SUB-SECOND SPEED 4D-CT IMAGE REGISTRATION USING UNSUPERVISED DEEP LEARNING

MSC THESIS REPORT

by

Thomas VAN DER MEULEN

in partial fulfillment of the requirements for the degree of

Master of Science
in Applied Physics,

at the Delft University of Technology,

to be defended on Monday January 16th, 2023 at 10:15.

Student number:	4593383
Supervisors:	dr. Zoltán Perkó
	Oscar Pastor Serrano
Thesis committee:	dr. Z. Perkó,
	dr.ir. M. C. Goorden,
	dr.ir. D. Lathouwers

Preface

This report is the resulting work of my thesis, which is the conclusion to my masters degree in Applied Physics at the Delft University of Technology.

First of all, I would like to thank Zoltán Perkó and Oscar Pastor Serrano for being my supervisors. I have phased some difficult times during the project, but all their guidance and support throughout this project has helped me greatly in finishing this project. Oscar, I wish you all the best in your new job and for your upcoming PhD defence.

A thanks to all members from both the Medical Physics & Technology group and Reactor Physics and Nuclear Materials group for all the nice lunch break conversations. A special thanks to my fellow master student in our cubical: Chris, Emma, Sylvie, Daphne and Ruben. I have enjoyed the campus walks, coffee breaks and eating Bueno together.

I would like to thank the members of the graduation committee for reading the thesis report and grading the project.

Lastly, This report is in remembrance of my grandfather, Opa Muis, who passed away during the time working on this thesis from the effects of cancer. I hope that my work contributes to the wider effort to find effective treatments for cancer patients.

Thomas van der Meulen
Den Haag, January 2023

Abstract

To evaluate the effect of interplay due to breathing of the patient during proton treatment of lung tumors Interplay dose calculation techniques have been proposed in literature. The proposed method requires the deformation vector field (DVF) to register dose distributions of different phases in the breathing cycle to a reference phase. The DVF is obtained by registering 4DCT lung scans between the phases. Current methods of image registration are too slow to make the interplay dose calculation techniques clinically feasible.

Advances in deep learning have allowed for models that predict the DVF in orders of magnitude quicker than traditional methods. In this research, two model architectures, previously applied for registration of brain MRI images, will be evaluated to predict the DVF between scans at different phases of a 4DCT lung scan. The quality of the registration is evaluated based on the mean absolute error between the images and contour metrics of organs including the Dice score, Hausdorff distance and the mean surface distance. In addition, the amount of grid folding was evaluated based on the number of voxels with a negative Jacobean determinant.

The first model architecture, VoxelMorph, is an unsupervised model with an U-net architecture. Two hyperparameters were varied: the maximum size of the DVF limited by a HardTanh, and secondly the weight of the loss function for the divergence of the DVF during training. The model performed poorly in predicting the DVF, the values of the DVF were too small. Varying the hyperparameter seems to have no significant impact on the prediction quality of the model. Limiting the maximum of the DVF prevents the registration of large deformations, which is not favourable.

The second model architecture has a multi-resolution approach. The images are downsampled to 1/2 and 1/4 the resolution. Multiple sub-network predict a DVF at each of the resolutions in a coarse to fine order. Each of the networks consisted of a feature encoder, residual blocks and a feature decoder. By upsampling and combining the multiple DVFs, the final DVF is obtained. Hyperparameter search is performed: The number of residuals blocks and their filters were varied. At first only for the coarses network, and later for all the networks. Lastly, an additional resolution was added to the model. The model was capable of predicting good-quality DVFs. Only varying the number of residual blocks and their filters for all resolutions resulted in a significant difference in the quality of the prediction.

Predictions are performed in 260 ± 4 ms and 24 ± 4 ms for the first and second architectures respectively. Which is faster than other deep learning methods found in literature, and significantly faster compared to traditional registration methods.

CONTENTS

List of Abbreviations

CC	Cross correlation.
CIN	Conditional Instance Normalization.
CIRM	Conditional image registration module.
CNN	Convolutional Neural Network.
CRN	CNN-based registration network.
CTV	Clinical Tumor Volume.
DIBH	Deep Inspiration Breath-Hold.
DSC	Dice similarity coefficient.
DVF	Deformation Vector Field.
GTV	Gross Tumor Volume.
HD	Hausdorff Distance.
HDF5	Hierarchical Data Format version 5.
HU	Hounsfield Units.
IMPT	Intensity Modulated Proton Therapy.
IQR	Inter Quartile Range.
LAP	Laplacian pyramid network.
LeakyReLU	Leaky Rectified Linear Unit.
MLP	Multi Layer Perceptrons.
MSD	Mean Surface Distance.
MSE	Mean Square Error.
PTV	Planning Tumor Volume.
TRE	Target Registration Error.
VM	VoxelMorph.

1

Introduction

In 2020, 20 million new cancer cases were diagnosed worldwide. Making it the leading cause of death worldwide. More than 50% of the patients receive radiation treatments. Currently, most patients are irradiated using photons, but the use of proton treatments is increasing due to the advantages such as finite range, highly local dose deposition and highly conformable dose distribution. Despite best efforts, in 2020, cancer accounts for nearly 10 million deaths. [1, 2].

The highly local dose deposition allows for better sparing of healthy tissue surrounding the tumor. For patients with lung tumors, breathing results in changing anatomy within the lungs. This can result in high-dose depositions at the wrong position. As a result, the actual dose uptake by the patient can differ from to the planned dose [3, 4].

Before treatment lung tumors patients, multiple computed tomography scans are made at different phases of the breathing cycle, called a four-dimensional computed tomography (4DCT) scan. By tracking the breathing of the patient during the treatment, the breathing phase can be determent at the time a proton spot is delivered. Combing the breathing phase with the delivered proton spot, dose calculation is performed using the correct scan from the 4DCT. To obtain the total delivered dose during the treatment, the dose of all the spots has to be combined. Firstly, the dose of spots within the same phase can be accumulated. Next, these accumulated doses have to deformed to a reference phase. The deformation is obtained using image registration. Image registration predicts an deformation vector field between images of the phase and the reference phase. This vector field is used to transform the accumulated dose from each phase to the reference phase. Next, all the transformed dose distributions are accumulated to get the final dose distribution.

Current methods of image registration use iterative methods to perform registrations and predict the deformation vector field. These current methods of image registration can take minutes. To make the dose accumulation methods faster, image registration has to be performed quicker [2].

Recent advances in deep learning have allowed for convolutional neural network architectures which are capable of predicting deformation vector fields within tens of milliseconds. These architectures have mainly been applied for the registration of magnetic resonance images of the brain [5, 6].

In this thesis, two convolutional neural network architectures will be adapted and evaluated to predict deformation vector fields between lung images of four-dimensional computed tomography images. The performance will be evaluated based on the image intensity as well as metrics for organ contours and landmarks and the registration speed.

In Chapter 2 background information regarding proton therapy, image registration and neural networks will be discussed. Chapter 3 gives an overview of the literature concerning image registration techniques. In Chapter 4 the used convolutional neural networks architectures will be discussed. Additionally, the used datasets, evaluation metrics and training routines are discussed. In Chapter 5 the results from the evaluated using the evaluation metrics will be given. Chapter 6 discusses these results and gives suggestions for further research. Chapter 7 will give a conclusive overview.

2
Background information

This chapter gives background information on the workings of proton therapy and the treatment workflow, image registration and breathing interplay. In addition, basic concepts and workings of neural networks are also discussed.

2.1. Proton therapy

In contrast to conventional radiation therapy, which uses photons to irradiate tumor tissue, proton therapy uses accelerated protons to deliver dose to the tumor tissue at a set range. The range is the distance the proton travels through the tissue. The dose deposited by protons increases sharply near the end of the particle range, known as the Bragg peak, and a rapid fall-off of the dose at the end of the range, as is illustrated in Figure 2.1. This local dose deposition makes protons suitable for cancer treatment since a high dose is delivered at the tumor tissue and it spares surrounding healthy tissue.

In contrast to protons, the dose fall-off of photons is only a few percent per centimetre, resulting in high doses for surrounding tissue [3]. This is also illustrated in Figure 2.1 as a red line.

Treatment of cancer using proton therapy is performed through a Pencil Beam Scanning (PBS) machine. The dose is delivered by a series of proton beamlets from multiple external positions at a certain energy level, referred to as spots. By altering the energy of the protons, the penetration depth can be adjusted. Using magnets inside of the machine, off-axis coverage is possible. The dose at each spot is adjusted using irradiation time [3].

To achieve a highly conformable dose distribution Intensity Modulated Proton Therapy (IMPT) is used. The total dose is delivered by thousands of beamlets from various directions. Each of the beamlets is adjusted individually, to jointly achieve a highly conformable final dose and better sparing of organs at risk [7].

2.2. Proton treatment workflow

The workflow for treating patients with proton therapy consists of multiple steps. First, anatomical information needs to be obtained to make a treatment plan. To obtain anatomical information about the tissue at various stages in the breathing cycle, multiple 3-dimensional computational tomography (3DCT) scans are made. This type of scan is referred to as a 4-dimensional CT (4DCT) scan.

A breathing cycle consists of an inhalation phase where air flows into the lungs, a brief pause, and an exhalation phase where air flows out of the lungs [9]. The movement during inhalation and exhalation are not identical, which is called breathing hysteresis. Due to the breathing hysteresis, scans are made at multiple discrete phases within the breathing cycle. For example, a breathing cycle of 8 phases consists of 0%, 25%, 50%, 75% and 100% inhale and 75%, 50%, 25% exhale phases [10].

Based on these scans, contours around the tumour volume are made. The visible tumour volume is defined as the Gross Tumor Volume (GTV). A margin of a few millimetres is added for microscopic extensions of the disease that are not visible on the scan, the enlarged volume is named the Clinical Tumor Volume (CTV). For photons, the volume is further enlarged to account for uncertainties such as setup errors, patient motion, and linear accelerator alignment errors. This results in the final volume used for planning: the Planning Target Volume (PTV) [11]. Proton therapy does not use a PTV, since the dose distribution can change substantially when the patients anatomy within the beam path is changed [12].

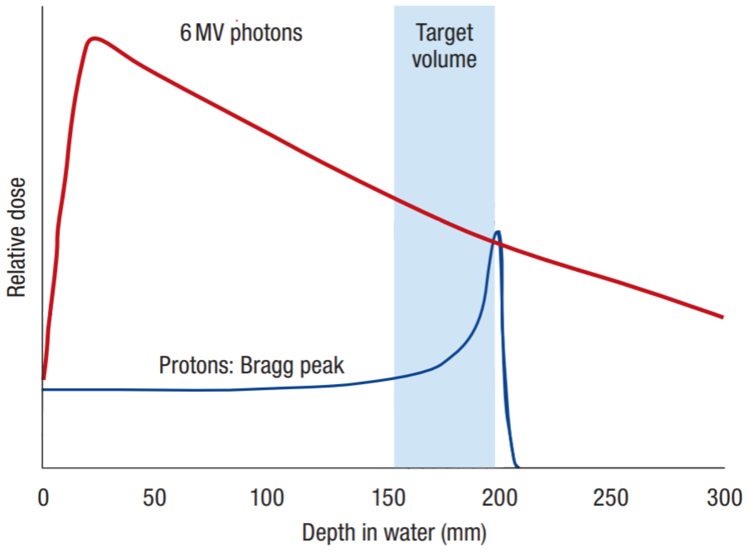

Figure 2.1: Relative dose as a function of depth in water for various radiation methods. The target volume is a shaded grey. The red line indicates treatment with photons. The dark-blue line illustrates the relative dose of a single proton, with a Bragg peak at the end of the path. Image taken from Barker et al. [8].

2.3. DOSE CALCULATION

Dose calculation methods are used to calculate the resulting dose distribution from the delivered spots. Current methods for dose calculations are either analytical methods or Monte Carlo methods.

Analytical methods use water as a reference medium, therefore a water-equivalent thickness of the tissue is used along the axial direction of a pencil beam [13]. As the beam traverses through the medium, the dose distribution diffuses in the lateral direction. There are two leading causes of diffuses: The first comes from the spread optical properties of the system. The second spread comes as a result of multi-coulomb scattering in the medium, which is depth-dependent. These diffuses are gaussian shaped. The total diffusion is the product of the two gaussian distributions [13]. Analytical methods assume a homogeneous medium in the lateral direction. This assumption leads to range degradation in complex and heterogeneous geometries such as air-tissue interfaces as present in the lung [13]. Analytical methods yield results at low computational costs.

Monte Carlo methods can yield highly accurate dose calculations. Many different particles are tracked through the geometry, simulating their trajectories, where the interaction of the particles is determined based on the sampling of a probability distribution. Examples of interactions are nuclear interactions, annihilation, scattering and creation of secondary particles [13]. Monte Carlo simulations also allows the modelling of uncertainties such as setup and range errors. A large number of particles need to be simulated to reach an acceptable statically precision, therefore the simulations have a high computational cost [14].

Novel approaches exists which use deep learning to predict the dose based on CT data and the beam energy. The results are comparable to results from Monte Carlo simulations. The computational time per pencil beam is 5.0 ± 4.9 ms down from 44 ± 12 seconds for conventional Monte Carlo simulations [2].

2.4. IMAGE REGISTRATION

Image registration is the process of finding the optimal one-to-one mapping of voxels from a moving image I_m to a fixed image I_f such that the transformed moving image represents the fixed image. The moving and fixed images consist of Ω_m and Ω_f voxels respectively, with size Ω^ζ in each direction with $\zeta \in \{x, y, z\}$ and each voxel is indicated by $\boldsymbol{p}$. Each of the voxels has a single intensity value. The transformations between I_m and I_f is given by $\boldsymbol{T_\theta}(\boldsymbol{p})$, were $\boldsymbol{\theta}$ are the transformation parameters [15]. The displacement of each voxel between I_m and I_f is given by the deformation vector field (DVF) ϕ [15, 16]. The DVF has the same size as the images but has three values per voxel, the displacement in each direction ζ.

2.4.1. TRANSFORMATION

The transformation, $\boldsymbol{T_\theta}(\boldsymbol{p})$, can be a linear transformation, examples of a linear transformation are translation, rotation, affine and shearing of the image. Alternatively, non-rigid transformations which are not linear can be applied such as the B-spline transform. Examples of rigid and non-rigid transformations are given in Figure 2.2. The fixed and moving images are divided into a grid of lines with spacing $\boldsymbol{\tau}$ resulting in $P_\zeta = \Omega^\zeta / \tau_\zeta$ points in each direction ζ. The crossings of grid lines are the control points, x_c, that are moved to transform the image. There are a total of $P_x \cdot P_y \cdot P_z$ control points.

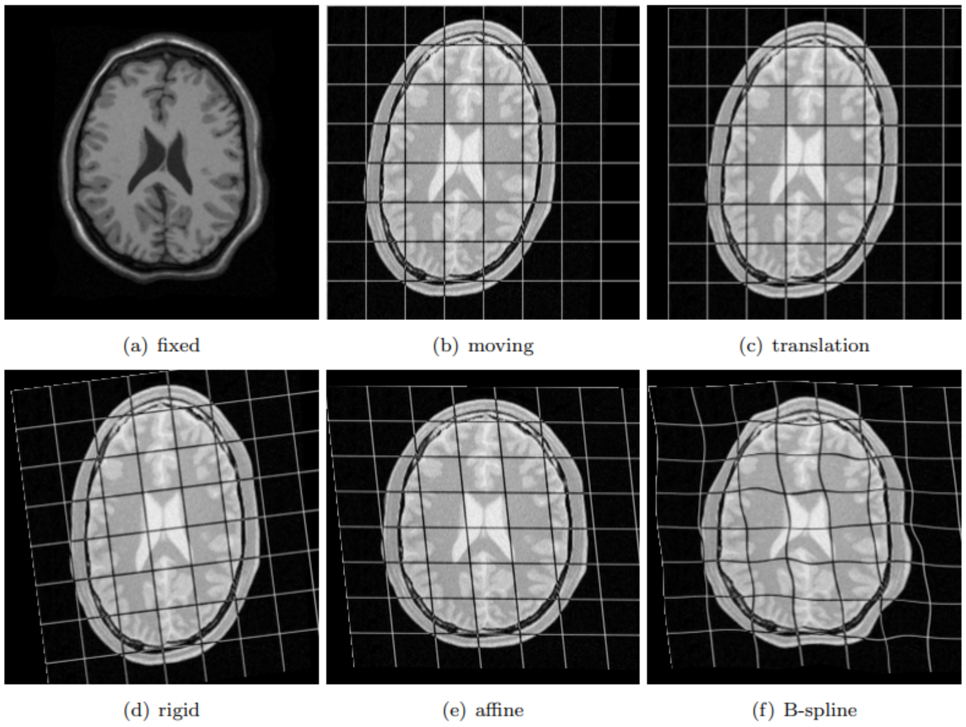

Figure 2.2: Overview of different transformations between fixed and moving image. (a) Fixed image I_f, (b) moving image I_m, (c) deformed image using translation, (d) deformed image using rigid transformation, (e) deformed image using affine transformation, (f) deformed image using B-spline transform. Images taken from S. Klein et al. [15]

A common non-rigid transformation is a B-spline parametrization. The transformation of voxel between the control points are interpolated using B-spline parametrization as given in Equation (2.1). It sums over control points which are within a set distance of the voxel, this distance is called the local support of the B-spline. Hence, the sum is over $\boldsymbol{x}_c \in \mathcal{N}_{\boldsymbol{x}}$, where $\mathcal{N}_{\boldsymbol{x}}$ are the points within the local support. By having a small local support this transformation can be calculated quickly.

$$\boldsymbol{T_\theta}(\boldsymbol{p}) = \boldsymbol{p} + \sum_{\boldsymbol{x}_c \in \mathcal{N}_{\boldsymbol{x}}} \boldsymbol{\theta}_{\boldsymbol{x}_c} \boldsymbol{\beta}^3 \left(\frac{\boldsymbol{p} - \boldsymbol{x}_c}{\boldsymbol{\tau}} \right) \tag{2.1}$$

$\beta^3(x)$ is the cubic multidimensional B-spline polynomial as given in Equation (2.2). The B-spline basis functions are piecewise continuous polynomials which allows for the cheap calculation of derivatives [17]. $\boldsymbol{\theta}_{\boldsymbol{x}_c}$ is the B-spline coefficient for each control point, with $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, ..., \boldsymbol{\theta}_{P_x \cdot P_y \cdot P_z})^T$, which are optimised to obtain the best registration.

$$\beta^3(x) = \begin{cases} \frac{2}{3} - |x|^2 + \frac{|x|^3}{2}, & 0 \leq |x| < 1 \\ \frac{(2-|x|)^3}{6}, & I \leq |x| < 2 \\ 0, & 2 \leq |x|. \end{cases} \tag{2.2}$$

2.4.2. ALIGNMENT QUALITY METRIC

The quality of alignment is defined by a cost function, the goal of the image registration is to minimize the cost function. Commonly used cost functions are the mean square error (MSE) and the normalised cross-correlation (CC). The MSE cost function is only suitable if the two images are with equal intensity. The MSE is defined in Equation (2.3).

$$\text{MSE}\left(\boldsymbol{\theta}; I_f, I_m\right) = \frac{1}{|\Omega_f|} \sum_{\boldsymbol{p} \in \Omega_f} \left(I_f\left(\boldsymbol{p}\right) - I_m\left(\boldsymbol{T}_{\boldsymbol{\theta}}\left(\boldsymbol{p}\right)\right)\right)^2 \tag{2.3}$$

The cross-correlation is robust against intensity differences between scans, and is defined by Equations (2.4) and (2.5)

$$\text{NCC}\left(\boldsymbol{\theta}; I_f, I_m\right) = \frac{\sum_{\boldsymbol{p_l} \in \Omega_f} \left(I_f\left(\boldsymbol{p_l}\right) - \overline{I_f}\right)\left(I_m\left(\boldsymbol{T_\theta}\left(\boldsymbol{p_l}\right)\right) - \overline{I_m}\right)}{\sqrt{\sum_{\boldsymbol{p_i} \in \Omega_f} \left(I_f\left(\boldsymbol{p_i}\right) - \overline{I_f}\right)^2 \sum_{\boldsymbol{p_j} \in \Omega_f} \left(I_m\left(\boldsymbol{T_\theta}\left(\boldsymbol{p_j}\right)\right) - \overline{I_m}\right)^2}}, \tag{2.4}$$

with

$$\overline{I_\alpha} = \frac{1}{|\Omega_\iota|} \sum_{\boldsymbol{p} \in \Omega_\iota} I_\alpha\left(\boldsymbol{p}\right) \text{ for } \iota = \{f, m\}. \tag{2.5}$$

If the two images are from different image modalities the mutual information cost function is best suitable. The mutual information cost function is defined in Equation (2.6).

$$\text{MI}\left(\boldsymbol{\theta}; I_f, I_m\right) = \sum_{m \in L_M} \sum_{f \in L_F} \rho(f, m; \boldsymbol{\theta}) \log_2\left(\frac{\rho(f, m; \boldsymbol{\theta})}{\rho_F(f; \boldsymbol{\theta})\rho_M(m; \boldsymbol{\theta})}\right) \tag{2.6}$$

Were ρ is the discrete joint probability between the images, and ρ_F and ρ_M are the marginal discrete probabilities for the fixed and moving image. The HU values of the fixed and moving image are divided into L_f and L_m histogram bins of width ω_f and ω_m [15].

2.4.3. PARAMETER OPTIMISER

The parameter vector $\boldsymbol{\theta}$ is updated in an iterative approach as given in Equation (2.7), where k is the current iteration, $\boldsymbol{a^k}$ the step size and $\boldsymbol{d^k}$ the search direction [15].

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \boldsymbol{a^k} \boldsymbol{d}^k, \quad k = 0, 1, 2, \cdots. \tag{2.7}$$

The search direction is determined using gradient descent, which takes the negative gradient of the cost function, C with respect to $\boldsymbol{\theta}^k$, as the search direction as given in Equation (2.8) [15].

$$\boldsymbol{d^k}\left(\boldsymbol{\theta}^k\right) = -\nabla_{\boldsymbol{\theta}^k} C \tag{2.8}$$

2.4.4. MULTI RESOLUTION APPROACH

Image registration is an iterative optimization process, often performed using a so-called multi-resolution strategy. There are two main methods for multi-resolution.

The first option is to first smooth and downsample the images. This results in images with fewer voxels, reducing the registration complexity. Performing this iteratively with increasingly finer images results in the final registration [15].

A second option is transformation complexity. The registration is first done with fewer degrees of freedom and then again with more degrees. An example is to first perform a rigid transformation to get a rough registration and then perform a nonrigid registration using e.g., B-spline [15].

2.4.5. DIFFEOMORPHIC TRANSFORMATION

A common issue with image registration is the folding of the registration grid over itself, which is not physical [18]. Therefore, the transformation should be diffeomorphic. A diffeomorphic transformation gives a smooth one-to-one mapping between the images and is invertible [19]. To obtain a diffeomorphic transformation, integration over time of a velocity vector field v has to be performed. This was proposed by following the method by Arsigny et al. [20] and the implementation by chen et al. [19] was used. The velocity field is assumed stationary, as is described by the ordinary differential equation given in Equation (2.9). The ordinary differential equation described the change of the DVF over time t with $t \in [0,1]$ [20].

$$\frac{\partial \phi^{(t)}}{\partial t} = v\left(\phi^{(t)}\right) \tag{2.9}$$

The integration is performed using the scaling and squaring method. In the scaling and squaring method, the DVF at time $t = 1$ is the exponent of v, $\phi^{(1)} = exp(v)$, where v is a Lie algebra member. As a result, the DVF is diffeomorphic and invertible [20].

The scaling and squaring method is performed by discretizing the time into T time steps. The initial DVF is given by Equation (2.10), were the stationary velocity field v is divided with a factor of 2^T, such that $v/2^T$ is close to zero [19].

$$\phi^{1/2^t} = \boldsymbol{p} + \frac{v(\boldsymbol{p})}{2^T} \tag{2.10}$$

Next, T recursive squaring steps are performed as given by Equation (2.11).

$$\phi^{1/2^{t-1}} = \phi^{1/2^t} \circ \phi^{1/2^t} \tag{2.11}$$

The resulting diffeomorphic DVF at $t = 1$, ϕ^1, is given by Equation (2.12).

$$\phi^1 = \phi^{1/2} + \phi^{1/2} \circ \phi^{1/2} \tag{2.12}$$

In practical application, the initial velocity field is scaled to $v/2^T$. Next, the squaring is performed for T times using the spatial transformation function. The use of interpolation in the transformation function can result in interpolation errors, as a result, some vectors in the DVF may not be invertible [19].

To evaluate grid folding, local volume changes can be observed by calculating the determinant of the Jacobian matrix of the DVF. If the determinant of the Jacobian matrix is larger than 1 the volume has increased, if the determinant is between 0 and 1 the volume has decreased. A negative determinant is nonphysical and is an indicator that grid folding has occurred [18].

2.5. BREATHING MONITORING

During treatment, it is important to monitor the breathing of the patient, for the timing of the delivery and quality assurance of the delivered dose. Monitoring is done using a breathing model, a breathing model approximates the relationship between the surrogate breathing data and the breathing phase of the patient [21]. There are various methods to obtain surrogate data, often markers are implanted inside of the patient near the region of interest for alignment of the patient at the start of the treatment. These markers can also be tracked using multiple X-ray beams during the treatment. The disadvantage of this method is the additional radiation dose for the patient [21]. Another method is by tracking external motion such as the point on the chest and abdomen. This can be done using optical tracking through LEDs and cameras and markers placed on the patient skin. There are also treatment setups available that include an MRI scanner. This allows for continuous imaging of soft tissue and does not have the disadvantage of additional radiation dose.

A common problem is variations between each breathing cycle, known as inter-cycle variations. There are multiple options to take this into account, the first option is to ignore the variations. This is possible if the variations are smaller than the required accuracy. A second option is to coach the breathing of the patient using visual or audio feedback which helps the patient breathe more regularly. A third option is to incorporate inter-cycle variations into the model [21].

2.6. INTERPLAY EFFECT

The Bragg peak of protons gives the ability for highly localization deposition of energy. This can result in highly con dose distributions. However, density changes along the beam path can impact the range of protons. These changes occur in the lung when the patient is breathing during the time that the proton beam is on. In addition, the target volume moves due to breathing. This can result in local over- and under-dosages of the target and surrounding tissue and results in distortions of the dose distribution. This effect is known as the breathing interplay effect [4, 22].

Currently, interplay mitigation techniques are available for treatment planning and during treatment delivery such as breath holding, beam gating and re-scanning. Currently, clinical practice for interplayer mitigation is done using robust treatment plans.

2.6.1. MITIGATION DURING TREATMENT PLANNING

One of the mitigation techniques at the treatment planning stage consists of enlarging the CTV into an internal target volume (ITV). This is done using the data from the 4DCT scan. The contours of CTV at all the phases are registered to a single phase reference, e.g. 0% inhale. The ITV is set as the union of all registered contours. This results in a larger area being irradiated [10, 23].

A secondary mitigation technique in treatment planning is 4DCT planning. The treatment plan is optimized for delivering the dose in all the phases of the 4DCT scan [24]. 4DCT planning yield more robust plans but is more time-consuming. Hence, currently most treatments allow the patient to breathe freely and increase the ITV to have a robust treatment plan.

2.6.2. MITIGATION DURING DELIVERY

BREATH HOLDING

In deep inspiration breath-hold (DIBH) the patient maintains a breath-hold for a prescribed period at approximately 100% inhale. During this period spots are delivered. This reduces the tumour motion during the delivery and reduces the needed margins for the PTV [25].

RE-SCANNING

A second method is re-scanning, here the dose for each voxel is not delivered at once, but in multiple passes. One could re-scan all the voxels within an energy level multiple times before proceeding to the next energy level, this is called layered re-scanning. Alternatively, one could do volumetric re-scanning. Where one scans all the voxels; in the volume first before re-scanning any voxels again [26]. There are two options to divide the total dose among the re-scanning passes.

The first option is to re-scan each voxels a preset amount of times. The delivered dose per rescan is the total dose per voxels divided by the set number of re-scans. This is called scaled repainting [27], and it is depicted in Figure 2.3a.

The second option is iso-layered repainting, where a maximum dose per visit is set. The voxels are revisited until the total dose per voxels is delivered. This could result in not all voxels being rescanned an equal number of times, since voxels with lower doses need fewer visits which is depicted in Figure 2.3b.

BEAM GATING

A third method is beam gating, where treatment is delivered within an interval of the breathing cycle. By using the breathing signal, it can be determined within which window a spot can be delivered. Usually, the end-exhale phase is chosen as the window since the tumour is most stable. Beam gating minimises the effect of respiration on motion but it increases the treatment time [21, 26].

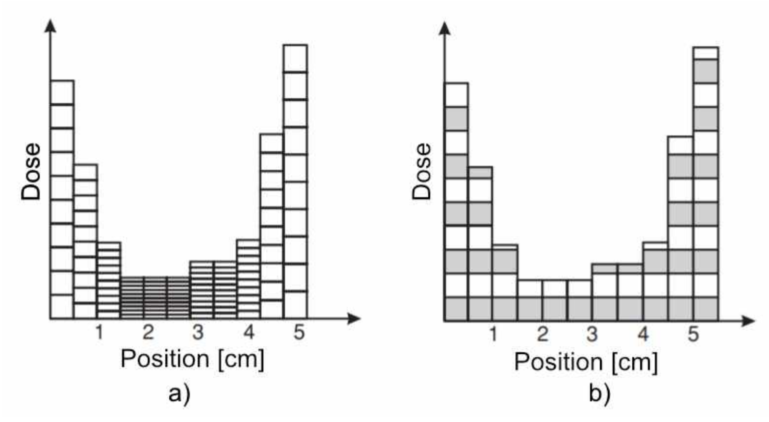

Figure 2.3: Different repainting methods: a) scaled repainting, each position is visited N times and a dose of 1/N is delivered. b) Iso-layered, for each visit a maximum fraction is delivered. Each voxels is revisited until the total dose is delivered. Taken and adapted from S. M. Zenklusen et al. [27].

2.6.3. INTERPLAY DOSE CALCULATION TECHNIQUES

There are approaches to calculating the effective total dose by combining the treatment plan with the breathing signal and the 4DCT data. Based on the treatment plan, the machine parameters, and the breathing signal, it is possible to determine in which breathing phase the patient was when a spot was delivered. The dose of each spot is accumulated in the appropriate phase. The dose distribution for each phase is then transformed to a reference phase through image registration, e.g., the 50% inhale phase. The dose registered distributions from each of the phases are combined to obtain the total dose distribution [10]. A schematic overview of this workflow is given in Figure 2.4.

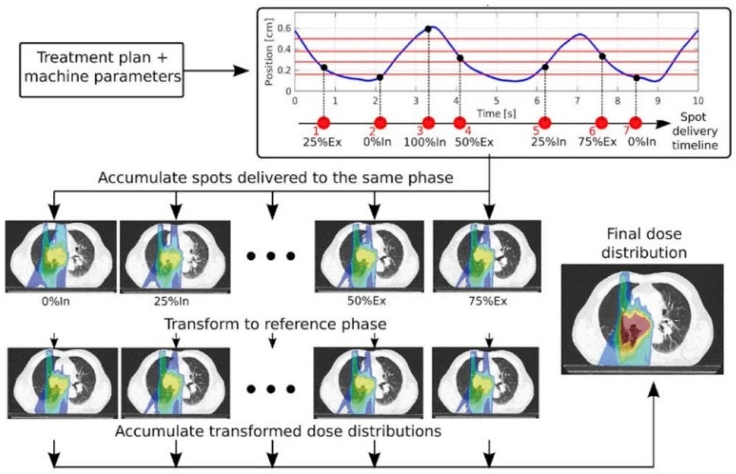
Treatment plan + machine parameters
Position [cm]
Time [s]
Spot delivery timeline
25%Ex
0%In
100%In
50%Ex
25%In
75%Ex
0%In
Accumulate spots delivered to the same phase
0%In
25%In
50%Ex
75%Ex
Final dose distribution
Transform to reference phase
Accumulate transformed dose distributions

Figure 2.4: Dose accumulation workflow: Based on the treatment plan, machine parameters and the breathing, the correct phase in which a spot was delivered is determined. The dose from each individual spot is accumulated in its respective phase. These accumulated doses are subsequently transformed into the reference phase and accumulated to give the final dose distribution. Retrieved from Oscar Pastor-Serrano et al. [10].

2.7. Artificial neural networks

Artificial neural networks are parametric non-linear models that learn the relationship between input and output samples. This is useful when the relationship is complex to describe or unknown. After sufficient learning of the relationship between known input and output pairs, the model is capable of predicting the output for unseen inputs.

The most basic network architecture is a fully connected dense neural network also known as multi-layer perceptrons (MLP). It consists of layers with neurons (also known as nodes), and all the neurons in each layer are connected to all the neurons in the next layer. The first layer of the network is the input layer, where the input data is fed into the network. The last layer of the network is the output layer, the outcome of this layer is the prediction of the network. The input and output layers are the only layers where the network interacts with the outside. Layers in between these are called hidden layers, as they are not visible to the outside. An illustration of a fully connected neural network is given in Figure 2.5.

Within each neuron a function is predefined. The neuron evaluates this function based on its input values and then passes the output to the neurons it is connected to in the next layer. The connection between the nodes have weights, which are trainable. By passing many examples of inputs and desired outputs, the network can adjust the weights to find the optimal values. This process is called training. Some parameters are not optimised during the training but are set manually, such as the number of layers and the number of neurons. The manually set parameters are called hyperparameters.

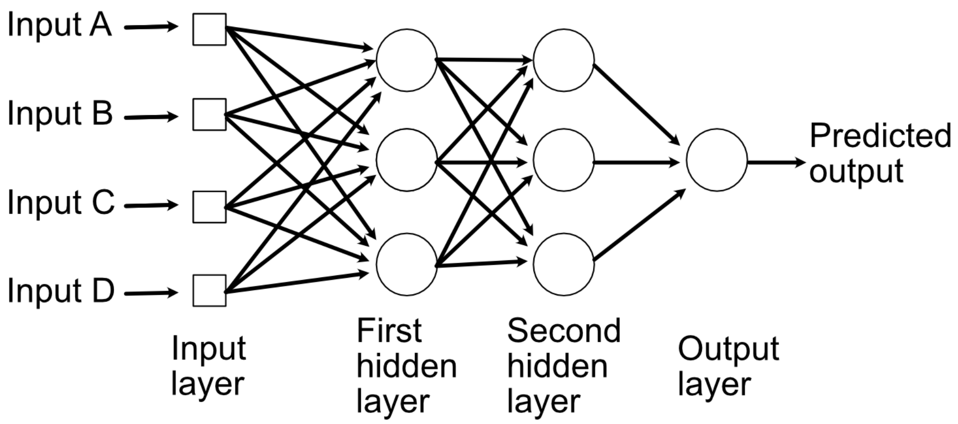

Figure 2.5: Illustration of a fully connected dense neural network. Neurons are represented as circles, and inputs as squares. Arrows represent weighted connections of neurons between different layers. Image adapted from van der Meulen [28].

2.7.1. Model training

To obtain the optimal weights for the network training is conducted. During training, many input and output sample pairs are given to the network. The input propagates through the network and an output is predicted. The predicted output is compared with the desired output using an error function. Next, the error function is a backpropagation and the weights of the network are updated based on the backpropagated error [29].

During the forward pass, the propagation direction is from left to right. Each node evaluates its function f as well as the derivative of the function $f\prime$ using the given input from the left. Both results are stored in the node. The output of the function is multiplied with a trainable weight w_{ij} and passed to the next node on the right. This is illustrated in Figure 2.6 [29].

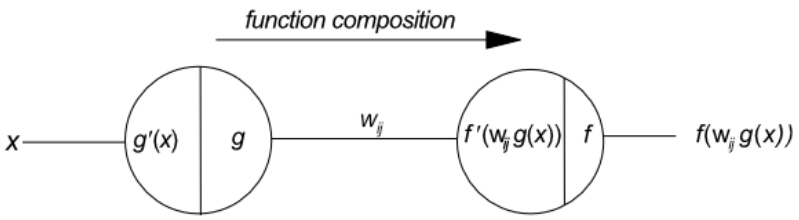

Figure 2.6: Illustration of a forward pass through two nodes with function g and f for input x. On the right side of each node, the nodes function is displayed, and on the right side, the derivative of the function is displayed. Image taken from Feldman et al. [29].

During the backward pass, the propagation direction is from right to left. A 1 is set as the input on the right, and the stored derivative is multiplied by the input. The resulting output is the derivative of the network with respect to the input. This is illustrated in Figure 2.7.

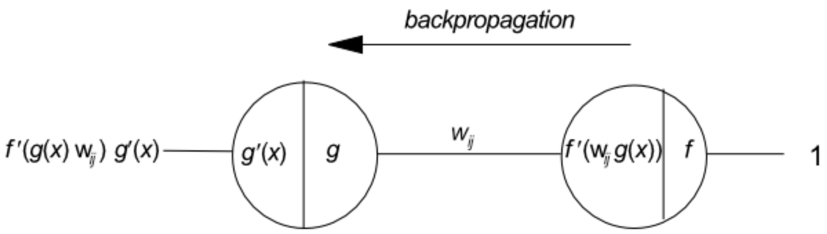

Figure 2.7: Illustration of backward pass through two nodes function g and f for input 1. Image taken from Feldman et al. [29].

Networks often have nodes which have multiple inputs and thus multiple weights. As a result, each node needs to compute and store the partial derivatives of the function f with respect to all inputs. This is illustrated in Figure 2.8.

To evaluate the performance of the network, one can compare the output of the last node, o_n with respect to the target output t_n. The error, E, between the target and the output is calculated using a loss function, for example, using the quadratic deviation given as $E = \frac{1}{2}(o_n - t_n)^2$. During the backward pass, each node calculates the backpropagated error δ [29]. Where δ is given in Equation (2.13) between the output of node i and input node j.

$$\delta_j = \frac{\partial E}{\partial o_i w_{ij}} \tag{2.13}$$

The partial derivative of the error E with respect to the weight w_{ij} between the output of node i and input node j is given in Equation (2.14).

$$\frac{\partial E}{\partial w_{ij}} = o_i \frac{\partial E}{\partial o_i w_{ij}} = o_i \delta_j \tag{2.14}$$

The adjustment of the weight is calculated using Equation (2.15), where γ is the learning rate.

$$\Delta w_{ij} = -\gamma \frac{\partial E}{\partial w_{ij}} = -\gamma o_i \delta_j \tag{2.15}$$

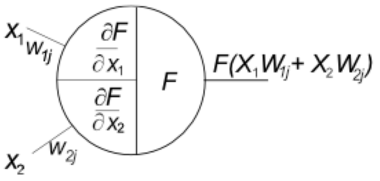
X_1
W_{1j}
$\frac{\partial F}{\partial x_1}$
$\frac{\partial F}{\partial x_2}$
F
$F(X_1W_{1j} + X_2W_{2j})$
X_2
W_{2j}

Figure 2.8: Illustration of a node with two inputs, the partial derivative of each input is stored in the node. Image taken from Feldman et al. [29].

By iteratively adjusting the weights w_{ij} the error is reduced, and the network converges to the optimal weights with minimal error.

When updating the network parameters during backpropagation, the gradients of the lower levels can become increasingly smaller. This could result in the parameters of the lower layers changing very little. This prohibits the network from converging to the optimal solution, this is known as the vanishing gradients problem [30].

For the training of neural networks, the available samples are split into three different datasets. A training dataset, which contains a majority of the samples, is used during the training of the model for finding the optimal weights. The second dataset is the validation dataset, these samples are used to evaluate the performance of the network during the training process. These samples are not used to update the weight. The third dataset is the testing dataset, this dataset is used after training to evaluate the performance of the network after training has finished.

During the training, samples are passed through the network in batches. By using batches the weights are updated based on the average error of multiple samples. This prevents large fluctuations. After all the training samples were passed through the network one training iteration is completed. This is called an epoch. After each epoch, the samples from the evaluation dataset are passed through the network to evaluate the performance of the network. This process is repeated for a set number of epochs.

2.7.2. DATA REPRESENTATION

The data within a neural network is represented as tensors, images are stored in tensors with the shape (N, C, D, H, W). Were D, H, W are the spatial depth, height and width of the image and C represents the number of channels, this gives the number of values per voxel. For example, the Hounsfield Unit (HU) for a CT scan only has a single value per voxel. Hence, the number of channels is one. For a vector field, the number of values per voxel is three, thus the number of channels is three. N is the number of samples.

2.7.3. CONVOLUTIONAL LAYER

For computer vision tasks, networks with layers of convolutional kernels are used. In contrast to a fully connected layer, convolutional neurons learn convolutions filters. The filters, also known as kernels, are 3-dimensional tensors where the values of each voxel is a trainable parameters.

The convolutional layers calculate the 3D cross-correlation between the input tensor and the convolutions filters. The resulting outputs are called feature maps and are high-level representations of the input data [31]. A 2D example of a convolution is displayed in Figure 2.9. Around the image zero padding is applied to increase the size of the tensor such that the feature map has the same shape as the input image.

The dot product is calculated between matrix elements within the receptive field, shown as the blue and red squares, of the input image and the convolution kernel. The output is stored in the feature map. The receptive field shifts along the image with a step size known as the stride. The elements of the kernel are trainable and are adjusted during backpropagation. The size of the filters is known as the kernel size which is a hyperparameter. The convolutional layer has a set number of kernels which is also a hyperparameter.

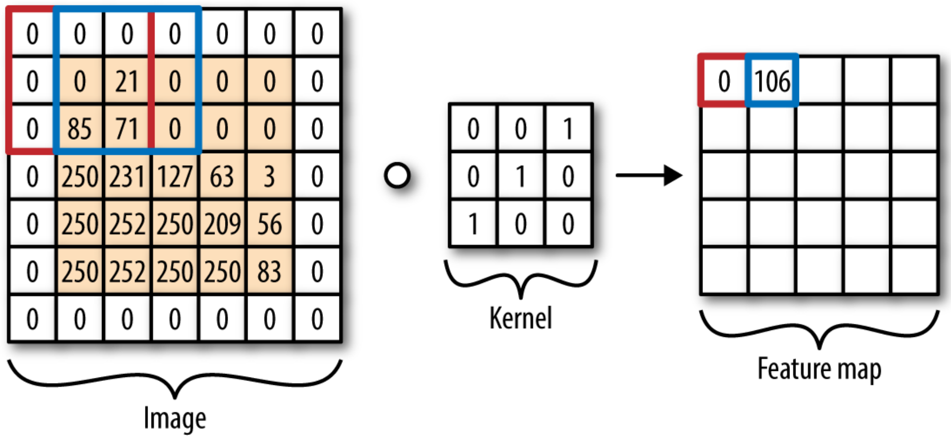

Figure 2.9: Illustration of a convolution in 2D. The input image is zero-padded such that the feature map has the same shape as the input image. The dot product between matrix elements of the input image within the receptive field, shown as the blue and red squares, and the kernel are calculated. The output of the dot product is stored in the feature map. The receptive field shifts along the image with a step size known as the stride. Image taken from B. Ramsundar et al. [32].

The feature maps are passed to the next convolution layer. There is no weight between these connections. Subsequent layers should produce higher-level representations of the input. This process can also be reversed, by transposed convolution where feature maps are used as input.

The total number of trainable parameters per convolution layer is the product of the kernel sizes times the number of filters plus a bias term. For example, a convolutional layer with 32 filters which have a kernel size of (3x3x3) has 865 trainable weights.

2.7.4. POOLING LAYER

To reduce the spatial dimension of a sample pooling can be applied. During pooling, the image is subsampled to reduce the spatial dimension of the image. The subsampling is performed on voxels within a receptive field. There are multiple sampling modes such as average sampling in which the mean value of all the voxels within the receptive field is used. A second option is max pooling in which the maximum value within the receptive field is used. Max pooling is the most used type of pooling [30]. An example of max pooling is shown in Figure 2.10. Pooling results in successive layers having increasingly coarser feature maps [5].

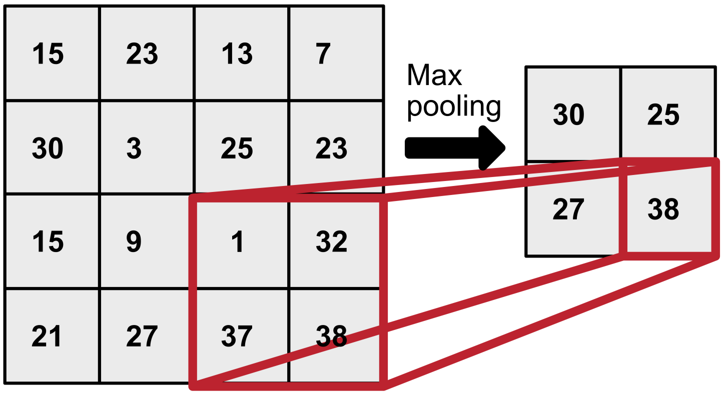

Figure 2.10: Illustration of max pooling in 2D with kernel size 2x2. The spatial dimension is reduced to half by taking the voxel with the maximum value within the 2x2 receptive field. Image taken from van der Meulen [28].

2.7.5. UPSAMPLING LAYER

To increase the spatial dimension upsampling can be applied. An example of upsampling is depicted in Figure 2.11. The number of voxels is increased in each direction and the points extra voxels are filled with either the nearest value or the values could be interpolated.

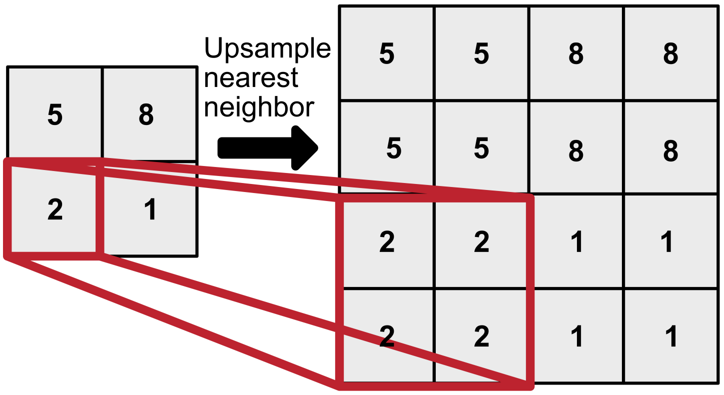

Figure 2.11: Illustration of nearest neighbour upsampling in 2D. The spatial dimension is doubled, and the points of extra voxels are filed with the nearest value.

2.7.6. GROUP NORMALISATION LAYER

Group normalisation has been shown to accelerate the convergence of the model during training. The input channels are divided into a set number of groups. Within the groups, the values are normalised according to Equation (2.16) where $\mathrm{E}[p]$ is the group mean and $\mathrm{Var}[p]$ the group variance. The values γ and β are channel-specific weights which are trainable. ϵ is a constant offset of 10^{-5} for numerical stability [33].

$$y = \frac{x - \mathrm{E}[p]}{\sqrt{\mathrm{Var}[p] + \epsilon}} * \gamma + \beta \tag{2.16}$$

2.7.7. NON-LINEAR ACTIVATION LAYER

In order to add non-linearity to the network, layers with activation functions are added. Commonly used activation functions are sigmoid functions, tanh, Hardtanh and (leaky) rectified linear unit functions [31].

The sigmoid activation function as defined in Equation (2.17) limits the output between 0 and 1 [31].

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{2.17}$$

The hyperbolic tangent limits the output between -1 and 1 and is defined as given Equation (2.18) [31].

$$\text{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.18}$$

The Hardtanh activation function as given in Equation (2.19) is linear within a set range $\pm\Gamma$ and has a constant value of $\pm\Gamma$ outside of this range.

$$\text{HardTanh}(x) = \begin{cases} \Gamma & \text{if } x > \Gamma \\ -\Gamma & \text{if } x < -\Gamma \\ x & \text{otherwise.} \end{cases} \tag{2.19}$$

The leaky rectified linear unit (LeakyReLU) function is defined in Equation (2.20), the function is linear for positive input but suppresses negative input towards zero, where α is small [34].

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise.} \end{cases} \tag{2.20}$$

2.7.8. LOSS FUNCTIONS

In order to find the optimal parameters for the convolutional filter, a loss function is needed to calculate the error. A loss function can be composed of multiple metrics, such as a similarity loss $\mathcal{L}_{sim}$ and a smoothness loss $\mathcal{L}_{smooth}$. Three relevant similarity loss functions for computer vision are the Mean Square Error, Normalised Cross-Correlation and Mutual Information. The mean square error is given in Equation (2.3). The normalised cross-correlation is robust against intensity differences between scans and is given in equations (2.4) and (2.5).

In addition to a similarity loss function, a regulation loss can also be used to have a smooth displacement field. The smooth function, $\mathcal{L}_{smooth}$, is defined as the spatial gradient of the displacement field with neighbouring voxels as given in Equation (2.21) [5].

$$\mathcal{L}_{smooth}(\phi) = \sum_{\mathbf{p}\in\Omega} \|\nabla\phi(\mathbf{p})\|^2 \tag{2.21}$$

As discussed in Section 2.4.5, folding of the registration grid over itself is a common issue. Therefore an additional loss function can be used to penalise the folding of the registration grid. The loss is based on the determinant of the Jacobian matrix of the DVF. A negative determinant is nonphysical and is an indicator that folding has occurred [18]. The loss function is defined in Equation (2.22) [35].

$$\mathcal{L}_{jac}(\phi) = \frac{1}{|\Omega|} \sum_{p\in\Omega} [ReLU\left(-\left|J_\phi(\mathbf{p})\right|\right)] \text{ with } \boldsymbol{J}_\phi(\mathbf{p}) = \nabla\phi(\mathbf{p}) \tag{2.22}$$

2.7.9. U-NET ARCHITECTURE

A commonly used architecture for convolutional neural networks is the U-net architecture as shown in Figure 2.12. It consists of an encoder part and a decoder part.

The encoder layers are represented as grey blocks. Each layer contains a convolutional layer to produce feature maps, group normalisation, a LeakyReLU activation function and a max pooling layer to reduce the spatial dimension of the image.

The decoder layers are represented by blue blocks. Each layer contains a transpose convolutional layer, group normalisation and a LeakyReLU activation function. To increase the spatial dimension nearest-neighbour upsampling is applied. Lastly, the filters are concatenated with the filter from the encoding step. These filters were passed over the black lines with arrows which represent skip connections. These connections also prevent vanishing gradients during propagation.

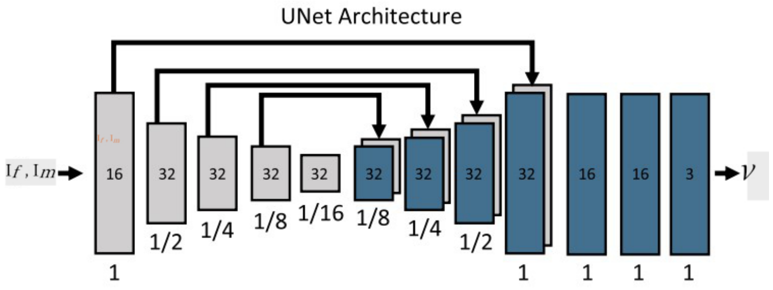

Figure 2.12: Schematic overview of UNet architecture. The network takes two images, I_m and I_f, as input. The network produces a velocity vector field (v). The boxes represent convolutional layers, and the value inside of the layers represents the number of kernels. The lines at the top represent skip connections. The number underneath is the relative spatial resolution of the volume. Retrieved from Balakrishnan et al. [5].

3
Literature overview

This chapter will give an literature overview of deep learning methods for image registration. The application of deep learning methods for image registration can be divided into two main categories: similarity metrics for traditional registrations and direct estimations of the DVF.

3.1. Similarity metrics

Deep learning has been applied to learn similarity metrics to evaluate the performance of the registration obtained by traditional iterative methods. Traditionally, manually crafted similarity metrics such as the MSE, CC and MI metrics are used. [36]. The deep learning models learn similarity metrics to asses the image performance of the registration. This results in a collaboration between traditional iterative image registration methods and performance evaluation using deep learning methods This approach is useful when performing registration between different image modalities [36].

3.2. Deformation vector field prediction

Deep learning has also been applied for the direct prediction of a DVF for registration between images. These methods can be divided into supervised and unsupervised methods. A majority of studies performed registration on images from an MRI modality, with CT being the second most used [37]. Registration of the brain is the most studied, followed by cardiac and lung [37].

Supervised learning methods

Supervised learning methods require the ground truth DVF to calculate the loss between the predicted DVF and the ground truth. The ground truth DVF can be obtained by manually performing registration or by generating an artificial DVF and applying the DVF to an image. As a result, only a limited number of samples are available for training. In addition, the ability of the network to learn accurate registrations is dependent on the quality of the ground truth DVF. Therefore, supervised learning methods have mainly been used for rigid registrations [36, 37].

When the ground truth DVF is unknown, other labelled d data such as contours or landmarks in the images can be used. The landmarks or contours are used to evaluate the quality of the registration. Obtaining landmarks or contours is time-consuming and has to be performed manually by a specialist such as a radiation oncologist, therefore limiting the amount of available data [36]. The quality of the landmarks and contours also impacts the quality of the prediction.

Unsupervised learning methods

Since the ground truth DVF is often unknown and scans with landmarks or contours are limited, many approaches use unsupervised methods. Unsupervised methods predict the DVF and apply the DVF to the moving image using resampling and then compare the registered image with the fixed image [37].

3.3. Registration of lung scans

This section will give an overview of others who have also performed image registration on 4DCT lung scans using deep learning models.

A commonly used dataset for evaluating the performance of the registration was the DIRLAB dataset from the Emory University School of Medicine (Atlanta, GA, USA) [38, 39]. The scans were obtained using a Discovery ST PET/CT scanner (GE Medical Systems, Waukesha, WI). The dataset contains 10

4DCT-lung scans with 300 manually placed landmarks between the end-exhalation and end-inhalation scans.

A commonly used metric for evaluating the performance is the Target Registration Error (TRE), as described by Equation (3.1).

$$\mathrm{TRE} = \frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{\phi_i} + \boldsymbol{l}_f^i - \boldsymbol{l}_m^i\right\|_2 \tag{3.1}$$

Where $\boldsymbol{l}_m^i$ is the landmark location in the moving image, $\boldsymbol{l}_f^i$ the landmark location in the fixed image, and $\boldsymbol{\phi_i}$ the displacement vector at the point $\boldsymbol{l}_M^i$ for landmark i. The TRE gives the average euclidean between n landmark pairs [40]. The mean TRE of the DIRLAB dataset before registration is 8.46 ± 5.48 mm. The three best scores for the TRE using traditional methods for registration were 1.43 ± 1.3 mm, 1.36 ± 0.99 mm, and 1.32 ± 1.24 mm [41].

De Vos et al. [42], proposed an unsupervised approach. The model takes patches of the images and predicts a DVF for each patch. The model, named ConvNet, consists of 3 blocks with a convolution and downsampling layer, followed by two convolutional layers and two fully connected layers to predict the DVF of each patch. The ConvNets were used in a multi-resolution approach from coarse to fine, multiple ConvNets were placed in sequence to predict a DVF on increasingly finer resolutions of the image. The images were resampled using the upsampled DVF predicted by the previous ConvNet on a coarser grid. The mean TRE was 2.64 ± 4.32 mm for the 300 landmarks using the multi-resolution network. The registration time was less than 1 second on an NVIDIA Titan-X GPU [41].

Sentker et al. [43], proposed a supervised network to predict the DVF using pretrained residual neural network (ResNet) blocks. The architecture consists of two separate encoders for each of the images. The features are then concatenated and passed to 5 blocks of pretrained ResNet, followed by a 20% dropout. Next are 10 more blocks of pretrained ResNet and followed by a second 20% dropout, and 5 more blocks of pretrained ResNet. The feature decoder which takes the features from the ResNet and also takes in the moving and fixed image is used to predict the DVF. The resulting TRE was 2.50 ± 1.16 mm. The registration time was a few seconds using an NVIDIA Titan-Xp GPU [41].

Fu et al. [44], used an unsupervised patch-based approach. Patches of 64x64x64 were extracted from the images. The patches were downsampled to 8x8x8 and a first network predicted a coarse DVF. The DVF is upsampled and the original moving patch is registered. Next, the registered patch and the fixed patch are down sampled to 32x32x32 and a finer DVF is predicted by a second network. All the DVFs for each of the patches are combined to obtain a coarse DVF and a fine DVF for the entire image. The resulting TRE was 1.59 ± 1.58 mm. The registration time was less than 1 minute using an NVIDIA Tesla V100 GPU [44, 41].

Sokooti et al. [40], proposed a supervised approach which was trained using artificial DVFs. Three architectures were evaluated. Firstly a U-net architecture with down-sampled images due to memory limitations. The second architecture was an adapted U-net network where the skip connection had additional convolution filters. Due to the increase in filters and memory limitations, the model was updated to have a patch-based approach. The last architecture was a multi-view model, patches of the images were passed to three pipelines at 1/4, 1/2 and full resolution of the patch and were later combined to predict a single DVF. The multi-view model was best performing with a TRE of 1.86 ± 2.12 mm. The registration time was less than 3 seconds using an NVIDIA Titan-Xp GPU [41].

Jiang et al. [45], proposed a multi-resolution unsupervised approach. The full images were downsampled to a coarser resolution and a CNN network predicted a coarse DVF. The predicted DVF was upsampled to the full resolution and the moving image was registered. These images were downsampled and passed to a network at a finer resolution to predict a secondary DVF. This was repeated for a third time. The DVFs predicted at different grid sizes were upsampled to the full resolution and combined to obtain the final DVF. Multiple scales were evaluated, the TRE where 1.75 ± 1.39 mm, 1.58 ± 1.19 mm, 1.56 ± 1.13 mm, for the 2-, 3-, and 4-scale models respectively. The registration time was less than 2 seconds using an NVIDIA Quadro P4000 GPU [41].

Fechter and Baltas [46], also proposed an unsupervised multi-view approach. Patches at the 1/2 and 1/4 and full resolution were passed to three identical U-net networks to predict the DVF at three resolutions. These patches were combined to get obtain a DVF for the full image at each resolution. The coarser DVF are upsampled and added to obtain the final DVF. The TRE was 1.83 ± 2.35 mm. The registration time using an NVIDIA Titan-Xp GPU was about 4 minutes [41].

4 Methods

First in this chapter the architecture and the loss function of the two models are introduces. Next, the used data sets are described, followed by the evaluation methods. Lastly, the training routines are discussed.

4.1. VoxelMorph

The first model was proposed in a paper by Balakrishnan et al. [5], an unsupervised convolutional neural network (CNN) framework is described. The framework learns a registration function between two images which predicts the DVF between these images. This framework is referred to as VoxelMorph (VM). It learns the parameterized registration function based on a collection of image pairs during the training. After training, registration of a new image pair can be performed in under a second using the learned registration function [5].

4.1.1. Network architecture

A schematic overview of the model is given in Figure 4.1. The model takes two 3D images as input, one fixed image and one moving image I_f and I_m respectively. The input shape of the images was set to 80x256x256 voxels. Beforehand, an affine alignment should be performed using conventional methods. The I_f and I_m images are concatenation into a 2-channel 4-D tensor. Then, the concatenated tensor is passed through a CNN with U-net architecture as described in Section 2.7.9. The network learns the function $g_\theta(I_f, I_m)$ to produce a velocity field v between the fixed and moving image. θ are the network parameters, in this case, the convolutions filters of the convolutional layers.

The U-net architecture consists of 3D convolutional layers with a kernel size of 3 and a stride of 1. Group normalisation and LeakyReLU activation layer with α set to 0.2 is applied after each convolution [5]. To reduce the spatial dimension max pooling is applied to the output of the activation layer. During decoding, the same convolutional layer, Group normalisation and activation layer are present. To double the spatial resolution, upsampling to the nearest neighbour is applied. The output is then concatenated with the filters from the encoding stage using the skip connections.

The range of the predicted velocity field was limited using a Hardtanh activation function as given in Equation (2.19) where Γ is the set range. The Hardtanh activation function with range Γ was placed after the U-net, this prevents large deformations. After the Hardtanh activation function an integration layer way present. The integration layer performed integration of the velocity vector v using scaling and squaring as described in Section 2.4.5 to obtain a diffeomorphic DVF ϕ.

Next, the DVF is used to spatially transform the moving image to the predicted image, $I_p = I_m \circ \phi$. The model is unsupervised since the ground truth DVF is unknown, and only the deformed and the fixed images are evaluated.

4.1.2. Loss function

To find the optimal values of θ, gradient descent is used to minimise the loss function. The loss function for the network is given in Equation (4.1). It consists of a similarity score between the predicted image and the fixed image $\mathcal{L}_{sim}(I_f, I_m \circ \phi)$, a regularization term to enforce a spatially smooth deformation, $\mathcal{L}_{smooth}(\phi)$ and a loss to penalise the folding of the registration field $\mathcal{L}_{jac}(\phi)$. Where ψ, λ and γ are hyper parameter.

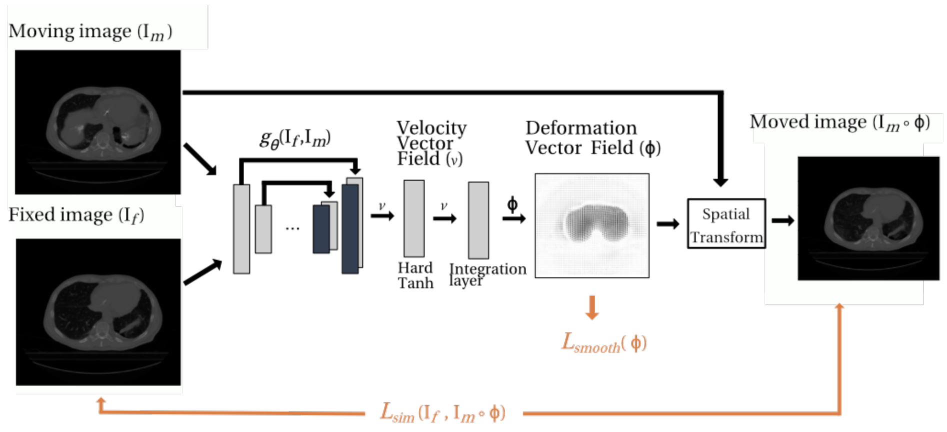
Moving image (I_m)
$g_\theta(I_f, I_m)$
Velocity Vector Field (v)
Deformation Vector Field (ϕ)
Moved image ($I_m \circ \phi$)
Fixed image (I_f)
Hard Tanh
Integration layer
Spatial Transform
$L_{smooth}(\phi)$
$L_{sim}(I_f, I_m \circ \phi)$

Figure 4.1: Schematic overview of the VoxelMorph model. The model takes two input images, the fixed image I_f and the moving image I_m. The model, g_θ has a U-net architecture and predicts a velocity field between these images. The velocity field is limited by a Hardtanh activation layer and an integration layer ensures a diffeomorphic DVF. The DVF is used to transform the moving image to the moved image. The moved image is compared to the fixed image using the similarity loss L_{sim}. Also, the smoothness of the DVF is evaluated by a regularization term L_{smooth}. Image adapted from Balakrishnan et al. [5].

$$\mathcal{L}_{VM}(I_f, I_m, \phi) = \psi\left(\mathcal{L}_{sim}(I_f, I_m \circ \phi) + \mathcal{L}_{sim}(I_m, I_f \circ -\phi)\right) + \lambda \mathcal{L}_{smooth}(\phi) + \gamma \mathcal{L}_{jac}(\phi) \tag{4.1}$$

The used cost functions for the similarity score, $\mathcal{L}_{sim}$, is the local cross-correlation as given in Equations (2.4) and (2.5). The similarity score is applied bi-directionally. $I_m \circ \phi$ and $I_f \circ -\phi$, this penalises if the transformation is not invertible and thus not diffeomorphic. The smoothness function, $\mathcal{L}_{smooth}$, is defined as the spatial gradient of the displacement field with neighbouring voxels as given in Equation (2.21). The used function for $\mathcal{L}_{jac}(\phi)$ is given in Equation (2.22).

4.1.3. VOXELMORPH ADAPTATIONS IN LITERATURE

Multiple options have been proposed to make increase the prediction made by Voxelmorph.

Pal et al. [18], proposed adding an additional cost function that penalises a negative determinant of the Jacobian. This would penalise grid folding. This proposal was implemented in the loss function in this thesis.

Kim et al. [47], proposed to double the network to train both the transform and the inverse transform separately.

Yongnan Zheng et al. [48], proposed an adaption to network architecture. In addition to the direct skip connection in the UNet architecture, it has additional skip connections which have convolutional layers with kernels of different sizes.

Hu et al. [49], adapted the network to have a multi-resolution approach. By down-sampling the image to three coarser grids and predicting DVFs at each resolution and later combining the DVFs. This showed an increase in the quality of the registration.

Hoopes et al. [50], describes expanding the model with a second network to predict the optimal hyperparameters for the registration network. This eliminates the need for manual hyperparameter optimization.

4.2. LAPLACIAN PYRAMID MODEL

The second model used is a Laplacian pyramid network (LAP) proposed by Mok et al. [6]. The model is also based on an unsupervised convolutional neural network that learns to predict the deformation vector field (DVF) between the image pairs. However, the Laplacian model has a multi-resolution approach, where multiple deformation vector fields are predicted at multiple resolutions. The multiple DVFs at increasingly finer grids allow for registration at increasingly finer details. The coarser DVF can correct for large deviations between the images and the finer DVFs correct for more detailed features [51].

4.2.1. Network architecture

The model consists of identical CNN-based registration networks (CRN) at various resolutions. A schematic overview of the Laplacian pyramid network is given in Figure 4.2.

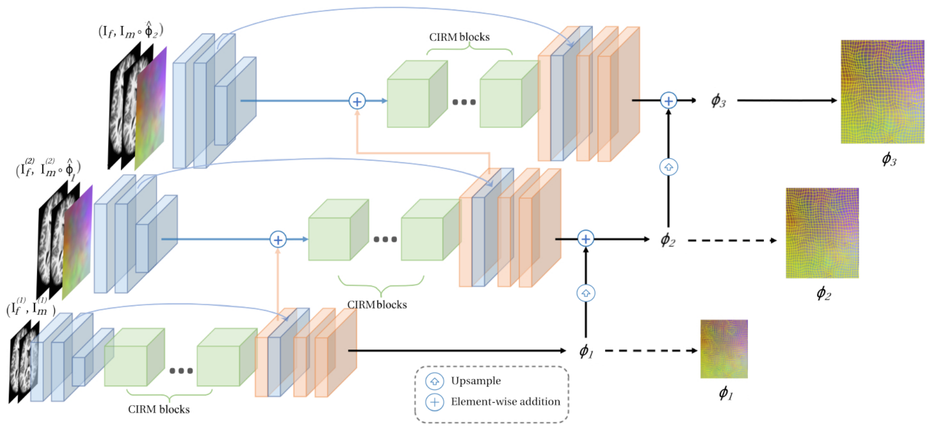

Figure 4.2: Schematic overview of the Laplacian Pyramid model. Three identical CNN-based registration networks (CRN) at increasingly finer grids. The first and coarses CRN is at the bottom. Each CRN consists of a feature encoder (Blue), Conditional Image Registration Module (CIRM) blocks (Green) and a feature decoder (orange). A skip connection is present between the feature encoder and decoder indicated by the blue arrow. The features from the CIRM are element-wise added to the features from the encoder of the finer CRN (orange arrow). The predicted DVF, ϕ, is upsampled and applied to the moving image of the finer CRN. The predicted DVF is upsampled and element-wise added to the finer predicted DVF. Image taken from Mok et al. [6].

First, the image pairs are downsampled using trilinear interpolation to obtain representations of the images at 1/2 and 1/4 of the original resolution. Next, the coarses image pair is used to predict a DVF at the coarses resolutions. This coarse DVF is upsampled to 1/2 the original resolution and applied to the moving image at 1/2 resolutions. These warped finer images are used to predict a finer DVF using a second CRN network at 1/2 the original resolution. Again, the finer DVF is upsampled and warps the moving image at the original resolution, which is then used to predict an even finer DVF. At last, the three DVF are all upsampled to the original resolution and combined to obtain the final DVF.

Each CRN consists of three components: a feature encoder, Conditional Image Registration Module (CIRM) blocks and a feature decoders. The feature encoder consists of a 3D convolutional layer with a kernel size of 3 and a stride of 1. This layer is flowed by a LeakyReLU layer with a value of 0.2 followed by an identical 3D convolutional layer. Then a third 3D convolutional layer is added with a kernel size of 3 and stride of 1 to halve the spatial resolution of the input.

The high-level embedding output from the feature encoders is fed into the CIRM blocks. A schematic overview of the CIRM architecture is given in Figure 4.3. The CIRMs consist of a Conditional Instance Normalization layer, a LeakyReLU layer with a value of 0.2, and a 3D convolutional layer with a kernel size of 3. This is repeated a second time and an additional LeakyReLU layer is added. A skip connection is present from the input until the last LeakyReLU layer.

The Conditional Instance Normalization (CIN) layer regulates the smoothness of the features and introduces a non-linearity and is simulair to group normalisation as discussed in section 2.7.6. the CIRM performs normalisation and shifting of the feature map, as described by Equation (4.2), where $\boldsymbol{w_i}$ represents the feature map, $\boldsymbol{i}$ is the channel and $\mu(\boldsymbol{w_i}), \sigma(\boldsymbol{w_i})$ are the channel-wise mean and standard deviation for channel i.

$$\boldsymbol{w}'_{\boldsymbol{i}} = \gamma_{\theta,i}(\boldsymbol{z}) \left(\frac{\boldsymbol{w_i} - \mu(\boldsymbol{w_i})}{\sigma(\boldsymbol{w_i})} \right) + \beta_{\theta,i}(\boldsymbol{z}) \tag{4.2}$$

The variables $\gamma_{\theta,i}, \beta_{\theta,i} \in \mathbb{R}$ are learned parameters for each CIRM. These variables are predicted using a separated mapping network based on a regularization parameter, ξ, as input. This mapping network consists of a 4-layer multilayer perceptron (MLP) with 48 neurons with a LeakyReLU activation layer with a value of 0.2 and the number of output features of the MLP is twice the number of filters of the CIRM [6]. Learning the variables for each CIRM allows for finding the optimal values for each CIRM at different depths. If a constant value for $\gamma_{\theta,i}, \beta_{\theta,i}$ is used for modulating the CIRM at different depths the performance becomes inconsistent [6].

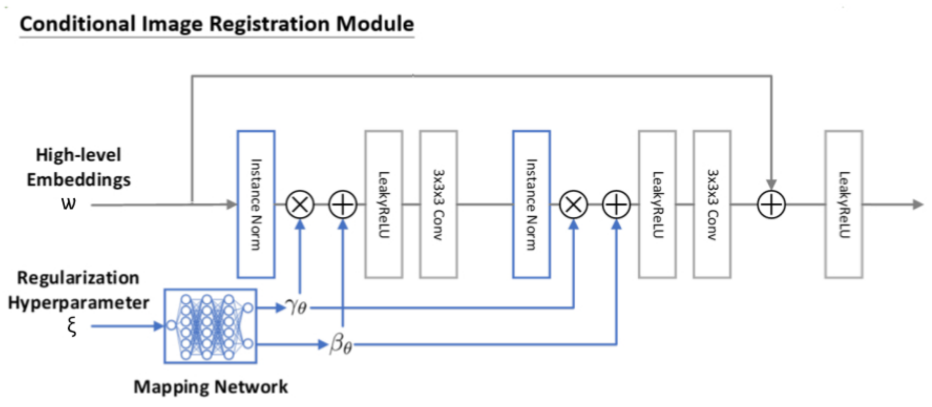

Figure 4.3: Schematic overview of the Conditional Instance Normalization layer (CIRM). The high-level embedding w is passed to a conditional instance normalization layer followed by a LeakyReLU layer with a value of 0.2 and a 3D convolutional layer with a kernel size of 3. This is repeated a second time. The output is concatenated with the input w via a skip connection and is passed through a LeakyReLU layer. The values for $\gamma_\theta, \beta_\theta$ of the conditional instance normalization layer are predicted using a mapping network based on a regularization parameter ξ. The mapping network consists of a 4-layer multilayer perceptron (MLP) with 48 neurons with a LeakyReLU activation layer with a value of 0.2. The number of output features of the MLP is twice the number of filters of the CIRM. Adapted from Mok et al. [6].

4.2.2. LOSS FUNCTION

To leverage the fact that a good alignment will yield high similarity values among all resolutions, a similarity pyramid framework is proposed. The similarity score is calculated on all the preceding grid sizes and summed with a weight for each level. Lower weights are assigned to coarser resolutions, since coarser levels are less sensitive to noise. Summing over all the grid levels avoids local minima during training on fine resolutions [6].

The resulting loss function for the Laplacian model is given in Equation (4.3), where K is the number of grids used, and k is the current grid level. The local cross-correlation as given in Equations (2.4) and (2.5) is used for the similarity score $\mathcal{L}_{sim}$. Equation (2.21) is used for the $\mathcal{L}_{smooth}$ loss. The penalty for folding of the deformation field, $\mathcal{L}_{jac}$, is given in Equation (2.22). Where ψ, λ and γ are hyper parameter.

$$\mathcal{L}_{LAP}(I_f, I_m, \phi) = \sum_{k\in[1,..,K]} -\frac{\psi}{2^{(K-k)}}\mathcal{L}_{sim}(I_f, I_m \circ \phi) + \frac{\lambda}{2^{(K-k)}}\mathcal{L}_{smooth}(\phi) + \gamma\mathcal{L}_{jac}(\phi) \tag{4.3}$$

4.3. IMAGE REGISTRATION USING ELASTIX

To compare the quality of the image registration, the image pairs from the test data are also registered using Elastix. The parameters for the registration were taken from a published study [52, 53]. The registration is a β-spline Transform between the moving and the fixed image pairs. The registration is multi-resolution on three resolutions, 1/4th, 1/2th and full resolution respectively. On each resolution 2000 iterations were performed to obtain the optimal registration. To evaluate the quality of the registration the mutual information metric was calculated for 5000 random samples [53].

4.4. DATA SET GENERATION

4.4.1. 4DCT DATA SET

For training and evaluation of the neural networks, three different data sets with 4DCT scans from non-small cell lung cancer patients were used. An overview of the datasets is given in Table 4.1.

• The first dataset is from the Cancer Imaging Archive containing 4DCT scans from non-small cell lung cancer patients [54]. Each 4DCT consists of 10 CTs at phases between 0 to 90% of the breathing cycle with increments of 10%. The 4DCT images were acquired using a Brilliance Big Bore 16-slice helical CT scanner by Philips Medical Systems, (Andover, MA, USA) [54]. The data set consists of 77 4DCT scans from twenty different patients. The number of scans per patient ranged between one and eight. Seven patients have gold coils implanted around the tumor as markers [54]. The scans consist of 512x512 voxels in the x and y directions with a spatial resolution of 0.97 mm. The spatial resolution in the axial direction is 3 mm, and the number of slices in the axial direction varied between the scans ranging from 77 to 149 [54]. The scans from this dataset have contours available, which were delineated by a single radiation oncologist. The delineated regions are the esophagus, the left and right lungs, the tumor and the heart [54]. However, some scans and phases only had a subset of these contours available.

• The second dataset consisting of seven 4DCT scans from seven patients with 10 phases per scan. These scans were obtained from the Léon Bérard Cancer Center & CREATIS lab, (Lyon, France) [55, 56, 57]. The scans consist of 512x512 voxels in the x and y directions with varying spatial resolutions in the x and y directions. The spatial resolution in the axial direction was 2 mm, and the number of slices in the z direction varies between the scans ranging between 140 and 187. These scans have landmarks available [55].

• The third dataset consisting of 10 4DCT-lung scans with 10 phases per scan, was obtained from the Emory University School of Medicine, (Atlanta, GA, USA) [38, 39]. The scans were obtained using a Discovery ST PET/CT scanner (GE Medical Systems, Waukesha, WI). Only 5 scans were used since these scans have a spatial resolution of 0.97 mm in the x and y direction and consist of 512x512 voxels. Whereas the other scans had only 256x256 voxels. The slices have a thickness of 2.5 mm with the number of slices ranging between 120 and 136. The scans have landmarks available [38, 39]. This dataset is the same as used for calculating the TRE between landmarks in Section 3.3.

Table 4.1: An overview of available data per dataset. Elastix indices if the dataset was used for comparison with registrations by Elastix.

	Dataset 1	Dataset 2	Dataset 3
Number of scans	77	7	5
Contours	✓	✗	✗
Landmarks	✗	✓	✓
Registerd using Elastix	✓	✓	✗

To reduce the GPU memory the number of grid points in the x and y plane was reduced to 256x256 voxels, and the 4DCT scans were resampled to a spatial resolution of 1.94 mm in the x and y plane and a resolution of 3 mm in the z-direction. If the resampled images had less than 256x256 grid points in the x and y plane, the image was padded in the x and y direction with air. If the number of grid points in the x and y plane was larger than 256x256, the image was cropped to 256x256 grid points. The resulting number of slices in the z direction from the resampled image was not altered. The resulting image has a shape of 256x256xNz, where Nz is the number of slices in the z-direction. The resampled images of all phases from a scan were stored together in a single Hierarchical Data Format version 5 (HDF5) file [58].

4.4.2. CONTOURS

Scans from the first dataset also include contours for organs. The contours of all the organs for each scan were stored in a single DICOM file. The contours are a set of points with an exact coordinate for each point. As the neural network only takes tensor indices, each point is sampled to the nearest voxel positions inside of the tensor. This introduces an error because the original contour points had exact coordinates, whereas in the tensor the coordinates are limited to the voxel positions. The tensor of each individual organ is stored in a separate sparse tensor.

4.4.3. LANDMARKS

The second and third datasets have landmarks available. Landmarks are points with an exact coordinate and were manually identified at each phase. An example of a landmark at various phases is shown in Figure 4.4.

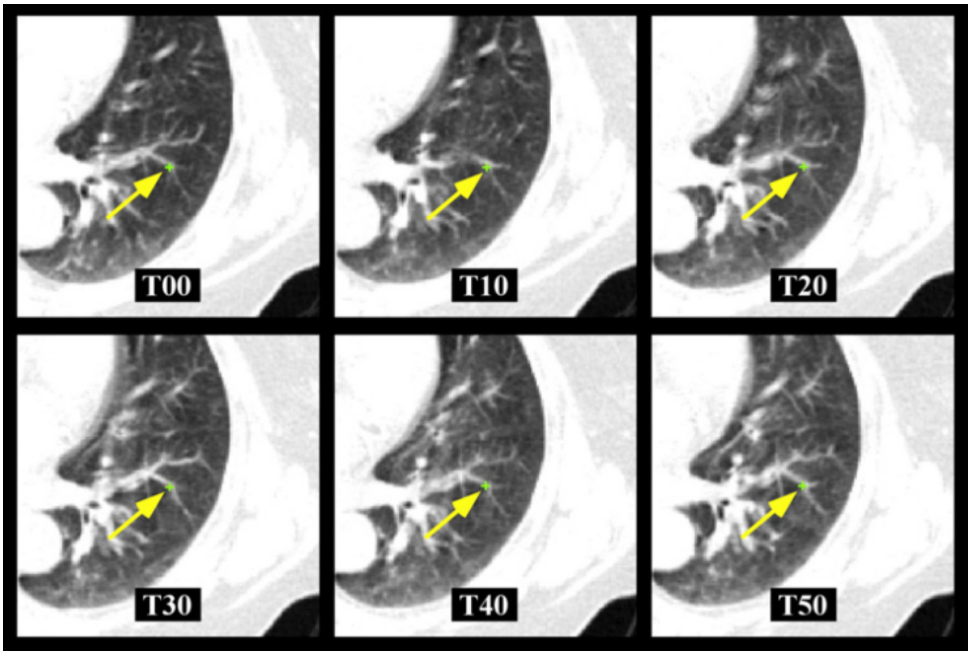

Figure 4.4: Example of a landmark in, green cross indicated by the yellow arrow, at 6 different phases of the breathing cycle. Image taken from Castillo et al. [38].

4.4.4. DATA AUGMENTATION

The voxel values of the three datasets ranged between -1000 and 3000 Hounsfield units (HU). The HU were rescaled to range between 0 and 1 values by increasing the values by 1000 and then dividing by 4000. To prevent overfitting of the network, data augmentation was applied before being fed to the neural networks. Augmentation was performed by randomly shifting the moving and fixed images in the x-y direction within a set range of ± 5 voxel in the x direction and $[0, 15]$ in the y direction. The number of available slices per scan was larger than the input of the network of 80 slices, and therefore a sub-volume of 80 slices was randomly selected.

4.5. NETWORK IMPLEMENTATION

For both the VM and LAP models the original Pytorch implementations are publicly available [59, 35]. These implementations of the networks were adapted and optimised to work with the 4DCT lung data.

4.5.1. GRADIENT ACCUMULATION

Due to the limited available GPU memory, only batches of a limited number of image pairs could be passed through the network simultaneously. Training with small batches could lead to large fluctuations and thus hinder training. To update the weights of the network based on more samples, gradient accumulation was used.

A set number of batches is passed through the network and their loss is calculated and divided by the set number of batches. Next, the loss gradients calculated and are accumulated. After all the set batches are passed, the weights are updated using the accumulated gradient.

4.5.2. TRAINING AND VALIDATION DATA

Each 4DCT scan consists of 10 phases per breathing cycle, resulting in 100 image pairs (90 combinations between phases plus 10 identity registrations) per scan. The training data consisted of 61 scans from 11 patients taken from dataset 1. This resulted in a total of 6100 image pairs. The validation set consisted of 500 image pairs from a single patient in dataset 1 with five 4DCT scans. Only CT data was used as input for training and validation, no contours or landmarks data was used during the training

4.5.3. TRAINING ROUTINE

VOXELMORPH

The Voxelmorph model was trained for 25 epochs with batches of two image pairs due to GPU memory limits. However, the gradients of ten batches were accumulated as described in Section 4.5.1 before updating the weights. This has an effective batch size of 20 image pairs. The set values of ψ was set to 2500, λ to 10, γ to 350000 and Γ was 10.

LAPLACIAN MODEL

The total training routine of the Laplacian Pyramid Networks consists of three consecutive training routines in a coarse-to-fine training scheme. First, only a single CIRM at the coarsest resolution on 1/4 is trained to predict a DVF. For the next training routine, an additional CIRM is added to the network at 1/2 the resolution level. The weights of the previously trained coarser CIRM are frozen for the first 200 image pairs in the first epoch to initialise the weights of the finer CIRM. During the third training routine, the third CIRM is added at the original resolution. Again, the weights of the previously trained coarser CIRM are frozen for the first 200 image pairs in the first epoch. The initial number of CIRM blocks was 5, with 32 filters each.

The CIRM at the coarsest level is trained for 40 epochs with a batch size of 20 image pairs. The second level was also trained for 40 epochs with a batch size of 5 image pairs and the finest level is trained for 25 epochs with a batch size of two image pairs. The set values of ψ was set to 1, λ to 4, γ to 350000.

Training, validation and testing of the networks were performed on a machine with an NVIDIA Tesla V100S GPU with 32GB of memory. This machine was available through DelftBlue, the high-performance computer cluster at the TU Delft [60].

4.6. EVALUATION METHODS

To compare the accuracy of the predictions made by the different models three evaluation methods were used.

4.6.1. MEAN ABSOLUTE ERROR

The first evaluation method is the mean absolute error. The mean absolute error is defined in Equation (4.4), which compares the image intensities in HU units. Where I_f is the fixed image, I_m is the moving image and ϕ is the DVF The function sums over all voxels p in the volume Ω. The intensity values of the fixed and moving images were rescaled from [0,1] to [-1000, 3000] on the HU scale.

$$\mathrm{MAE}(I_f, I_m \circ \phi) = \frac{1}{|\Omega|} \sum_{\mathbf{p}\in\Omega} |I_f(\mathbf{p}) - [I_m \circ \phi](\mathbf{p})| \tag{4.4}$$

4.6.2. REGISTRATION GRID FOLDING

As discussed in Section 2.4.5 a common issue with image registration is the folding of the registration grid over itself, this results in the transformation not being diffeomorphic [18]. Therefore the determinant of the Jacobian matrix from the deformation vector field is calculated for each voxel point. Then the fraction of grid points with negative determinate is calculated. This is mathematically formulated in Equation (4.5).

$$\mathrm{JAC}(\phi) = \frac{1}{|\Omega|} \sum_{\mathbf{p}\in\Omega} \left(\left|J_\phi(\mathbf{p})\right| \leq 0\right) \text{ with } \boldsymbol{J}_\phi(\mathbf{p}) = \nabla\phi(\mathbf{p}) \tag{4.5}$$

4.6.3. EVALUATION USING CONTOURS

As stated in Section 4.4.1, some scans have contours available for the esophagus, the left and right lung, the tumor, and the heart. The network is trained with an image dimension of 80x256x256 voxels, whereas the original 4DCT images and the contours have a dimension of 80x512x512. Therefore the predicted DVF is upsampled in the x and y direction using trilinear interpolation to have the same size as the contours. The contours are deformed using the upsampled DVF and the PyTorch grid sample function to the nearest neighbour [61].

To evaluate the deformed contour the Hausdorff Distance (HD), the mean surface distance (MSD) and the surface dice similarity coefficient (DSC) are calculated. These metrics were calculated using a publicly available surface distance metrics package [62, 63]. An illustration of each metric is given in Figure 4.5.

The Hausdorff Distance gives the maximum distance between the nearest points from the two contours, A and B. The function calculates the maximum distance for a point in the contour A to its nearest point in contour B. This is formulated in Equation (4.6). The Hausdorff Distance is the maximum from Equation (4.6) for $h(A,B)$ and $h(B,A)$ as given in Equation (4.7), were a and b are points in contour A and B respectively [64].

$$h(A,B) = \max_{a\in A}\min_{b\in B}\|a-b\| \tag{4.6}$$

$$HD = \max(h(A,B), h(B,A)) \tag{4.7}$$

The mean surface distance gives the average distance between points on the contours. For each point on the contour A the closed point on contour B is determined, and these results are averaged. The same is done in the opposite direction, and both results combined with a weight of 1/2. This is formulated in Equation (4.8) where $q(A,B)$ is given in Equation (4.9) [64].

$$MSD = \frac{1}{2}(q(A,B) + q(B,A)) \tag{4.8}$$

$$q(A,B) = \frac{1}{A}\sum_{a\in A}\min_{b\in B}\|a-b\| \tag{4.9}$$

For the overlap of organs the volumetric Dice similarity coefficient (DSC) is commonly used. The volumetric DSC compares the overlap between the volumes enclosed by the contour, which requires the generation of a mask enclosed for each layer. The surface Dice Similarity Coefficient (surface DSC) asses the overlap between the outer surfaces of the contours. It is defined as twice the union of the surface of the two contours, normalised by the sum of the surface of the two contours, this is given in Equation (4.10), where the verticle bars, |, denote the surfaces of the contour [64]. The value of the surface DSC ranges between 0 and 1, where 0 is no overlap of the contours, and 1 is a perfect overlap within the acceptable tolerance [63]. Since the contours are stored as discrete voxels and the DVF is upsampled from 2mm to 1 mm and a tolerance of 1 mm is allowed in each of the directions. In this report, the surface DSC is used.

$$DSC = \frac{2|A\cap B|}{|A|+|B|} \tag{4.10}$$

4.6.4. EVALUATION OF PREDICTION TIME

Besides the accuracy of the registration, prediction time is also important, since long prediction times limit the practical use of the network. The average time to predict the DVF on both a CPU and a GPU is measured and compared.

4.7. PERFORMED SIMULATIONS

4.7.1. VOXELMORPH MODEL

During early simulations of the VM model, the predicted DVF would diverge. Therefore a HardTanh activation layer, as given in Equation (2.19), is added after the U-net and before the integration step.

This limits the maximum value of the vectors in the DVF. The range of the linear region, Γ, is evaluated for values of 5,6,8,10,15,20.

Additional simulations were formed with a range of values for λ as a weight for $\mathcal{L}_{smooth}(\phi)$ loss function. Simulated values for λ where powers of then, 10^n with n (ranging between 0 and 5), and additional values of 500, 5000, and 50000. The value of Γ for the HardTanh was 10. These models were individually trained as described in Section 4.5.3.

4.7.2. LAPLACIAN PYRAMID MODEL

For the Laplacian pyramid model, the number of CIRM blocks as well as the number of filters was varied for the coarsest level. The tested number of CIRM blocks was 5,7,9 and the number of filters was 32, 48, and 80. This resulted in nine different models.

First, only the coarsest level was varied, since the additional computational cost is limited due to the coarser resolution. All other levels had the original setting of 5 CIRM blocks with 32 filters. Second, the number of CIRM blocks and filters for all three levels was varied.

The last simulation was expanding the model to have a fourth grid size at 1/8 the resolution. This level had the same number of residual CIRM blocks and filters as the original model, namely 5 residual blocks and 32 filters. Adding a fourth grid size increases the model size and the computational cost. However, the additional grid size is on a coarse grid the computational is cheap. These models were individually trained as described in Section 4.5.3.

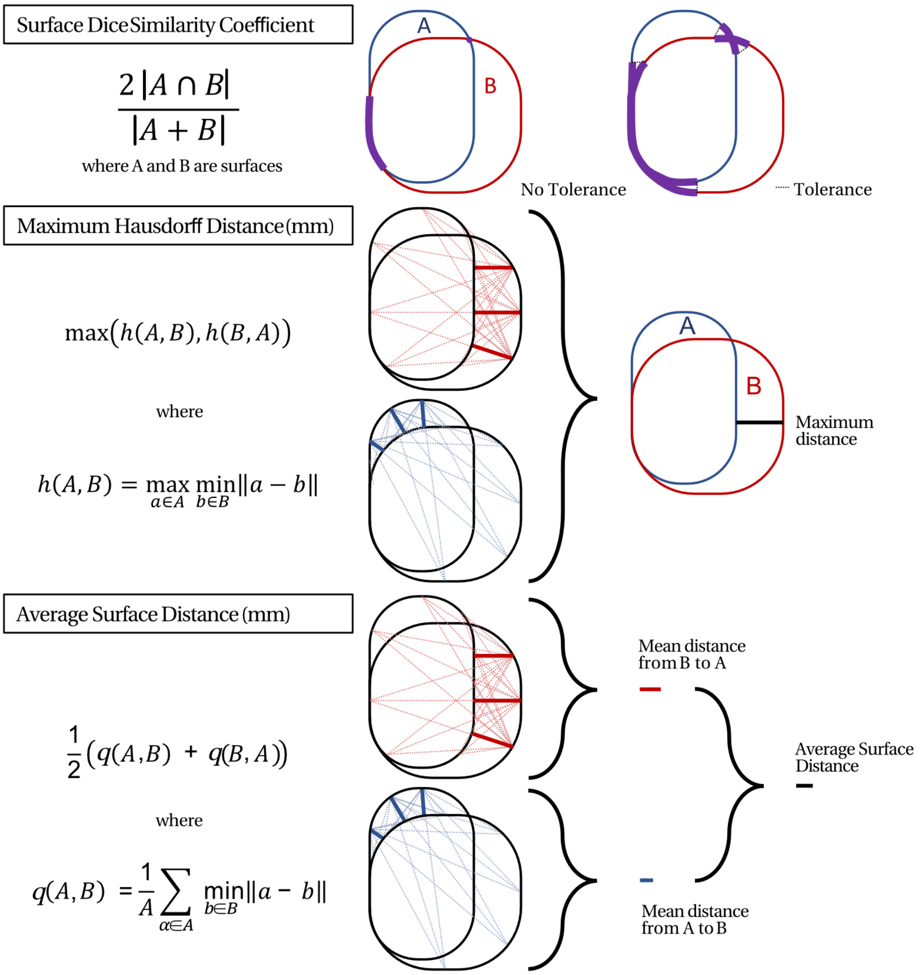
Surface Dice Similarity Coefficient
$\frac{2\,|A \cap B|}{|A + B|}$
where A and B are surfaces
A
B
No Tolerance
Tolerance
Maximum Hausdorff Distance (mm)
$\max(h(A,B), h(B,A))$
where
$h(A,B) = \max_{a\in A} \min_{b\in B} \|a - b\|$
A
B
Maximum distance
Average Surface Distance (mm)
$\frac{1}{2}(q(A,B) + q(B,A))$
where
$q(A,B) = \frac{1}{A} \sum_{a\in A} \min_{b\in B} \|a - b\|$
Mean distance from B to A
Average Surface Distance
Mean distance from A to B

Figure 4.5: Illustration and formula showing the Surface Dice Similarity Coefficient, Hausdorff Distance and Average Surface Distance for contour evaluation between contours A and B in 2D. For the surface Dice, the purple section indicates accepted overlap, without tolerance (left) and with tolerance (right). The Hausdorff distance is the maximum distance between points contour A and B indicated with the black line on the right. And the Average Surface Distance between the contours. Illustration taken from K.J. Kiser et al. [64].

5
Results

The model of the performed simulations described in Section 4.7 are evaluated using the metrics described in section 4.6. The evaluation is performed using the three data sets described in section 4.4.1. From dataset 1 500 unseen image pairs were used in the testing set. Testing data from the second dataset consists of 700 image pairs, and from the third 500 image pairs were used.

Please note that the centre value of the box gives the median value. The box gives the 25% to 75% interval, known as the interquartile range (IQR). The whiskers extents to 1.5 * IQR values. Circles represent are outliers. The tables give the average value and the standard deviation.

5.1. VoxelMorph Model

5.1.1. Results varying the HardTanh value

The calculated mean absolute error in HU units of the three testing datasets for various values of Γ for the HardTanh is given in Table 5.1 and displayed in Figure 5.1.

Table 5.1: Mean absolute error in HU units between the prediction and the fixed image on three testing datasets for various values of Γ for the VM model. Calculated as described in Section 4.6.1.

Γ value	Dataset 1	Dataset 2	Dataset 3	Average
Baseline	19.50 ± 9.02	18.22 ± 7.77	23.41 ± 10.54	20.06 ± 9.26
5	16.76 ± 6.92	16.52 ± 5.17	19.38 ± 6.35	17.36 ± 6.29
6	17.43 ± 6.30	17.37 ± 4.71	20.28 ± 5.92	18.16 ± 5.80
8	16.32 ± 6.46	17.52 ± 4.84	19.56 ± 6.30	17.62 ± 6.01
10	17.85 ± 7.04	17.03 ± 5.27	21.03 ± 7.14	18.38 ± 6.67
15	17.17 ± 6.63	16.29 ± 4.81	19.77 ± 6.09	17.53 ± 6.04
20	16.98 ± 6.70	16.70 ± 4.53	19.55 ± 5.76	17.55 ± 5.85

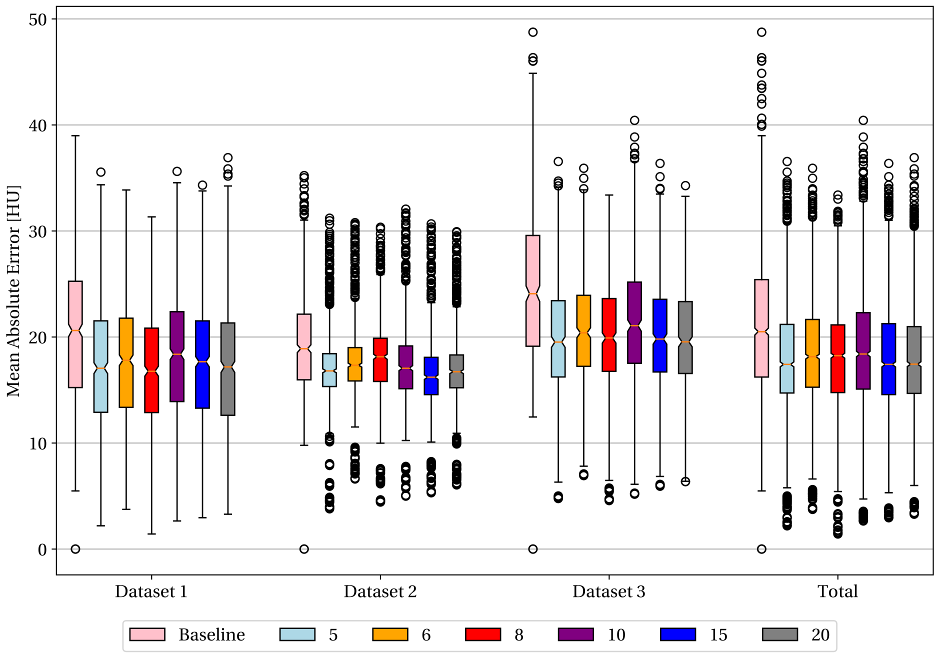

Figure 5.1: Mean absolute error in HU units between the prediction and the fixed image for various values of Γ for the VM model. Calculated as described in Section 4.6.1. Values are given in Table 5.1.

The ratio of voxels in the DVF with a negative Jacobian determinant for various values of Γ for the HardTanh is given Table 5.2 and shown in Figure 5.2.

Table 5.2: Ratio of voxels with a negative Jacobian determinant for various values of Γ for the HardTanh of the VM model. Calculated as described in Section 4.6.2.

Γ value	Dataset 1	Dataset 2
5	$1.764\cdot10^{-5}\pm3.866\cdot10^{-5}$	$3.884\cdot10^{-5}\pm9.709\cdot10^{-5}$
6	$2.232\cdot10^{-5}\pm7.261\cdot10^{-5}$	$2.287\cdot10^{-5}\pm4.788\cdot10^{-5}$
8	$2.232\cdot10^{-5}\pm7.261\cdot10^{-5}$	$2.287\cdot10^{-5}\pm4.788\cdot10^{-5}$
10	$2.155\cdot10^{-6}\pm1.211\cdot10^{-5}$	$2.001\cdot10^{-6}\pm1.428\cdot10^{-5}$
15	$1.942\cdot10^{-5}\pm4.110\cdot10^{-5}$	$4.065\cdot10^{-5}\pm9.267\cdot10^{-5}$
20	$7.821\cdot10^{-5}\pm1.747\cdot10^{-4}$	$7.058\cdot10^{-5}\pm1.381\cdot10^{-4}$

Γ value	Dataset 3	Average
5	$2.812\cdot10^{-5}\pm3.726\cdot10^{-5}$	$2.821\cdot10^{-5}\pm6.687\cdot10^{-5}$
6	$8.594\cdot10^{-5}\pm1.444\cdot10^{-4}$	$3.927\cdot10^{-5}\pm9.516\cdot10^{-5}$
8	$5.830\cdot10^{-4}\pm6.143\cdot10^{-4}$	$3.878\cdot10^{-4}\pm5.077\cdot10^{-4}$
10	$1.256\cdot10^{-5}\pm3.308\cdot10^{-5}$	$4.836\cdot10^{-6}\pm2.094\cdot10^{-5}$
15	$1.039\cdot10^{-5}\pm1.813\cdot10^{-5}$	$2.486\cdot10^{-5}\pm6.349\cdot10^{-5}$
20	$2.955\cdot10^{-5}\pm4.219\cdot10^{-5}$	$6.259\cdot10^{-5}\pm1.383\cdot10^{-4}$

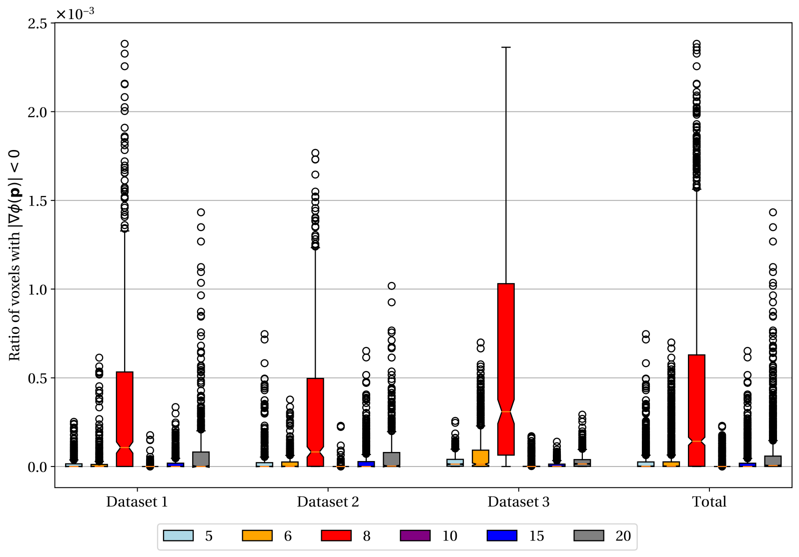

Figure 5.2: Ratio of voxels with a negative Jacobian determinant for various values of Γ for the VM model. Calculated as described in Section 4.6.2. Values are given in Table 5.2.

The calculated dice scores are given in Table 5.3 and are depicted in Figure 5.3. The calculated mean surface distances are given in Table 5.4 and displayed in Figure 5.4. The Hausdorff results are given in Table 5.5 and displayed in Figure 5.5.

Table 5.3: The Dice score between the prediction and the fixed contour for various values of Γ for the HardTanh for the VM model. Calculated as described in Section 4.6.3

Γ value	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	0.89 ± 0.08	0.83 ± 0.12	0.83 ± 0.12	0.79 ± 0.13	0.76 ± 0.17
5	0.89 ± 0.08	0.88 ± 0.08	0.87 ± 0.09	0.82 ± 0.11	0.77 ± 0.14
6	0.90 ± 0.08	0.87 ± 0.09	0.86 ± 0.09	0.82 ± 0.11	0.78 ± 0.15
8	0.90 ± 0.08	0.88 ± 0.09	0.87 ± 0.09	0.82 ± 0.10	0.79 ± 0.15
10	0.87 ± 0.07	0.88 ± 0.08	0.87 ± 0.09	0.81 ± 0.10	0.74 ± 0.13
15	0.90 ± 0.08	0.86 ± 0.10	0.85 ± 0.10	0.81 ± 0.12	0.78 ± 0.15
20	0.90 ± 0.08	0.87 ± 0.09	0.86 ± 0.10	0.82 ± 0.11	0.78 ± 0.16

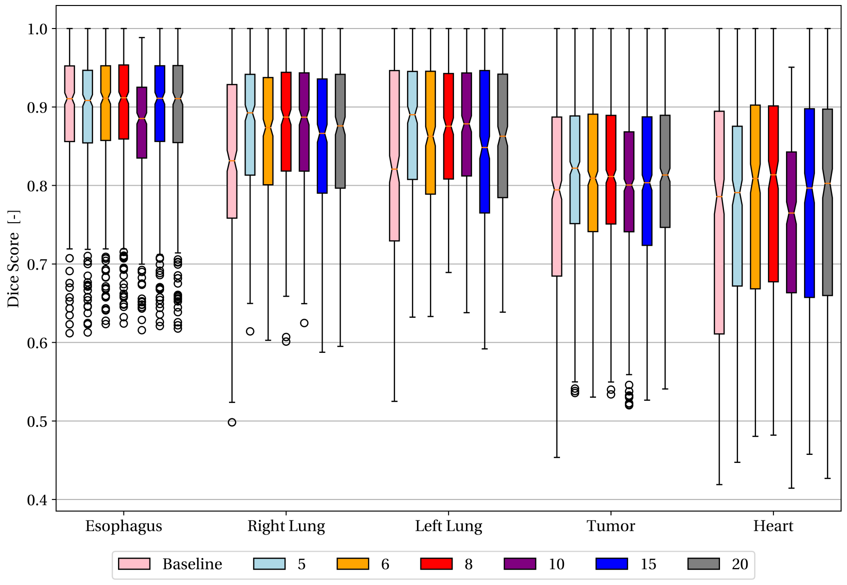

Figure 5.3: The Dice score between the prediction and the fixed contour for various values of Γ for the VM model. Calculated as described in Section 4.6.3. Values are given in Table 5.3.

Table 5.4: Mean surface distance in mm between the prediction and the fixed contour for various values of Γ for the HardTanh of the VM model. Calculated as described in Section 4.6.3.

Γ value	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	0.41 ± 0.34	0.92 ± 0.80	0.80 ± 0.73	0.71 ± 0.46	0.98 ± 0.79
5	0.43 ± 0.35	0.63 ± 0.66	0.58 ± 0.55	0.62 ± 0.37	0.90 ± 0.62
6	0.40 ± 0.34	0.64 ± 0.60	0.60 ± 0.52	0.62 ± 0.36	0.85 ± 0.65
8	0.41 ± 0.34	0.61 ± 0.62	0.56 ± 0.49	0.62 ± 0.36	0.85 ± 0.63
10	0.55 ± 0.35	0.61 ± 0.57	0.56 ± 0.45	0.67 ± 0.34	1.03 ± 0.61
15	0.40 ± 0.33	0.66 ± 0.60	0.61 ± 0.50	0.64 ± 0.38	0.88 ± 0.66
20	0.41 ± 0.35	0.69 ± 0.75	0.63 ± 0.62	0.61 ± 0.36	0.89 ± 0.70

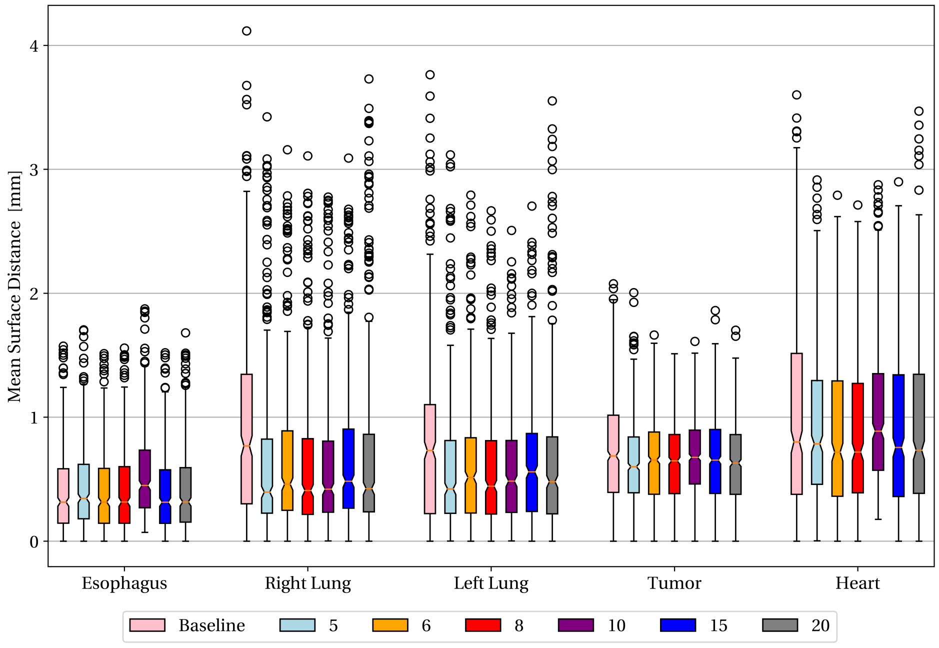

Figure 5.4: The mean surface distance in mm between the prediction and the fixed contour for various values of Γ. Calculated as described in Section 4.6.3. Values given in Table 5.4.

Table 5.5: The Hausdorff distance in mm between the prediction and the fixed contour for various values of Γ for the HardTanh of the VM model. Calculated as described in Section 4.6.3.

Γ value	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	8.23 ± 4.48	17.49 ± 9.27	16.79 ± 10.55	7.33 ± 3.94	13.62 ± 5.72
5	8.40 ± 4.00	16.98 ± 7.33	16.76 ± 9.80	7.45 ± 3.43	13.98 ± 4.79
6	8.23 ± 4.48	17.24 ± 8.70	16.37 ± 10.40	7.11 ± 3.65	13.61 ± 5.71
8	8.21 ± 4.46	17.75 ± 8.79	16.35 ± 10.40	7.10 ± 3.63	13.61 ± 5.70
10	8.82 ± 4.00	17.46 ± 7.59	16.82 ± 9.83	7.61 ± 3.29	14.29 ± 4.89
15	8.22 ± 4.47	17.14 ± 8.79	16.43 ± 10.39	7.12 ± 3.68	13.61 ± 5.72
20	8.16 ± 4.44	16.98 ± 8.42	16.24 ± 10.40	7.09 ± 3.58	13.60 ± 5.70

5

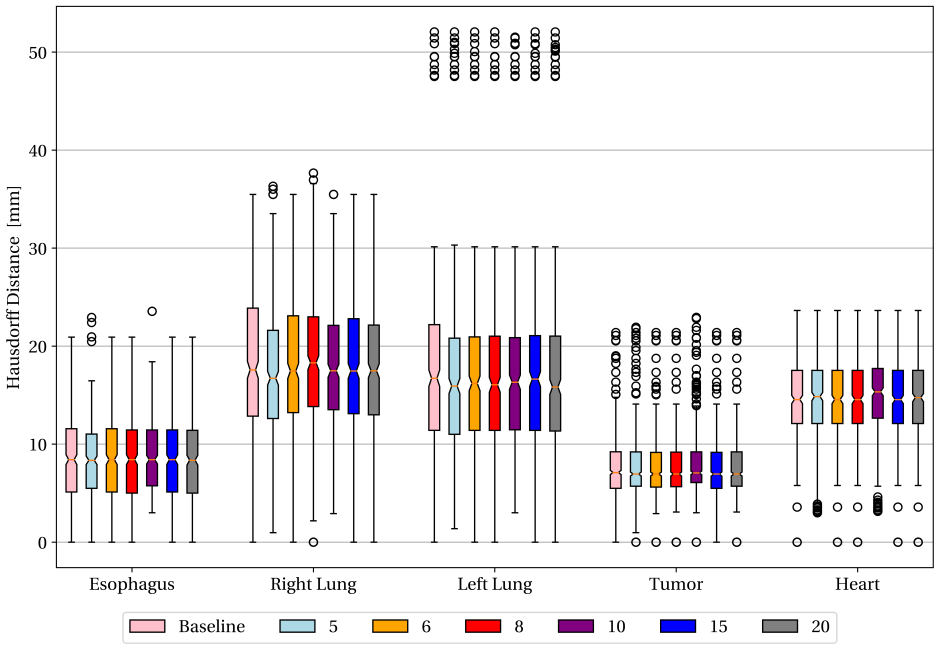

Figure 5.5: The Hausdorff distance in mm between the prediction and the fixed contour for various values of Γ. Calculated as described in Section 4.6.3. Values given in Table 5.5.

5.1.2. RESULTS VARYING LAMBDA VALUE WITH HARDTANH(10)

The value of λ was varied for the Voxelmorph model. The value of Γ for the HardTanh was set to 10. The mean absolute error in HU units for the three test sets are given in Table 5.6 and are displayed in Figure 5.6.

Table 5.6: Mean absolute error in HU units between the prediction and the fixed image for various values of λ for the VM model with HardTanh value Γ of 10. Calculated as described in Section 4.6.1.

Hardtan value	Dataset 1	Dataset 2	Dataset 3	Average
Baseline	19.50 ± 9.02	18.22 ± 7.77	23.41 ± 10.54	20.06 ± 9.26
1	16.80 ± 6.28	18.39 ± 4.97	20.02 ± 6.20	18.23 ± 5.95
10	16.58 ± 6.49	16.76 ± 4.90	19.89 ± 6.33	17.52 ± 6.08
100	17.85 ± 7.04	17.03 ± 5.27	21.03 ± 7.14	18.38 ± 6.67
500	16.84 ± 6.69	16.70 ± 4.88	19.46 ± 6.69	17.48 ± 6.20
1000	17.12 ± 6.62	18.13 ± 4.88	19.82 ± 6.04	18.20 ± 5.97
10000	17.66 ± 7.41	17.03 ± 5.75	20.76 ± 7.79	18.24 ± 7.12
50000	16.27 ± 6.14	17.04 ± 4.60	20.39 ± 6.28	17.47 ± 5.95
100000	17.55 ± 6.68	17.72 ± 4.69	20.83 ± 6.22	18.47 ± 6.06

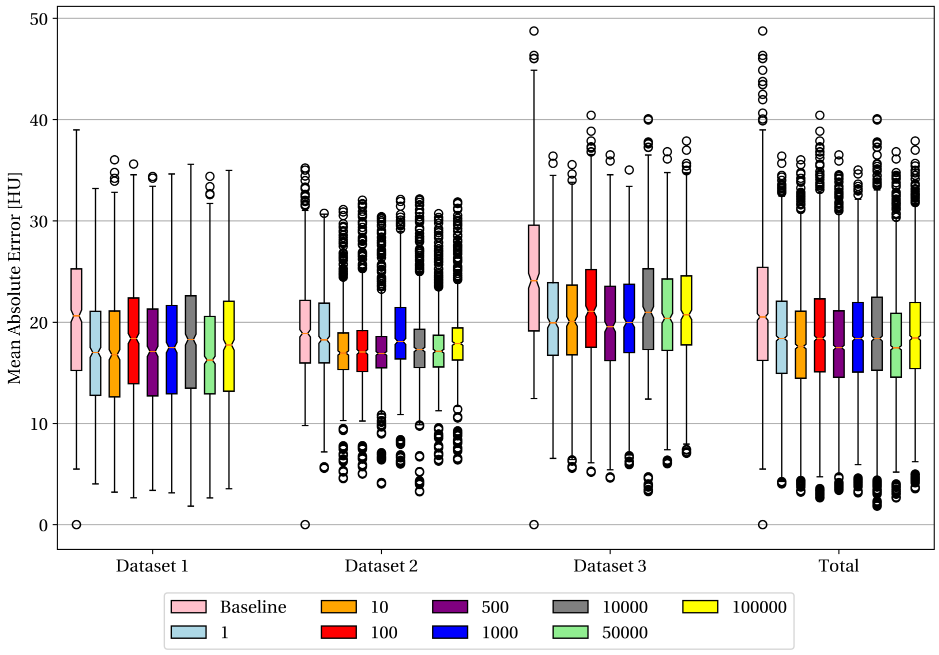

Figure 5.6: Calculated mean absolute error in HU units between the prediction and the fixed image for three test sets for various values of the gradient weight for the VM model with HardTanh value Γ set to 10. Values given in Table 5.6.

The ratio of voxels in the DVF with a negative Jacobian determinant for various values of λ are given in Table 5.7 and are displayed in Figure 5.7.

Table 5.7: Ratio of voxels with a negative Jacobian determinant for various values of λ for the VM model with HardTanh value Γ of 10. Calculated as described in Section 4.6.2.

Hardtan value	Dataset 1	Dataset 2	Dataset 3	Average
1	$6.904 \cdot 10^{-5} \pm 1.245 \cdot 10^{-4}$	$1.120 \cdot 10^{-4} \pm 2.048 \cdot 10^{-4}$	$3.128 \cdot 10^{-5} \pm 5.718 \cdot 10^{-5}$	$7.491 \cdot 10^{-5} \pm 1.518 \cdot 10^{-4}$
10	$4.739 \cdot 10^{-4} \pm 6.863 \cdot 10^{-4}$	$3.583 \cdot 10^{-4} \pm 5.748 \cdot 10^{-4}$	$1.972 \cdot 10^{-4} \pm 2.234 \cdot 10^{-4}$	$3.585 \cdot 10^{-4} \pm 5.658 \cdot 10^{-4}$
100	$2.155 \cdot 10^{-6} \pm 1.211 \cdot 10^{-5}$	$2.001 \cdot 10^{-6} \pm 1.428 \cdot 10^{-5}$	$1.256 \cdot 10^{-5} \pm 3.308 \cdot 10^{-5}$	$4.836 \cdot 10^{-6} \pm 2.094 \cdot 10^{-5}$
500	$1.017 \cdot 10^{-4} \pm 2.214 \cdot 10^{-4}$	$3.784 \cdot 10^{-5} \pm 9.720 \cdot 10^{-5}$	$6.514 \cdot 10^{-5} \pm 1.270 \cdot 10^{-4}$	$6.854 \cdot 10^{-5} \pm 1.629 \cdot 10^{-4}$
1000	$3.181 \cdot 10^{-5} \pm 6.559 \cdot 10^{-5}$	$7.100 \cdot 10^{-5} \pm 1.485 \cdot 10^{-4}$	$4.660 \cdot 10^{-5} \pm 9.262 \cdot 10^{-5}$	$5.014 \cdot 10^{-5} \pm 1.107 \cdot 10^{-4}$
10000	$2.469 \cdot 10^{-6} \pm 1.626 \cdot 10^{-5}$	$6.894 \cdot 10^{-7} \pm 3.580 \cdot 10^{-6}$	$2.270 \cdot 10^{-7} \pm 1.252 \cdot 10^{-6}$	$1.223 \cdot 10^{-6} \pm 1.017 \cdot 10^{-5}$
50000	$1.575 \cdot 10^{-7} \pm 9.833 \cdot 10^{-7}$	$5.601 \cdot 10^{-6} \pm 2.273 \cdot 10^{-5}$	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$1.878 \cdot 10^{-6} \pm 1.318 \cdot 10^{-5}$
100000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000

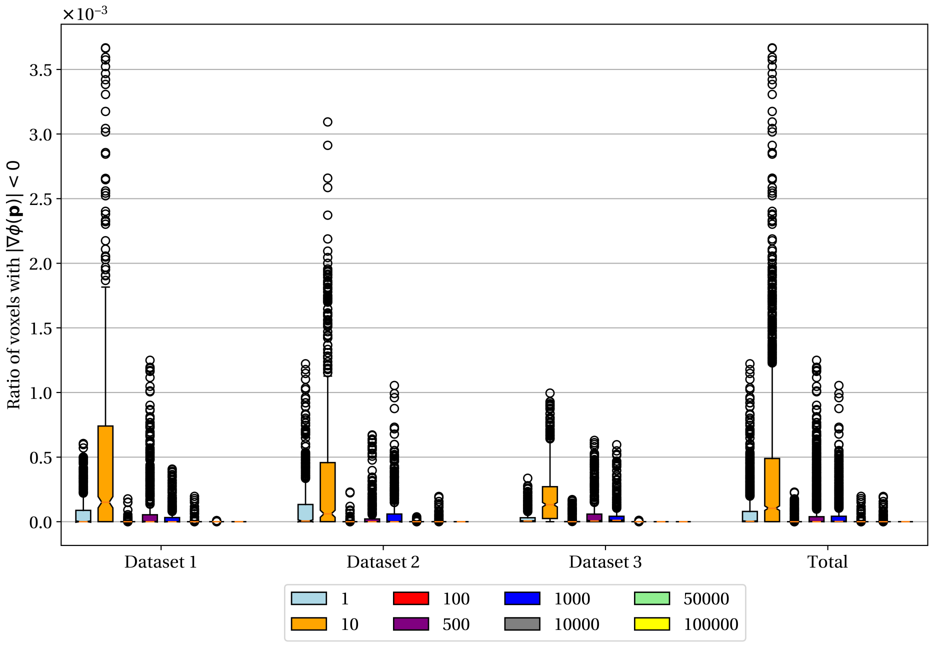

Figure 5.7: Ratio of voxels with a negative Jacobian determinant for various values of λ for the VM model with HardTanh value Γ set to 10. Calculated as described in Section 4.6.2. Values given in Table 5.7.

The calculated dice scores are given in Table 5.8 and are depicted in Figure 5.8. The calculated mean surface distances in mm are given in Table 5.9 and displayed in Figure 5.9. The Hausdorff results in mm are given in Table 5.10 and displayed in Figure 5.10.

Table 5.8: The Dice score between the prediction and the fixed contour for various values of λ for the VM model with HardTanh value Γ of 10. Calculated as described in Section 4.6.3.

λ value	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	0.89 ± 0.08	0.83 ± 0.12	0.83 ± 0.12	0.79 ± 0.13	0.76 ± 0.17
1	0.90 ± 0.08	0.86 ± 0.09	0.85 ± 0.10	0.82 ± 0.11	0.78 ± 0.16
10	0.90 ± 0.08	0.88 ± 0.08	0.88 ± 0.09	0.83 ± 0.10	0.79 ± 0.15
100	0.87 ± 0.07	0.88 ± 0.08	0.87 ± 0.09	0.81 ± 0.10	0.74 ± 0.13
500	0.90 ± 0.08	0.87 ± 0.09	0.86 ± 0.10	0.82 ± 0.11	0.78 ± 0.15
1000	0.90 ± 0.08	0.88 ± 0.09	0.87 ± 0.09	0.82 ± 0.10	0.78 ± 0.15
10000	0.90 ± 0.08	0.85 ± 0.11	0.84 ± 0.11	0.81 ± 0.12	0.77 ± 0.16
50000	0.90 ± 0.08	0.86 ± 0.10	0.85 ± 0.11	0.81 ± 0.11	0.77 ± 0.16
100000	0.90 ± 0.08	0.85 ± 0.10	0.85 ± 0.11	0.81 ± 0.11	0.78 ± 0.16

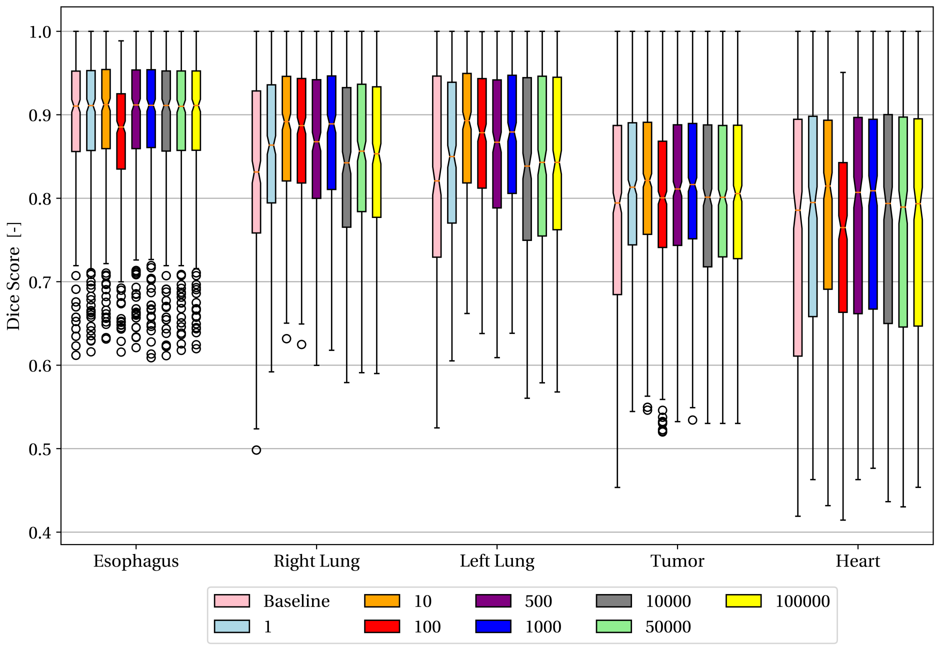

Figure 5.8: The Dice score between the prediction and the fixed contour for various values of λ for the VM model with HardTanh value Γ set to 10. Calculated as described in Section 4.6.3. Values are given in Table 5.8.

Table 5.9: Mean surface distance in mm the prediction and the fixed contour for various values of λ for the VM model with HardTanh value Γ of 10. Calculated as described in Section 4.6.3.

λ value	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	0.41 ± 0.34	0.92 ± 0.80	0.80 ± 0.73	0.71 ± 0.46	0.98 ± 0.79
1	0.40 ± 0.34	0.72 ± 0.69	0.66 ± 0.61	0.61 ± 0.36	0.88 ± 0.67
10	0.40 ± 0.34	0.60 ± 0.66	0.56 ± 0.55	0.60 ± 0.35	0.85 ± 0.64
100	0.55 ± 0.35	0.61 ± 0.57	0.56 ± 0.45	0.67 ± 0.34	1.03 ± 0.61
500	0.40 ± 0.34	0.66 ± 0.61	0.60 ± 0.50	0.61 ± 0.35	0.88 ± 0.66
1000	0.40 ± 0.34	0.59 ± 0.57	0.55 ± 0.48	0.61 ± 0.35	0.86 ± 0.64
10000	0.41 ± 0.35	0.82 ± 0.78	0.71 ± 0.66	0.65 ± 0.40	0.90 ± 0.71
50000	0.41 ± 0.34	0.73 ± 0.70	0.68 ± 0.63	0.64 ± 0.38	0.90 ± 0.70
100000	0.40 ± 0.33	0.73 ± 0.68	0.66 ± 0.58	0.63 ± 0.37	0.88 ± 0.66

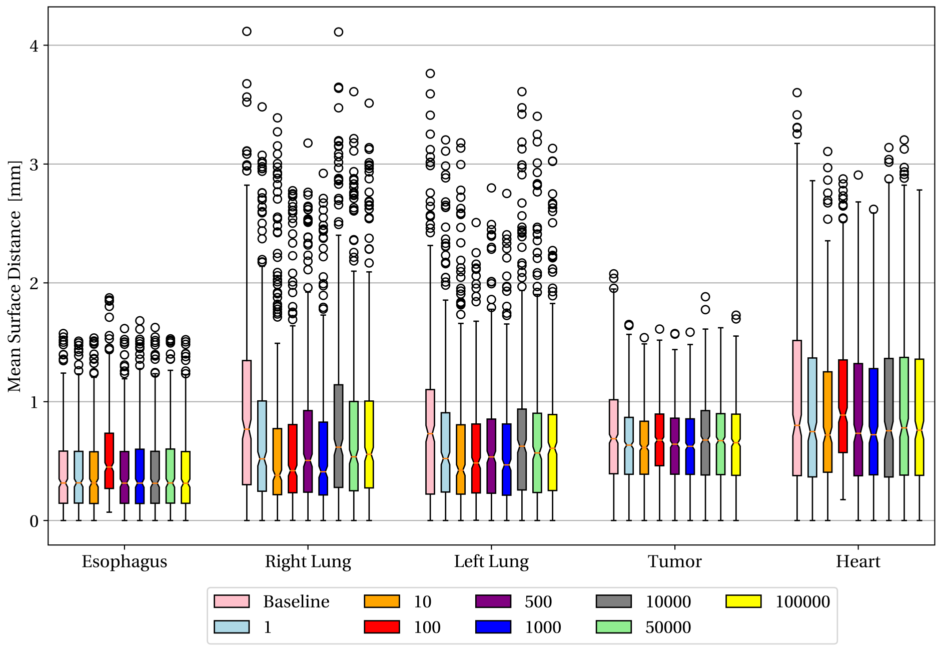

Figure 5.9: The mean surface distance in mm between the prediction and the fixed contour for various values of λ for the VM model with HardTanh value Γ set to 10. Calculated as described in Section 4.6.3. Values given in Table 5.9.

Table 5.10: The Hausdorff distance in mm between the prediction and the fixed contour for various values of λ for the VM model with HardTanh value Γ of 10. Calculated as described in Section 4.6.3.

λ value	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	8.23 ± 4.48	17.49 ± 9.27	16.79 ± 10.55	7.33 ± 3.94	13.62 ± 5.72
1	8.23 ± 4.48	17.37 ± 8.93	16.48 ± 10.42	7.19 ± 3.73	13.60 ± 5.70
10	8.19 ± 4.44	16.95 ± 8.29	16.22 ± 10.35	7.11 ± 3.62	13.60 ± 5.69
100	8.82 ± 4.00	17.46 ± 7.59	16.82 ± 9.83	7.61 ± 3.29	14.29 ± 4.89
500	8.22 ± 4.47	17.84 ± 9.01	16.63 ± 10.44	7.12 ± 3.68	13.60 ± 5.71
1000	8.21 ± 4.47	17.04 ± 8.11	16.32 ± 10.32	7.08 ± 3.62	13.63 ± 5.65
10000	8.23 ± 4.47	17.40 ± 9.17	16.62 ± 10.49	7.17 ± 3.71	13.62 ± 5.72
50000	8.21 ± 4.46	17.05 ± 8.87	16.43 ± 10.44	7.13 ± 3.66	13.61 ± 5.71
100000	8.22 ± 4.47	17.27 ± 9.09	16.58 ± 10.45	7.12 ± 3.70	13.60 ± 5.71

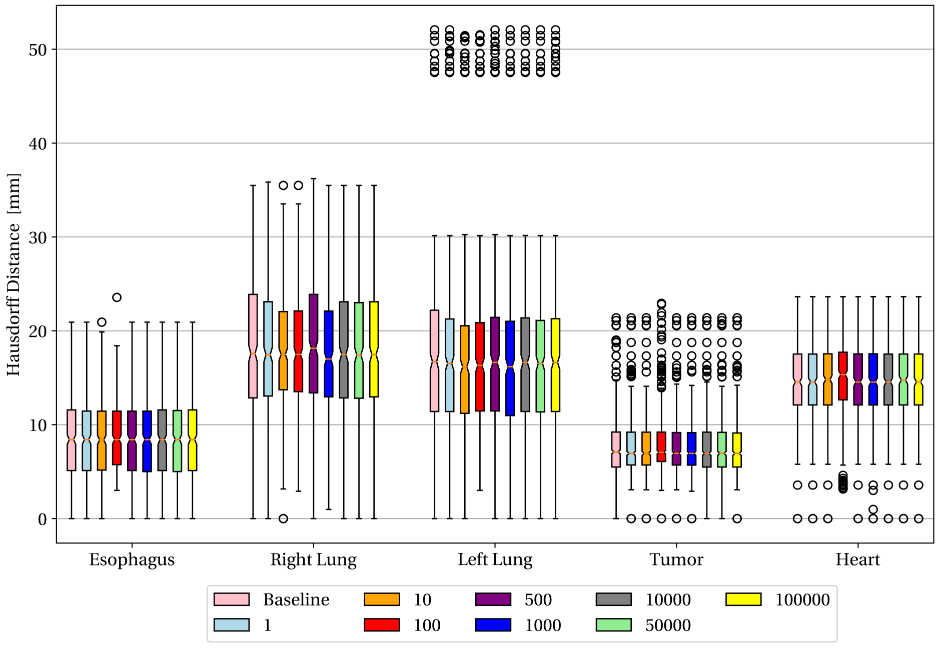

Figure 5.10: The Hausdorff distance in mm between the prediction and the fixed contour for various values of λ for the VM model with HardTanh value Γ set to 10. Calculated as described in Section 4.6.3. Values given in Table 5.10.

5.2. Laplacian Pyramid Model

5.2.1. Vary layers and filters for coarsest level

The number of CIRM blocks and filters of the coarsest grid size varied, the other two grid sizes used the original implementation of 5 CIRM blocks with 32 filters. The mean absolute error in HU units between the predicted and the fixed images was calculated as described in Section 4.6.1. The results are given in Table 5.11 and are depicted in Figure 5.11.

Table 5.11: Mean absolute error in HU units for various number of CIRM blocks and filters for the coarses layer. Other layers have 5 CIRM blocks with 32 filters. Calculated as described in Section 4.6.1.

CIRM blocks	Filters	Dataset 1	Dataset 2	Dataset 3	Average
Baseline		19.50 ± 9.02	18.22 ± 7.77	23.41 ± 10.54	20.06 ± 9.26
9	80	13.99 ± 6.38	13.94 ± 4.82	15.64 ± 5.20	14.41 ± 5.59
7	80	13.83 ± 6.37	13.76 ± 5.00	15.29 ± 5.34	14.19 ± 5.66
5	80	13.83 ± 6.40	13.88 ± 4.97	15.66 ± 5.58	14.33 ± 5.75
9	48	13.83 ± 6.36	13.81 ± 4.99	15.14 ± 5.31	14.16 ± 5.65
7	48	13.92 ± 6.66	13.95 ± 4.73	15.20 ± 5.43	14.27 ± 5.71
5	48	13.80 ± 6.27	14.10 ± 4.76	15.37 ± 5.17	14.32 ± 5.50
9	32	13.59 ± 6.44	13.92 ± 5.06	15.10 ± 5.41	14.11 ± 5.73
7	32	13.65 ± 6.38	13.39 ± 4.96	14.95 ± 5.20	13.90 ± 5.62
5	32	12.75 ± 5.60	13.50 ± 4.72	15.35 ± 5.36	13.71 ± 5.33

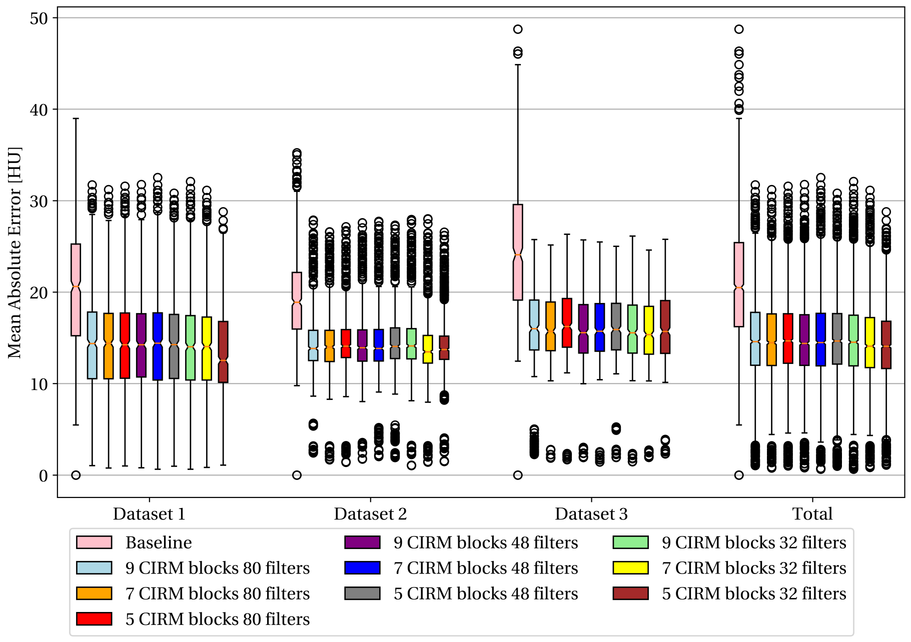

Figure 5.11: Calculated mean absolute error in HU units for the LAP model with varying number of CIRM blocks and filters in the coarsest level. Calculated as described in Section 4.6.1. Values given in Table 5.11.

The ratio of voxels where the DVF has a negative Jacobian determinant is calculated as described in Section 4.6.2. The results are given in Table 5.12.

Table 5.12: Ratio of voxels with a negative Jacobian determinant, for various number of CIRM blocks and filters for the coarsest layer of the LAP model. Other layers have 5 CIRM blocks with 32 filters. Calculated as described in Section 4.6.2.

CIRM blocks	Filters	Dataset 1	Dataset 2
9	80	$8.719 \cdot 10^{-8} \pm 3.294 \cdot 10^{-7}$	$5.338 \cdot 10^{-7} \pm 1.382 \cdot 10^{-6}$
7	80	$1.038 \cdot 10^{-7} \pm 3.219 \cdot 10^{-7}$	$6.673 \cdot 10^{-7} \pm 1.473 \cdot 10^{-6}$
5	80	$5.886 \cdot 10^{-8} \pm 2.436 \cdot 10^{-7}$	$4.455 \cdot 10^{-7} \pm 1.250 \cdot 10^{-6}$
9	48	$1.458 \cdot 10^{-7} \pm 4.705 \cdot 10^{-7}$	$1.745 \cdot 10^{-6} \pm 4.606 \cdot 10^{-6}$
7	48	$3.815 \cdot 10^{-8} \pm 2.508 \cdot 10^{-7}$	$1.945 \cdot 10^{-7} \pm 5.314 \cdot 10^{-7}$
5	48	$1.054 \cdot 10^{-7} \pm 4.697 \cdot 10^{-7}$	$9.817 \cdot 10^{-7} \pm 2.730 \cdot 10^{-6}$
9	32	$3.079 \cdot 10^{-8} \pm 1.263 \cdot 10^{-7}$	$4.747 \cdot 10^{-7} \pm 1.118 \cdot 10^{-6}$
7	32	$2.880 \cdot 10^{-7} \pm 1.051 \cdot 10^{-6}$	$1.648 \cdot 10^{-6} \pm 4.072 \cdot 10^{-6}$
5	32	$4.905 \cdot 10^{-9} \pm 3.643 \cdot 10^{-8}$	$5.760 \cdot 10^{-7} \pm 1.440 \cdot 10^{-6}$

CIRM blocks	Filters	Dataset 3	Average
9	80	$6.275 \cdot 10^{-7} \pm 1.143 \cdot 10^{-6}$	$3.939 \cdot 10^{-7} \pm 1.069 \cdot 10^{-6}$
7	80	$7.149 \cdot 10^{-7} \pm 2.584 \cdot 10^{-6}$	$4.722 \cdot 10^{-7} \pm 1.635 \cdot 10^{-6}$
5	80	$9.800 \cdot 10^{-7} \pm 3.072 \cdot 10^{-6}$	$4.437 \cdot 10^{-7} \pm 1.792 \cdot 10^{-6}$
9	48	$1.777 \cdot 10^{-6} \pm 6.459 \cdot 10^{-6}$	$1.164 \cdot 10^{-6} \pm 4.414 \cdot 10^{-6}$
7	48	$7.614 \cdot 10^{-7} \pm 4.263 \cdot 10^{-6}$	$2.861 \cdot 10^{-7} \pm 2.235 \cdot 10^{-6}$
5	48	$7.214 \cdot 10^{-7} \pm 2.636 \cdot 10^{-6}$	$5.904 \cdot 10^{-7} \pm 2.192 \cdot 10^{-6}$
9	32	$3.971 \cdot 10^{-7} \pm 1.094 \cdot 10^{-6}$	$2.907 \cdot 10^{-7} \pm 9.068 \cdot 10^{-7}$
7	32	$1.197 \cdot 10^{-6} \pm 3.446 \cdot 10^{-6}$	$1.028 \cdot 10^{-6} \pm 3.161 \cdot 10^{-6}$
5	32	$2.518 \cdot 10^{-7} \pm 1.241 \cdot 10^{-6}$	$2.803 \cdot 10^{-7} \pm 1.109 \cdot 10^{-6}$

For four patients from dataset 1 contours metrics were calculated. The results for dice scores are given in Table 5.13 and are depicted in Figure 5.12.

Table 5.13: Calculated Dice score for various organs for various number of CIRM blocks and filters for the coarsest layer of the LAP model. Other layers have 5 CIRM blocks with 32 filters. Calculated as described in Section 4.6.3.

CIRM blocks	Filters	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline		0.89 ± 0.08	0.83 ± 0.12	0.83 ± 0.12	0.79 ± 0.13	0.76 ± 0.17
9	80	0.90 ± 0.06	0.92 ± 0.07	0.91 ± 0.08	0.84 ± 0.09	0.78 ± 0.13
7	80	0.91 ± 0.06	0.91 ± 0.07	0.90 ± 0.08	0.84 ± 0.09	0.78 ± 0.13
5	80	0.91 ± 0.06	0.91 ± 0.07	0.90 ± 0.08	0.84 ± 0.09	0.78 ± 0.13
9	48	0.91 ± 0.06	0.91 ± 0.07	0.90 ± 0.08	0.84 ± 0.09	0.78 ± 0.13
7	48	0.91 ± 0.06	0.91 ± 0.07	0.90 ± 0.08	0.85 ± 0.09	0.78 ± 0.13
5	48	0.91 ± 0.06	0.91 ± 0.07	0.91 ± 0.08	0.84 ± 0.09	0.78 ± 0.12
9	32	0.91 ± 0.06	0.91 ± 0.07	0.90 ± 0.08	0.85 ± 0.09	0.78 ± 0.13
7	32	0.91 ± 0.06	0.92 ± 0.06	0.91 ± 0.08	0.84 ± 0.09	0.78 ± 0.13
5	32	0.91 ± 0.06	0.91 ± 0.07	0.91 ± 0.08	0.84 ± 0.09	0.77 ± 0.12

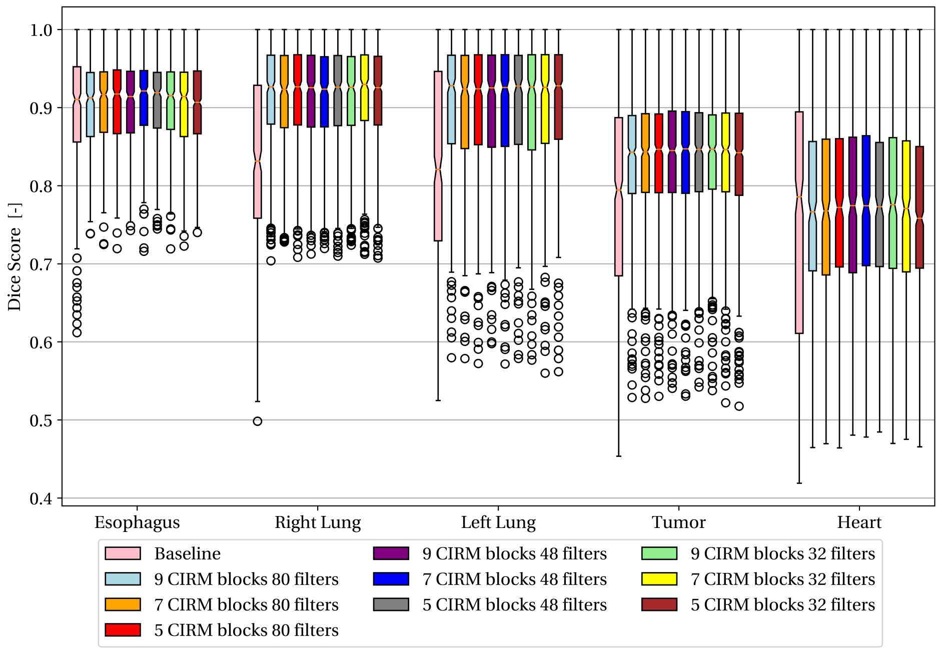

Figure 5.12: Calculated dice score for organs from four patients from dataset 1 for LAP model with varying number of CIRM blocks and filters in the coarsest level for the LAP model. Calculated as described in Section 4.6.3. Values given in Table 5.13.

The results for the mean surface distances in mm are given in Table 5.14 and displayed in Figure 5.13.

Table 5.14: Calculated mean surface distance in mm for various organs for various number of CIRM blocks and filters for the coarsest layer for the LAP model of the LAP model. Other layers have 5 CIRM blocks with 32 filters. Calculated as described in Section 4.6.3.

CIRM blocks	Filters	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline		0.41 ± 0.34	0.92 ± 0.80	0.80 ± 0.73	0.71 ± 0.46	0.98 ± 0.79
9	80	0.39 ± 0.25	0.41 ± 0.36	0.41 ± 0.32	0.57 ± 0.32	0.87 ± 0.48
7	80	0.38 ± 0.23	0.43 ± 0.38	0.42 ± 0.34	0.57 ± 0.32	0.87 ± 0.48
5	80	0.38 ± 0.24	0.42 ± 0.36	0.42 ± 0.33	0.56 ± 0.32	0.86 ± 0.48
9	48	0.38 ± 0.23	0.42 ± 0.37	0.41 ± 0.32	0.56 ± 0.32	0.86 ± 0.48
7	48	0.37 ± 0.22	0.43 ± 0.39	0.42 ± 0.34	0.56 ± 0.32	0.85 ± 0.47
5	48	0.37 ± 0.22	0.41 ± 0.37	0.41 ± 0.32	0.56 ± 0.32	0.86 ± 0.47
9	32	0.38 ± 0.23	0.41 ± 0.36	0.41 ± 0.33	0.56 ± 0.31	0.86 ± 0.47
7	32	0.39 ± 0.24	0.40 ± 0.34	0.41 ± 0.32	0.56 ± 0.32	0.88 ± 0.49
5	32	0.38 ± 0.22	0.42 ± 0.36	0.41 ± 0.31	0.56 ± 0.32	0.87 ± 0.46

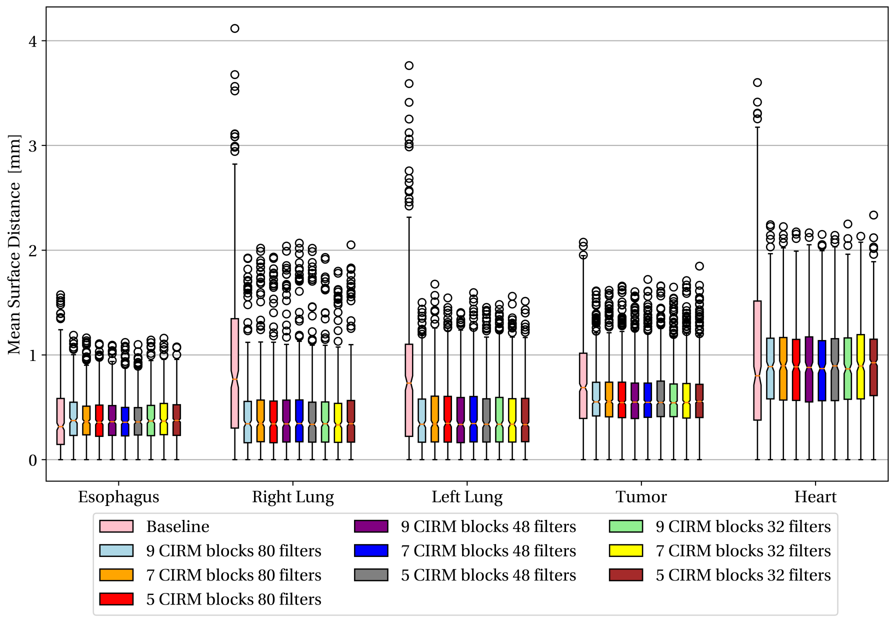

Figure 5.13: Calculated mean surface distances in mm for organs from four patients from dataset 1 for models with varying number of CIRM blocks and filters for the coarsest layer for the LAP model. Calculated as described in Section 4.6.3. Values given in Table 5.14.

The Hausdorff results in mm are given in Table 5.15 and displayed in Figure 5.14.

Table 5.15: Calculated Hausdorff distance in mm for various organs for various number of CIRM blocks and filters for the coarsest layer for the LAP model. Other layers have 5 CIRM blocks with 32 filters. Calculated as described in Section 4.6.3.

CIRM blocks	Filters	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline		8.23 ± 4.48	17.49 ± 9.27	16.79 ± 10.55	7.33 ± 3.94	13.62 ± 5.72
7	80	6.76 ± 3.35	15.71 ± 7.84	15.67 ± 10.03	7.19 ± 3.85	13.32 ± 5.48
9	80	6.70 ± 3.38	15.75 ± 7.97	15.73 ± 10.05	7.15 ± 3.80	13.38 ± 5.52
5	80	6.86 ± 3.45	15.57 ± 7.86	15.64 ± 10.00	7.19 ± 3.81	13.32 ± 5.53
9	48	6.66 ± 3.36	15.77 ± 8.03	15.73 ± 10.09	7.18 ± 3.79	13.34 ± 5.49
7	48	6.64 ± 3.33	15.79 ± 8.03	15.76 ± 10.07	7.15 ± 3.79	13.30 ± 5.50
5	48	6.72 ± 3.32	15.69 ± 7.86	15.74 ± 10.03	7.24 ± 3.85	13.33 ± 5.53
9	32	6.70 ± 3.29	15.79 ± 7.98	15.73 ± 10.08	7.13 ± 3.76	13.30 ± 5.48
7	32	6.73 ± 3.38	15.56 ± 7.79	15.69 ± 10.08	7.19 ± 3.85	13.38 ± 5.54
5	32	6.50 ± 3.12	15.83 ± 7.87	15.74 ± 9.89	7.07 ± 3.61	13.48 ± 5.24

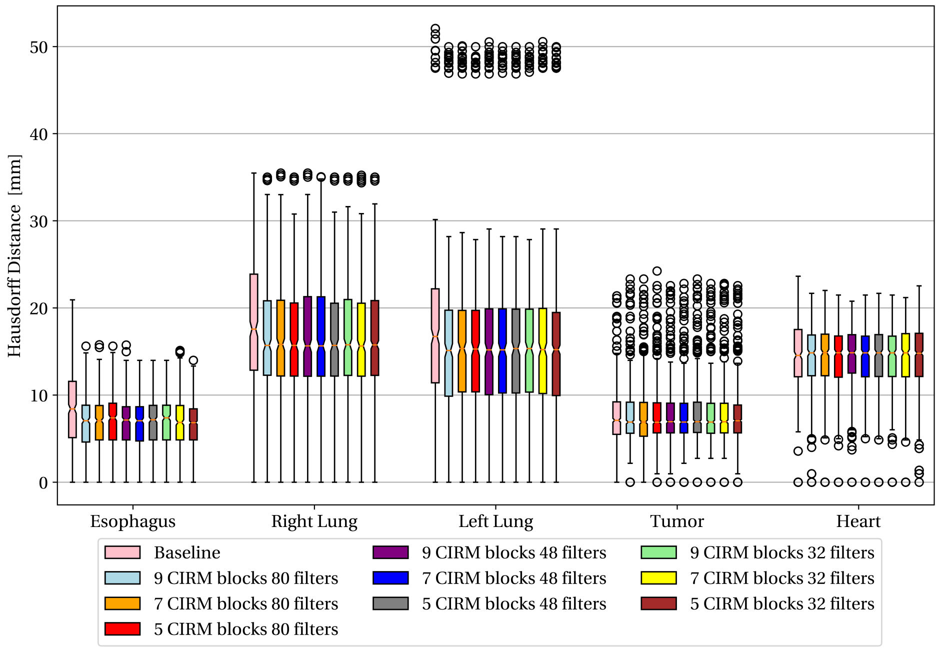

Figure 5.14: Calculated Hausdorff distance in mm for organs from four patients from dataset 1 for models with varying number of CIRM blocks and filters for the coarsest layer for the LAP model. Calculated as described in Section 4.6.3. Values given in Table 5.15.

5.2.2. VARY LAYERS AND FILTERS FOR ALL LEVELS

The number of CIRM blocks and filters was varied for all grid sizes of the LAP model. The mean absolute error between in HU units the predicted and the fixed images was calculated as described in Section 4.6.1. The results are given in Table 5.16 and are depicted in Figure 5.15.

Table 5.16: Mean absolute error in HU units for networks with various number of CIRM blocks and filters for the LAP model. Calculated as described in Section 4.6.1.

CIRM blocks	Filters	Dataset 1	Dataset 2	Dataset 3	Average
Baseline		19.50 ± 9.02	18.22 ± 7.77	23.41 ± 10.54	20.06 ± 9.26
7	80	18.99 ± 6.18	20.78 ± 4.93	22.91 ± 5.09	20.68 ± 5.67
5	80	16.44 ± 6.46	16.88 ± 5.15	19.90 ± 6.33	17.51 ± 6.14
9	48	18.65 ± 5.70	21.81 ± 5.44	23.21 ± 5.92	21.01 ± 5.97
7	48	19.13 ± 6.75	22.68 ± 5.80	22.31 ± 5.50	21.27 ± 6.31
5	48	17.43 ± 6.33	18.03 ± 4.78	19.67 ± 5.81	18.24 ± 5.73
9	32	16.54 ± 7.09	16.59 ± 5.69	18.61 ± 6.46	17.10 ± 6.50
7	32	13.30 ± 5.80	14.65 ± 4.81	15.96 ± 5.25	14.50 ± 5.41
5	32	12.75 ± 5.60	13.50 ± 4.72	15.35 ± 5.36	13.71 ± 5.33

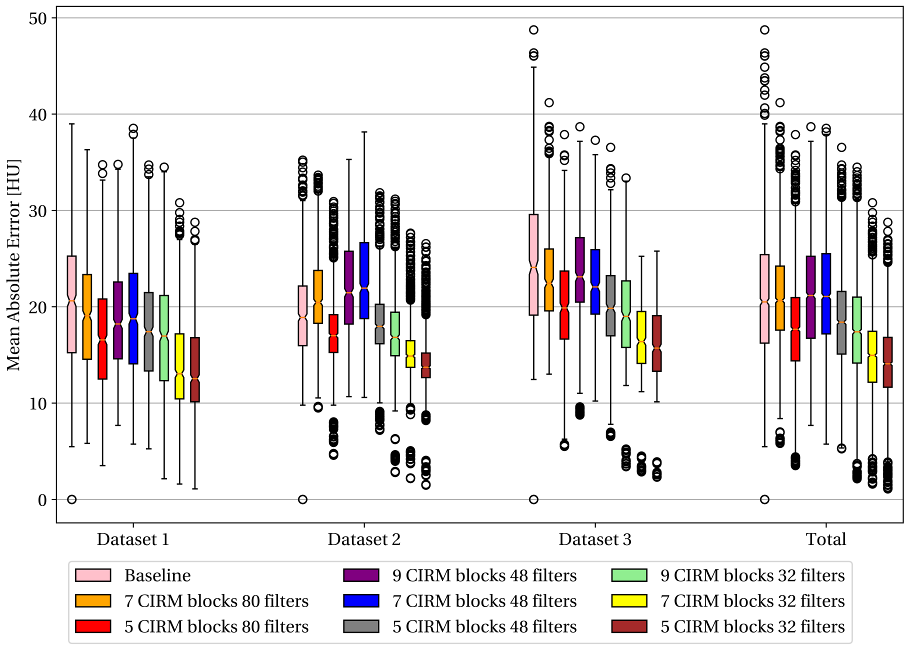

Figure 5.15: Calculated mean absolute error in HU units for three datasets for models with varying number of CIRM blocks and filters for all levels of the LAP model. Calculated as described in Section 4.6.1. Values given in Table 5.16.

The ratio of voxels where the DVF has a negative Jacobian determinant is calculated as described in Section 4.6.2. The results are given in Table 5.17.

Table 5.17: Ratio of voxels with a negative Jacobian determinant for various number of CIRM blocks and filters for all layers of the LAP model. Calculated as described in Section 4.6.2.

CIRM blocks	Filters	Dataset 1	Dataset 2
7	80	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$2.371 \cdot 10^{-6} \pm 8.201 \cdot 10^{-6}$
5	80	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$1.567 \cdot 10^{-6} \pm 2.988 \cdot 10^{-6}$
9	48	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$1.744 \cdot 10^{-8} \pm 1.282 \cdot 10^{-7}$
7	48	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$8.174 \cdot 10^{-8} \pm 4.964 \cdot 10^{-7}$
5	48	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$
9	32	$1.196 \cdot 10^{-5} \pm 3.847 \cdot 10^{-5}$	$7.666 \cdot 10^{-6} \pm 2.363 \cdot 10^{-5}$
7	32	$5.450 \cdot 10^{-8} \pm 2.103 \cdot 10^{-7}$	$1.564 \cdot 10^{-6} \pm 4.318 \cdot 10^{-6}$
5	32	$4.905 \cdot 10^{-9} \pm 3.643 \cdot 10^{-8}$	$5.760 \cdot 10^{-7} \pm 1.440 \cdot 10^{-6}$

CIRM blocks	Filters	Dataset 3	Average
7	80	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$8.735 \cdot 10^{-7} \pm 5.107 \cdot 10^{-6}$
5	80	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$5.773 \cdot 10^{-7} \pm 1.965 \cdot 10^{-6}$
9	48	$3.815 \cdot 10^{-10} \pm 8.521 \cdot 10^{-9}$	$6.525 \cdot 10^{-9} \pm 7.837 \cdot 10^{-8}$
7	48	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$3.012 \cdot 10^{-8} \pm 3.039 \cdot 10^{-7}$
5	48	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$	$0.000 \cdot 10^{0} \pm 0.000 \cdot 10^{0}$
9	32	$1.625 \cdot 10^{-5} \pm 7.999 \cdot 10^{-5}$	$1.151 \cdot 10^{-5} \pm 4.946 \cdot 10^{-5}$
7	32	$1.379 \cdot 10^{-6} \pm 2.927 \cdot 10^{-6}$	$9.591 \cdot 10^{-7} \pm 3.102 \cdot 10^{-6}$
5	32	$2.518 \cdot 10^{-7} \pm 1.241 \cdot 10^{-6}$	$2.803 \cdot 10^{-7} \pm 1.109 \cdot 10^{-6}$

The calculated dice scores are given in Table 5.18 and are depicted in Figure 5.16. The calculated mean surface distances in mm are given in Table 5.19 and displayed in Figure 5.17. The Hausdorff results in mm are given in Table 5.20 and displayed in Figure 5.18.

Table 5.18: Calculated Dice score for various organs for networks with various number of CIRM blocks and filters for the LAP model. Calculated as described in Section 4.6.3.

CIRM blocks	Filters	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline		0.89 ± 0.08	0.83 ± 0.12	0.83 ± 0.12	0.80 ± 0.14	0.76 ± 0.17
7	80	0.88 ± 0.08	0.88 ± 0.09	0.87 ± 0.09	0.84 ± 0.10	0.77 ± 0.15
5	80	0.90 ± 0.08	0.89 ± 0.08	0.89 ± 0.09	0.84 ± 0.10	0.77 ± 0.14
9	48	0.89 ± 0.08	0.88 ± 0.08	0.89 ± 0.08	0.84 ± 0.10	0.76 ± 0.14
7	48	0.89 ± 0.08	0.89 ± 0.08	0.89 ± 0.08	0.84 ± 0.09	0.75 ± 0.13
5	48	0.88 ± 0.08	0.89 ± 0.07	0.89 ± 0.08	0.85 ± 0.08	0.74 ± 0.14
9	32	0.90 ± 0.08	0.90 ± 0.08	0.90 ± 0.08	0.86 ± 0.09	0.78 ± 0.14
7	32	0.90 ± 0.06	0.91 ± 0.07	0.90 ± 0.08	0.85 ± 0.09	0.77 ± 0.12
5	32	0.91 ± 0.06	0.91 ± 0.07	0.90 ± 0.08	0.85 ± 0.09	0.77 ± 0.12

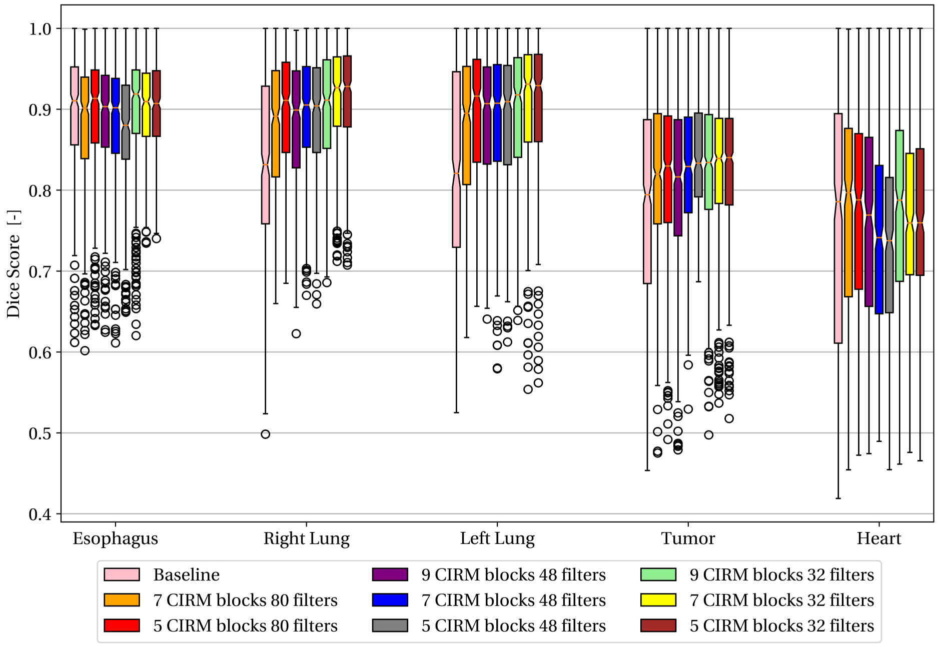

Figure 5.16: Calculated dice score for organs from four patients from dataset 1 for models with varying number of CIRM blocks and filters for all levels of the LAP model. Calculated as described in Section 4.6.3. Values given in Table 5.18.

Table 5.19: Calculated mean surface distance in mm for various organs for networks with various number of CIRM blocks and filters for the LAP model. Calculated as described in Section 4.6.3.

CIRM blocks	Filters	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline		0.41 ± 0.34	0.92 ± 0.80	0.80 ± 0.73	0.71 ± 0.46	0.98 ± 0.79
7	80	0.50 ± 0.26	0.61 ± 0.54	0.57 ± 0.49	0.62 ± 0.33	0.94 ± 0.58
5	80	0.39 ± 0.30	0.51 ± 0.46	0.47 ± 0.39	0.60 ± 0.35	0.87 ± 0.57
9	48	0.42 ± 0.27	0.56 ± 0.44	0.49 ± 0.34	0.63 ± 0.37	0.91 ± 0.50
7	48	0.45 ± 0.33	0.51 ± 0.42	0.47 ± 0.36	0.57 ± 0.32	0.97 ± 0.51
5	48	0.48 ± 0.28	0.50 ± 0.40	0.48 ± 0.38	0.55 ± 0.30	1.01 ± 0.52
9	32	0.39 ± 0.30	0.49 ± 0.43	0.44 ± 0.36	0.57 ± 0.32	0.86 ± 0.54
7	32	0.39 ± 0.23	0.40 ± 0.34	0.40 ± 0.31	0.58 ± 0.32	0.87 ± 0.47
5	32	0.38 ± 0.22	0.42 ± 0.36	0.41 ± 0.31	0.58 ± 0.32	0.87 ± 0.46

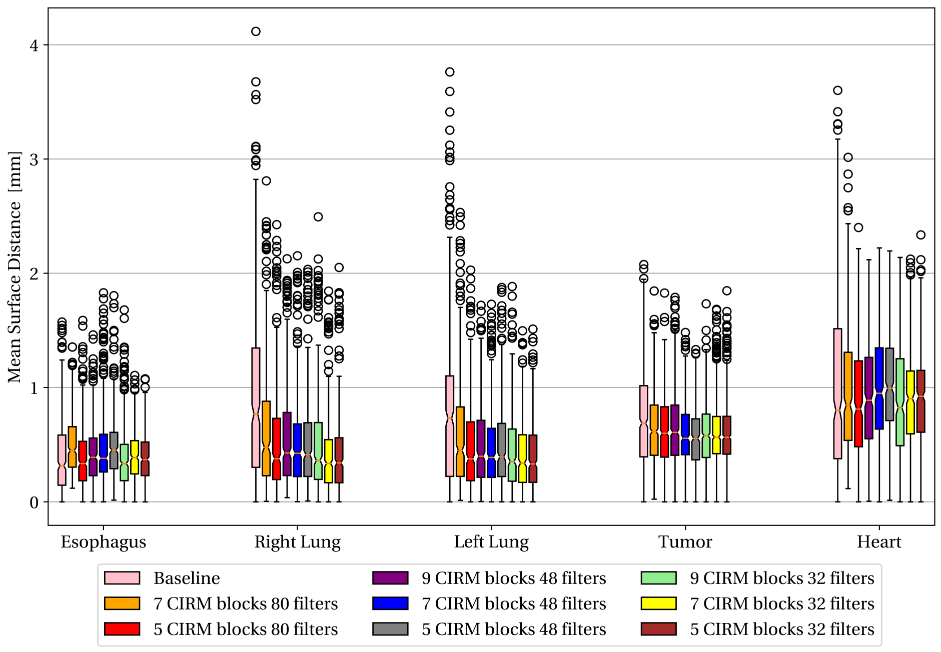

Figure 5.17: Calculated mean surface distance in mm for organs from four patients from dataset 1 for models with varying number of CIRM blocks and filters for all levels. Calculated as described in Section 4.6.3. Values given in Table 5.19.

Table 5.20: Calculated Hausdorff distance in mm for various organs for networks with various number of CIRM blocks and filters. Calculated as described in Section 4.6.3.

CIRM blocks	Filters	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline		8.23 ± 4.48	17.49 ± 9.27	16.79 ± 10.55	7.33 ± 3.94	13.62 ± 5.72
7	80	7.52 ± 3.40	17.30 ± 8.24	16.60 ± 9.93	7.36 ± 3.31	13.76 ± 4.91
5	80	7.22 ± 3.67	16.13 ± 8.18	16.09 ± 10.01	7.29 ± 3.74	13.53 ± 5.52
9	48	7.17 ± 3.12	16.55 ± 7.32	16.26 ± 9.58	7.45 ± 3.42	13.77 ± 4.70
7	48	7.60 ± 3.50	16.59 ± 7.95	16.13 ± 9.84	6.80 ± 2.93	13.88 ± 4.72
5	48	7.18 ± 3.25	16.25 ± 7.86	16.08 ± 9.92	6.79 ± 2.91	13.96 ± 4.88
9	32	7.02 ± 3.60	16.19 ± 8.16	15.81 ± 10.14	7.13 ± 3.72	13.40 ± 5.40
7	32	6.54 ± 3.27	15.78 ± 7.86	15.58 ± 9.93	7.28 ± 3.79	13.33 ± 5.47
5	32	6.48 ± 3.13	15.77 ± 7.91	15.69 ± 9.91	7.28 ± 3.84	13.43 ± 5.29

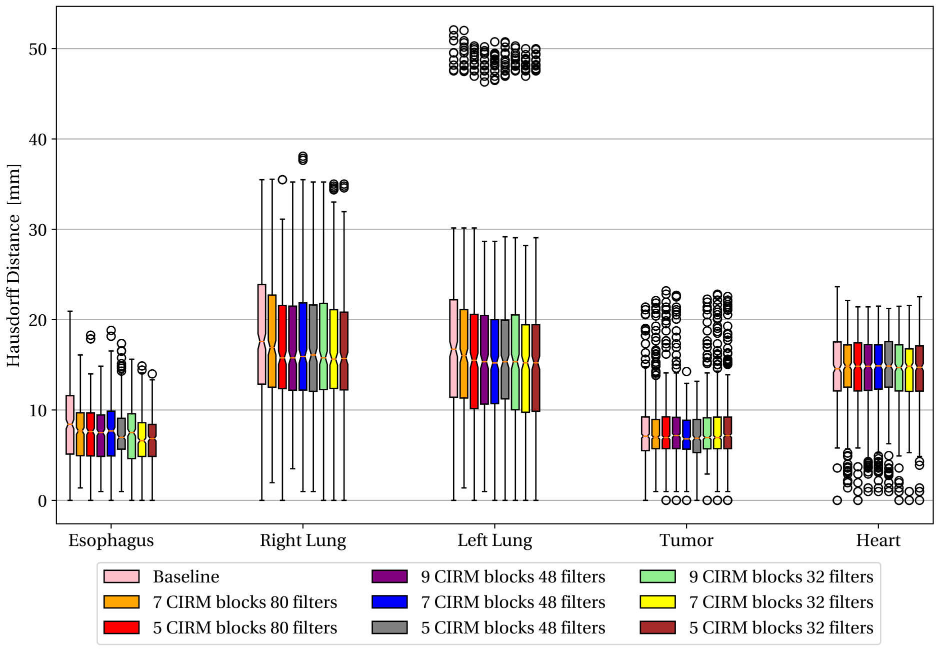

Figure 5.18: Calculated Hausdorff distance in mm for organs from four patients from dataset 1 for models with varying number of CIRM blocks and filters for all levels of the LAP model. Calculated as described in Section 4.6.3. Values given in Table 5.20.

5.2.3. 4 GRID SIZES

The third variation of the Laplacean model was adding a 4th grid size of 5 CIRM blocks with 32 filters.

The results for the mean absolute error in HU units and the Jacobian are given in Table 5.21.

Table 5.21: The calculated Mean absolute error and Ratio of voxels with a negative Jacobian determinant for LAP model with four grid sizes. Calculated as described in Sections 4.6.1 and 4.6.2.

	Mean Absolute Error	Jacobian
Dataset 1	14.15 ± 6.28	$9.610 \cdot 10^{-7} \pm 4.060 \cdot 10^{-6}$
Dataset 2	14.43 ± 4.68	$2.258 \cdot 10^{-6} \pm 7.559 \cdot 10^{-6}$
Dataset 3	16.31 ± 5.35	$1.890 \cdot 10^{-6} \pm 5.976 \cdot 10^{-6}$
Total	14.82 ± 5.56	$1.683 \cdot 10^{-6} \pm 6.070 \cdot 10^{-6}$

The contour metric results are given in Table 5.22.

Table 5.22: The calculated Mean absolute error and Ratio of voxels with a negative Jacobian determinant for LAP model with four grid sizes. Calculated as described in Section 4.6.3.

	Mean surface distance [mm]	Hausdorff distance [mm]	Dice Score
Esophagus	0.38 ± 0.24	6.82 ± 3.47	0.91 ± 0.06
Right Lung	0.42 ± 0.38	15.75 ± 7.96	0.91 ± 0.06
Left Lung	0.41 ± 0.32	15.62 ± 10.08	0.90 ± 0.08
Tumor	0.55 ± 0.31	7.13 ± 3.76	0.85 ± 0.09
Heart	0.84 ± 0.47	13.29 ± 5.53	0.78 ± 0.12

5.3. ELASTIX RESULTS

The calculated MAE and the ratio of voxels with a negative Jacobian determinant for dataset 1 and 2 are given in Table 5.23.

Table 5.23: The calculated mean absolute error and ratio of voxels with a negative Jacobian determinant for the Elastix deformations. Calculated as described in Sections 4.6.1 and 4.6.2.

	Mean Absolute Error [HU]	Jacobian
Dataset 1	15.50 ± 5.13	$5.440 \cdot 10^{-7} \pm 7.822 \cdot 10^{-7}$
Dataset 2	18.51 ± 3.97	$2.860 \cdot 10^{-7} \pm 6.460 \cdot 10^{-7}$

5.3.1. Comparison between models

The prediction from the VM, LAP and Elastix are compared. The compared models are the VM model with a Γ value of 15 and λ of 100. The LAP model consists of 5 CIRM blocks with 32 filters. The models are compared to Elastix with the parameters as described in Section 4.3. No image registration was performed for dataset 3 using Elastix. Also, no contours were evaluated using Elastix. Slices of the fixed, moving and predicted images for three patients predicted using the VM and LAP model are given in Figures 5.19 to 5.22. Additional images are given in Appendices A.1 and A.2.

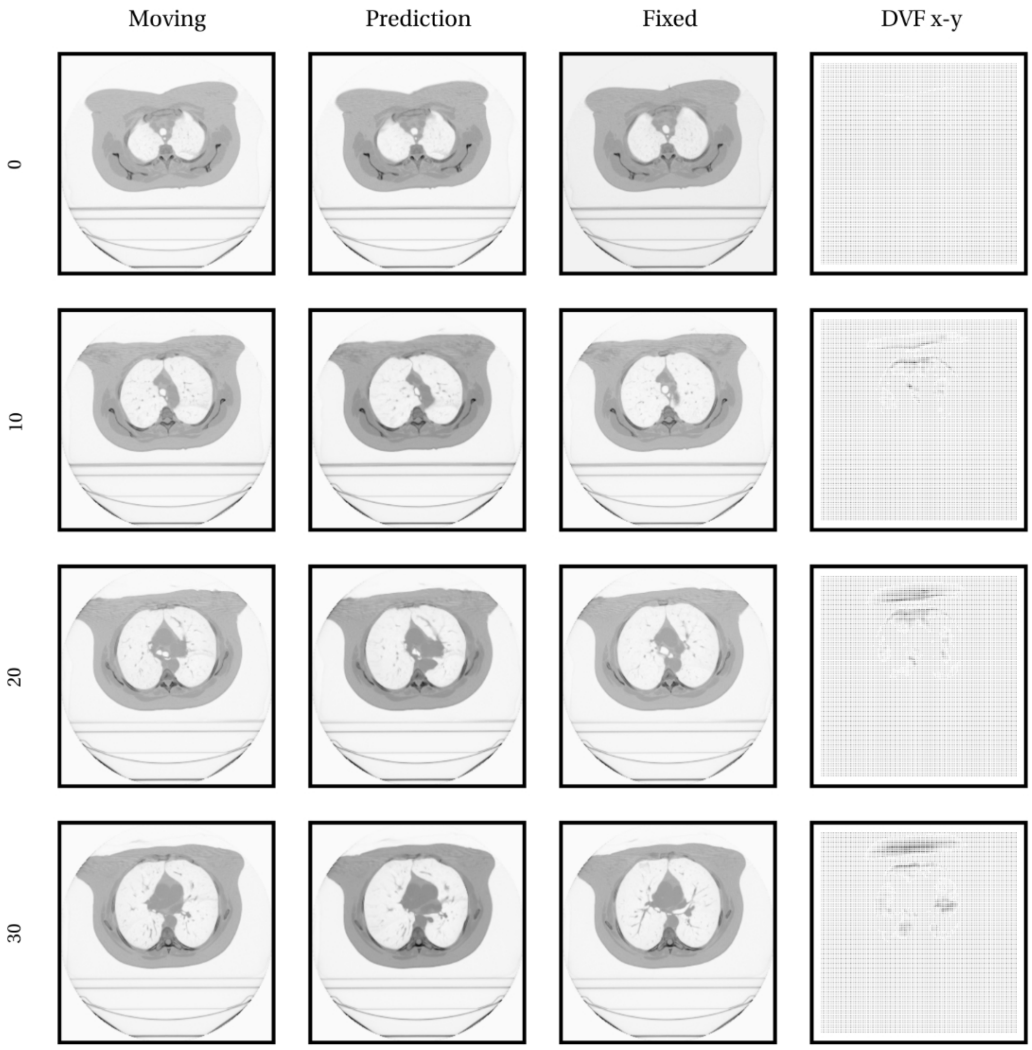

Figure 5.19: Overview of upper slices for the fixed, moving and predicted image of scan 132 from 0 to 50 phase. Prediction by the VM model with Γ set to 10 and λ is 100.

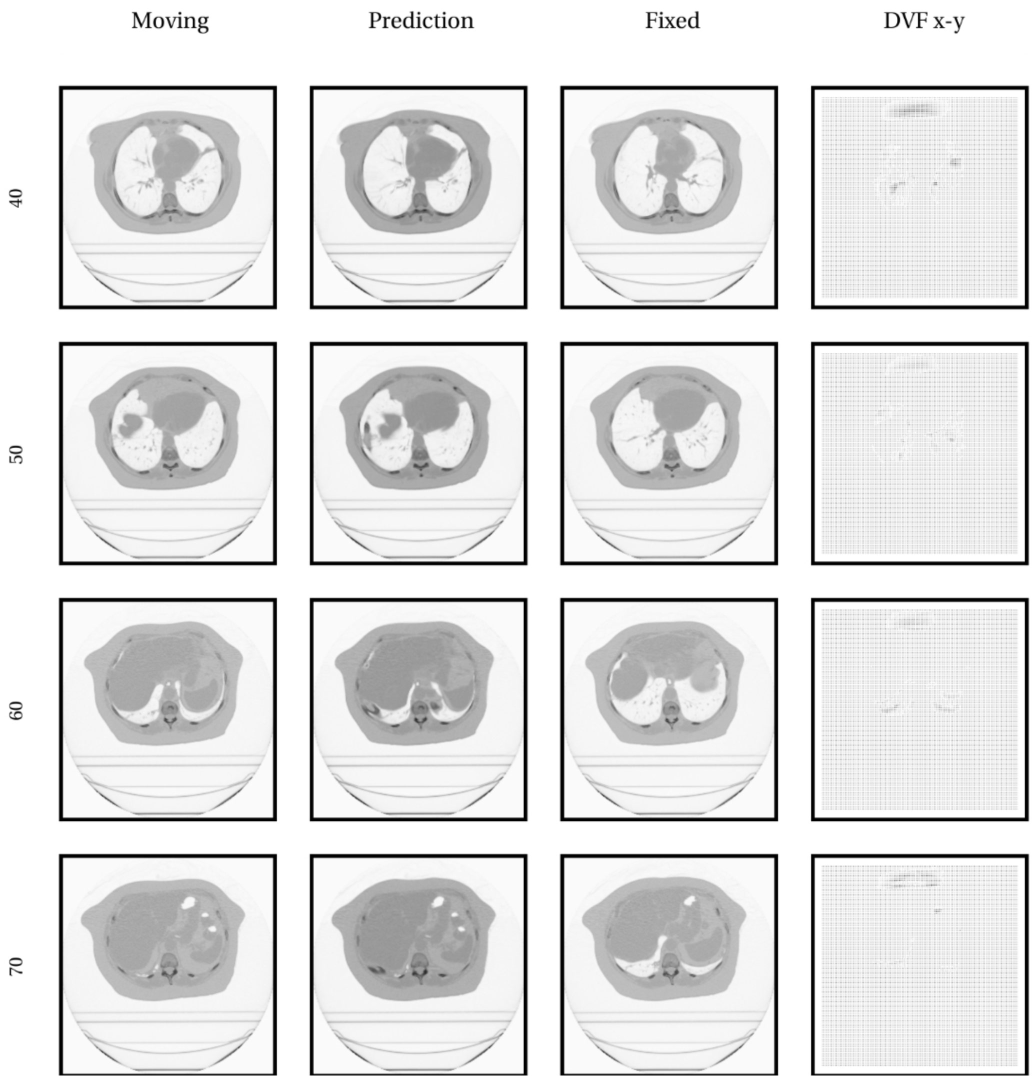

Figure 5.20: Overview of lower slices for the fixed, moving and predicted image of scan 132 from 0 to 50 phase. Prediction by the VM model with Γ set to 10 and λ is 100.

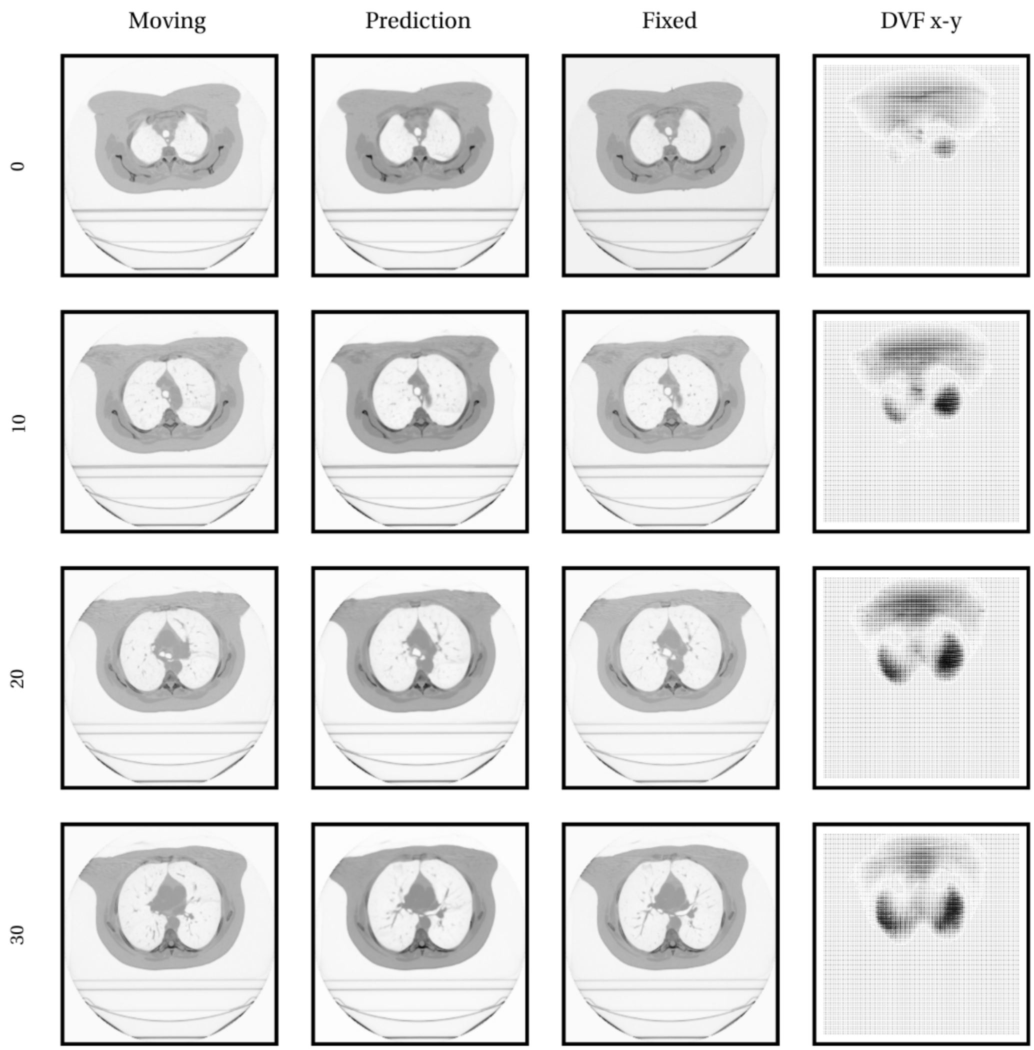

Figure 5.21: Overview of upper slices for the fixed, moving and predicted image of scan 132 from 0 to 50 phase. Prediction by the LAP model with 5 CIRM blocks of 32 filters.

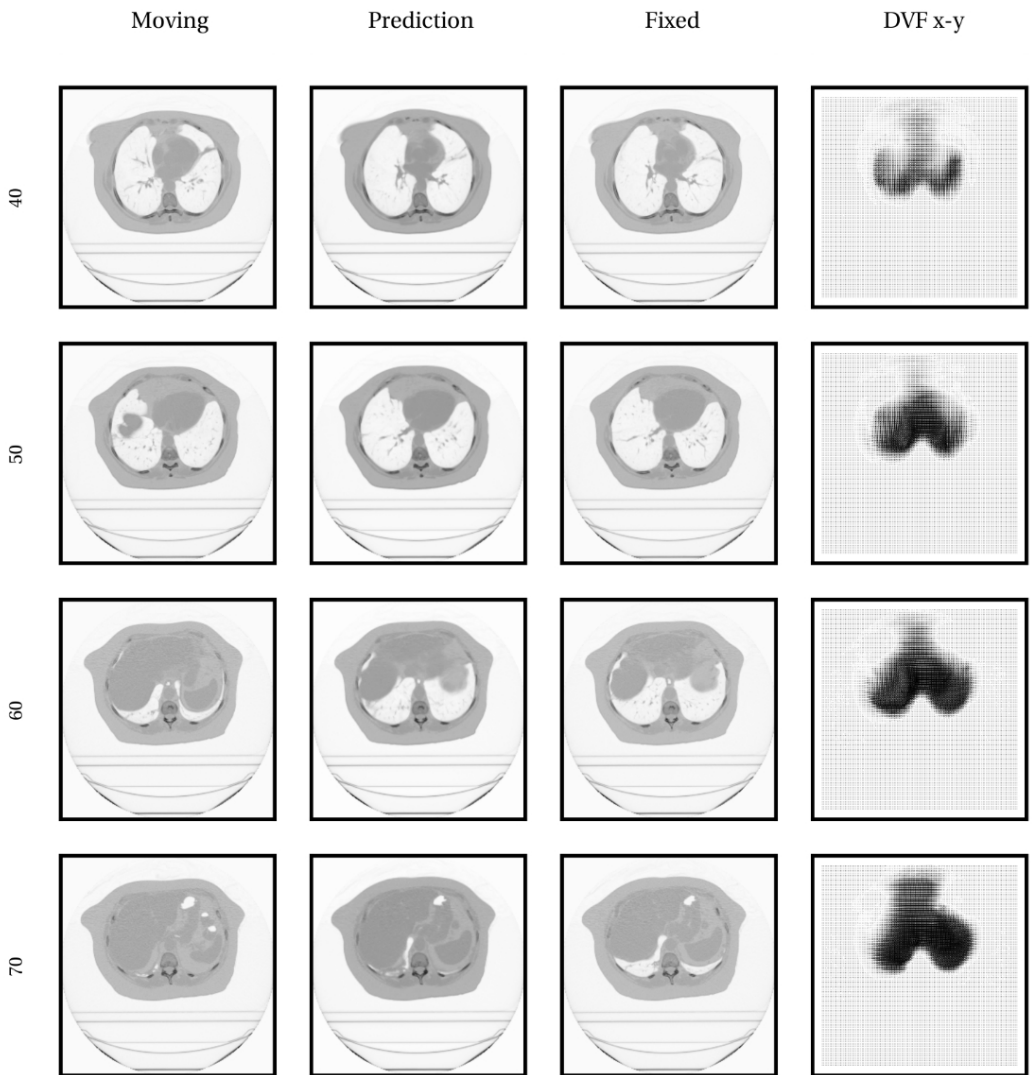

Figure 5.22: Overview of lower slices for the fixed, moving and predicted image of scan 132 from 0 to 50 phase. Prediction by the LAP model with 5 CIRM blocks of 32 filters.

The mean absolute error for the three methods are given in Table 5.24 and are displayed in Figure 5.23.

Table 5.24: Mean absolute error in HU units between the prediction and the fixed image for the registration methods. Calculated as described in section 4.6.1.

Method	Dataset 1	Dataset 2	Dataset 3	Average
Baseline	19.50 ± 9.02	18.22 ± 7.77	23.41 ± 10.54	20.06 ± 9.26
VoxelMorph	17.17 ± 6.63	16.29 ± 4.81	19.77 ± 6.09	17.53 ± 6.04
Laplacian model	12.75 ± 5.60	13.50 ± 4.72	15.35 ± 5.36	13.71 ± 5.33
Elastix	15.50 ± 5.13	18.51 ± 3.97		

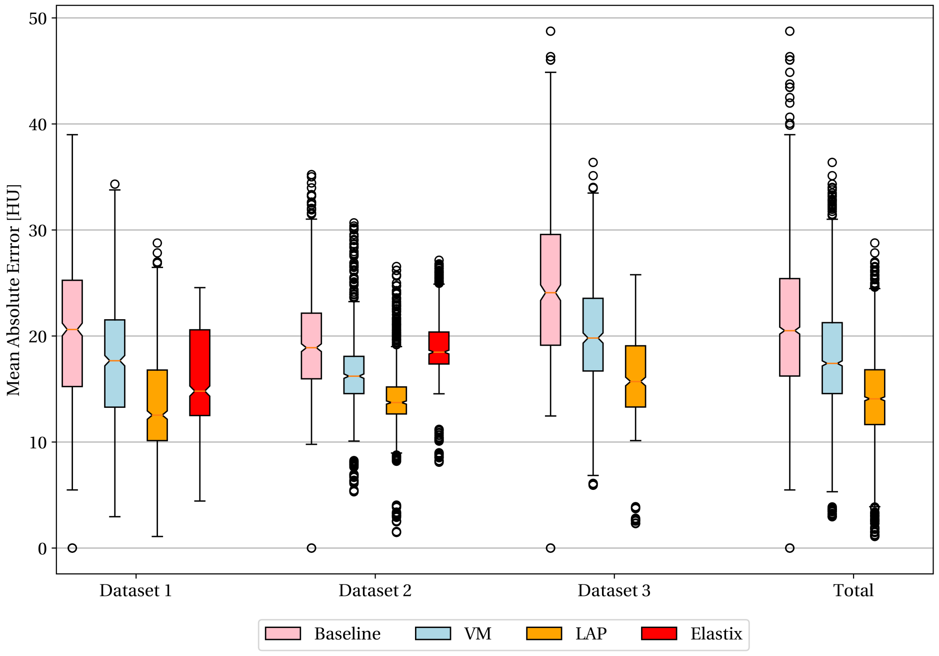

Figure 5.23: Calculated mean absolute error in HU units between the prediction and the fixed image for three datasets for three registration methods. Calculated as described in Section 4.6.1. Values given in Table 5.24.

The ratio of voxels in the DVF with a negative Jacobian determinant for three registration methods is given in Table 5.25.

Table 5.25: Ratio of voxels with a negative Jacobian determinant for the registration methods. Calculated as described in Section 4.6.2.

Method	Dataset 1	Dataset 2	Dataset 3	Average
VM	$1.942 \cdot 10^{-5} \pm 4.110 \cdot 10^{-5}$	$4.065 \cdot 10^{-5} \pm 9.267 \cdot 10^{-5}$	$1.039 \cdot 10^{-5} \pm 1.813 \cdot 10^{-5}$	$2.486 \cdot 10^{-5} \pm 6.349 \cdot 10^{-5}$
LAP	$4.905 \cdot 10^{-9} \pm 3.643 \cdot 10^{-8}$	$5.760 \cdot 10^{-7} \pm 1.440 \cdot 10^{-6}$	$2.518 \cdot 10^{-7} \pm 1.241 \cdot 10^{-6}$	$2.803 \cdot 10^{-7} \pm 1.109 \cdot 10^{-6}$
Elastix	$1.252 \cdot 10^{-5} \pm 5.850 \cdot 10^{-5}$	$5.169 \cdot 10^{-7} \pm 3.857 \cdot 10^{-6}$		

The calculated Dice score are given in Table 5.26 and are depicted in Figure 5.24. The calculated mean surface distance are given in Table 5.27 and displayed in Figure 5.25. The Hausdorff results are given in Table 5.28 and displayed in Figure 5.26.

Table 5.26: Calculated Dice score for various organs for the registration methods. Calculated as described in Section 4.6.3.

Method	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	0.89 ± 0.08	0.83 ± 0.12	0.83 ± 0.12	0.79 ± 0.13	0.76 ± 0.17
VM	0.90 ± 0.08	0.86 ± 0.10	0.85 ± 0.10	0.81 ± 0.12	0.78 ± 0.15
LAP	0.91 ± 0.06	0.91 ± 0.07	0.91 ± 0.08	0.84 ± 0.09	0.77 ± 0.12

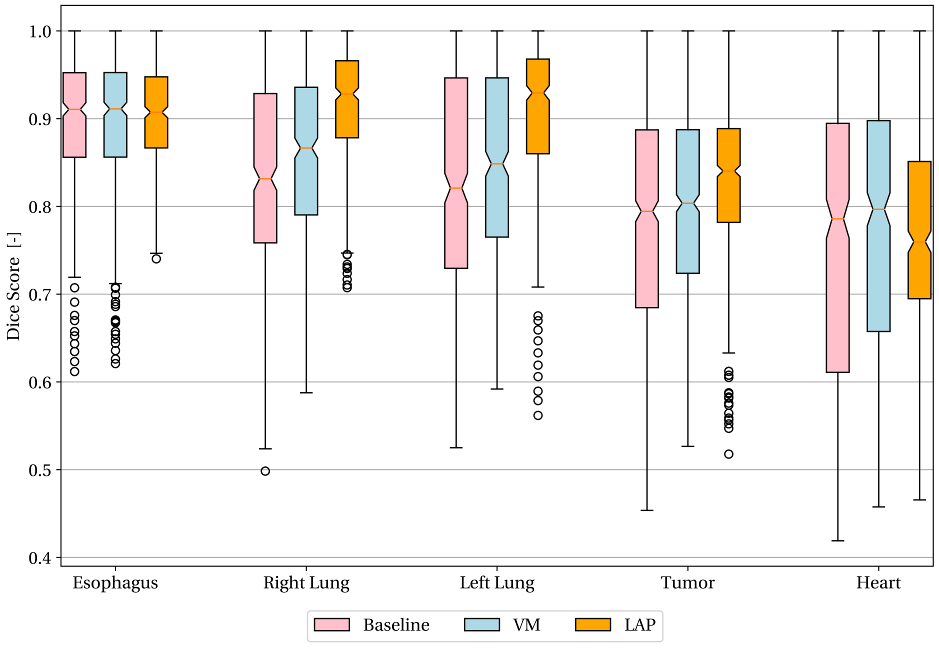

Figure 5.24: The Dice score between the prediction and the fixed contour for three registration methods. Calculated as described in Section 4.6.3. Values are given in Table 5.26.

Table 5.27: Mean surface distance in mm between the prediction and the fixed contour for the registration methods. Calculated as described in Section 4.6.3.

Method	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	0.41 ± 0.34	0.92 ± 0.80	0.80 ± 0.73	0.71 ± 0.46	0.98 ± 0.79
VM	0.40 ± 0.33	0.66 ± 0.60	0.61 ± 0.50	0.64 ± 0.38	0.88 ± 0.66
LAP	0.38 ± 0.22	0.42 ± 0.36	0.41 ± 0.31	0.58 ± 0.32	0.87 ± 0.46

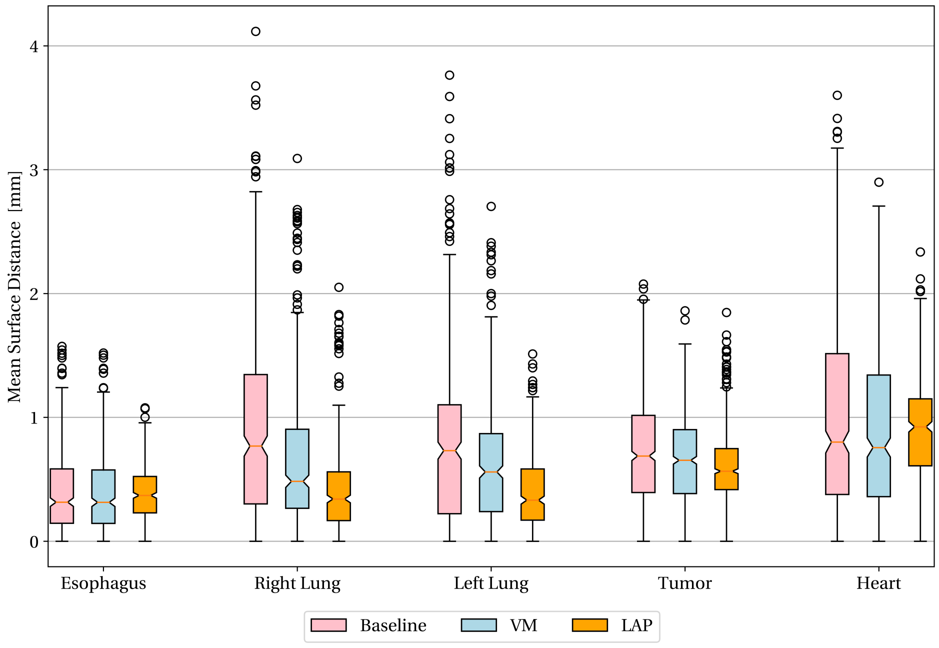

Figure 5.25: The mean surface distance in mm between the prediction and the fixed contour for three registration methods. Calculated as described in Section 4.6.3. Values given in Table 5.27.

Table 5.28: Calculated Hausdorff distance in mm for various organs for the registration methods. Calculated as described in Section 4.6.3.

Method	Esophagus	Right Lung	Left Lung	Tumor	Heart
Baseline	8.23 ± 4.48	17.49 ± 9.27	16.79 ± 10.55	7.33 ± 3.94	13.62 ± 5.72
VM	8.22 ± 4.47	17.14 ± 8.79	16.43 ± 10.39	7.12 ± 3.68	13.61 ± 5.72
LAP	6.48 ± 3.13	15.77 ± 7.91	15.69 ± 9.91	7.28 ± 3.84	13.43 ± 5.29

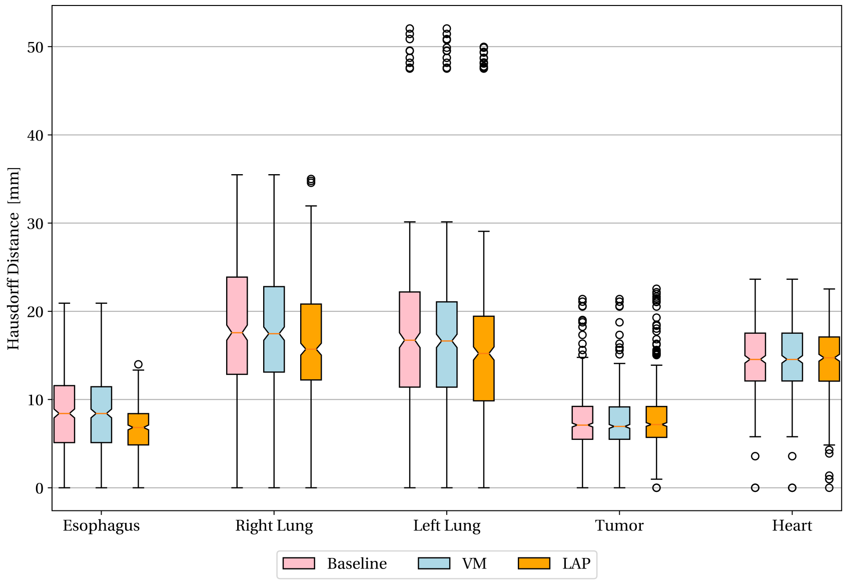

Figure 5.26: The Hausdorff distance in mm between the prediction and the fixed contour for three registration methods. Calculated as described in Section 4.6.3. Values given in Table 5.28.

The time to predict the DVF on both an Intel i7-8550u CPU and NVIDIA Tesla V100S GPU for the three registration methods are given in Table 5.29.

Table 5.29: Prediction times of DVF per method for both the Intel i7-8550u CPU and NVIDIA Tesla V100S GPU. Elastix was not performed on the GPU.

Method	CPU	GPU
Elastix	248 ± 11 s	
VM	23 ± 1 s	260 ± 4 ms
LAP	19 ± 1 s	24 ± 4 ms

6
Discussion

In this chapter, the results given in Chapter 5 from each of the performed simulations will be discussed. This will be followed by recommendations for additional research.

6.1. VoxelMorph model

6.1.1. Varying Hardtanh value

Comparing the results of varying Γ for the Hardtanh range given in Section 5.1.1. No significant difference is observed between the models performance based on the MAE. When comparing the ratio of voxels with a negative Jacobian determinant, the value for the model with a Γ of 8 is an order of magnitude higher than the other models. For the model with Γ set to 20, the ratio of voxels with a negative Jacobian determinant was double the value compared to the others. Among the other models, there is no significant difference in the ratio of voxels with a negative Jacobian determinant. Comparing the contour metrics gives no significant difference between the models in any of the metrics.

As a result, there is no significant impact on the performance by varying the value of the Γ, except for the higher voxels with a negative Jacobian determinant Γ value of 8 and 20.

Since the ground through DVF is unknown, choosing the maximum value of the DVF is difficult. The value of Γ limits the values of the DVF, as a result, large deformations are not possible. If contours or landmarks are available the maximum value of Γ could be based on the maximum Hausdorff distance between the moving and fixed images. Overall, it is better to have a large value for Γ as long as the values of the predicted DVF do not diverge towards infinity.

6.1.2. Varying λ value

The results of varying the weight of the regulation loss λ are given in Section 5.1.2. When comparing the models based on the MAE no significant difference is observed. Comparing the results observed for the ratio of voxels with a negative Jacobian determinant, the model trained with λ is 10 has a larger value. Comparing the contour metrics gives no significant difference between the models in either of the metrics.

Increasing the value of λ penalizes the gradient of the DVF. It would be expected that the gradient of the DVF decreases and the DVF would become more suppressed for increasing values of λ. A DVF which is suppressed too much can result in a decrease of the quality of the registration.

Overall, varying the hyperparameter for Γ and λ seems to have no significant impact on the models performance.

6.2. Laplacian pyramid model

6.2.1. Vary coarses layer

The results of varying the number of CIRM blocks and filters for the coarses level are given in Section 5.2.1. When comparing the models based on the MAE no significant difference is observed. Also, no significant difference is observed for the ratio of voxels with a negative Jacobian determinant. Comparing the contour metrics, for the Dice score, the mean surface distance and the Hausdorff distance there is no significant difference between the models.

6.2.2. VARY ALL LAYER

The results of varying all the levels of the LAP models are given in Section 5.2.2. When comparing the MAE for the models, one observes significant differences between the models. The best-performing model is the base model of 5 CIRM blocks with 32 filters.

When comparing the ratio of voxels with a negative Jacobian determinant large variations are observed. Models with 48 filters are the best performing. The model with 9 layers and 32 filters is the worst-performing model. When comparing the models based on contour metrics no significant difference is observed.

Varying the number of layers and filters does not result in better results compared to the base model of 5 CIRM blocks with 32 filters. The increase in the number of layers and filters increases the number of network parameters. It could be that these models need more training epochs to converge to the optimal values. However, these larger models also take longer to train and use more memory.

6.2.3. 4 GRID SIZES

The results for the models with 4 grid sizes are given in Section 5.2.3. The results do not show an improvement for the MAE compared to the 3-grid model. In addition, the results for the ratio of voxels with a negative Jacobian determinant are worse than the model with 3 grid sizes. The results of the contour metrics are similar for both models.

The addition of a coarser 4th grid size increases the number of parameters in the model but does not improve the quality of the predicted DVF.

Based on the evaluated metrics it can be concluded that changing the number of layers and filters does not significantly improve the performance compared to the base model of 5 CIRM blocks with 32 filters. The addition of a 4th grid size also does not significantly improve the performance of the model.

6.3. COMPARISON OF METHODS

The VM model with a Γ value of 15 and λ of 100 and the LAP model with 5 CIRM blocks of 32 filters are compared. The models are also compared to the registration performed using Elastix with the parameters as described in Section 4.3.

Based on the MAE results given in Section 5.3.1 the LAP model performs better than the VM and Elastix in all data sets. The VM model performs similarly to Elastix based on the MAE.

The VM model has the poorest score when comparing the ratio of voxels with a negative Jacobian determinant. The LAP model performs best and the ratio of voxels with a negative Jacobian determinant is two orders of magnitude smaller compared to the VM model. Elastix performance differs largely between the two datasets. When comparing the contour metrics, the LAP models perform best for the Esophagus and the Lungs. For the tumor and the heart, there is no significant difference between the models.

Prediction of three scans made by the VM and LAP model are displayed in Appendices A.1 and A.2. When comparing the results visually, it is observed that the DVF of VM is minimal. Also, the predicted image is hardly deformed compared to the moving image. The VM model is thus unable to predict good-quality DVF between lung scans. Comparing the results from the LAP model, a much larger DVF is visible. Also, the predicted image seems to match the fixed image better. Therefore the LAP model is suitable for predicting good-quality DVF between lung scans.

The time to predict the DVF by the VM and the LAP models on a CU were 23 ± 1 seconds and 19 ± 1 seconds respectively. The prediction times of both the VM and LAP models are significantly shorter compared to Elastix with 248 ± 11 seconds. On the GPU the prediction times were 260 ± 4 ms and 24 ± 4 ms for the VM and LAP models respectively. Although the VM has 327635 parameters compared to the LAP model which has 1515488 parameters, the VM model takes an order of magnitude longer compared to the LAP model on the GPU. Both the VM and LAP models are faster than other deep learning methods described in Section 3.3. However, the time is dependent on the size of the scans, and the type of GPU used.

6.4. RECOMMENDATIONS

For further research on applying deep learning for DVF prediction the following points are suggested:

- To compare the results obtained in this thesis to others, such as given in Section 3.3, the TRE could be calculated for the landmarks from the DIRLAB dataset. This was attempted but, due to lack of time, was not finished and therefore not in this report.
- In addition to evaluating the quality of the DVF using the registration on the 4DCT images, the predicted DVF could also be used to perform the interplay dose calculation as the method described in Section 2.3. The resulting dose distribution could then be compared with a dose distribution calculated with a DVF obtained with conventional registration techniques.
- The performance of the model is dependent on the training of the model, since neural networks optimise their convolutional filters by training from samples. Additional training on 4DCT scans from different datasets will help to optimise the convolutional filters further and improve the quality of the registration. As a result, the model could become more versatile for scans from unseen machines and patients.

7 Conclusion

7.1. Conclusion

In this report, two model architectures have been evaluated to predict the DVF between scans at different phases of a 4DCT lung scan. The DVF can be used to evaluate the effect of interplay due to breathing of the patient during proton treatment of lung tumors.

The findings of this report can be summarized as:

- The Voxelmorph model is unable to predict good-quality DVFs which could be used for image registration. Hyperparameter tuning of the maximum size of the DVF limited by a HardTanh did not result in a significant increase in the quality of the prediction.

  Varying the loss weight of the divergence of the DVF during training also did not result in a significant increase in the performance of the model. Therefore, in the current state, Voxelmorph is not usable for DVF prediction between 4DCT lung scans.
- The LAP model was capable of predicting good-quality DVFs. Hyperparameter tuning of the number of CIRM blocks and filters did only yield better results when varying all the grid sizes. Also, the addition of a 4th grid size did not yield better results.
- The prediction times for both models were faster than other deep learning methods found in literature, and significantly faster compared to traditional registration methods.
- To better compare the results with other approaches found in the literature, the TRE could be calculated for the landmarks from the DIRLAB dataset.
- Combing the fast prediction of the DVF using the LAP model in combination with the deep learning dose calculation method could result in interplay dose calculation on timescales which are clinically feasible.

7.2. Applications of fast deformation vector fields prediction

The LAP model allows for fast prediction of the DVF between the 4DCT phases in 24 ± 4 ms. The deep learning dose calculation method described in Section 2.3 performs dose calculation in 5.0 ± 4.9 ms per pencil beam. The interplay dose calculation described in Section 2.6.3 could be speedup by combining the LAP model and the deep learning dose calculation.

BIBLIOGRAPHY

[1] WHO - International Agency for Research on Cancer. *World fact-sheets 2020*. https://gco.iarc.fr/today/data/factsheets/populations/900-world-fact-sheets.pdf[Online - accessed 2022-03-17], 2020.

[2] O. Pastor-Serrano and Z. Perkó. "Learning the Physics of Particle Transport via Transformers". In: *arXiv* (Sept. 2021). DOI: 10.48550/arXiv.2109.03951.

[3] A.C. Moreno et al. "Intensity modulated proton therapy (IMPT) – The future of IMRT for head and neck cancer". en. In: *Oral Oncology* 88 (Jan. 2019), pp. 66–74. ISSN: 1368-8375. DOI: 10.1016/j.oraloncology.2018.11.015.

[4] K.M. Kraus et al. "Dosimetric consequences of tumour motion due to respiration for a scanned proton beam". In: *Physics in Medicine and Biology* 56.20 (Sept. 21, 2011), pp. 6563–6581. ISSN: 0031-9155. DOI: 10.1088/0031-9155/56/20/003.

[5] G. Balakrishnan et al. "VoxelMorph: A Learning Framework for Deformable Medical Image Registration". In: *IEEE Transactions on Medical Imaging* 38.8 (Feb. 2019), pp. 1788–1800. DOI: 10.1109/TMI.2019.2897538.

[6] T.C.W. Mok et al. *Large Deformation Diffeomorphic Image Registration with Laplacian Pyramid Networks*. June 2020. DOI: 10.48550/ARXIV.2006.16148.

[7] H. M. Kooy and C. Grassberger. "Intensity modulated proton therapy". en. In: *The British Journal of Radiology* 88.1051 (July 2015), p. 20150195. ISSN: 0007-1285. DOI: 10.1259/bjr.20150195.

[8] C. Barker et al. "An introduction to proton beam therapy". In: (Oct. 2019). DOI: 10.12968/hmed.2019.80.10.574.

[9] *Breathing cycle and regulation - Osmosis*. 2022. URL: https://www.osmosis.org/learn/Breathing%5C_cycle%5C_and%5C_regulation (visited on 12/02/2022).

[10] O. Pastor-Serrano et al. "How should we model and evaluate breathing interplay effects in IMPT?" In: *Physics in Medicine and Biology* 66.23 (Nov. 23, 2021), p. 235003. ISSN: 0031-9155. DOI: 10.1088/1361-6560/ac383f.

[11] T. L. Phillips et al. *Leibel and Phillips Textbook of Radiation Oncology*. en. Saunders, 2010. ISBN: 9781416058977.

[12] P.C. Park et al. "A beam-specific planning target volume (PTV) design for proton therapy to account for setup and range uncertainties". In: *International Journal of Radiation Oncology Biology Physics* 82.2 (Feb. 2012), e329–e336. DOI: 10.1016/j.ijrobp.2011.05.011.

[13] H. Paganetti. *Proton Therapy Physics*. en. CRC Press, Apr. 19, 2016. ISBN: 978-1-4398-3645-3.

[14] K. Souris et al. "Technical Note: Monte Carlo methods to comprehensively evaluate the robustness of 4D treatments in proton therapy". en. In: *Medical Physics* 46.10 (Aug. 31, 2019), pp. 4676–4684. ISSN: 0094-2405. DOI: 10.1002/mp.13749.

[15] S. Klein and M. Staring. *Elastix, The Manual Elastix 4.9.0*. Ed. by https://usermanual.wiki/Document/elastix490manual.1389615963/view. [Online - accessed 2022-03-07], 2018.

[16] I. J. Chetty and M. Rosu-Bubulac. "Deformable Registration for Dose Accumulation". en. In: *Seminars in Radiation Oncology* 29.3 (July 2019), pp. 198–208. ISSN: 1053-4296. DOI: 10.1016/j.semradonc.2019.02.002.

[17] M. Unser. "Splines: a perfect fit for signal and image processing". In: *IEEE Signal Processing Magazine* 16.6 (Nov. 1999), pp. 22–38. DOI: 10.1109/79.799930.

[18] S. Pal et al. "Towards Positive Jacobian: Learn to Postprocess Diffeomorphic Image Registration with Matrix Exponential". In: *arXiv* (Feb. 2022). DOI: 10.48550/ARXIV.2202.00749.

[19] J. Chen et al. "TransMorph: Transformer for unsupervised medical image registration". In: *Medical Image Analysis* 82 (Nov. 2022), p. 102615. DOI: 10.1016/j.media.2022.102615.

[20] Vincent Arsigny et al. “A Log-Euclidean Framework for Statistics on Diffeomorphisms”. In: *Medical Image Computing and Computer-Assisted Intervention MICCAI 2006*. Springer Berlin Heidelberg, June 2006, pp. 924–931. DOI: 10.1007/11866565_113.

[21] J. R. McClelland et al. “Respiratory motion models: A review”. en. In: *Medical Image Analysis* 17.1 (Jan. 2013), pp. 19–42. ISSN: 1361-8415. DOI: 10.1016/j.media.2012.09.005.

[22] A. Protik et al. “The impact of breathing amplitude on dose homogeneity in intensity modulated proton therapy”. en. In: *Physics and Imaging in Radiation Oncology* 3 (July 2017), pp. 11–16. ISSN: 2405-6316. DOI: 10.1016/j.phro.2017.07.004.

[23] H. A. Shih et al. “Internal target volume determined with expansion margins beyond composite gross tumor volume in three-dimensional conformal radiotherapy for lung cancer”. en. In: *International Journal of Radiation Oncology, Biology and Physics* 60.2 (Oct. 2004), pp. 613–622. ISSN: 0360-3016. DOI: 10.1016/j.ijrobp.2004.05.031.

[24] M. Engelsman et al. “Four-dimensional proton treatment planning for lung tumors”. en. In: *International Journal of Radiation Oncology, Biology and Physics* 64.5 (Apr. 2006), pp. 1589–1595. ISSN: 0360-3016. DOI: 10.1016/j.ijrobp.2005.12.026.

[25] J. Hanley et al. “Deep inspiration breath-hold technique for lung tumors: the potential value of target immobilization and reduced lung density in dose escalation”. en. In: *International Journal of Radiation Oncology, Biology and Physics* 45.3 (Oct. 1999), pp. 603–611. ISSN: 0360-3016. DOI: 10.1016/s0360-3016(99)00154-6.

[26] A. Schätti et al. “The effectiveness of combined gating and re-scanning for treating mobile targets with proton spot scanning. An experimental and simulation-based investigation”. In: *Physics in Medicine and Biology* 59.14 (June 23, 2014), pp. 3813–3828. ISSN: 0031-9155. DOI: 10.1088/0031-9155/59/14/3813.

[27] S.M. Zenklusen et al. “A study on repainting strategies for treating moderately moving targets with proton pencil beam scanning at the new Gantry 2 at PSI”. In: *Physics in Medicine and Biology* 55.17 (Aug. 11, 2010), pp. 5103–5121. ISSN: 0031-9155. DOI: 10.1088/0031-9155/55/17/014.

[28] T. van der Meulen. *BSc thesis: Optimising convolutional neural networks for attenuation map estimation based on SPECT data*. Delft University of Technology, 2020.

[29] J. Feldman and R. Rojas. *Neural Networks - A Systematic Introduction*. en. Springer London, Limited, 1996. ISBN: 978-3-642-61068-4.

[30] A. Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Unsupervised learning techniques*. en. 2019. ISBN: 9781492032649.

[31] A. R. Jha. *Mastering Pytorch*. en. 2021. ISBN: 978-1789614381.

[32] B. Ramsundar et al. *Tensorflow for Deep Learning*. en. O’Reilly Media, 2018. ISBN: 9781491980453.

[33] *GroupNorm – PyTorch 1.12 documentation*. 2022. URL: https://pytorch.org/docs/stable/generated/torch.nn.GroupNorm.html (visited on 10/05/2022).

[34] *LeakyReLU – PyTorch 1.12 documentation*. 2022. URL: https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html?highlight=nn%5C%20leakyrelu%5C#torch.nn.LeakyReLU (visited on 10/05/2022).

[35] *Conditional Deformable Image Registration with Convolutional Neural Network*. 2022. URL: https://github.com/cwmok/Conditional%5C_LapIRN (visited on 11/02/2022).

[36] G. Haskins et al. “Deep learning in medical image registration: a survey”. In: (Jan. 2020). DOI: 10.1007/s00138-020-01060-x.

[37] S. Abbasi et al. “Medical image registration using unsupervised deep neural network: A scoping literature review”. In: (Mar. 2021). DOI: 10.1016/j.bspc.2021.103444.

[38] E. Castillo et al. “Four-dimensional deformable image registration using trajectory modeling”. In: *Physics in Medicine and Biology* 55.1 (Dec. 2009), pp. 305–327. DOI: 10.1088/0031-9155/55/1/018.

[39] *4DCT - Emory School of Medicine*. 2009. URL: https://med.emory.edu/departments/radiation-oncology/research-laboratories/deformable-image-registration/downloads-and-reference-data/4dct.html (visited on 11/02/2022).

[40] H. Sokooti et al. *3D Convolutional Neural Networks Image Registration Based on Efficient Supervised Learning from Artificial Deformations.* Aug. 2019. DOI: 10.48550/ARXIV.1908.10235.

[41] F. Yabo et al. "Deep learning in medical image registration: a review". In: *Physics in Medicine & Biology* 65.20 (Oct. 2020), 20TR01. DOI: 10.1088/1361-6560/ab843e.

[42] B. D. de Vos et al. "A deep learning framework for unsupervised affine and deformable image registration". In: *Medical Image Analysis* 52 (Feb. 2019), pp. 128–143. DOI: 10.1016/j.media.2018.11.010.

[43] T Sentker et al. "GDL-FIRE-4D: Deep Learning-Based Fast 4DCT Image Registration". In: *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018.* Springer International Publishing, Sept. 2018, pp. 765–773. DOI: 10.1007/978-3-030-00928-1_86.

[44] F. Yabo et al. "LungRegNet: An unsupervised deformable image registration method for 4D-CT lung". In: *Medical Physics* 47.4 (Feb. 2020), pp. 1763–1774. DOI: 10.1002/mp.14065.

[45] J. Zhuoran et al. "A multi-scale framework with unsupervised joint training of convolutional neural networks for pulmonary deformable image registration". In: *Physics in Medicine & Biology* 65.1 (Jan. 2020), p. 015011. DOI: 10.1088/1361-6560/ab5da0.

[46] T. Fechter and D. Baltas. *One Shot Learning for Deformable Medical Image Registration and Periodic Motion Tracking.* July 2019. DOI: 10.48550/ARXIV.1907.04641.

[47] B. Kim et al. "CycleMorph: Cycle consistent unsupervised deformable image registration". en. In: *Medical Image Analysis* 71 (July 2021), p. 102036. ISSN: 1361-8415. DOI: 10.1016/j.media.2021.102036.

[48] Y. Zheng et al. "Deformable registration of chest CT images using a 3D convolutional neural network based on unsupervised learning". en. In: *Journal of Applied Clinical Medical Physics* 22.10 (Sept. 10, 2021), pp. 22–35. ISSN: 1526-9914. DOI: 10.1002/acm2.13392.

[49] X. Hu et al. "Dual-Stream Pyramid Registration Network". In: *Lecture Notes in Computer Science.* Springer International Publishing, Oct. 2019, pp. 382–390. DOI: 10.1007/978-3-030-32245-8_43.

[50] A. Hoopes et al. *Learning the Effect of Registration Hyperparameters with HyperMorph.* Mar. 2022. DOI: 10.48550/ARXIV.2203.16680.

[51] A.Hering et al. "mlVIRNET: Multilevel Variational Image Registration Network". In: *Lecture Notes in Computer Science.* Springer International Publishing, Oct. 2019, pp. 257–265. DOI: 10.1007/978-3-030-32226-7_29.

[52] H. Sokooti et al. "Accuracy Estimation for Medical Image Registration Using Regression Forests". In: *Lecture Notes in Computer Science.* Springer International Publishing, Oct. 2016, pp. 107–115. DOI: 10.1007/978-3-319-46726-9_13.

[53] *Github ElastixModelZoo/models/Par0049 - SuperElastix/ElastixModelZoo.* 2020. URL: https://github.com/SuperElastix/ElastixModelZoo/tree/master/models/Par0049 (visited on 11/16/2022).

[54] G. D. Hugo et al. "Data from 4D Lung Imaging of NSCLC Patients". In: (Oct. 2016). DOI: 10.7937/K9/TCIA.2016.ELN8YGLE.

[55] *Validation Data for Deformable Image Registration - Léon Bérard Cancer Center & CREATIS lab, Lyon, France.* 2019. URL: https://www.creatis.insa-lyon.fr/rio/dir%5C_validation%5C_data (visited on 10/12/2022).

[56] J. Vandemeulebroucke et al. "Spatiotemporal motion estimation for respiratory-correlated imaging of the lungs". In: *Medical Physics* 38.1 (Dec. 2010), pp. 166–178. DOI: 10.1118/1.3523619.

[57] J. Vandemeulebroucke et al. "The POPI-model, a point-validated pixel-based breathing thorax model". In: *XVth international conference on the use of computers in radiation therapy (ICCR).* Vol. 2. Citeseer. 2007, pp. 195–199.

[58] *Hierarchical Data Formats - What is HDF5?* 2020. URL: https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5 (visited on 10/13/2022).

[59] *voxelmorph: Unsupervised Learning for Image Registration.* 2022. URL: https://github.com/voxelmorph/voxelmorph (visited on 11/02/2022).

[60] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 1)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1. 2022.

[61] *torch.nn.functional.grid_sample – PyTorch 1.12 documentation*. 2022. URL: https://pytorch.org/docs/stable/generated/torch.nn.functional.grid%5C_sample.html (visited on 10/06/2022).

[62] *deepmind/surface-distance: Library to compute surface distance based performance metrics for segmentation tasks*. 2018. URL: https://github.com/deepmind/surface-distance (visited on 11/08/2022).

[63] S. Nikolov et al. *Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy*. Mar. 2018. DOI: 10.48550/ARXIV.1809.04430.

[64] K. J. Kiser et al. "Novel Autosegmentation Spatial Similarity Metrics Capture the Time Required to Correct Segmentations Better Than Traditional Metrics in a Thoracic Cavity Segmentation Workflow". In: *Journal of Digital Imaging* (May 2021). DOI: 10.1007/s10278-021-00460-3.

A

Appendix

A.1. VOXELMORPH IMAGES

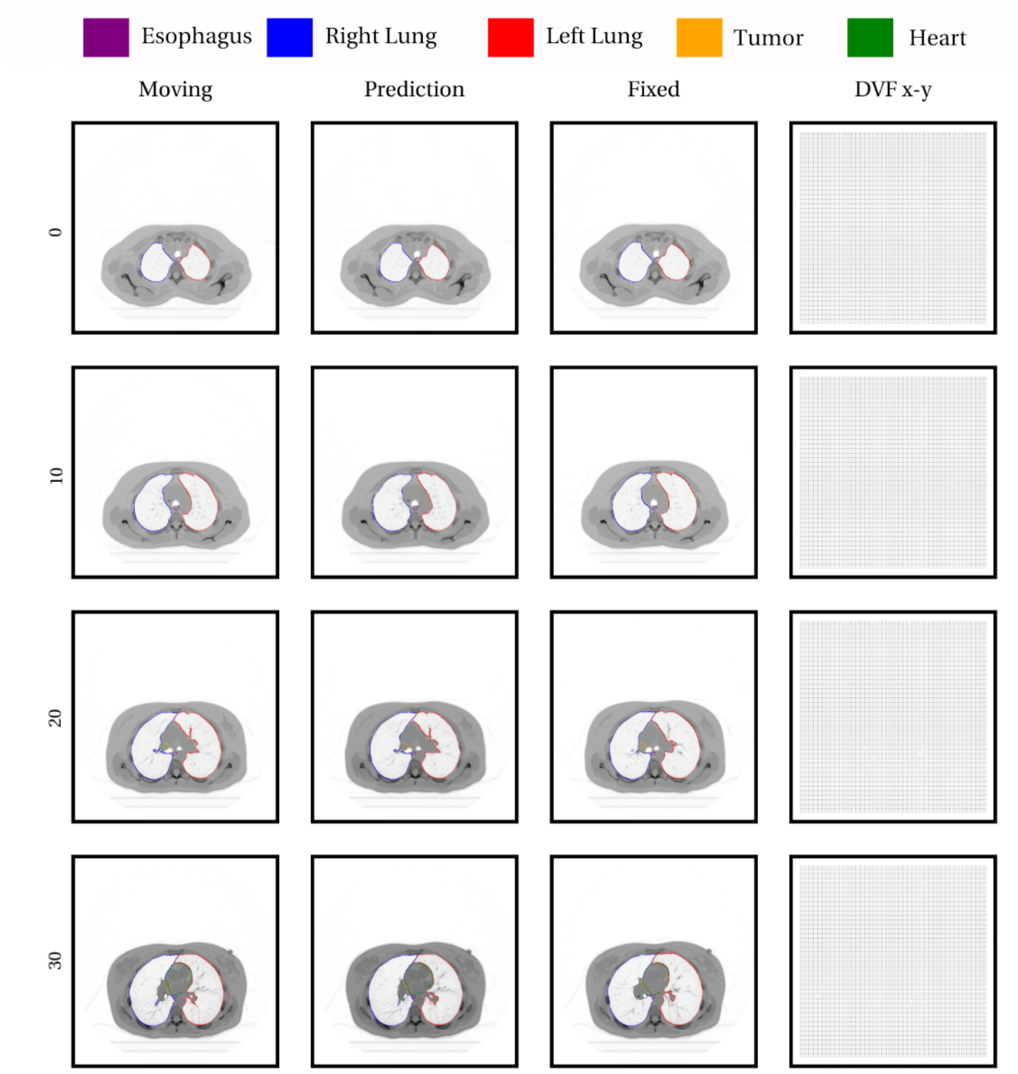

Figure A.1: Overview of upper slices for the fixed, moving and predicted image of scan 101 from 0 to 6x0 phase. Prediction by the VM model with Γ is 10 and λ is 100.

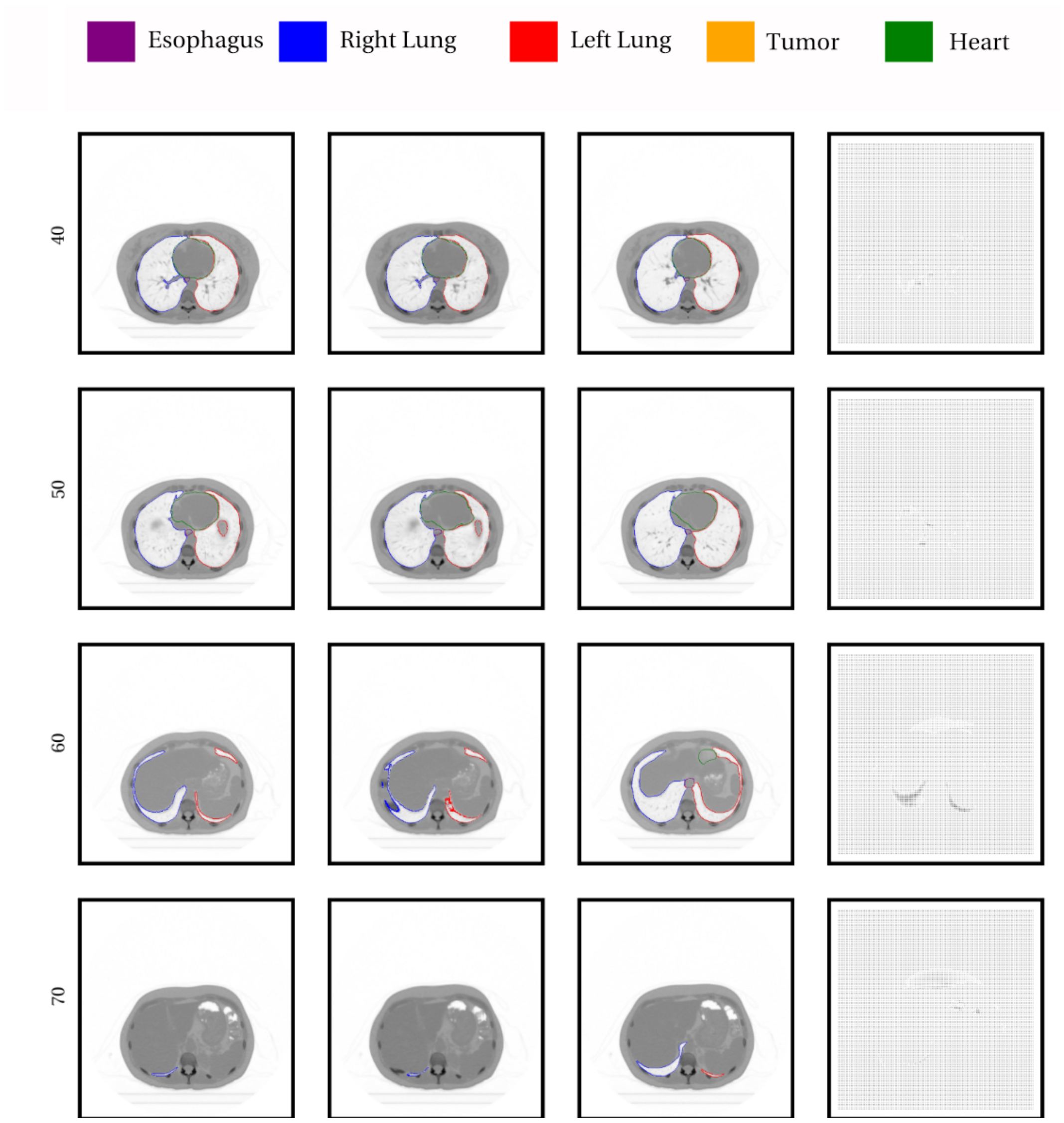

Figure A.2: Overview of lower slices for the fixed, moving and predicted image of scan 101 from 0 to 60 phase. Prediction by the VM model with Γ is 10 and λ is 100.

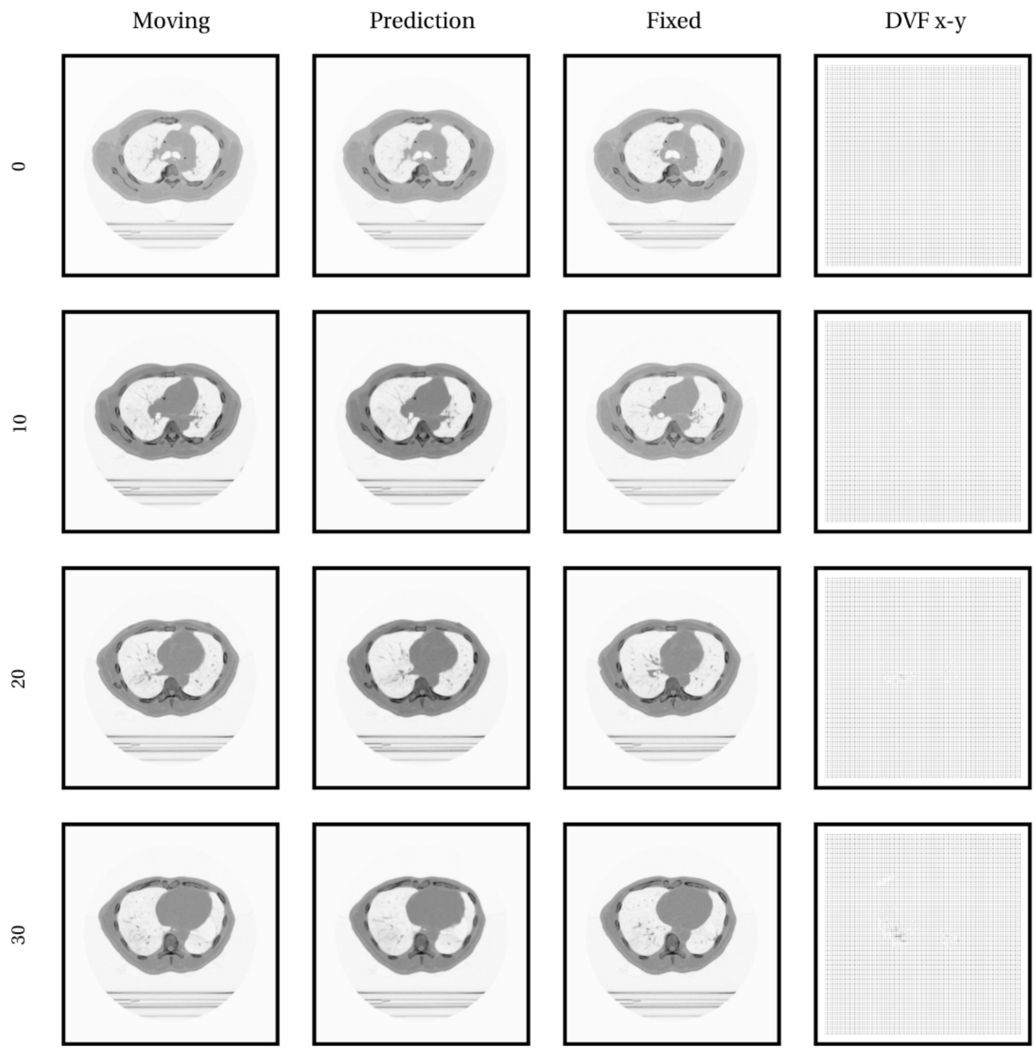

Figure A.3: Overview of upper slices for the fixed, moving and predicted image of scan 124 from 0 to 40 phase. Prediction by the VM model with Γ is 10 and λ is 100.

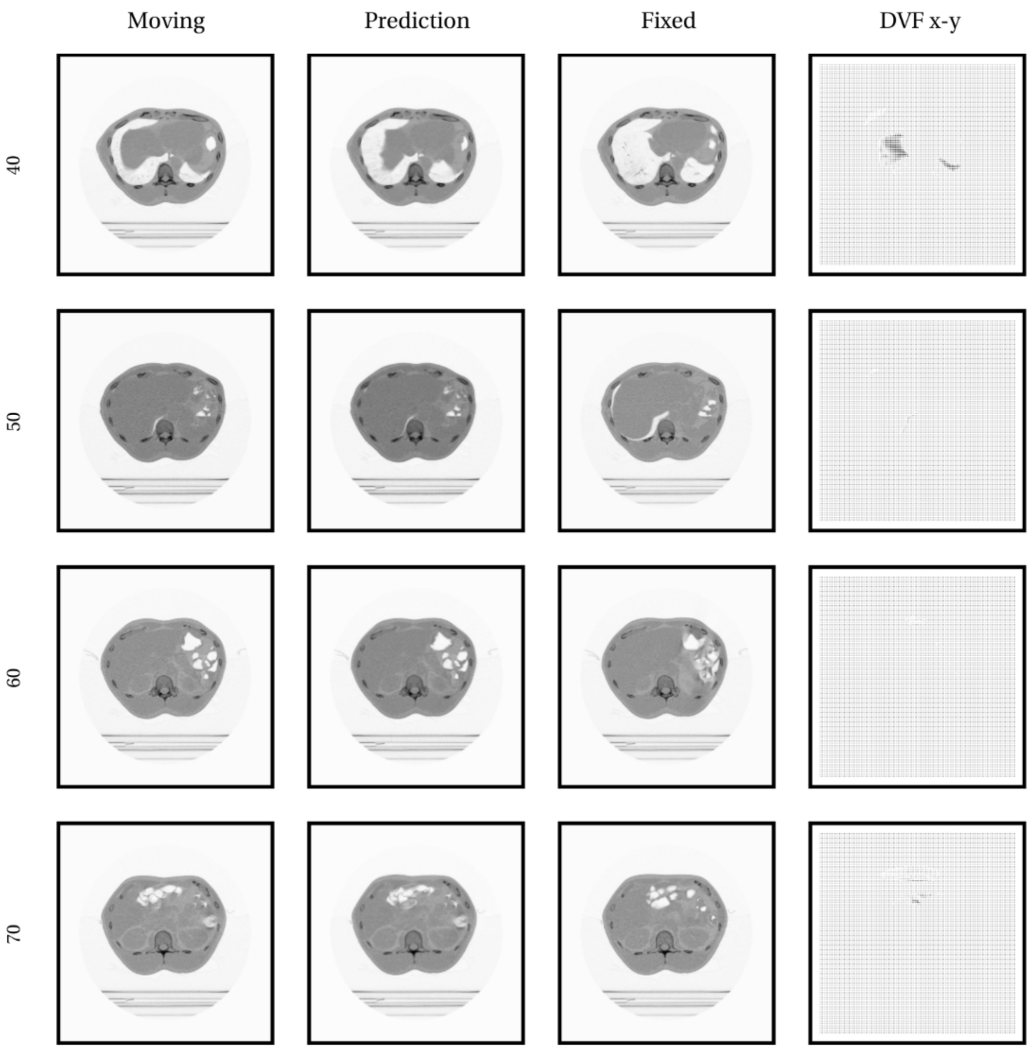

Figure A.4: Overview of lower slices for the fixed, moving and predicted image of scan 124 from 0 to 40 phase. Prediction by the VM model with Γ is 10 and λ is 100.

A.2. LAPLACIAN PYRAMID IMAGES

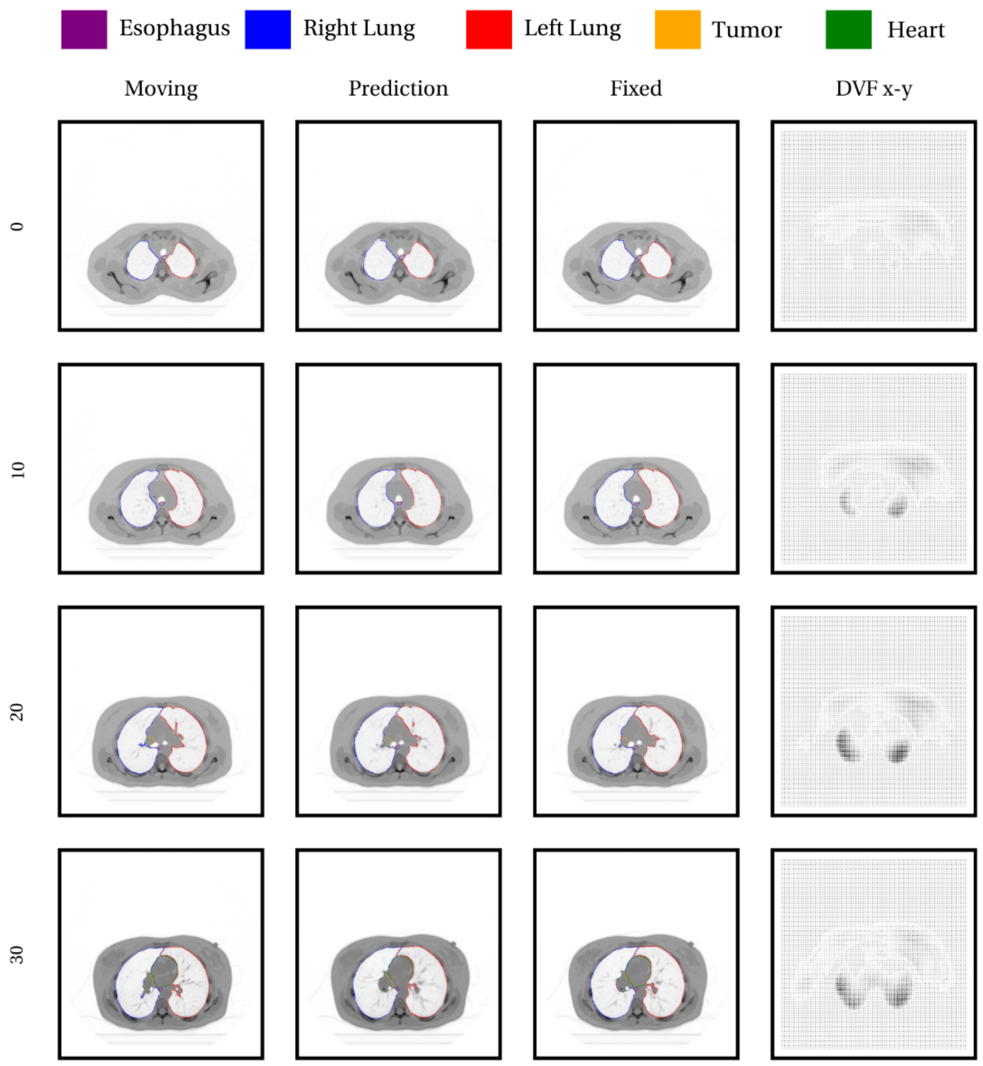

Figure A.5: Overview of upper slices for the fixed, moving and predicted image of scan 101 from 0 to 60 phase. Prediction by the LAP model with 5 CIRM blocks of 32 filters.

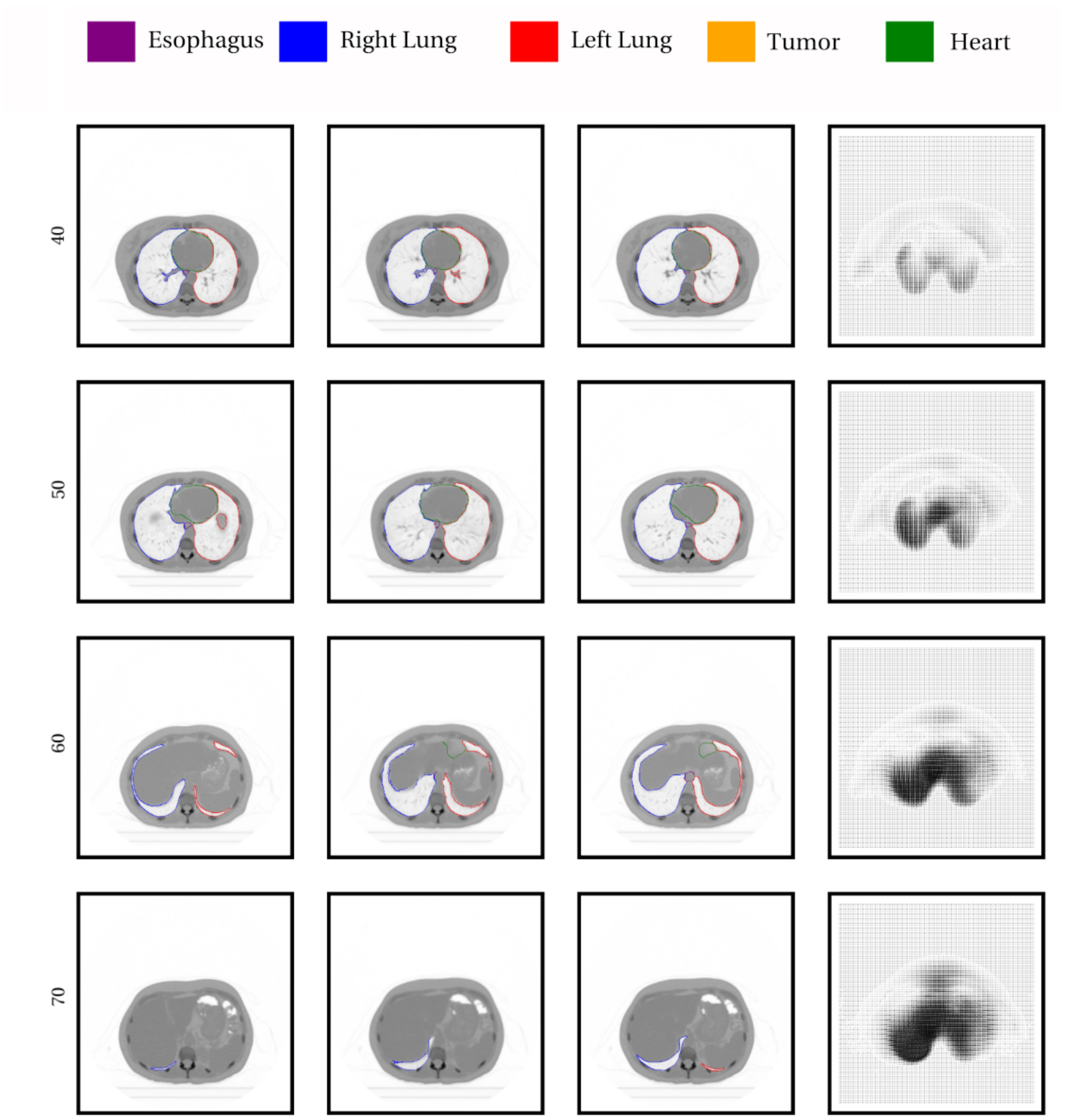

Figure A.6: Overview of lower slices for the fixed, moving and predicted image of scan 101 from 0 to 60 phase. Prediction by the LAP model with 5 CIRM blocks of 32 filters.

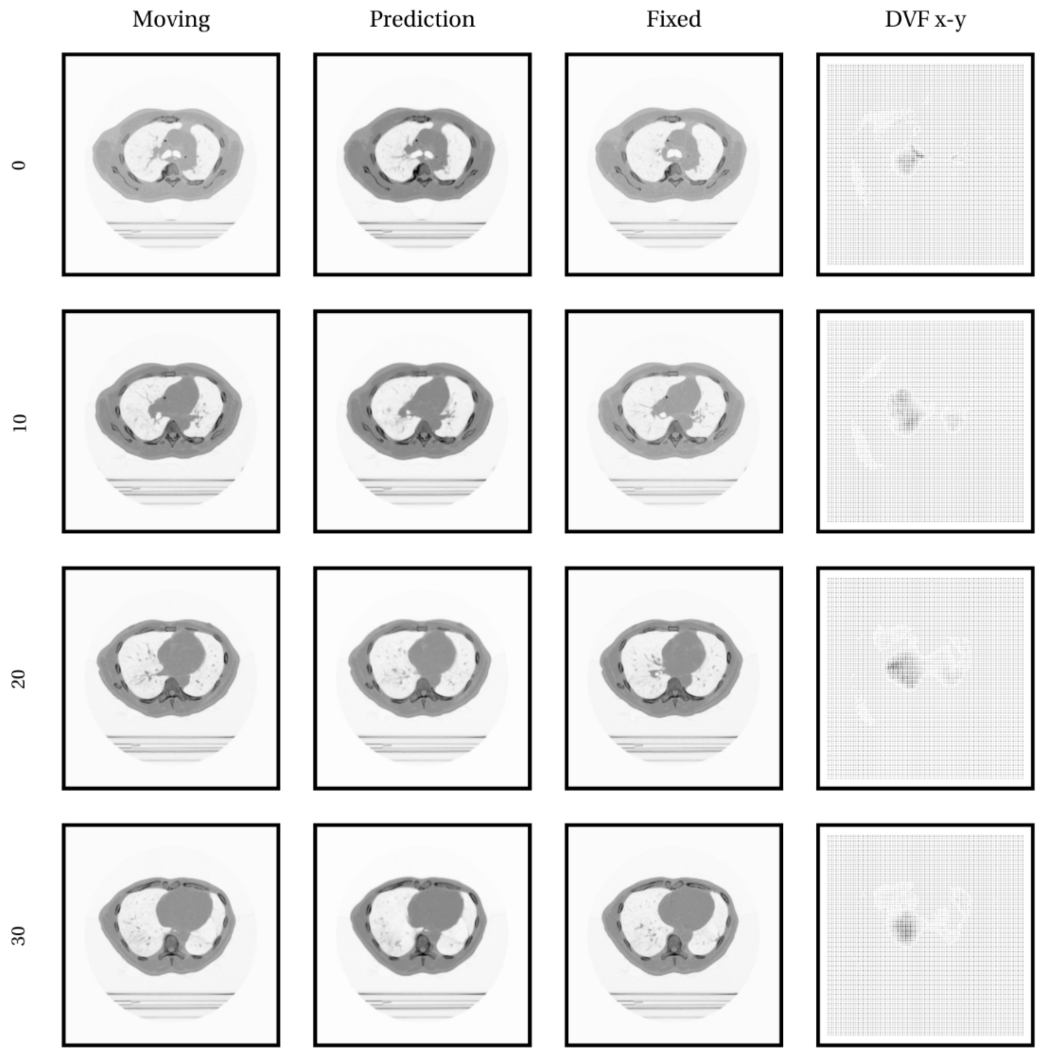

Figure A.7: Overview of upper slices for the fixed, moving and predicted image of scan 124 from 0 to 40 phase. Prediction by the LAP model with 5 CIRM blocks of 32 filters.

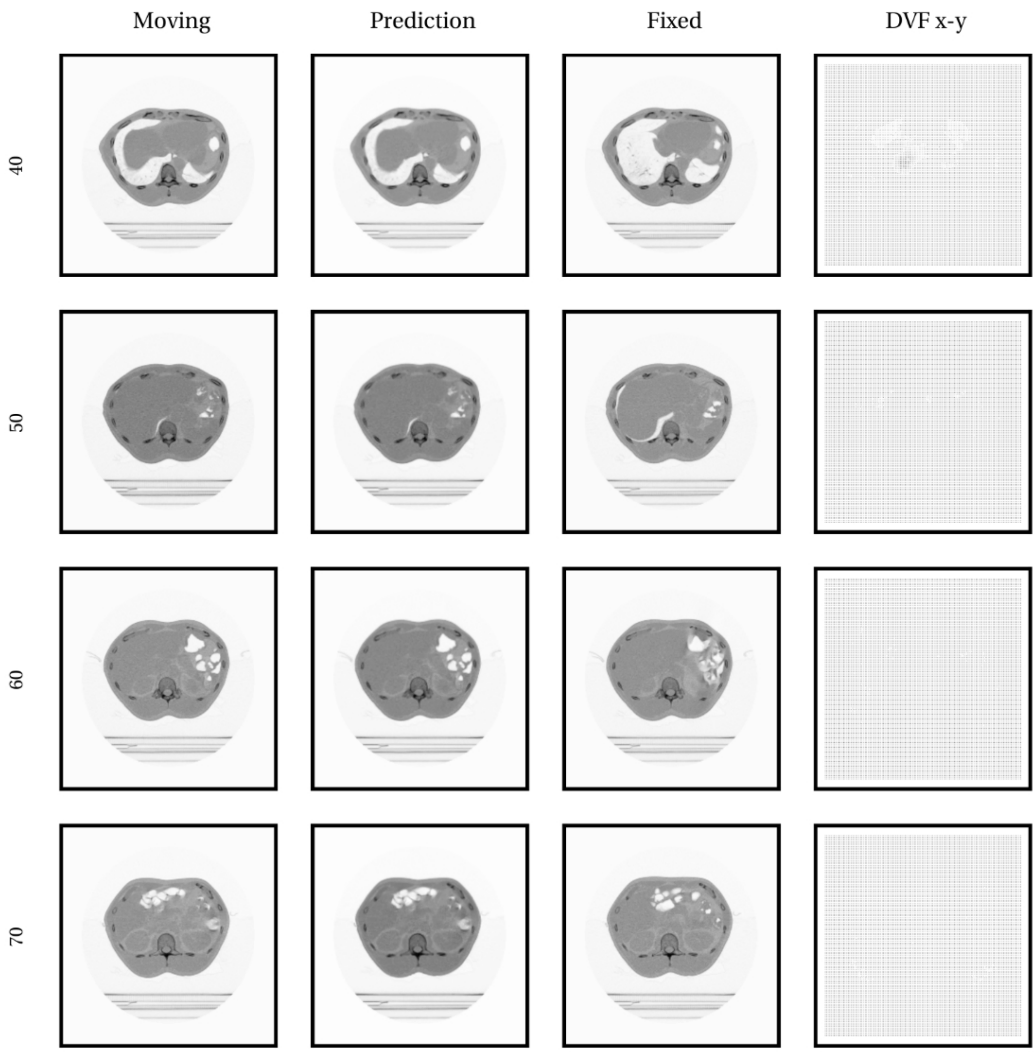

Figure A.8: Overview of lower slices for the fixed, moving and predicted image of scan 124 from 0 to 40 phase. Prediction by the LAP model with 5 CIRM blocks of 32 filters.