 0.2$

Figure 5.10: Matrices computed in step 1

The Math

This time, the argument passed to the Follow-Point-Process will be the sequence of displacement distances. If μ_f is the mean-distance-per-point, the following equation applies:

$$\mu_f = \frac{1}{D-1} \sum_{i=0}^{D-2} P_0^i(||V_i^{i+1}||) \tag{5.4}$$

where :

- $||.||$ designates the element-wise euclidean norm of the matrix set as argument.
- V_i^{i+1} is the dense displacement field between frame i and $i+1$ that follows the definition of equation 4.6

The mean-distance-per-point can only be computed using $D-1$ frames because the displacement field of the last frame cannot be evaluated.

The binary mask function M, that filters its input matrix to set zeros where the average displace-

ment is lower than τ_{dist}, is defined as :

$$M(A) = \begin{cases} (A)_{(i,j)} & \text{if } (\mu_f)_{(i,j)} > \tau_{dist} \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

5.3.3 Binary Mask Creation

At this stage, the binary mask function M is applied to μ_G to obtain a matrix that contains the gradient magnitudes only for the points that are moving enough, and zeros for the other points.

A threshold must be set for the new matrix ($M(\mu_G)$) in order to discard points that are not on edges. The threshold used here is also one of the parameters of the method, and is referred to as τ_{Sobel} in the table of parameters in Appendix B. However, if $M(\mu_G)$ were to be thresholded directly, the value of τ_{Sobel} would be influenced by k, the value of the Sobel filter used[2]. This is a source of bias as those two parameters should vary independently. To counter that issue, $M(\mu_G)$ was first mapped to a known interval, $[0, 10[$, so that the threshold could be applied on that interval.

5.4 Step 2 - Mean Angle Difference Thresholding

Now that the moving edges are located, the identification of those that are moving in a specific direction can start. The working assumption here, is that the desired movement direction of an interface should be close to it's gradient direction. This is because the closer those two directions are, the bigger the amplitude on the navigator is. This is further explained in Hint 4.

To identify the points of such interfaces, the same procedure as in step 1 was used. A sequence of matrices is created such that each one contains, for every pixel, the angle difference between the two directions in a particular frame. That sequence is then passed as argument to the Follow-Point-Process to compute the **mean-angle-difference-per-point** (μ_{δ_ϕ}).

5.4.1 Angle Difference Matrix

The angle-difference (δ_ϕ) is computed as a function of the angle of the movement direction (ϕ_f, "f" for *flow*) and the angle of the gradient direction (ϕ_g). To define the angle-difference function, one must not forget that the movement vector will be of opposite direction during inhalation and exhalation. The angle difference however, should not be sensitive to this inversion. This is because the movement direction of an interface only influences the direction in which a crossing

[2] According to the definition of a Sobel filer given in section 4.2.2, the bigger the size of the filter, the larger the range of values produced.

point moves on a navigator, not the amplitude it produces. For this reason, the angle-difference function is not a simple difference between ϕ_f and ϕ_g. Hint 6 further details this.

The angle-difference function, which was chosen instead, is to consider two crossing lines defined by the gradient and the movement vectors, and to consider δ_ϕ to be the smallest of the two crossing angles appearing. Figure 5.11 shows three different cases of δ_ϕ.

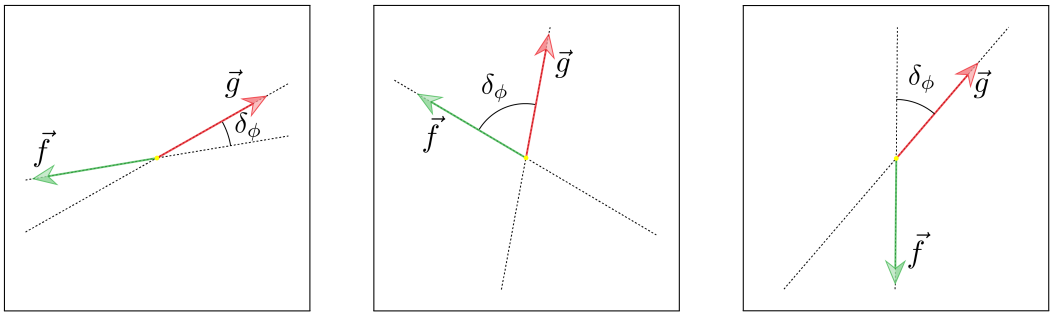


Figure 5.11: Examples of angle difference

The Math

With the assumption that the gradient and movement angles are given in degrees, the formula of the the angle difference that was chosen 6 is the following :

$$\delta\left(\phi_f, \phi_g\right) = min\left[(360 + \phi_f - \phi_g) \mod 180,\ (360 + \phi_g - \phi_f) \mod 180\right] \tag{5.6}$$

By defining the matrices of the gradient and movement angles for the i-th frame as Φ_i^g and Φ_i^f, and by defining the corresponding error matrix by Δ_i^Φ, equation 5.6 take the following matrix form :

$$\Delta_i^\Phi = min\left[\left(360 - \Phi_i^f + \Phi_i^g\right) \mod 180,\ \left(360 - \Phi_i^g + \Phi_i^f\right) \mod 180\right] \tag{5.7}$$

Where the *min* function is the element-wise minimum function.
Finally the expression of the mean angle difference matrix is :

$$\mu_{\Delta^\Phi} = \frac{1}{D}\sum_{i=0}^{D-1} P_0^i\left(\Delta_i^\Phi\right) \tag{5.8}$$

Hint 6 : Why is the angle-difference not a simple subtraction?

The two figures below show both an edge at two different moments in time. The blue edge is captured at time t, while the gray one is captured at time $t+1$. The patient inhales in Figure 5.12 and exhales in Figure 5.13.

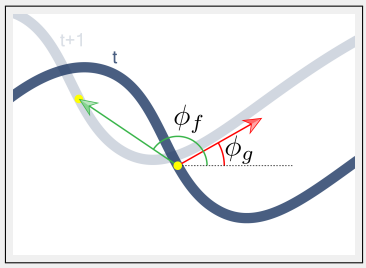


Figure 5.12: Inhalation

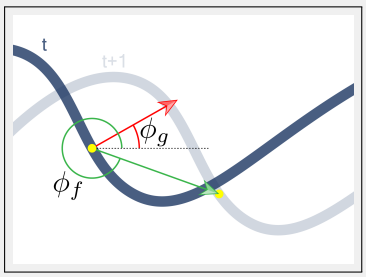


Figure 5.13: Exhalation

The edge at time t (the black one) moves to the left top corner during inhalation and to the opposite corner during exhalation. This produces two flow vectors $\vec{f_i}$ and $\vec{f_e}$ of opposite directions. The gradient's direction however stays the same. Therefore, a simple difference between the two angles would result in a difference of 180 degrees between an angle difference computed during inhalation and an angle difference computed during exhalation.

5.4.2 Binary Mask Creation

Once the sequence of matrices is computed, and passed as argument to the Follow-Point-Process, a matrix is obtained, where each element represents the average angle difference for a particular point. A threshold is then applied to that matrix to create a binary mask. The threshold that is applied is one of the parameters of the method and is referred to as ϕ_{max} in the table of parameters in Appendix B. In our implementation, ϕ_{max} was set to 35 degrees.

The binary mask defined in this step is finally compared to the one that was defined at the end of step 1 leading to another binary mask which is the output of step 2. This mask identifies the points that are on edges that move in a direction that is close to the gradient direction of the edge.

In the implementation of the method, rather than computing the angle difference matrix for every point, it was decided to only compute those angle differences for points that were considered as relevant at the end of step 1. Doing so increases the algorithm's efficiency.

5.5 Step 3 - CT Correspondence

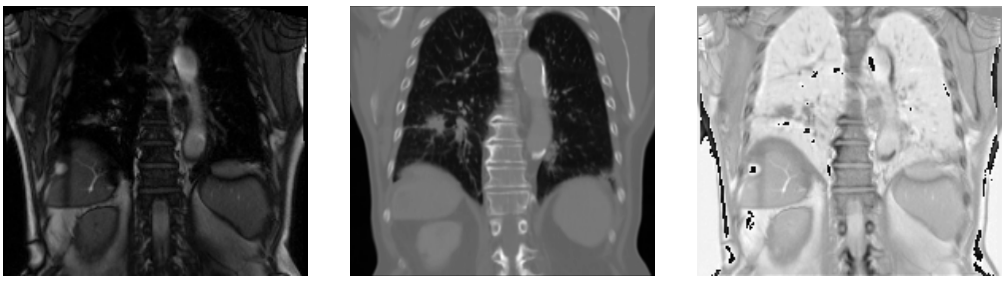
Step 3 consists of removing the points that are not located on an edge appearing on CT modality. As already mentioned in section 2, the CT sequence is composed of only 10 different CT frames that are played in a loop. It means there is no exact match between a CT frame and a MRI frame. As a consequence, knowing the coordinates of a particular point across the MRI sequence does not provide precisely where that same point is located in the 10 CT images. To counter that issue, the following two steps are applied :

1. A new CT sequence is recreated that has the same length as the MRI sequence. For every MR frame, the CT frame that is considered to be the closest in term of respiratory phase is selected.
2. The gradient magnitudes of the relevant points in that new CT sequence are analyzed to determine if they are located on an edge appearing in a CT modality, or not.

5.5.1 Mean CT Gradient Matrix

Creation of a New CT Sequence

Comparing CT and MR images (Figure 5.14) is more complicated than it may seem because they are composed of two different signal types. If both images were of MR or CT type, one image could be subtracted from the other, and at the error produced could be used to evaluate how close they are from each other. This can not be done to compare MR and CT images, as the difference would result in an image like in Figure 5.14c.



(a) MRI (b) CT (c) Difference

Figure 5.14: Subtraction of CT and MR images

A measuring tool particularly well suited for comparing two signals of different modalities is the **Mutual Information**. Our assumption here, is that the mutual information between a MR image and a CT scan should be greater if two images are closer in terms of respiratory phase. By selecting the CT frame that maximises the mutual information, and by doing this for every MR

frame, one can define the new CT sequence, referred to as CT′, as follows:

$$\mathrm{CT}'_i = \underset{\mathrm{CT} \in \mathcal{S}}{\operatorname{argmax}} \; \boldsymbol{I}(\mathrm{MR}_i;\mathrm{CT}) \tag{5.9}$$

where:

- $\mathcal{S}$ is the set of the 10 CT frames.
- $\boldsymbol{I}$ is the mutual information.
- CT'_i and MRI_i are respectively the i-th frame of CT′ and the i-th frame of the MR sequence.

5.5.2 Creation of the Mean Gradient Matrix

A Sobel filter is applied to all the CT′ frames to obtain an estimate of the CT gradient magnitudes. Applying the Follow-Point-Process directly to that sequence of matrices would not produce good results because the displacement fields computed on basis of the MR sequence (which are the ones that are used to follow the points in the Follow-Point-Process) and the CT′ sequence are not all perfectly aligned. Some of the MR images will be coupled to a CT image that is close in term of respiratory phase. This is shown in Figure 5.15.

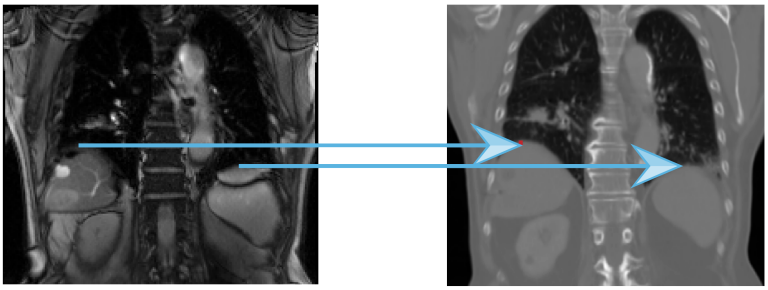


Figure 5.15: Good MR/CT Matching

Some of the MR images will have a breathing period that is too distant from any of the 10 available CT images. They will therefore be mapped to a CT image that is not the ideal one. This case is shown in Figure 5.16.

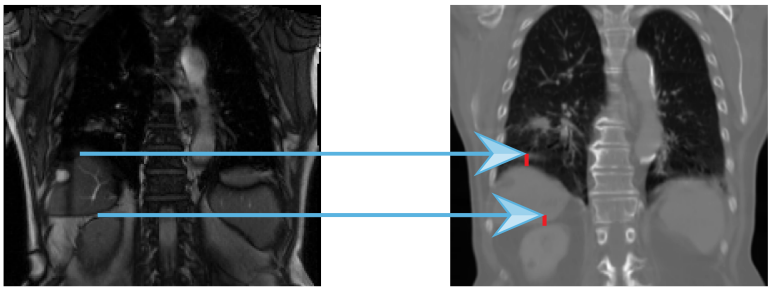


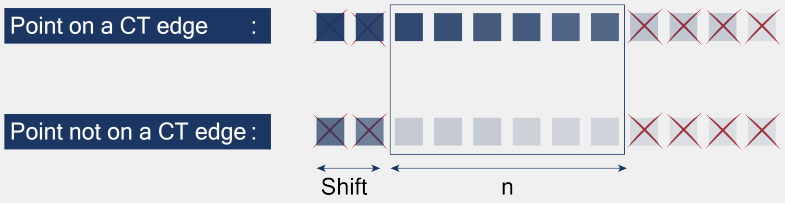
Figure 5.16: Good MR/CT Matching

Rather than computing the mean-CT-gradient-magnitude by considering all the CT′ gradients of a point, it was decided to first sort those CT′ gradients for each point and to compute the mean-CT-gradient-magnitude only on the basis of the shifted n highest values. This is explained more in detail in Hint 7. The shift and n are parameters of the method and are referred to as s_{CT} and n in the table of parameters in Appendix B.

Hint 7 : Explanation of "*shifted* n *highest*" gradient magnitudes

It is assumed that a point that refers to an anatomical element that is located on a CT edge should be mapped more frequently to a pixel that is located on an edge in the CT′ sequence, than a point that refers to an anatomical element that is not located on a CT edge. This is illustrated in the figure below. Each square represents the gradient magnitudes that are found by following a point with the FFP in the CT′ sequence. The shade indicates the level of magnitude of the gradient. A point that is located on a CT edge, is sometimes mapped on a pixel where the gradient magnitude is low. Also, a point that is not located on a CT edge is sometimes mapped on a pixel where the gradient magnitude is high.

By sorting the values of the gradient and by computing the mean on the shifted n highest, the gradients that are more representative of the nature of the point will be taken into account.



This alternative method to compute the means is very effective because it produces values that allows to segregate better the points that are located on CT edges, from those that are not. The effectiveness of this alternative method is shown in the figure below.

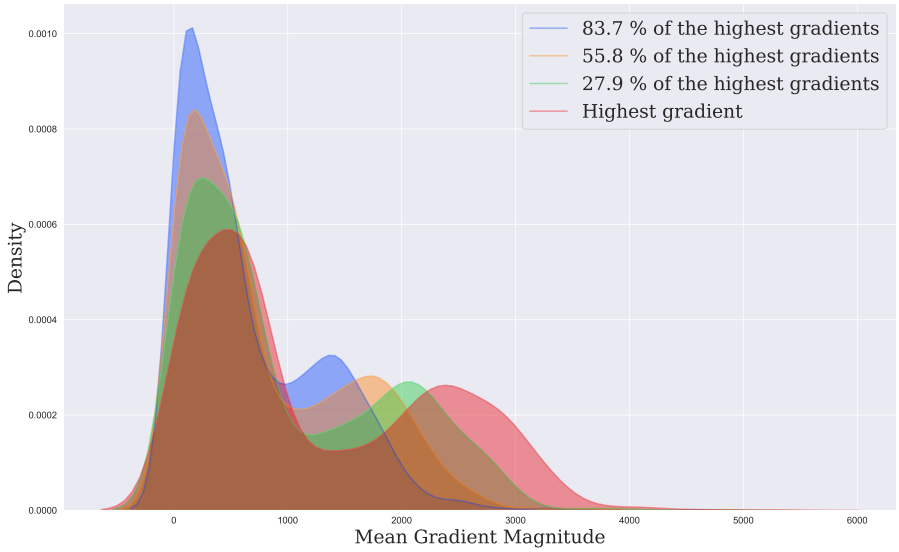


Figure 5.17: Probability Distributions of CT gradient for various values of n

Figure 5.17 shows the probability distribution of the mean for several different values of n. The shift applied to generate this distribution is constant and has a value of 3. The figure shows that the smaller the value for n, the better the two families are segregated. Conversely, when n increases, the two families tend to merge. The distribution of the mean obtained by using the Follow-Point-Process corresponds to the curve where n is maximum because it is the case where all the gradients are taken into account.

By default, n i set to 15, because it provides balance between differentiation and correctness, and s_{CT} is set to 3.

Hint 8 : Why choosing $n = 15$ instead of $n = 1$?

The smaller the value for n, the better the segregation between the two families. Why then chose $n = 15$ and not just $n = 1$?

Considering the way CT′ was build, it could happen that a point that is not located on a CT edge is mapped to a point that is on a CT edge in several CT′ frames. However, the probability of this happening 15 times is low. Therefore, $n = 15$ seemed to be an optimal choice.

5.5.3 Binary Mask Creation

Now that two families of points are segregated, a threshold is applied to the mean-CT-gradient-magnitude matrix in order to only keep the points that are on CT edges. An identical procedure as the one applied at step 1 was chosen for thresholding. The mean-CT-gradient-magnitudes are first mapped to a known interval, [0; 10[, and a threshold, τ_{CT}, is then applied on that interval. As a reminder, this procedure enables to make the parameters k (Sobel filter size) and τ_{CT} vary independently. By default, τ_{CT} was set to 3.

5.6 Step 4 - Group Size Thresholding

Throughout step 1, 2 and 3, points were selected that are on edges appearing in the CT and in the MRI modality, and that are moving in a specific direction. In step 4, points will be removed when they do not belong to groups that are big enough. To do so, points that are located on the same edge must be identified. The term "group" is used to refer to a collection of points located on the same edge. Defining these groups is the first stage in step 4.

Once the groups are defined, they are filtered so as to only keep those that contain a large number of points. The rationale is, that if a point is relevant, there should be several other points near it that are also relevant. (See Hint 9)

The data processed in this fourth step is a binary matrix delivered by step 3, where an element is set to "1" if it corresponds to a point that remained in the selection after steps 1, 2 and 3, and is set to "0" otherwise.

5.6.1 How to Define Groups ?

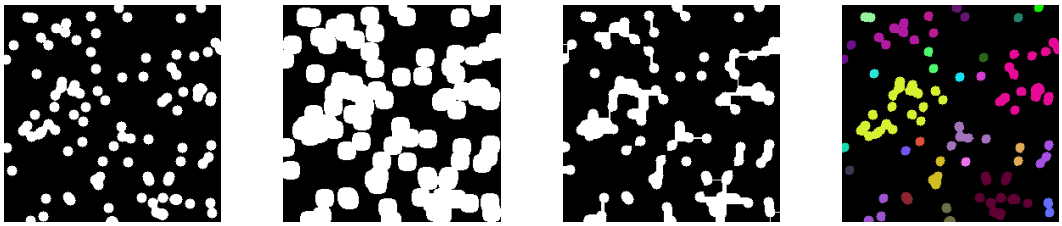
A simple way to define a group would be to focus on points that are touching each other. However, errors could have divided sets of pixels situated on a same edge into several distinct subsets. To re-unite subsets that were separated wrongly, a function called **closing** is used. This function consists of two sub-functions.

1. **Dilation** : sets a pixel at (i,j) to the maximum value among all pixels in the neighborhood. Neighborhood being defined as a square of a certain size centered on (i,j). Dilation enlarges bright regions and shrinks dark regions.[3]
2. **Erosion** : sets a pixel at (i,j) to the minimum over all pixels in the neighborhood centered

[3] According to the definition given on the Skimage website [19]

at (i,j). Erosion shrinks bright regions but keeps the connections that emerged through dilation.

This process of closing connects subsets that are close enough, thereby allowing to build groups. The "*close enough*" criteria can be adjusted by changing the size of the neighborhood. The neighborhood size is one of the parameters of the method and is referred as s in the parameter table in Appendix B. By default, s was set to 4. An example of the closing process in action is shown in Figure 5.18. The function that was used in this project is the `morphology.closing` function of the Skimage [19] open source library.



(a) Initial (b) After Dilatation (c) After Erosion (d) Groups generated

Figure 5.18: Closing Process

With the closing function, groups are established with pixels that are touching each other. They form the re-united subset of relevant points. This is shown through the different colors in Figure 5.18(d).

5.6.2 Discarding Too Small Groups

Now that each group is identified, groups that are too small can easily removed.

The minimum size of groups is one of the parameters of the method and is referred to τ_{blob} in the table of parameters in Appendix B. In our implementation, τ_{blob} value was set to 30 pixels.

Hint 9 : Why should relevant points appear together?

Inline with the definition, a relevant point p must be located on an edge and its mean-angle-difference must less than a threshold.

When isolating a particular frame of the MRI sequence, because p is located on an edge, points near p will tend to have the same gradient direction as p, as shown in Figure 5.19. Therefore :

$$\phi_f^i \simeq \phi_f^p \qquad \forall\, i \text{ near } p$$

Other points near p will tend to move in the same direction as p. This is due to the fact that when inhaling, organs are deformed in an elastic way, and not randomly. This is what is illustrated in Figure 5.20. Therefore:

$$\phi_g^i \simeq \phi_g^p \qquad \forall\, i \text{ near } p$$

The definition of the angle difference implies that :

$$\delta\left(\phi_f^i, \phi^i\right) \simeq \delta\left(\phi_f^p, \phi^p\right) \qquad \forall\, i \text{ near } p \tag{5.10}$$

Therefore, if p is relevant, points near p should be relevant too.

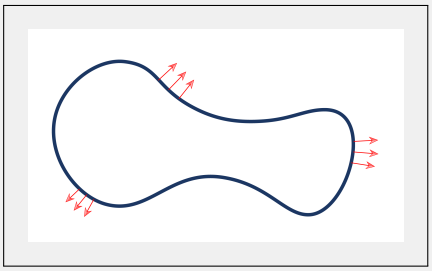


Figure 5.19

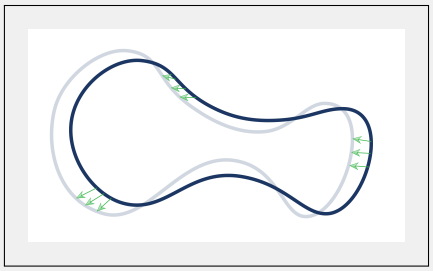


Figure 5.20

5.7 Step 5 - Navigators Positioning

With the completion of step 4, the final set of relevant regions is identified and the final step of the method, Navigator-Positioning phase, can start. A navigator is placed for each region. One navigator is sufficient because each region represents an edge. Each navigator is defined by the coordinates of its two end points. The output of step 5, which is also the output of the entire method, will consist in a list of couples of 2D coordinates.

Since a navigator is a line segment, the task of finding its two endpoints was subdivided in :

- Determining a line
- Determining the two extremes between which the interface movement occurs on that line

5.7.1 Line Estimation

The equation of a line can be parametrised by two variables : the coordinates of a reference point though which the line passes, and the second is the inclination of the line towards the reference point.

Reference Point

For each line, the reference point is defined as the mean position of the center of mass for the corresponding group, also called centroid. This choice is justified by the fact that when the edge associated to the group moves, it will increase the chance that points belonging to the group continue intersecting the navigator.

It is worth noting that if a border point of the group had been chosen as reference point, it could have happened that, due to the movement, the edge completely shifts outside the range of the navigator.

In the implementation part of the method, finding the center of mass was done using the `regionprops.centroid` function of the skimage library [19].

Line Inclination

Intuitively, the line should be perpendicular to the edge. A way of computing that direction is to fit a multivariate normal distribution for each group, and to select, as inclination, the direction of the smallest covariance component. Fitting a multivariate distribution corresponds to considering each point of a group as being a sample that is drawn from a 2D Gaussian distribution, and

estimating the Gaussian distribution that represents the best the group.

The contour line of a Gaussian distribution is an ellipse that is parametrised by an orientation and an eccentricity. Only the orientation is needed to estimate the inclination of the line. The `regionprops` class of Skimage provides a function that, on basis of a group of points, computes the orientation of the ellipse having the same second moment as the group. This is the function that was used in the Python module.

Figure 5.21 shows an example of an ellipse that is fitting a group of points.

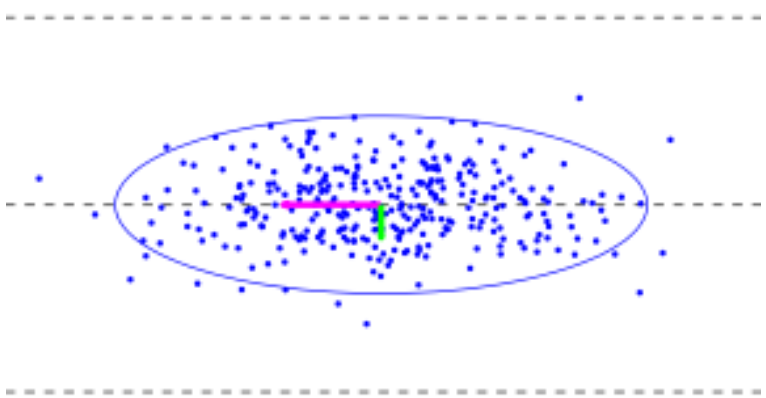


Figure 5.21: 2D Gaussian Distribution Fitting [20]

5.7.2 Endpoints Estimation

Navigators should be long enough so that during the respiration, the crossing edge always stays within the navigator. A simple way to determine the length of the navigators is to compute the distance that separates the two farthest apart positions taken by the centroids along the MR sequence.

The exact distance can be computed in $\mathcal{O}(D^2)$, where D is the length of the MR sequence. To increase the efficiency of the overall method, it was decided to compute an approximation of that distance instead. Positions taken by the centroid are first projected on the line defined here above, and the maximum distance is estimated by the distance that separates the two extreme projected points. This estimation is done in $\mathcal{O}(D)$.

Finally, a security margin of 10 pixels was added to those distances.

This closes the explanation of the method. As already mentioned, a Python implementation has also been developed and is available as open source code on GitHub at the following repository `https://github.com/RomainPattyn/MasterThesis`. In the next chapter, the results of our validation protocol are detailed.

Chapter 6

Validation Protocol

The final chapter of this master thesis presents a validation protocol for the Auto-Navigator method described in chapter 5. The goal is to evaluate how much the method meets its requirements. This chapter is divided into 3 sections:

1. Validation Criteria
2. Validation Protocol
3. Results

6.1 Validation Criteria

As the Auto-Navigator method intends to replace, or at least facilitate, the action of the doctor, the participation of a doctor has been organised in the validation process. Real medical images were submitted to a doctor and also processed through the Auto-Navigator method. The results were compared.

It is important to note that a lot of different navigators can be considered as good. The navigator positioning task is not a problem that has a unique solution. Knowing this, two different measurements were explored to evaluate the quality of the navigators. The first measurement puts a number on the percentage of navigators identified by the method that were considered as good. The second measurement counts how many of the method placed navigators were identical, or at least partly identical to the ones placed by the doctor, identical being a navigator placed on the same edge.

6.2 Validation Protocol

The proposed validation protocol is given as follows :

- The images are first processed through the Auto-Navigator method to obtain a set of navigators.

  This leads to a set of navigators that will be defined as $\mathcal{S}_A$ (A for Auto-Navigator)

- The images are then submitted to a doctor that is asked to position navigators manually.

  This leads to another set of navigators that will be defined as $\mathcal{S}_D$ (D for Doctor)

- The method-placed navigators, $\mathcal{S}_A$, are shown to the doctor, that is asked to identify the wrong ones.

  This leads to another set of navigators that will be defined as $\mathcal{S}_{GA}$ (GA for Good Auto-Navigator)

- Based on $\mathcal{S}_A$ and $\mathcal{S}_{GA}$, the first validation measure, τ_1, is computed. It represents the ratio of doctor-placed-navigators that were found by the method, i.e. :

$$\tau_1 = \frac{|\mathcal{S}_{GA}|}{|\mathcal{S}_A|}$$

- Based on $\mathcal{S}_A$ and $\mathcal{S}_D$), the second validation measure, τ_2, is computed. It represents the ratio of doctor-placed-navigators that identified by the method, i.e. :

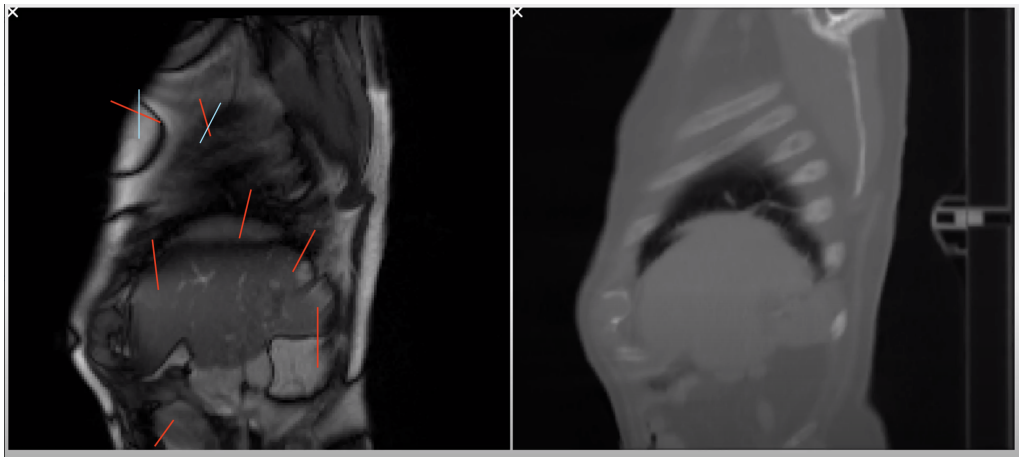
$$\tau = \frac{|\mathcal{S}_D \cap \mathcal{S}_A|}{|\mathcal{S}_D|}$$

6.3 Results

16 sets of CT and MR images, corresponding to 3 patients were used as cases on which the validation process was run. 12 of those 16 sets were sagittal planes, and the 4 other sets were coronal[1] planes.

[1] Sagittal and coronal planes are anatomical planes that are used to image a body from different angles. A sagittal plane corresponds to the plane that divides the patient body in a left and a right parts. A coronal plane corresponds to the plane that divides the patient body in a front and a back parts.

Here below is an example of what was received in return from the doctor as part of the protocol. The red navigators are the ones that the Auto-Navigator method placed, the light blue crosses represent the ones the doctor has rejected.



By averaging all the τ_1's obtained for each set of images, the method reached an average τ_1 of 0.78, meaning that, on average, 78% of the navigators that were placed by the method were correct.

By averaging all the τ_2's obtained for each set of images, the method reached an average τ_2 of 0.38, meaning that, among all the navigators that were placed, 38% were identical to ones placed by the doctor.

All the results are provided in Figure 6.1 and are illustrated in Figure 6.2.

processed sequence	short label	# nav. placed by algo.	# nav. placed by doc.	# identical nav.	# nav. kept by doc	# nav. rejected by doc
p16-3-1	i	1	5	1	1	0
p16-3-2	j	1	5	1	1	0
p16-3-3	k	1	5	1	1	0
p16-1-1	f	2	5	0	2	0
p19-22-2	m	2	7	2	2	0
p19-3-1	n	2	6	0	1	1
p16-2-2	h	3	6	0	2	1
p19-3-2	o	3	5	1	2	1
p19-3-3	p	3	5	1	2	1
p1-3-2	e	6	4	1	5	1
p16-2-1	g	6	5	3	3	3
p1-3-1	d	7	4	3	5	2
p19-22-1	l	7	6	4	6	1
p0-2-2	b	8	4	3	5	3
p0-2-1	a	10	5	5	10	0
p0-3-3	c	10	4	3	5	5

Figure 6.1: Table of Values

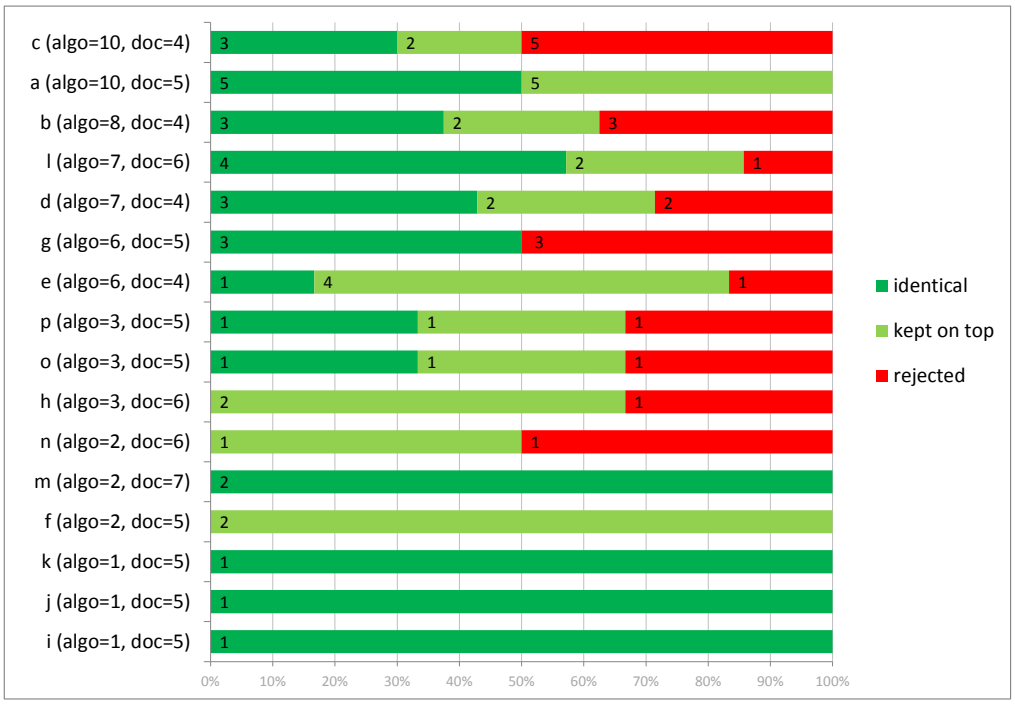


Figure 6.2: Results obtained on the 16 instances.

The information of the table in Figure 6.1 was used to build the bar-chart in Figure 6.2.

How to read the chart

* Each line corresponds to a single set of MR/CT sequences (a line in the previous table).
* Each bar represents in %, 3 types of information : the number of navigators positioned by the method that were also positioned by the doctor (identical - dark green), the number of navigators positioned by the method that were rejected by the doctor (rejected - red), and the number of navigators proposed by the method that were not found by the doctor but were still considered as valid by the doctor (kept on top - light green).
* The figures appearing in the bars correspond to the actual numbers of navigators concerned. The information in % is to be found on the horizontal axis.

Comments on the results

- For every instance, the method has found navigators that were validated by the doctors. This is a first good result.
- For 11 instances out of 16, the doctor has validated navigators which she had not identified at first.
- For 10 instances out of 16, some of the navigators were rejected by the doctor. The percentage of rejected navigators varies from 15 to 50%.

6.4 Improvement of the Validation Protocol

One should not jump to quickly to conclusion looking at the results of this protocol. It was noticed, after implementation, that important information was missing. An aspect which was not considered in the protocol relates to the minimum number of navigators required for each case. For instance, the entry `m(algo=2, doc=7))` in Figure 6.2, shows that for case `m`, all the navigators placed by the method were also placed by the doctor. However, one can see that the doctor placed seven navigators while only two were placed by the method. Having this in mind, can it be concluded that the method worked well on that instance?

Also, a better validation protocol would have been to incorporate navigators provided by the Auto-Navigator method directly into Dasnoy's algorithm in order to evaluate their performance on a dry-run simulating the MR/CT movement replication. An automated parameter tuning procedure optimising the quality of the movement replication could be an interesting extension to this master thesis.

Part IV

Appendix

Appendix A

User Guide for the Python Module

The implementation of the method presented in chapter 5 is available on GitHub at the following repository : `https://github.com/RomainPattyn/MasterThesis`. The project was coded using python3 and all the tools and functions that were used are open source. The required libraries are : cv2, matplotlib, numpy, scipy, seabor and skimage. If, as explained in section 5.1.3.2, one wants to use Brox's algorithm for the dense displacement field estimation, an extra library will be needed : PyFlow. That library can be downloaded at : `https://github.com/pathak22/pyflow`.

Python files are extensively documented and should be easy to understand. The code follows indeed the structure explained in chapter 5. Each step of the method is handled by a different function call in the code. This appendix is intended to facilitate the comprehension of the organisation of the project and will not enter into code details. For more explanations about lines of code, please refer to the comments in the python files, or contact me at romain.pattyn@student.uclouvain.be.

The module is divided into the three following files :

- **ImageProcessing** : File containing a class implementing the code described in chapter 5.
- **VideoViewer** : File containing a class that facilitates the visualisation of results provided by the ImageProcessing module.
- **FlowMemory** : File that facilitates and improves the efficiency of the use of the displacement fields.

I recommend to use the module through a Jupyter notebook because it eases the visualisation of videos.

A.1 Project Description

The variables, listed in Appendix B, that parametrise the entire method can be modified in the file `Libraries/example/data/vars/variables`. The only variable that is modifiable directly in the function calls is the number of navigators wanted.

The module provides two functions : `GET_REGIONS` and `GET_NAVIGATORS`. Those two functions take as input both the sequence of MR images and the sequence of CT images. Before any computations, the CT images are resized to the dimensions of the MR images. The output of those two functions is therefore representing coordinates in the coordinate system of the MR images.

- `GET_REGIONS` : That function executes the first 4 steps of the method in chapter 5 and corresponds therefore to the Region-Selection phase. It identifies regions that are similar in the both modalities, and inside which there is movement. The output of that function consists in a tuple where the first element is a binary mask identifying the relevant points, and the second is a list containing the coordinates of the relevant points.
- `GET_NAVIGATOR` : That function executes all the 5 steps of the method. It outputs a list containing couples of coordinates that represents the endpoints of the navigators. As a reminder, the number of navigator that the function will output can be modified in the file containing the variables, through the variable : `number_groups`

At each step of the method, those two functions send a binary mask to the `VideoVieuwer` so that the successive results can be visualized afterwards. Some information shown can be changed, for instance : showing the displacement field or not, changing the wanted steps to be shown and showing the navigators or not. A video can be saved in the mp4 format.

A.2 How to Use the Python Module

An example of use of the module is provided inside the project through a Jupyter notebook in the file : `Libraries/example/main.ipynb`.

The module enables the display of intermediate graphs and illustrations. By default the module does not show anything. To activate that option, the optional argument `show_figures` must be set to true inside the function calls of `GET_REGIONS` and `GET_NAVIGATORS`. The module also enables to save those images inside the folder `Libraries/example/images`. To do so the optional

argument `save_images` must be set to true.

A.2.1 Import the Module

The module can be loaded by adding the path leading to the library, to the `PYTHON_PATH`.

```
import os
import cv2
import sys
import time
import importlib
import pickle as pkl
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import Video, HTML

sys.path.append(os.path.abspath("./../"))

from ImageProcessing import GET_NAVIGATORS
from ImageProcessing import GET_REGIONS
import ImageProcessing as IP

def reload_library():
    importlib.reload(IP)
    print("Library reloaded")
```

A.2.2 Load the Data

Two pickle files containing the MR sequence and the CT sequence related to a patient are provided in the project in the `Libraries/data` folder. The two sets of images can be loaded as follows :

```
def getDataSequences(data_path):
    with open(data_path, 'rb') as handle:
        images = pkl.load(handle)
    return images

MRI_images = getDataSequences("./data/MRI_images.p")
CT_images = getDataSequences("./data/CT_images.p")
```

A.2.3 Region Identification

The following example shows how to get a binary mask corresponding to the relevant regions. It also shows how one can print the relevant points directly on the first frame of the MR sequence.

```
binary_mask, point_coord = GET_REGIONS(MRI_images, CT_images)

h, w = point_coord.T
ref_img = cv2.cvtColor(MRI_images[0].copy(), cv2.COLOR_GRAY2RGB)
img = ref_img.copy()
img[h, w] = np.array([173, 255, 47])
img = cv2.addWeighted(img, 0.6, ref_img, 1 - 0.6, 0)

plt.figure(figsize=(8, 5)), plt.imshow(img), plt.axis("off"), plt.show()
```

A.2.4 Navigator Identification

The format of the output of the `GET_NAVIGATOR` function is illustrate in the example below. The list of couples returned is such that the first element of the coordinates corresponds to the height of the image and the second element corresponds to the width.

```
for ((x1, y1), (x2, y2)) in GET_NAVIGATORS(MRI_images, CT_images):
    print("From ({0}, {1})\tto ({2}, {3})".format(x1, y1, x2, y2))
```

A.2.5 VideoViewer

The following example shows how to call the `GET_NAVIGATOR` function by implicitly instantiating the `ImageProcessing` class. As said earlier, this is necessary in order to modify and save a video showing the successive results computed by the method.

```
reload_library()
DataSet = IP.ImageProcessing(MRI_images[10:], CT_images)
DataSet.get_navigators()

DataSet.vv_deactivate_layer('flow')
DataSet.vv_change_layer_opacity('following_points', 0.5)

video_path = "./video/example.mp4"
DataSet.export_video(video_path, slower=3)

Video(video_path)
```

Appendix B

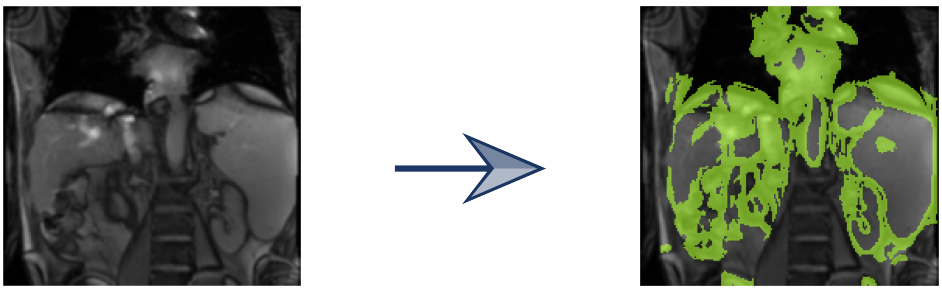
Table Of Parameters

Method of Chapter 5			
Parameters	**Value**	**Step**	**Description**
k	3	1	Size of the Sobel filter used.
τ_{dist}	0.5	1	Minimum average distance that a pixel has to move from a frame to the next one.
τ_{Sobel}	3	1	Value between 0 and 10 that thresholds the average MRI gradient magnitude. 0 corresponds to taking all the points, 10 corresponds to taking none.
ϕ_{max}	35	2	Maximum average angle difference between the gradient direction and the movement direction.
n	15	3	Number of highest gradients considered in the computation of the mean CT gradient magnitudes
s_{CT}	3	3	Shift used during the computation of the n highest gradients.
τ_{CT}	3	3	Value between 0 and 10 that thresholds the average CT gradient magnitude. 0 corresponds to taking all the points, 10 corresponds to taking none.
s	4	4	Size of the square used during the closing process
τ_{blob}	30	4	Minimum size of the interesting edges

Appendix C

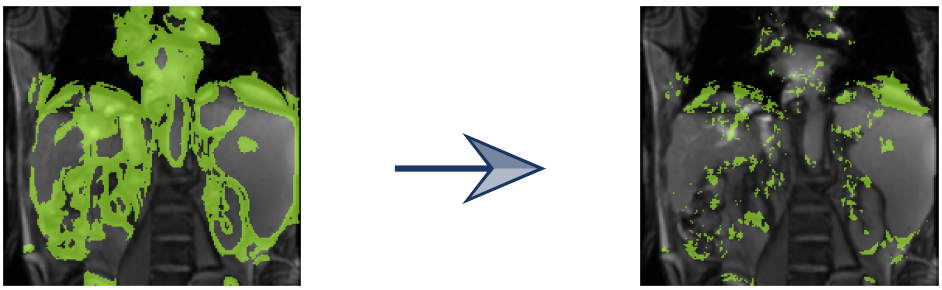
Results Obtained On Other Patients

C.1 Patient 1



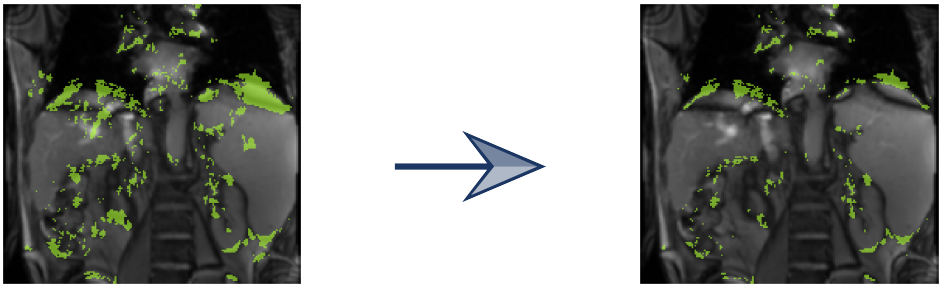
(a) Before step 1

(b) After step 1



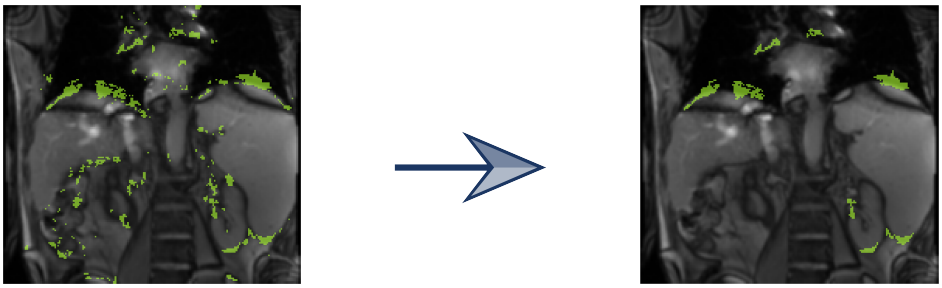
(a) Before step 2

(b) After step 2



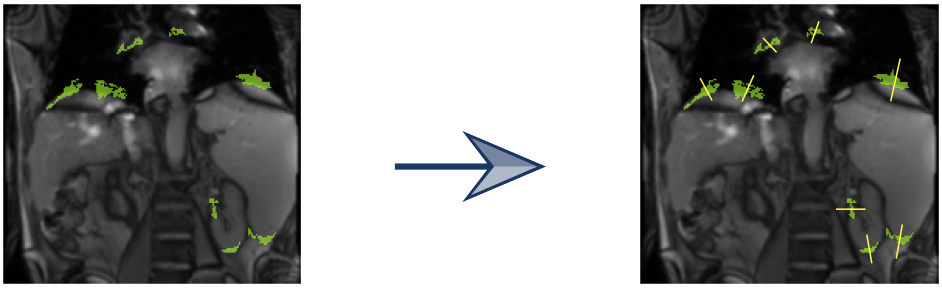
(a) Before step 3

(b) After step 3



(a) Before step 4

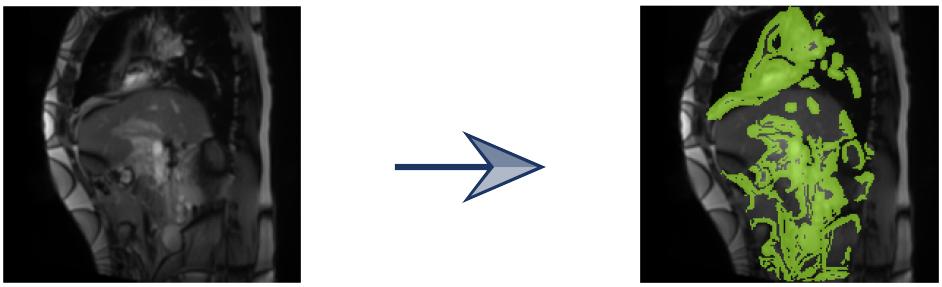
(b) After step 4



(a) Before step 5

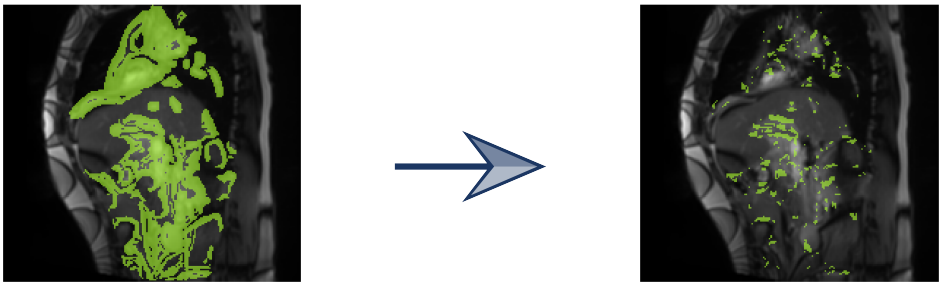
(b) After step 5

C.2 Patient 2



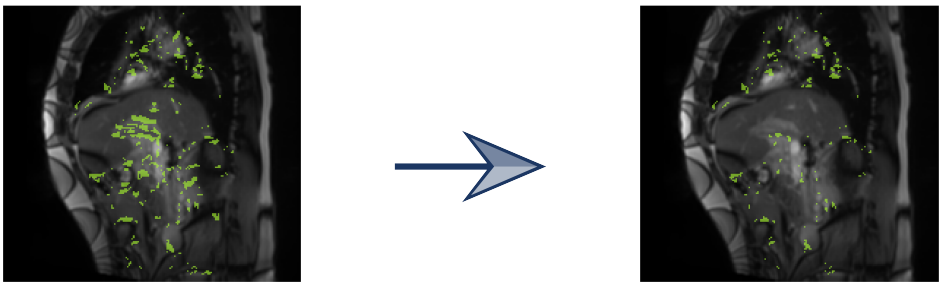
(a) Before step 1

(b) After step 1



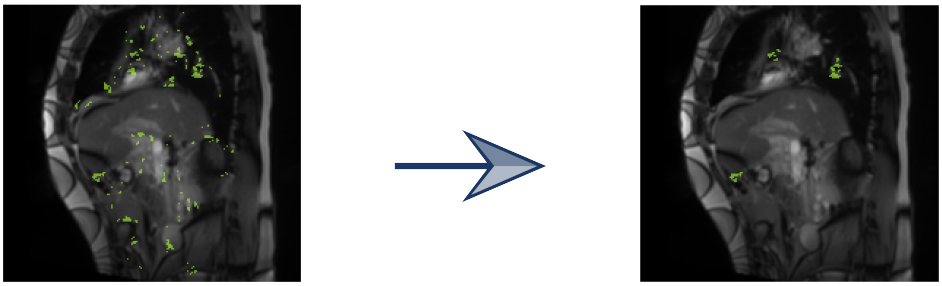
(a) Before step 2

(b) After step 2



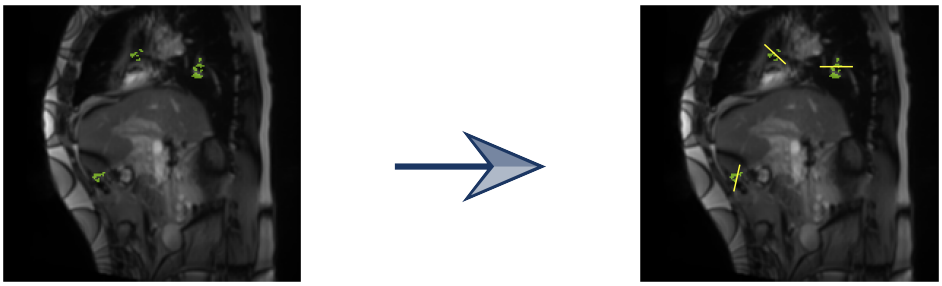
(a) Before step 3

(b) After step 3



(a) Before step 4

(b) After step 4



(a) Before step 5

(b) After step 5

Appendix D

Photon and Proton Beams

Photon beams and Proton beams follow two different energy deposition patterns.

This is mainly due to the fact that photons are particles with a mass equal to zero, while protons are particles with a mass greater than zero.

Figure D.1 shows the curves representing the energy dose deposition by photons (yellow curve) and protons (dotted-blue curve) in function of the depth they reach in the patient's body. Note that photons are referred to as X-rays on the figure.

A photon (yellow curve) deposits most of its energy on the first tissues and then progressively deposits less and less energy the more it penetrates the tissues.

A proton (dotted-blue curve) deposits most of its energy deeper in the patient's body and in a narrower window before stopping. The peak on the proton curve is known as the Bragg Peak.

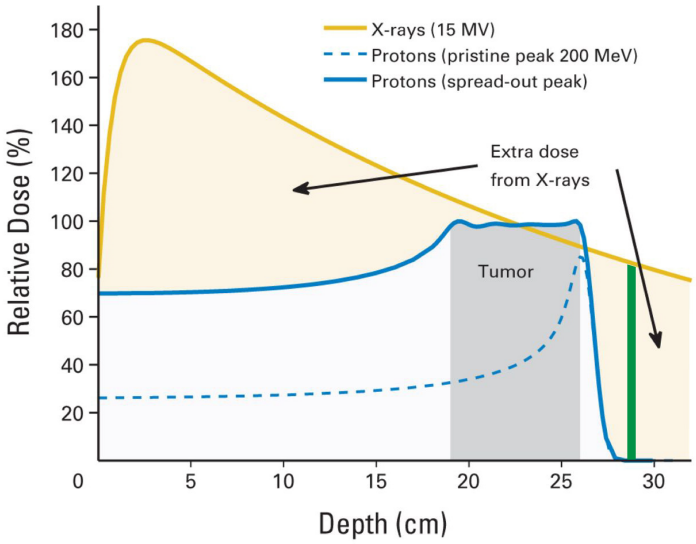


Figure D.1: Relative depth dose depositions of photons versus protons [21]

As shown on the picture, proton therapy is more localised and damages less healthy cells than photon therapy.

However, a drawback of proton therapy is that it is difficult to estimate with high accuracy the depth at which the Bragg Peak occurs. This increases the complexity of targeting correctly a tumor. In particle therapy, if the tumor location is not estimated correctly the radiations can completely miss the tumor area. This is the case for instance, if the tumor is located after the Bragg peak (as shown by the green area on Figure D.1).

Countermeasures have been developed to reduce the impact of the tumor location error in particle therapy, but they can not always be applied. To make sure that the entire tumorous area is covered, the beam projects particles in a way that makes their Bragg Peaks occur at different depths. This leads to the plain blue curve on Figure D.1.

Photon therapy is not as sensitive to errors in the tumor location as proton therapy. Figure D.1 highlights that independently of the location of the tumor, a certain amount of radiations will be delivered to the tumor.

Bibliography

[1] "Cancer." https://ourworldindata.org/cancer.

[2] S. Gianfaldoni, R. Gianfaldoni, U. Wollina, J. Lotti, G. Tchernev, and T. Lotti, "An overview on radiotherapy: From its history to its current applications in dermatology," 2017.

[3] "Radiation therapy to treat cancer." https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy.

[4] "The difference between chemotherapy and radiotherapy." https://treatcancer.com/blog/difference-chemotherapy-radiation/.

[5] https://wikimedi.ca/wiki/ULaval:MED-1202/Cancer#/media/Fichier:Diagram_showing_how_you_have_internal_radiotherapy_for_lung_cancer_CRUK_160.svg.

[6] https://www.shutterstock.com/g/Tomacco?searchterm=radiotherapie.

[7] "Proton-beam therapy versus photon-beam therapy: The debate continues." https://www.lungcancernews.org/2017/09/29/proton-beam-therapy-versus-photon-beam-therapy-the-debate-continues/.

[8] "What to expect from the proton therapy market in 2019-2020." https://www.itnonline.com/content/what-expect-proton-therapy-market-2019-2020.

[9] "Md anderson - mr-linac." https://www.mdanderson.org/publications/cancer-frontline/merging-an-mri-with-a-linear-accelerator-allows-greater-precisio.h00-159222567.html.

[10] S. Goossens, F. Senny, J. Lee, G. Janssens, and X. Geets, "Assessment of tumor motion reproducibility with audio-visual coaching through successive 4d ct sessions," 2014.

[11] L. Lin, K. Souris, M. Kang, A. Glick, H. Lin, S. Huang, K. Stu tzer, G. Janssens, E. Sterpin, and J. Lee, "Evaluation of motion mitigation using abdominal compression in the clinical implementation of pencil beam scanning proton therapy of liver tumors," 2017.

[12] G. Van Ooteghem, D. Dasnoy-Sumell, M. Lambrecht, G. Reychler, G. Liistro, E. Sterpin, and X. Geets, "Mechanically-assisted non-invasive ventilation: A step forward to modulate and to improve the reproducibility of breathing-related motion in radiation therapy," 2019.

[13] D. Dasnoy, K. Souris, and G. Van Ooteghem, "Continuous real time 3d motion reproduction using dynamic mri and precomputed 4dct deformation fields," 2017.

[14] http://of-eval.sourceforge.net/.

[15] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," *In: Image analysis*, vol. 2749, pp. 363–370, 06 2003.

[16] "Opencv website." https://opencv.org.

[17] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping.," In: Pajdla T., Matas J. (eds) Computer Vision - ECCV 2004. Lecture Notes in Computer Science, vol 3024. Springer, Berlin, Heidelberg.

[18] S. Hermann and R. Werner, "High accuracy optical flow for 3d medical image registration using the census cost function," In: Klette R., Rivera M., Satoh S. (eds) Image and Video Technology. PSIVT 2013. Lecture Notes in Computer Science, vol 8333. Springer, Berlin, Heidelberg, 2014.

[19] "scikit-image website." https://scikit-image.org.

[20] https://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/.

[21] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4152713/figure/F1/.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**