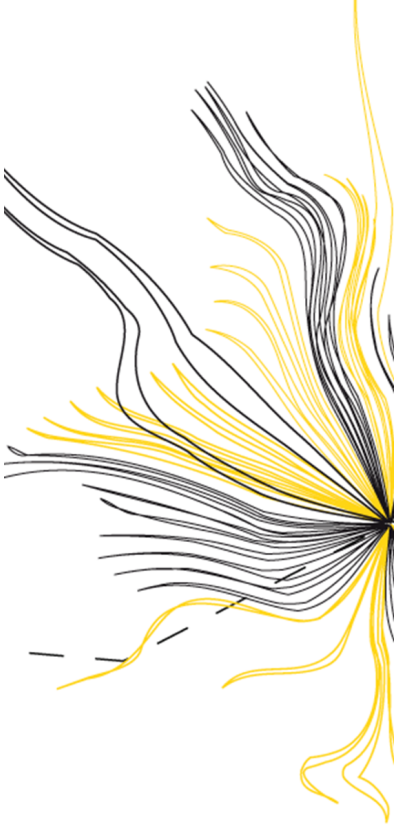


UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

Neural implicit representations for diffeomorphic medical image registration

Jesse Zwienenberg

M.Sc. Thesis
August 2021

Supervisors:
Christoph Brune
Jelmer Wolterink

Applied Analysis group
Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

Image registration is the task of finding a transformation that aligns given images. Typically, this is solved using discrete image registration algorithms. These discrete algorithms are inherently limited by the resolution of the images. Rather than representing (3D-) images explicitly using a discrete set of pixels (or voxels), they can also be represented implicitly by a neural network. Not only the images can be implicitly represented, but also the transformation itself. One of the advantages of the implicit representations in registration models is that the resulting transformation is defined on any coordinate in the continuous image domain, rather than only on a discrete set of coordinates. This means there is no need for interpolation anywhere, and the image derivatives can easily be computed analytically rather than through finite differences. Also, both the transformation and the images can be sampled at arbitrarily high resolutions. In this thesis, we propose two models for medical image registration in a continuous setting, a small deformation model and a diffeomorphic model. For both models, the image-pair and the transformation are continuous differentiable functions parametrized by neural networks. The models are evaluated on a dataset of 3D CT scans with manually identified landmarks.

Keywords: medical image registration, neural implicit representations, unsupervised learning, diffeomorphism, periodic activation functions

CONTENTS

Chapter 1

Introduction

1.1 Medical image registration

Medical image registration is the practice of spatially transforming one image in such a way that the underlying anatomical structures align with a given reference image. Aligning two (or more) images can be useful for several tasks in the medical field. Registering multiple scans from one particular patient taken in the span of a few seconds can be used for modeling the spatio-temporal behavior of certain tissues. This behavior is especially important for detecting complications regarding, for example, cardiac and respiratory motion. Registration can also be useful to model long-term deformations. For instance, the growth or shrinkage of lesions can be measured by registering scans taken multiple days apart.

The images do not necessarily have to be of the same patient, or even of the same modality. For instance, different modalities can depict different aspects of a certain tissue. Registering these images makes it possible to accurately combine information derived from these different modalities. For this thesis, however, we consider the scenario of intra-patient single-modal registration.

In this thesis, we look at medical image registration through a continuous lens. In order to facilitate a continuous approach, the images need to be represented in a continuous way. For this, we use neural implicit representations.

1.2 Neural implicit representation

Generally, any digital image is represented as a discrete grid of pixels. A color (or intensity/density) is assigned to each of these pixels and together they form the image as it appears on a screen. Even though the images themselves are discrete, they represent a continuous real-world scene. In image analysis, it is common to work with this underlying continuous real-world distribution of which the digital image is a discretization. Dealing with a continuous distribution rather than a discrete set offers many advantages. This continuous setting allows spatial derivatives of the images to exist. Also, the fact that the image is defined everywhere on the continuous domain makes it possible to perform operations such as a coordinate transformation.

However, this continuous distribution is unknown in practical cases. So in order to apply any knowledge gained from the analysis of the continuous distribution, techniques such as interpolation and finite differences are necessary. These techniques generally introduce errors and properties in the continuous setting do not always directly translate to the discrete setting, especially when the discrete sampling is diffuse. This incentivizes the usage of a continuous representation of images, *i.e.* a map of any coordinate in the continuous image domain to a

color (or intensity/density).

Since, of course, the images have to be stored in finite memory, these continuous maps need to be described by finitely many parameters. A natural way of designing a parametrized function is by using a neural network. By the universal approximation theorem [1], any well-behaved function can be described with arbitrarily high accuracy by an appropriate neural network. So, in this thesis, the implicit representations are parametrized by neural networks. These implicitly defined signal representations parametrized by neural networks are referred to as neural implicit representations.

Neural implicit representations offer several advantages over the traditional discrete representation. Since the image is defined on the continuous domain it can be sampled at an arbitrary resolution, even allowing resolutions higher than the one of the original image. Furthermore, the memory required to store the image depends solely on the complexity of the signal, whereas discrete images are limited in efficiency by the grid resolution. So, for neural implicit representations, the memory requirement is determined by the size of the network that is necessary to represent the image accurately. This required size can vary significantly between different images and will largely depend on the presence of fine details. Another advantage is that continuous derivatives exist, as long as differentiable activation functions are used. As will be demonstrated later, this allows the image derivatives to be computed analytically in a straightforward manner, without having to resort to finite difference methods as is required for discrete images.

This neural implicit representation is not only useful for representing images, but can be used to represent any type of signal. It can, for instance, be used to represent vector fields, which is useful for describing the transformation in the image registration problem.

1.3 Related work

Recently, there has been some research into the usage of neural implicit representations for novel view synthesis for 3D scenes. Most of the recent papers in this area build on the work of NeRF (neural radiance fields) [2]. This work proposes learning (implicit) neural radiance fields from 2D data from limited camera angles. Constructing novel 2D views of the 3D scene is done by integrating along light rays. Since this is done in a differentiable way the whole process is compatible with gradient descent methods for optimization.

One of the papers that build on the idea of neural radiance fields is NeRF in the Wild [3]. Here the NeRF model is extended to work with unconstrained photo collections. Since the data used is uncontrolled, the objects in the images do not always look the same. The discrepancies include things such as differences in lighting and people walking in front of the object. To allow the model to function in this setting it is used in conjunction with a separate model which is able to handle the photometric discrepancies.

Furthermore, several papers address the extension of NeRF models to a dynamic setting in which the scene is allowed to change over time. These models can be used to produce videos of non-static scenes in which the camera angle can be set to any arbitrary value within a certain domain after the data has already been collected. Two of the papers that address this challenge are D-NeRF [4] and Nerfies [5]. Both of them propose similar ideas where a neural network is used to (implicitly) capture a deformation field, which allows for a dynamic (time-dependent) scene. This has a close connection to image registration, where we are interested in finding a deformation field that aligns images.

1.4 Contribution

In this thesis, we propose to use neural implicit representations for images as well as the transformation in image registration, thereby allowing resolution-agnostic small deformation and diffeomorphic registration models. We provide two models that use these neural implicit representations and evaluate their performance on a 3D CT scan dataset.

1.5 Thesis outline

In the next chapter, we start by giving a mathematical overview of medical image registration. We describe several options for the loss function, including regularization, and describe the optimization process. We close off the chapter by briefly discussing what it means for registration to be diffeomorphic.

In the third chapter, we discuss the neural implicit representation. Then we describe two solutions for overcoming the low-frequency bias of standard MLPs. We conclude the chapter with some experiments.

The fourth chapter contains our models for the registration of medical images. We separate between a small deformations model and a diffeomorphic model.

In the fifth chapter, we break down the experiments performed using our models and present the results.

Finally, in the last chapter, we present the main conclusions and discuss the possible limitations of our approach. We conclude this thesis by presenting some ideas for future work.

Chapter 2

Image Registration

2.1 Overview

Image registration is the task of finding the optimal spatial transformation that aligns two images. By convention, the two images that are registered are called the moving image and the fixed image. The moving image M is the image that is to be transformed. The fixed image F, on the other hand, serves as the target of the transformation on the moving image. After applying the transformation on the moving image, the goal is a perfect correspondence with the fixed image.

Both images can be described by a function that maps image coordinates to intensity or color values. For this thesis we only consider grayscale images, so both functions $F(x)$ and $M(x)$ map the n-d image coordinates to intensity values in the interval $[0, 1]$. This leads to the following description of the images:

$$\begin{aligned} M &: \Omega_M \subset \mathbb{R}^n \to [0,1] \\ F &: \Omega_F \subset \mathbb{R}^n \to [0,1]\,. \end{aligned} \tag{2.1}$$

Here, Ω_M and Ω_F denote the domain of image coordinates of the moving and fixed image respectively. In this thesis, we consider the normalized image domain $[-1, 1]^n$ for both Ω_M and Ω_F. Consequently, Ω_M and Ω_F are the same and will be referred to as the image domain Ω. In practice, the images are only defined on finitely many coordinates within this domain, as there are only finitely many pixels. However, we assume the image to be defined on the entire domain. In general, this is realized by using spline interpolation, but in this thesis we use neural implicit representations to define the images on the entire image domain. We discuss these neural implicit representations in more detail in chapter 3.

For image registration the objective is finding a transformation Φ on the image coordinates:

$$\Phi : \Omega_F \to \Omega_M. \tag{2.2}$$

The objective of image registration is finding an optimal $\hat{\Phi}$ such that the object located at coordinate x in image F corresponds to the object at coordinate $\hat{\Phi}(x)$ in image M. Assuming all objects look the same in both images, and that the transformation $\hat{\Phi}$ perfectly describes the correspondence between objects across the images, we have that:

$$\left(M \circ \hat{\Phi}\right)(x) = F(x) \qquad \forall x \in \Omega. \tag{2.3}$$

In general, there are some cases where objects are not expected to have the same appearance across different images. For example, in multi-modal image registration the appearances of the two images are not the same. However, for this thesis, we focus on the cases where the moving and fixed image have a similar appearance, which is most often the case for single-modal intra-patient registration.

2.2 Loss functions

To find the optimal transformation $\hat{\Phi}$ one could define the following minimization problem:

$$\hat{\Phi} = \underset{\Phi}{\text{argmin}}\, \mathcal{L}\left(M \circ \Phi, F\right). \tag{2.4}$$

Here $\mathcal{L}$ denotes the loss function. In this case, the loss function can be any function that measures similarity, *i.e.* a function that is low when $M \circ \Phi$ and F are similar, and high when they are dissimilar. An example of such a function is the L^p-norm.

This problem is, in general, ill-posed in the sense that no unique solution can be provided. A famous example highlighting this ill-posedness is the 'aperture problem'. This problem refers to the fact that the motion of a homogeneous contour is locally ambiguous. A common approach for solving ill-posed problems is to use regularization. For this particular problem, we can add a regularization term to the loss function that penalizes unrealistic transformation:

$$\begin{aligned}\hat{\Phi} &= \underset{\Phi}{\text{argmin}}\, \mathcal{L}\left(\mathcal{F}, \mathcal{M}, \Phi\right) \\ &= \underset{\Phi}{\text{argmin}}\, \mathcal{L}_{data}\left(M \circ \Phi, F\right) + \alpha \mathcal{L}_{reg}\left(\Phi\right)\end{aligned} \tag{2.5}$$

Here, α is a tunable regularization parameter that determines the proportionality between the data term and the regularization term. The higher the α, the smoother the solution, but this smoothness may come at the cost of violations in the data term.

For the data term we can use the mean squared error (MSE), which is a commonly used similarity measure. This term is defined as follows:

$$\mathcal{S}^{\text{MSE}}\left(\mathcal{M} \circ \Phi, F\right) = \frac{1}{|\Omega|} \int_{\Omega} \left[\mathcal{F}(x) - \left(\mathcal{M} \circ \Phi\right)(x)\right]^2 dx. \tag{2.6}$$

One thing that should be noted is that this only works because we use fixed and moving images that are similar in appearance. In other cases the intensity values can not be compared directly. In these cases, other similarity measures can be used such as cross-correlation and mutual information. These similarity measures do not require very similar appearances and hence are robust to variations in intensity between images.

2.3 Regularization

The goal of the regularization terms is to enforce smoothness on the solution to make it more realistic, and to disambiguate the solution. There are several possibilities for regularization. Here we present several regularization terms that we have considered. Several terms use the deformation u. This is simply the difference vector between the original and the transformed coordinate. So, the transformation can be described as adding the deformation to the input coordinate:

$$\Phi(x) = u(x) + x. \tag{2.7}$$

For clarity we omit the input x of the functions Φ and u everywhere inside the integrals, *e.g.* Φ is written to denote $\Phi(x)$.

A simple regularization term is $\mathcal{S}^{\text{jac}}$, which uses the Jacobian determinant:

$$\mathcal{S}^{\text{jac}}[\Phi] = \int_{\Omega} |1 - \det \nabla \Phi| dx. \tag{2.8}$$

A negative Jacobian determinant indicates folding and the loss of invertibility. A Jacobian determinant is also an indicator for expansion and shrinkage, where the Jacobian determinant is exactly the expansion factor. In practice, shrinkage and expansion occur, however, we want to limit great deviations from 1. This regularization term enforces smoothness and also penalizes the solution for not being diffeomorphic.

Several more complex regularization terms are proposed in the work of Ruthotto [6], the first one being the diffusion term $\mathcal{S}^{\mathsf{diff}}$:

$$\mathcal{S}^{\mathsf{diff}}[\Phi] = \frac{1}{2}\sum_{i=1}^{d}\int_{\Omega} |\nabla\Phi_i - x_i|^2\, dx = \frac{1}{2}\sum_{i=1}^{d}\int_{\Omega} |\nabla u_i|^2\, dx. \tag{2.9}$$

A limitation of this regularization term is that the components of the transformation are decoupled, so solutions might not be physically meaningful. The next term is the elastic term $\mathcal{S}^{\mathsf{elas}}$:

$$\begin{aligned} \mathcal{S}^{\mathsf{elas}}[u] &= \int_{\Omega} \nu(\mathsf{tr}V)^2 + \mu\,\mathsf{tr}\left(V^2\right) dx, \\ \text{with} \quad V = V(u) &= \left(\nabla u + \nabla u^\top\right)/2. \end{aligned} \tag{2.10}$$

In contrast to the previous term, here there is some interplay between the different components of the transformation. However, its functionality is limited to small strains $\|u\| \ll 1$. A term that works better with larger displacements is $\mathcal{S}^{\mathsf{quad}}$:

$$\begin{aligned} \mathcal{S}^{\mathsf{quad}}[\Phi] &= \int_{\Omega} \nu(\mathsf{tr}E)^2 + \mu\,\mathsf{tr}\left(E^2\right) dx, \\ \text{with} \quad E = E(u) &= \left(\nabla u + \nabla u^\top + \nabla u^\top \nabla u\right)/2. \end{aligned} \tag{2.11}$$

The fact that this term works significantly better than the previous term in cases of large displacements comes at the cost that a unique optimal solution is not guaranteed to exist anymore. Unlike $\mathcal{S}^{\mathsf{jac}}$, both $\mathcal{S}^{\mathsf{elas}}$ and $\mathcal{S}^{\mathsf{quad}}$ do not control tissue compression and expansion in any way.

The following term $\mathcal{S}^{\mathsf{Ogden}}$, on the other hand, does take into account the compression and expansion of volume:

$$\mathcal{S}^{\mathsf{Ogden}}[\Phi] = \int_{\Omega} \frac{1}{2}\alpha_1|\nabla u|^2 + \alpha_{\mathsf{a}}\|\,\mathsf{cof}\,\nabla\Phi\|_F^2 + \alpha_{\mathsf{v}}\psi_{\mathsf{O}}(\mathsf{det}\,\nabla\Phi)dx. \tag{2.12}$$

Here, $\|\cdot\|_F^2$ is the squared Frobenius norm. For $\alpha_a > 0$ there is a bias towards transformations that reduce the surface areas, while $\alpha_a = 0$ would complicate existence theory. Since in general $\psi_{\mathsf{O}}(v) \neq \psi_{\mathsf{O}}(1/v)$ this functional is not inverse consistent. Swapping the moving and fixed image changes the transformation between the two, which is not desirable.

The final regularization term is the hyper-elastic term $\mathcal{S}^{\mathsf{hyper}}$:

$$\mathcal{S}^{\mathsf{hyper}}[\Phi] = \int_{\Omega} \frac{1}{2}\alpha_1|\nabla u|^2 + \alpha_{\mathsf{a}}\phi_{\mathsf{c}}(\mathsf{cof}\,\nabla\Phi) + \alpha_{\mathsf{v}}\psi(\mathsf{det}\,\nabla\Phi)dx, \tag{2.13}$$

with the convex functions:

$$\phi_{\mathsf{c}}(C) = \sum_{i=1}^{3} \mathsf{max}\left\{\sum_{j=1}^{3} C_{ji}^2 - 1, 0\right\}^2 \text{ and } \psi(v) = \frac{(v-1)^4}{v^2}.$$

This regularization term covers all the drawbacks that are previously mentioned.

2.4 Optimization

There are many software packages dedicated to iteratively optimizing the transformation Φ given an image pair F and M, ***e.g*** Elastix [7] and FAIR [8]. In general, this transformation is discretely defined, so the optimization comes down to finding the optimal deformation vectors at all pixel coordinates. A commonly used optimizer is gradient descent, in which steps are taken repeatedly in the opposite direction of the gradient of the loss function.

For this thesis, the transformation is implicitly represented. So, instead of directly optimizing the output of $\Phi(x)$ on a finite set of points, we optimize it across the entire domain Ω by adapting the weights of the network used to describe the transformation. For the optimization we use Adam [9], an extension to stochastic gradient descent. Adam uses adaptive per-parameter learning rates, adapted based on the first and second moment of the gradient.

Even though we use stochastic gradient descent to optimize a neural network, this method is not like typical deep learning methods, as it does not generalize to unseen images. The network is 'trained' for one specific image pair. There is still a generalizing aspect, in the sense that the network predicts the transformation even for coordinates that do not appear in the training, *i.e.* the coordinates in between pixel locations. However, when faced with a new image pair, a previously trained network is not useful, as its only use is describing the transformation for the image pair that it is optimized for.

2.5 Diffeomorphic image registration

Diffeomorphic image registration is important for many medical applications. In diffeomorphic image registration, the solution space of the transformation is constrained to the space of diffeomorphisms between Ω_F and Ω_M This constraint ensures certain desirable properties, such as the preservation of topology.

A map f between two manifolds is called a diffeomorphism when f is a bijection and both f and its inverse f^{-1} are differentiable. In general, a solution Φ to equation (2.4) is not necessarily a diffeomorphism, so extra measures have to be taken to ensure a diffeomorphic solution.

In order to limit our feasible solutions to the set of diffeomorphic transformations between the two images, we look at transformations induced by a velocity field v. Let $t = 0$ describe the time at which the moving image is taken, and $t = 1$ the time at which the fixed image is taken. Now let $v(x, t)$ describe the velocity of the object located at location x at time t, defined for all $x \in \Omega$ and $t \in [0, 1]$. A temporal dependency is added to Φ as well, so now $\Phi(x, t)$ is used to describe the movement on the interval $t \in [0, 1]$, where $\Phi(x, 1)$ represents the full deformation between the moving image and the fixed image. Note that $v(x, t)$ is nothing more than the derivative of $\Phi(x, t)$ with respect to time. This allows the following reformulation of the problem:

$$M\left(\Phi\left(x,1\right)\right) = F\left(x\right) \tag{2.14}$$

$$\begin{aligned} &\text{with} \quad \frac{d\Phi}{dt}\left(x,t\right) = v\left(\Phi\left(x,t\right),t\right) \\ &\text{and} \quad \Phi\left(x,0\right) = x. \end{aligned}$$

For clarity the temporal inputs can be moved to the subscript position:

$$\frac{d\Phi}{dt}(x) = \left(v_t \circ \Phi_t\right)(x). \tag{2.15}$$

By taking the integral it can be written as:

$$\Phi_t(x) = x + \int_0^t (v_\tau \circ \Phi_\tau)(x) d\tau. \tag{2.16}$$

Looking at Φ defined in this manner, it is possible to construct the inverse function Φ^{-1}:

$$\Phi_t^{-1}(x) = x - \int_0^t \left(v_\tau \circ \Phi_\tau^{-1}\right)(x) d\tau. \tag{2.17}$$

So, there exists an inverse for any $x \in \Omega$. Under the condition that Ω is closed under the operation Φ, the existence of the inverse implies that Φ is a bijection. When, for instance, the deformation is 0 at the boundary of Ω, we have that Ω is closed under Φ, hence Φ is a bijection from Ω onto itself.

A discrete approximation of the integral in equation (2.16) can be computed as follows:

$$\begin{aligned} &\Phi^{(0)} = Id \\ &\Phi^{(t)} = \Phi^{(t-1)} + \frac{1}{T} v^{(t-1)}\left(\Phi^{(t-1)}\right) \quad \text{for } 1 \leq t \leq T. \end{aligned} \tag{2.18}$$

In this discrete context $\Phi^{(t)}$ is used to denote $\Phi(x, \frac{t}{T})$, meaning $\Phi^{(T)}$ forms an approximation of the final transformation $\Phi(x, 1)$.

A popular alternative for the integration of velocity fields is scaling and squaring, and several papers report good results using this technique, such as DARTEL [10] and VoxelMorph [11]. It relies on scaling down the velocity field and repeatedly squaring it. This method requires the velocity field to be static (time-invariant) and gives the exact same approximation as the scheme in (2.18). However, scaling and squaring only requires $\log_2(T)$ steps to compute the final solution, where the regular scheme requires T steps. It leverages the fact that for static velocity fields the transformation satisfies the following equation: $\Phi^{(t)} \circ \Phi^{(s)} = \Phi^{(t+s)}$. For scaling and squaring the scheme looks like:

$$\begin{aligned} &\Phi^{(1)} = Id + \frac{1}{T} v \\ &\Phi^{(2^t)} = \Phi^{(2^{t-1})} \circ \Phi^{(2^{t-1})} \quad \text{for } 1 \leq t \leq \log_2(T). \end{aligned} \tag{2.19}$$

So, the main advantage of scaling and squaring is that it is very fast, however, it only works for static velocity fields.

Regularization for the velocity field

When using time-dependent velocity fields without any further constraints, the solution is definitely not unique. So, we can use regularization to disambiguate the solution and enforce the velocity field to change as little as possible over time. This results in the following definition of $\mathcal{S}^{\text{time}}$, as a function of the velocity field v:

$$\mathcal{S}^{\text{time}}[v] = \int_0^1 \int_\Omega \| \frac{\partial}{\partial t} v(x,t) \|_2 dx\, dt. \tag{2.20}$$

Chapter 3

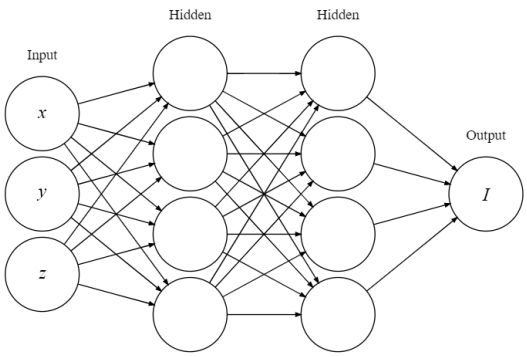
Neural Implicit Representation

3.1 Overview

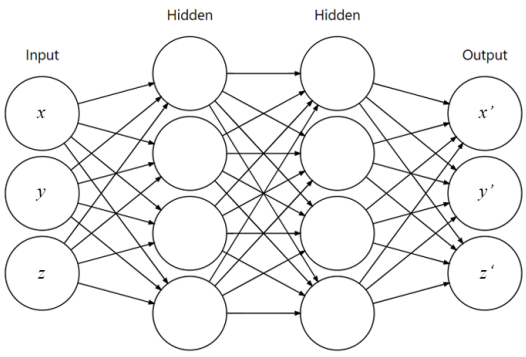
Generally, any digital image is represented by a discrete set of pixels. This offers only a discretized view of the real-world scene that is continuous. To represent images in a continuous way we have to parametrize a function that maps any image coordinate from the continuous image domain to an intensity value. For this, we use a fully connected feedforward neural network, which is often referred to as a multilayer perceptron (MLP). An example of an MLP for the implicit representation of an image is depicted in Figure 3.1a. Not only images can be represented by an MLP, but also the coordinate transformation Φ that we wish to find in the image registration problem. Figure 3.1b shows an example of an MLP that can represent such a transformation.

By learning an implicit representation we get a function that is defined on the entire continuous image domain. Also, since we use MLPs, we automatically end up with a differentiable function and its derivatives can be easily computed. Since we now have a continuous and differentiable representation instead of a discrete one, techniques such as interpolation and finite differences become completely redundant.

Completely omitting techniques such as interpolation and finite differences offers a big advantage, as these techniques introduce errors. With implicit representations, the discrepancy between theory and practice depends solely on how accurately the implicit representation describes the true underlying distribution.



(a) MLP for implicit representation of a grayscale 3D image.



(b) MLP for implicit representation of a coordinate transformation.

Figure 3.1: Visualization of example MLPs for the implicit representation of (a) images and (b) coordinate transformations.

(a) Target (b) Standard MLP (c) Positional encoding (d) SIREN

Figure 3.2: Implicit representations of an image using different networks. Figure (a) displays the target image. Figures (b), (c) and (d) display the output using a standard MLP, a standard MLP with basic positional encoding, and a SIREN network respectively. All networks were optimized for 500 iterations over the full image. The peak signal-to-noise ratio (PSNR) is given in the top-left corner of the images. A higher PSNR indicates a higher similarity to the target image.

3.2 Overcoming the low-frequency bias

In Figure 3.2b we see the result of fitting an image using a standard MLP with ReLU (rectified linear unit) activation functions. Clearly, the result is a low-frequency approximation of the image and it lacks high-frequency details. There are several papers that address this spectral bias, such as the work of Rahaman *et al*. [12]. The conclusion is that, in general, standard MLPs are inherently biased towards a low-frequency output making them unsuitable for implicitly representing signals of higher frequency without any adaptations.

In the work of Tancik *et al.* [13], this spectral bias of MLPs is studied using knowledge from the neural tangent kernel literature. They claim that as the width of the layers tends to infinity and the learning rate tends to zero, a standard MLP converges to the solution of kernel regression using a neural tangent kernel (NTK). It is shown that in kernel regression using the NTK, the i^{th} component of the absolute prediction error in the eigenbasis of the NTK decays approximately exponential at the rate $\alpha\lambda_i$, where α is the learning rate and λ_i is the i^{th} eigenvalue. Since high-frequency components correspond to the lower eigenvalues, they will converge slower. For standard MLPs the eigenvalues decay rapidly, causing extremely slow convergence for the high-frequency components.

We discuss two different solutions for overcoming the spectral bias, namely positional encoding and periodic activation functions.

Positional encoding

In NeRF [2] and many other subsequent works, positional encoding is used to overcome the spectral bias. With positional encoding the input of the network is transformed using a so-called positional encoding layer. This layer uses a Fourier feature map to map the input x of the network onto the surface of a higher dimensional hypersphere. The output $p(x)$ of a positional encoding layer is defined as follows:

$$\begin{aligned} p(x) &= [p_1(x_1), \cdots p_n(x_n)]^\mathsf{T}, \\ p_i(x_i) &= [\sin(B_{i,1}x_i), \cos(B_{i,1}x_i), \cdots, \sin(B_{i,L}x_i), \cos(B_{i,L}x_i)]^\mathsf{T}. \end{aligned} \tag{3.1}$$

Here, $B \in \mathbb{R}^{n \times L}$ is a parameter, and $L \in \mathbb{N}$ determines the dimensionality of the output of the positional encoding layer. The output of this layer functions as the input for the original neural

network. The size of the input for the neural network effectively goes from n to $2L \cdot n$. We can see that the positional encoding layer does indeed map the input to the surface of a $2L \cdot n$-dimensional hypersphere, as $\|p(x)\|_2 = \sqrt{L}$ regardless of x and B.

For standard positional encoding, as is used in NeRF [2], the value of B is chosen as $B_{i,j} = 2^j \pi$. In the work of Tancik *et al.* [13], another option for B is proposed that yields improved results compared to the standard B. Here each entry of B is sampled from a zero-mean normal distribution.

Figure 3.2c depicts the implicit representation of an image with the help of positional encoding. It already gives a significant improvement compared to a standard MLP, however it is still easily distinguishable from the original image.

Periodic activation functions

A commonly used activation function is the ReLU activation function. A typical neural network Φ using this activation function is defined as follows:

$$\Phi(x) = (\phi_n \circ \phi_{n-1} \circ \ldots \circ \phi_1)(x), \quad \phi_i(x) = \mathsf{max}(W_i x + b_i, 0). \tag{3.2}$$

Here ϕ_i represents the i^{th} layer of the network, with weight matrix W_i and bias vector b_i. As previously mentioned, these networks have an inherent bias to low-frequency solutions. However, this bias can be combated by changing the activation function. In SIREN [14], the usage of a sine as periodic activation function is proposed. This yields the following function for the neural network Φ with input x:

$$\Phi(x) = (\phi_n \circ \phi_{n-1} \circ \ldots \circ \phi_1)(x), \quad \phi_i(x) = \mathsf{sin}\left(\omega\left(W_i x + b_i\right)\right). \tag{3.3}$$

Here, the ω is a hyper-parameter that can be tuned. For values of ω that are close to 1, this network performs quite similarly to the ReLU-network. However, increasing ω lets high-frequency components of the solution converge faster. The optimal value will differ depending on the problem, but SIREN reports $\omega = 30$ to work well across a wide range of different tasks.

These periodic activation functions allow neural networks to learn higher frequency details in a similar manner as positional encoding. With positional encoding, we have a positional encoding layer at the start of the network where the input is passed through several sine functions. In the network of SIREN, every single layer of the network resembles the positional encoding layer.

A big advantage of SIREN is that it can represent infinitely many higher-order derivatives. On the other hand, the derivatives of order 2 and higher of an MLP using a ReLU activation function are always 0.

In Figure 3.2 we see that SIREN performs way better than a standard MPL, and is able to capture the high-frequency details of the target image. Even compared to the approach with positional encoding, the periodic activation functions offer a big improvement. For this reason, we use SIREN-networks for the neural implicit representations in this thesis.

3.3 In practice

In Figure 3.3, we see the evolution of the output of the network during optimization. Every iteration the output resembles the target image more closely. Already within 100 iterations, the output is difficult to distinguish from the target image. Only the details of the highest frequency are not yet fully present at that time.

(a) Target (b) 1 iteration (c) 3 iterations (d) 10 iterations (e) 30 iterations (f) 100 iterations

Figure 3.3: Evolution of the implicit representation during optimization, using SIREN.

The implicit representation must match the original image on the coordinates of the grid on which the original image is defined. However, it is also important that the implicit representation generalizes well, *i.e.* the implicit representation should reasonably predict the intensity value for points that do not lie exactly on the grid.

Figure 3.4 visualizes how well the network generalizes to unseen coordinates. We can see that the capability of generalizing heavily depends on the value of ω. For high values of ω the network overfits on the image used for training. In Figure 3.4f we see that for $\omega = 70$ the output of the network is basically noise that happens to correspond to the training image at the training coordinates.

For low values of ω, the high-frequency details take a long time to converge. But as discussed previously, a value for ω that is too high leads to overfitting. The optimal value for ω depends on the training image. For the example in Figure 3.4, the training data is of very low frequency, as it is limited by its low resolution. For practical applications, the optimal value will generally be higher than the optimal value for this example.

(a) Ground truth (b) Training data (c) $\omega = 10$ (d) $\omega = 30$ (e) $\omega = 50$ (f) $\omega = 70$

Figure 3.4: Implicit representation on unseen coordinates for different values of ω. Figure (a) shows the ground truth image and (b) shows a lower resolution version of the same image that is used for the training. Figures (c), (d), (e) and (f) display the learned implicit representations for different values of ω, using the training data from Figure (b). All of these representations are practically indistinguishable from the training data (b) when sampled only at the training coordinates, all having a mean squared error $< 10^{-5}$. However, when sampled at a higher resolution, as is done in the figures, we see that not all representations are very similar to the ground truth in (a). Especially for higher values of ω the network overfits on the training coordinates and does a poor job at generalizing to unseen coordinates.

Chapter 4

Mathematical models

In this chapter, we explain the architectures of the registration models used for the experiments in this thesis. Each model acts as a deformation function $\Phi : \Omega \to \Omega$ that maps a spatial coordinate x from the image domain $\Omega \subset \mathbb{R}^n$ to another spatial coordinate y in the same domain Ω.

We differentiate between two types of models. For the first model, we use an MLP to directly represent the deformation u. The transformation Φ is then given by $\Phi(x) = u(x) + x$. We call this model $\mathcal{D}$. For the second model, we use an MLP to represent a time-dependent velocity field instead. This velocity field is then integrated to produce the deformation. This model will be denoted by $\mathcal{V}$.

As discussed in section 3.2, standard MLP's inherently struggle to represent high-frequency details. Even though the general motion between images can often be captured by a deformation of a low frequency, there are many occasions of very detailed local motions which require a higher frequency to be described. In order to adequately describe these local motions, we need our MLP to be capable of representing high-frequency details. For this, we use a SIREN-network as previously discussed in section 3.2.

4.1 Small deformation model

The model $\mathcal{D}$ that uses a network to directly represent the deformation u is displayed in figure 4.1. The model is straightforward and consists of only a single network. The input coordinate x is fed directly to the network $\Psi_{\mathcal{D}}$. The output gives us the deformation u which gives us the transformation Φ after adding the identity, $\Phi = u + Id$.

The network takes only a singular coordinate x as input, instead of a whole grid. However, in practice, a batch of coordinates is propagated through the network simultaneously in order to compute multiple outputs in parallel.

This model does not necessarily provide a diffeomorphic registration. There are regularization techniques that will reduce the likelihood of violating the diffeomorphic properties in the pre-

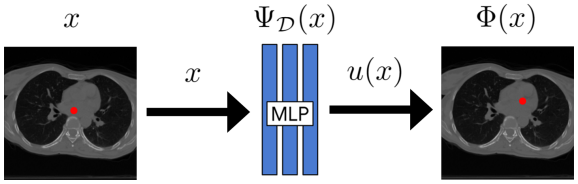


Figure 4.1: Registration-model $\mathcal{D}$.

dicted Φ. However, a perhaps more effective method of obtaining a diffeomorphic registration is predicting a deformation through integrating a velocity field, as discussed in section 2.5. This is what prompts the next model.

4.2 Diffeomorphic model

For a diffeomorphic approach to image registration we want to predict the deformation Φ by integrating a velocity field. For this purpose we use an MLP $\Psi_{\mathcal{V}}$ to describe the time-dependant velocity field. The network takes as input a spatial coordinate x and a time t and it will return the prediction of the velocity field $v(x,t)$ at this coordinate. By doing this we obtain an implicit representation of the velocity field v, that is defined for any $x \in \Omega$ and $t \in [0,1]$. Using this representation of the velocity field we can compute the solution of the following scheme for a discrete approximation of the integral, as previously defined in section 2.5:

$$\begin{aligned} \Phi^{(0)} &= Id \\ \Phi^{(t)} &= \Phi^{(t-1)} + \frac{1}{T} v^{(t-1)} \circ \Phi^{(t-1)} \quad \text{for } 1 \leq t \leq T. \end{aligned} \tag{4.1}$$

For our model, we opt to not use scaling and squaring. The main reason for this is that it requires spatial interpolation. When the velocity field is only defined on a grid, spatial interpolation is required regardless of the integration method. However, since we have a velocity field that is defined for any spatial coordinate, spatial interpolation is not a requirement anymore. Hence we prefer to stay away from spatial interpolation, as the lack of interpolation is one of the big advantages of using implicit representations. Another advantage of not using scaling and squaring is that it allows us to use dynamic (time-dependant) velocity fields. This puts less of a restriction on the solution space, as it does not limit us to solely deformations that can easily be expressed using a static velocity field.

An overview of the model is displayed in figure 4.2. In essence, the MLP $\Psi_{\mathcal{V}}(x,t)$ is evaluated T times in a row. For each MLP the input is slightly different. The input for the i^{th} MLP is (x_{i-1}, t_{i-1}). The t-component of the input is gradually increased each step, with $t_i = i/T$. The x-component of the input for the MLPs is determined by the scheme in (4.1), with $x_i = \Phi^{(i)}(x_0)$, or in other words, the predicted transformation of the coordinates up to time t_i. The output of each MLP is the velocity field $v(x,t)$, which is used in the scheme to determine the input for the next MLP.

Propagating through the network multiple times to compute the final transformation is very slow, so implementing the model like this is not very practical. It is, however, interesting to see how the results compare to the small deformations model.

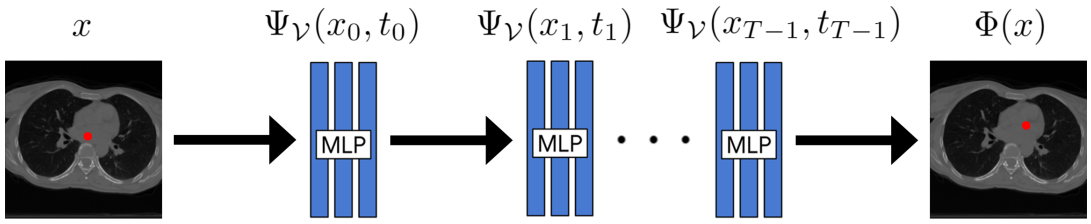


Figure 4.2: Registration-model $\mathcal{V}$.

4.3 Implementation

What separates our models from existing ones is that the images are implicitly represented by neural networks. This allows us to use arbitrary coordinates during training, instead of only the coordinates of pixels. Also, a major advantage of the neural implicit representations lies in the implementation of regularization. The regularization depends on derivatives of the images, so traditional models require numerical methods such as finite differences to compute the regularization terms. On the other hand, for our model the image derivatives can be easily computed, as neural networks are designed to be easily differentiable. This makes the implementation of regularization trivial. For instance, the Jacobian matrix ∇u can be computed per row using the automatic differentiation package from PyTorch as follows:

```
for i in range(n):
    jacobian[i, :] = torch.autograd.grad(network_output[i], image_coordinate)
```

Here n denotes the dimensions of the data. For our experiments we have $n = 3$, since we consider 3D CT scans. Since $\Phi(x) = u(x) + x$, the Jacobian matrix $\nabla\Phi$ can be computed by adding the $n \times n$ identity matrix to ∇u. All the regularization terms mentioned in section 2.3 follow directly from these two Jacobian matrices.

Chapter 5

Results

In this section, we present the results of our registration models on 3D CT scan data. Before evaluating the performance of the model on this data, we motivate our choice for the value of ω based on an experiment. We also take a look at the visual results of regularization in our model.

When measuring the similarity between images, the error is better visualized on the logarithmic scale. That is why we use the peak signal-to-noise ratio (PSNR) instead of the MSE in the figures. For images with a maximum intensity of 1 the PSNR is defined as:

$$\text{PSNR} = -10 \log_{10} (\text{MSE}) . \tag{5.1}$$

A high PSNR value indicates a low MSE, *i.e.* a great similarity between the images.

5.1 Implementation

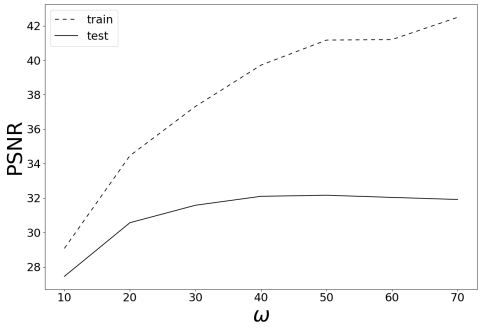
We implemented our model using PyTorch, making use of the automatic differentiation package. For the implicit representations of the images, we opt for an MLP with 3 hidden layers of width 512. We set the learning rate to 10^{-4}. The batch size is 10^6 coordinates. For both registration models the MLPs have 3 hidden layers of width 512, a learning rate of 10^{-6} and a batch size of 5000. For model $\mathcal{V}$, we compute the integral of the velocity field in 16 steps, *i.e.* $T = 16$.

The training is done on an NVIDIA Quadro RTX 6000 24GB GPU. For the implicit representation of the images the training occurs at a rate of 20.1 iterations per second, or 2,010,000 coordinates per second. Registration model $\mathcal{D}$ has a rate of 16.9 iterations per second, or 84,500 coordinates per second. Finally, model $\mathcal{V}$ has a rate of 2.3 iterations per second, or 11,500 coordinates per second.

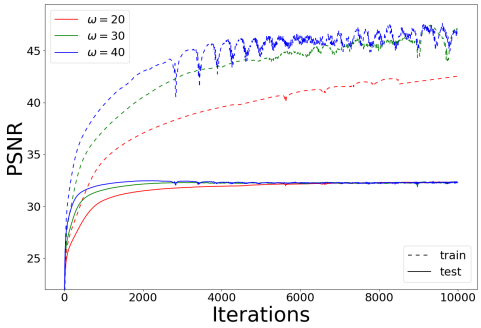
5.2 Implicit representation

As mentioned in section 3.3, it is important to choose the correct value for ω when using SIREN to represent a signal. This ω depends on the data, most importantly on the presence of high-frequency details in the data. For the sake of our experiments, we are interested in a value of ω that works well for representing CT scans of the lungs.

In order to find the optimal value for ω, we trained neural networks using a scaled-down version of a 3D CT scan of the lungs, part of the DIR-Lab data set. By scaling down the image we optimize the network using only a subset of the available voxels. The voxels that are not used during this optimization give us the ground-truth information that we can use to evaluate the ability of the network to generalize. We scaled down the image by a factor of 2 in all three dimensions, meaning only an eighth of the original image is used during optimization.



(a) Train and test loss after 1000 iterations for different values of ω.



(b) Train and test loss over time for different values of ω.

Figure 5.1: Performance of SIREN on both the training and the test data for different values of ω.

In Figure 5.1a we see the performance of a SIREN network on both the training and test coordinates after 1000 iterations. We can see that training performance is increasing in ω, but the performance on the test coordinates does not improve by increasing ω to a value higher than 40. However, this graph does not necessarily tell us anything about the convergence of the networks.

For the convergence behavior we look at Figure 5.1b. For all three values of ω depicted in the figure, the network converges to the same PSNR-value. A higher value of ω seems to increase the rate of convergence. The actual results for the different values of ω are depicted in Figure A.1 in the appendix.

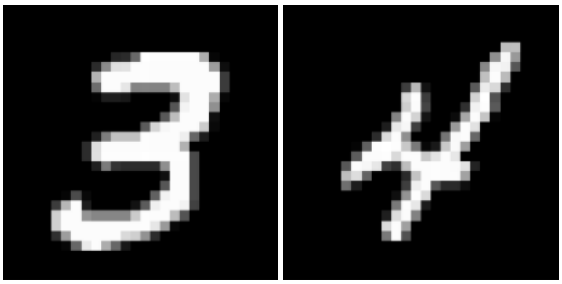
In conclusion, the results are very similar for many different values of ω and there is no clear reason to deviate from $\omega = 30$, which was reported in the SIREN paper as a value that generally works well.

5.3 Regularization

Now, we look at the effects of the different regularization terms by registering two simple images from the MNIST data set of handwritten digits [15]. These images are, of course, very different from the data set of 3D CT scans that we use for the other experiments. However, using these simple images gives us a rough idea of the effect of the regularization terms. We chose an image of the digits 3 and 4 for the moving and fixed image respectively, both depicted in Figure 5.2.

We intentionally picked fixed and moving images that do not have an obvious transformation between them. This way we can get a good indication of the behavior of the regularization terms. Does it allow unrealistic transformations in order to be able to fit the data, or does it enforce smoothness at the cost of not fitting the data perfectly?

In Figure 5.3 we see the registration results for all regularization terms, using model $\mathcal{D}$ and several different values for α. A grid overlay is placed on the images to show how the images are deformed. The grid is colored using a color gradient. If, for instance, the registration algorithm would flip the image the colors would indicate this, while a unicolored grid would not. The original coloring of the grid can be inferred from the right-most column, where deformations are minimal.



(a) Moving image. (b) Fixed image.

Figure 5.2: Fixed and moving image from MNIST data set of handwritten digits [15].

The left-most column shows the results for $\alpha = 0$, which are, of course, the same for all terms. In this column, the moving image is perfectly recreated by transforming the moving image. However, the manner in which the image is transformed is very chaotic, and the colored overlay does not at all resemble a grid anymore. It is clear that this is undesired behavior for the registration. On the other hand, in the column for $\alpha = 1$ the transformations are overly smooth, to the point where the moving image is barely deformed at all. The optimal value of alpha for all terms lies somewhere in between.

Figure 5.3: A visualization of the effects all regularization terms.

We can see that for many of the regularization terms there is unwanted local behavior. Most notably apparent with $\mathcal{S}^{\text{Jac}}$ and $\mathcal{S}^{\text{quad}}$. For these terms, there are some irregular artifacts present, even for values of α as high as 0.1. Furthermore, with exception of the $\mathcal{S}^{\text{hyper}}$, many of the resulting deformations contain unwanted and unnecessary oscillations.

As motivated in section 2.3, $\mathcal{S}^{\text{hyper}}$ covers certain weaknesses that the other terms have. Also, the behavior of this term in Figure 5.3 looks more realistic compared to other terms. The solutions with the hyper-elastic regularization term approach the fixed image while maintaining a realistic transformation. Especially the global behavior of this term seems more desirable than most other terms. Consequently, we use the hyper-elastic regularization term $\mathcal{S}^{\text{hyper}}$ for the rest of our experiments.

The smoothness of the solution is clearly influenced by the regularization term, but also by the parameter ω. In Figure 5.4 we see the interplay of ω and α, when using hyper-elastic regularization. We can see that the solutions do depend on the choice of ω, especially for low values of α. However, for higher values of α, changes in ω do not change the solution as much. Later, we evaluate the optimal values for α and ω based on the performance on a 3D CT scan data set.

In figure 5.5 we see an example of the impact of regularization in an example of registering 3D CT-scan data. It shows the solution that model $\mathcal{D}$ gives us with and without the use of hyper-elastic regularization. Without regularization, the transformation is not realistic at all. The transformation is not smooth and has very strange local behavior in some places.

By looking at the determinant of the Jacobian in 5.5b we see that it is far from 1 at many places. It reaches values up to 4 on this particular slice, indicating extreme expansion. This determinant is even negative in some regions, meaning that folding is happening there.

On the other hand, the registration with regularization gives a transformation that looks much more natural. Also, the determinant of the Jacobian is much more stable. In this slice the Jaco-

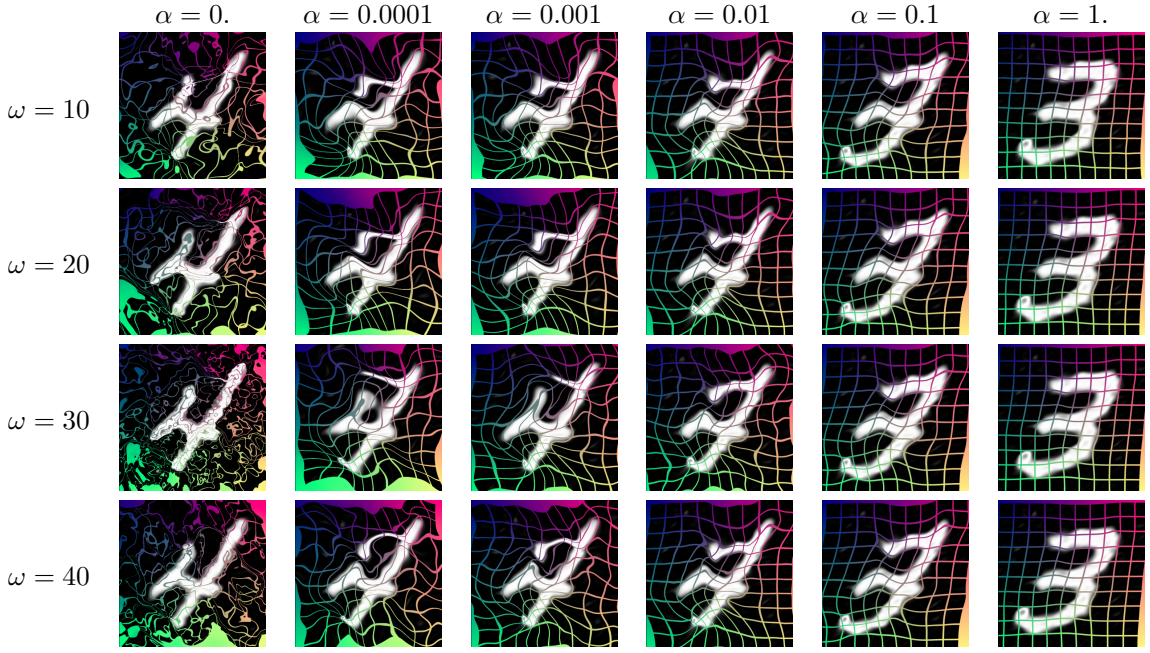
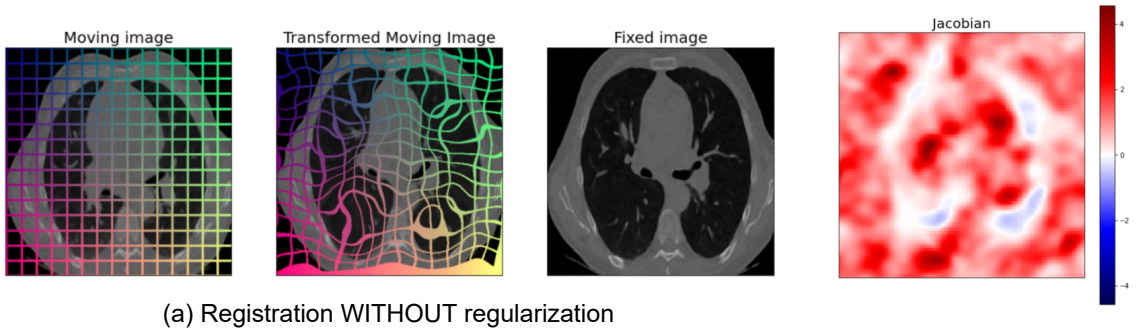


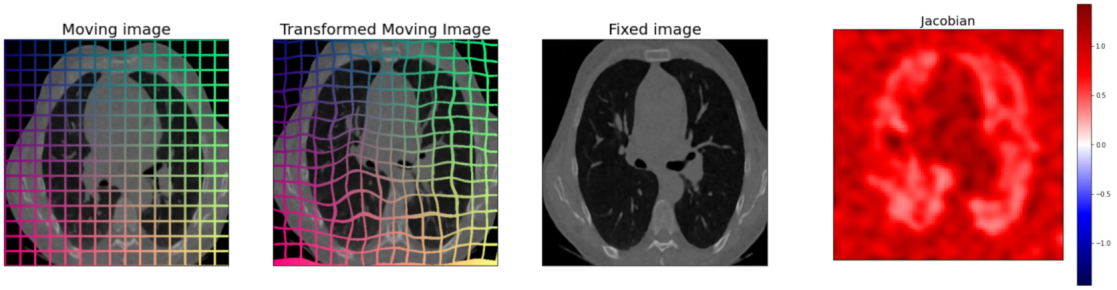
Figure 5.4: Effect of hyper-parameters α and ω.

bian determinant is close to 1 in most places, indicating that only slight expansion and shrinkage occurs. So, even though the similarity between $M \circ \Phi$ and F is comparable across both depicted solutions, the solution with regularization is much more realistic.



(a) Registration WITHOUT regularization

(b) Jacobian



(c) Registration WITH regularization

(d) Jacobian

Figure 5.5: Example showing the impact of regularization on real 3D CT scan data from the Learn2Reg Challenge [16].

5.4 Evaluating performance

For evaluating the performance of the registration models we use the DIR-Lab data set [17], containing 3D CT scan data of the lungs. We specifically use the 4DCT data set, containing 10 sets of thoracic CT images, acquired as part of the radiotherapy planning process for the treatment of thoracic malignancies at the University of Texas M. D. Anderson Cancer Center in Houston, TX. The first five sets contain images with a resolution of $256 \times 256 \times 100$ and the resolution of the images in the last five sets is $512 \times 512 \times 128$. Along with this data set, 300 manual landmarks are provided that can be used to evaluate the accuracy of the registration. For more details, see the DIR-Lab website: https://www.dir-lab.com/.

The average registration error in millimetres e is computed in "snap-to-voxel" fashion, rounding the predicted landmark position to the nearest integer voxel position. Using $\lfloor x \rceil$ to denote rounding x to the nearest integer voxel position, the average registration error of the predicted transformation Φ is computed as follows:

$$e = \frac{1}{N} \sum_{i=1}^{N} \sqrt{d_x^2 \left\lfloor \Phi(x_i^{(F)}) - x_i^{(M)} \right\rceil^2 + d_y^2 \left\lfloor \Phi(y_i^{(F)}) - y_i^{(M)} \right\rceil^2 + d_z^2 \left\lfloor \Phi(z_i^{(F)}) - z_i^{(M)} \right\rceil^2}. \tag{5.2}$$

Here, $x_i^{(F)}$ and $x_i^{(M)}$ denote the x location of the i^{th} landmark in the fixed and moving image respectively. The constants d_x, d_y and d_z are the voxel dimensions in millimetres.

First, we are interested in the best values for both α and ω. In order to make an appropriate choice for these hyper-parameters, we perform a grid search. For both model $\mathcal{D}$ and $\mathcal{V}$ we test various combinations of α and ω by optimizing the network for 1000 iterations on the first image pair in the data set. The goal is to find the optimal combination of these hyper-parameters.

Tables 5.1 and 5.2 display the average registration error e for registration models $\mathcal{D}$ and $\mathcal{V}$ respectively, after 1000 iterations, for the different combinations of α and ω. The values reported in Tables 5.1 and 5.2 are visualized in Figure 5.6. We can see that for both model $\mathcal{D}$ and $\mathcal{V}$, the surfaces are almost completely convex. Also the results in both tables are very similar. Only for low values of α there is a significant difference, where the model $\mathcal{V}$ performs better. This makes sense, as the architecture of model $\mathcal{V}$ inherently regularizes the solution.

For both models, the optimal values for α and ω based on this experiment are 0.01 and 30 respectively. These are the values that we use for evaluating the models on the entire data set.

For comparison, we consider two different models of which the performance on the DIR-Lab data set is known. We refer to these models as isoPTV [18] and CNN [19].

The model isoPTV is a variational method that uses isotropic total variation as the regularization term. A numerical solution is computed using the alternating direction method of multipliers (ADMM). This solution consists of a displacement vector for every pixel in the image. For more details on this method we refer to [18].

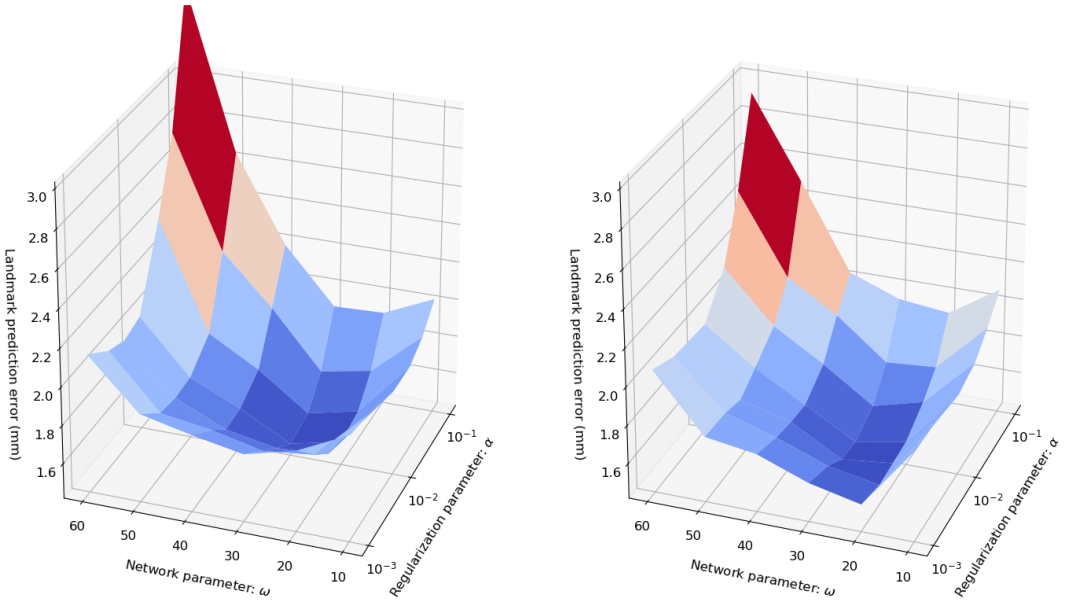
The model CNN uses a convolutional neural network to predict the transformation between two images. The network takes two images as input and gives as output three maps for the x, y,

	$\omega = 10$	$\omega = 20$	$\omega = 30$	$\omega = 40$	$\omega = 50$	$\omega = 60$
$\alpha = 0$	2.09(1.33)	2.04(1.44)	2.11(1.40)	2.07(1.49)	2.19(1.63)	2.40(1.87)
$\alpha = 0.001$	1.97(1.29)	1.84(1.42)	1.77(1.24)	1.82(1.33)	1.88(1.56)	2.14(1.72)
$\alpha = 0.0025$	1.94(1.29)	1.68(1.40)	1.65(1.27)	1.71(1.30)	1.75(1.58)	2.03(1.68)
$\alpha = 0.005$	1.91(1.26)	1.58(1.32)	1.56(1.28)	1.61(1.30)	1.73(1.56)	1.99(1.70)
$\alpha = 0.01$	1.86(1.28)	1.54(1.31)	**1.48(1.28)**	1.62(1.33)	1.74(1.57)	2.02(1.76)
$\alpha = 0.025$	1.86(1.30)	1.57(1.31)	1.51(1.29)	1.70(1.42)	1.85(1.65)	2.42(2.08)
$\alpha = 0.05$	1.92(1.38)	1.69(1.37)	1.64(1.39)	1.94(1.54)	2.20(1.91)	2.79(2.22)
$\alpha = 0.1$	2.03(1.50)	1.91(1.49)	1.90(1.54)	2.19(1.75)	2.64(2.16)	3.47(2.57)

Table 5.1: Landmark error: α versus ω, registration model $\mathcal{D}$ after 1000 iterations. The reported value denotes the average error in millimeters (and standard deviation).

	$\omega = 10$	$\omega = 20$	$\omega = 30$	$\omega = 40$	$\omega = 50$	$\omega = 60$
$\alpha = 0$	1.92(1.21)	1.61(1.31)	1.70(1.40)	1.79(1.49)	1.89(1.54)	2.17(1.80)
$\alpha = 0.001$	1.89(1.23)	1.56(1.28)	1.62(1.39)	1.72(1.45)	1.76(1.52)	2.06(1.76)
$\alpha = 0.0025$	1.88(1.23)	1.52(1.28)	1.57(1.38)	1.67(1.38)	1.75(1.54)	2.00(1.71)
$\alpha = 0.005$	1.86(1.24)	1.53(1.30)	1.50(1.33)	1.65(1.40)	1.72(1.52)	1.99(1.66)
$\alpha = 0.01$	1.84(1.25)	1.54(1.31)	**1.49(1.32)**	1.64(1.38)	1.79(1.55)	1.98(1.72)
$\alpha = 0.025$	1.88(1.32)	1.62(1.31)	1.55(1.38)	1.73(1.41)	1.89(1.67)	2.16(1.93)
$\alpha = 0.05$	1.97(1.40)	1.71(1.38)	1.69(1.44)	1.90(1.51)	2.06(1.79)	2.48(2.16))
$\alpha = 0.1$	2.08(1.48)	1.91(1.47)	1.94(1.53)	2.04(1.68)	2.49(2.12)	2.92(2.18)

Table 5.2: Landmark error: α versus ω, registration model $\mathcal{V}$ after 1000 iterations. The reported value denotes the average error in millimeters (and standard deviation).



(a) Model $\mathcal{D}$ (b) Model $\mathcal{V}$

Figure 5.6: Surface plots of the performance for different values of α and ω, visualization of Tables 5.1 and 5.2

and z components of a thin-plate spline transformation grid. The network is trained on random synthetic transformations that have been applied to CT images of the lungs. For more details regarding this method we refer to [19].

The registration error for our models $\mathcal{D}$ and $\mathcal{V}$ are shown alongside the models isoPTV and CNN in Table 5.3. The models $\mathcal{D}$ and $\mathcal{V}$ are trained for 100,000 iterations on each of the 10 image pairs in the data set. The results of models $\mathcal{D}$ and $\mathcal{V}$ are nearly equal on all the data. Only on 4DCT8, the model with the largest displacement, there is a significant difference, where model

	Model $\mathcal{D}$	Model $\mathcal{V}$	isoPTV[1]	CNN[2]	Displacement[3]	Observers[4]
4DCT1	1.18(1.20)	1.19(1.21)	0.76(0.90)	1.65 (0.89)	4.01 (2.91)	0.85 (1.24)
4DCT2	1.31(1.52)	1.30(1.53)	0.77(0.89)	2.26 (1.16)	4.65 (4.09)	0.70 (0.99)
4DCT3	3.01(2.70)	2.98(2.71)	0.90 (1.05)	3.15 (1.63)	6.73 (4.21)	0.77 (1.01)
4DCT4	2.93(2.66)	2.93(2.68)	1.24 (1.29)	4.24 (2.69)	9.42 (4.81)	1.13 (1.27)
4DCT5	2.69(2.77)	2.69(2.77)	1.12 (1.44)	3.52 (2.23)	7.10 (5.14)	0.92 (1.16)
4DCT6	4.46(3.29)	4.42(3.26)	0.85 (0.89)	3.19 (1.50)	11.10 (6.98)	0.97 (1.38)
4DCT7	6.52(5.63)	6.51(5.62)	0.80 (1.28)	4.25 (2.08)	11.59 (7.87)	0.81 (1.32)
4DCT8	9.52(8.31)	9.32(8.29)	1.34 (1.93)	9.03 (5.08)	15.16 (9.11)	1.03 (2.19)
4DCT9	4.37(2.73)	4.38(2.76)	0.92 (0.94)	3.85 (1.86)	7.82 (3.99)	0.75 (1.09)
4DCT10	3.59(3.58)	3.57(3.66)	0.82 (0.89)	5.07 (2.31)	7.63 (6.54)	0.86 (1.45)
Average	3.96	3.93	0.95	4.02	8.52	0.88

Table 5.3: Landmark error: models evaluated for all ten image sets from the DIR-Lab 4DCT data set. The reported values denote the average error in millimetres (and standard deviation). [1]Algorithm using isotropic total variation regularization [18]. [2] Model using supervised learning with convolutional neural networks [19]. [3]The displacement column displays the average displacements of the objects in the fixed and moving images. [4]The observer column shows the average repeat registration error of the experts.

$\mathcal{V}$ performs slightly better.

When we compare our models to the other models, we see that our performance is comparable to the performance of model CNN. On the entire data set, our models perform on average slightly better than model CNN. Especially for the cases of small displacements, our model compares favorably.

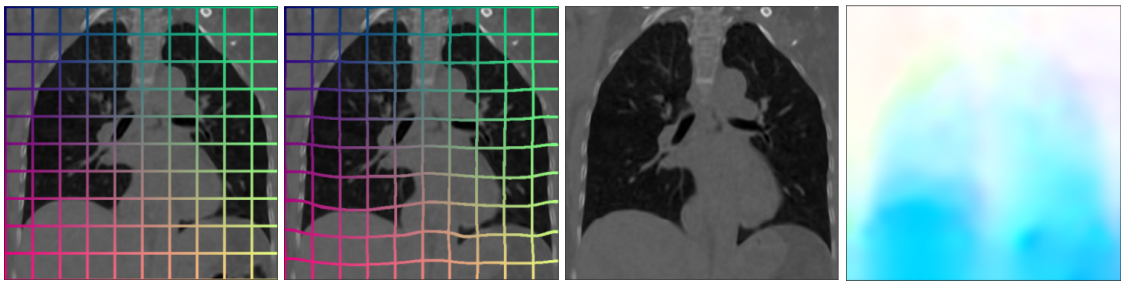
In general, our models perform reasonably well on the data with small displacements. However, both models struggle on the data with the larger displacements. Especially compared to the model isoPTV, the average registration errors of our models are significantly larger. The model isoPTV gets very close to the repeat registration error of the experts that provided the landmark information, reported in the 'Observers' column.

To possibly improve the performance for larger displacements, in-between images can be used. The DIR-Lab data set provides a sequence of six images. The first and last images of this sequence constitute the moving and fixed image respectively. The other four images still can be used to guide the registration between the moving and fixed image. We can also incorporate this into model $\mathcal{V}$ by adding the following term to the loss:

$$\mathcal{S}^{\text{in-between}} = \gamma \sum_{i=1}^{4} \int_{\Omega} \left[\mathcal{I}_i(x) - \left(\mathcal{M} \circ \Phi^{(i \cdot T/5)} \right)(x) \right]^2 dx. \tag{5.3}$$

Here $\mathcal{I}_i$ denotes (the implicit representation of) the i^{th} in-between image and γ is the hyper-parameter deciding how heavily dissimilarities to these images are punished. Recalling equation (2.18), $\Phi^{(T)}$ denotes the final deformation between the moving and fixed image. The term $\mathcal{S}^{\text{in-between}}$ enforces that the deformation $\Phi^{(t)}$ is similar to the in-between images for appropriate values of t.

Using the in-between images for either model did not significantly improve the performance. This leaves model $\mathcal{V}$ as our best performing model, slightly edging out $\mathcal{D}$ on average. Figure 5.7 depicts the registration resulting from this model on 4DCT1. See the appendix for more figures regarding the experiments.



(a) $\mathcal{M}$ (b) $\mathcal{M} \circ \Phi$ (c) $\mathcal{F}$ (d) Φ

Figure 5.7: Images of Φ from model $\mathcal{V}$.

Chapter 6

Discussion and future work

6.1 Conclusion

In this thesis, we proposed to use neural implicit representations in image registration for both the images and the transformation. We presented two simple models that use these implicit representations, a small deformations model $\mathcal{D}$ and a diffeomorphic model $\mathcal{V}$. These models yield differentiable images and transformations that can be sampled at arbitrarily high resolutions. The neural implicit representations allow derivatives to be computed analytically which makes the implementation of regularization trivial.

The goal was to explore the idea of integrating neural implicit representations into models for medical image registration, not to provide a method performing on par with current state-of-the-art image registration methods. The neural implicit representations of the images used in our experiments did not represent the image perfectly. The data in the dataset that we used was too detailed for our specific implementation. Despite this limitation, our model does provide an improved alignment with average errors lower than half of the average displacement in most cases.

6.2 Discussion

The neural implicit representations make it so that there are no errors introduced due to interpolation or the approximation of derivatives. For any coordinate, we can analytically compute its contribution to the loss. However, one thing that still has to be taken into account is how well the implicit representations of the images approximate the scene in reality. In the experiments, these approximations were far from perfect, as there was a significant loss. The fact that there is a significant loss in the representation of the images leaves room for improvement. Because of limitations in time and GPU memory, the images are represented using neural networks with approximately 500,000 weights, while the images in the experiments contain between 6,000,000 and 33,000,000 voxels. Within our implementation, this compression does cause the loss of some information. We tested deeper neural networks with a similar amount of weights, this did however not improve the performance.

Using neural implicit representations for image registration could overcome the limitations of discrete image registration. Discrete methods are inherently limited by the resolution of the data. Consequently, the potential for continuous methods is higher than discrete methods can ever be. Also, the fact that interpolation is required for transforming the images means that information can get lost between transformations. In the discrete case, given a perfectly accurate inverse transformation Φ^{-1}, performing Φ and Φ^{-1} in succession would, in general, not give back the exact original image due to interpolation. In the continuous setting, however, the

result would be the exact original image.

The potential of the accuracy of registration is very high for neural implicit representations. However, the speed of registration is a point of concern. Without conditioning a network on other data, optimizing a model per image pair is very time-consuming. Also, the requirement for learning an implicit representation for every image slows down the registration. Doing these things more efficiently is crucial for the speed of registration.

6.3 Future work

Generalizing the model for one-shot registration

For several medical tasks real-time registration is useful. This would require generalizing the model so one could one-shot register any input image-pair. In VoxelMorph [20], a model is proposed for this purpose. The model takes two images as input and gives the deformation field as output. Doing this in a continuous setting would require the model to output the weights of a neural network that represents the deformation. Also, the input images should be replaced with implicit representations, *i.e.* the inputs for this model are the weights of the networks representing the respective images.

Generalizing the model also opens up the opportunity for semi-supervised learning, using the landmark information during training where available.

Coarse to fine

The registration models presented in this thesis performed relatively poorly on data with large displacements. Aside from improving the neural representations, a possible improvement could be made by mimicking a pyramid scheme, as often used in traditional variational methods for image registration. In a setting with implicit representations, one could start training with a very low value for ω. A low value for ω will lead to a low-frequency solution, so this way the network would be able to catch the general large motions, but struggle to represent the smaller details in the transformation. During training the ω can be gradually increased, allowing the network to represent smaller details every iteration. Ideally, this would improve the registration of the larger movements. However, it could be the case that implementing a scheme like this would not lead to an improvement in the learning of the larger movements, while slowing down the learning of the smaller details for no benefit at all.

Integrating without discretizing

The diffeomorphic approach discussed in 2.5 does in reality not guarantee the preservation of topology because of the discretization in time. Since the implicit representations allow us to work in a continuous setting it might be interesting to think about ways to analytically compute the integral of the velocity field. Ideas similar to AutoInt [21] might be interesting, however, for our problem the integral is not along a straight line, which complicates the integration.

Finding a better way of integrating the velocity field could also speed up the training process significantly, as the current implementation is very slow.

REFERENCES

[1] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[2] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[3] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv preprint arXiv:2008.02268*, 2020.

[4] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[5] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

[6] Lars Ruthotto. *Hyperelastic image registration*. PhD thesis, PhD thesis, 2012.

[7] Stefan Klein, Marius Staring, Keelin Murphy, Max A Viergever, and Josien PW Pluim. Elastix: a toolbox for intensity-based medical image registration. *IEEE transactions on medical imaging*, 29(1):196–205, 2009.

[8] Jan Modersitzki. *FAIR: flexible algorithms for image registration*. SIAM, 2009.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] John Ashburner. A fast diffeomorphic image registration algorithm. *Neuroimage*, 38(1):95–113, 2007.

[11] Adrian V Dalca, Guha Balakrishnan, John Guttag, and Mert R Sabuncu. Unsupervised learning for fast probabilistic diffeomorphic registration. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 729–738. Springer, 2018.

[12] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[13] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

[14] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[15] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[16] Alessa Hering, Keelin Murphy, and Bram van Ginneken. Lean2reg challenge: Ct lung registration-training data, 2020.

[17] Richard Castillo, Edward Castillo, Rudy Guerra, Valen E Johnson, Travis McPhail, Amit K Garg, and Thomas Guerrero. A framework for evaluation of deformable image registration spatial accuracy using large landmark point sets. *Physics in Medicine & Biology*, 54(7):1849, 2009.

[18] Valery Vishnevskiy, Tobias Gass, Gabor Szekely, Christine Tanner, and Orcun Goksel. Isotropic total variation regularization of displacements in parametric image registration. *IEEE transactions on medical imaging*, 36(2):385–395, 2017.

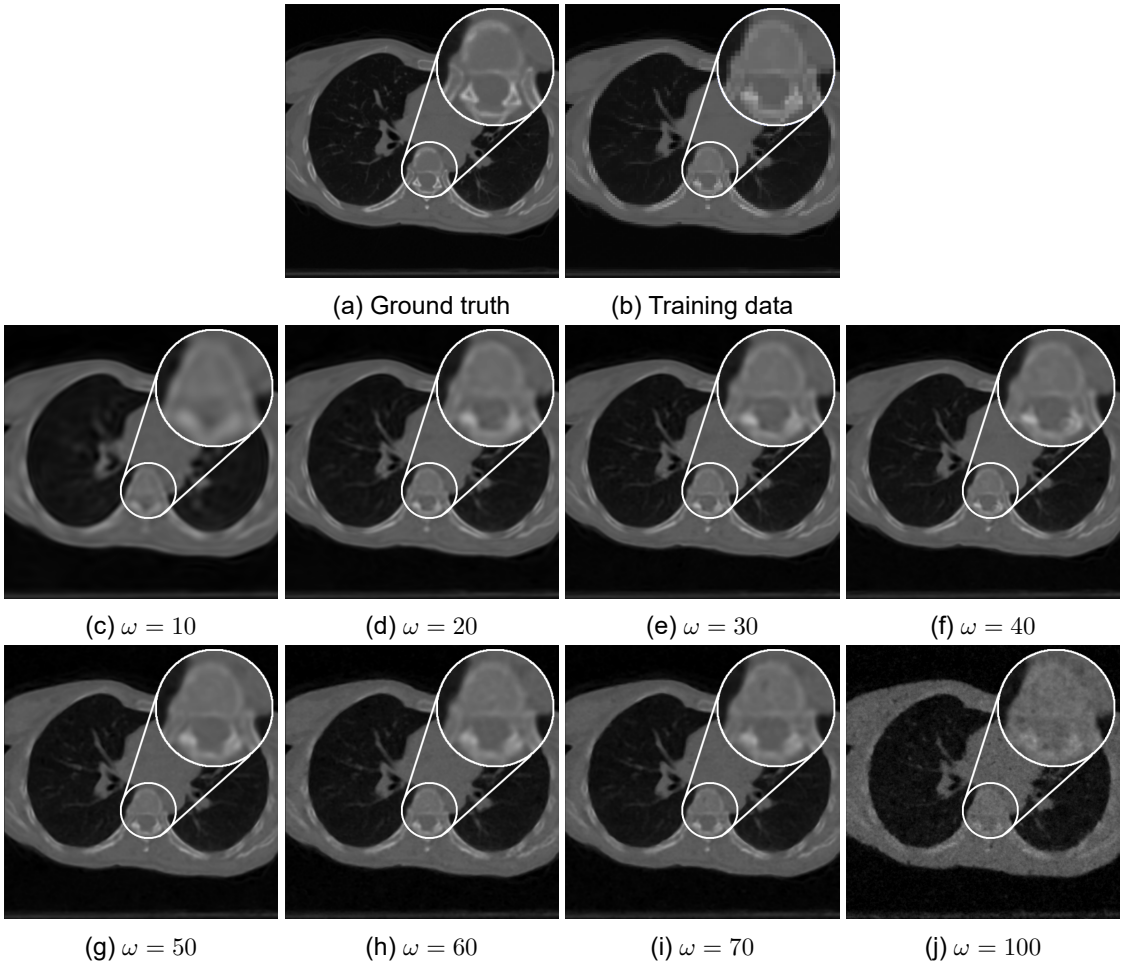
[19] Koen AJ Eppenhof, Maxime W Lafarge, Pim Moeskops, Mitko Veta, and Josien PW Pluim. Deformable image registration using convolutional neural networks. In *Medical Imaging 2018: Image Processing*, volume 10574, page 105740S. International Society for Optics and Photonics, 2018.

[20] Guha Balakrishnan, Amy Zhao, Mert R Sabuncu, John Guttag, and Adrian V Dalca. Voxelmorph: a learning framework for deformable medical image registration. *IEEE transactions on medical imaging*, 38(8):1788–1800, 2019.

[21] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. *arXiv preprint arXiv:2012.01714*, 2020.

Appendix A

Supplementary figures

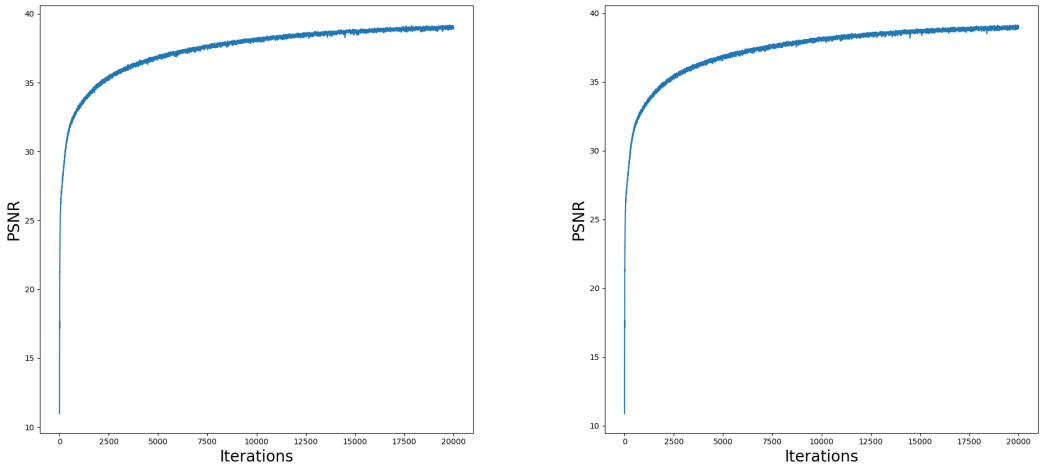


(a) Ground truth (b) Training data

(c) $\omega = 10$ (d) $\omega = 20$ (e) $\omega = 30$ (f) $\omega = 40$

(g) $\omega = 50$ (h) $\omega = 60$ (i) $\omega = 70$ (j) $\omega = 100$

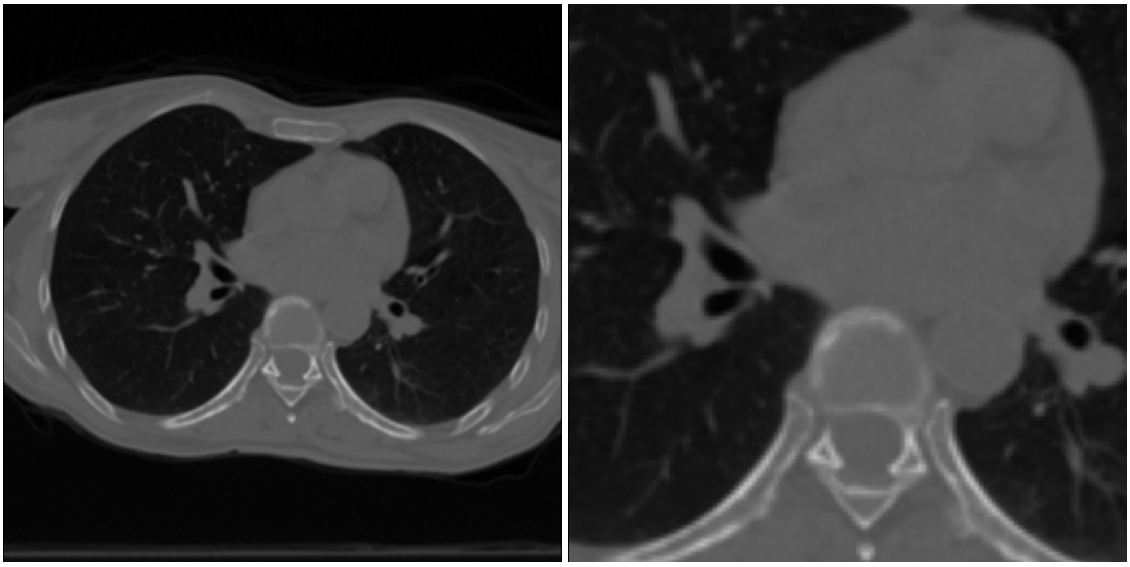
Figure A.1: Implicit representation on unseen coordinates for different values of ω. Figure (a) shows the ground truth image and (b) shows a lower resolution version of the same image that is used for the training for this particular figure. Figures (c) - (j) display the learned implicit representations for different values of ω, using the training data from Figure (b). For a very low value of ω as in (c) the result is too smooth and lacks high-frequency details. For higher values the results do contain details of higher frequency. However, as ω keeps increasing the results become more 'grainy', especially visible in (j).



(a) Moving image.

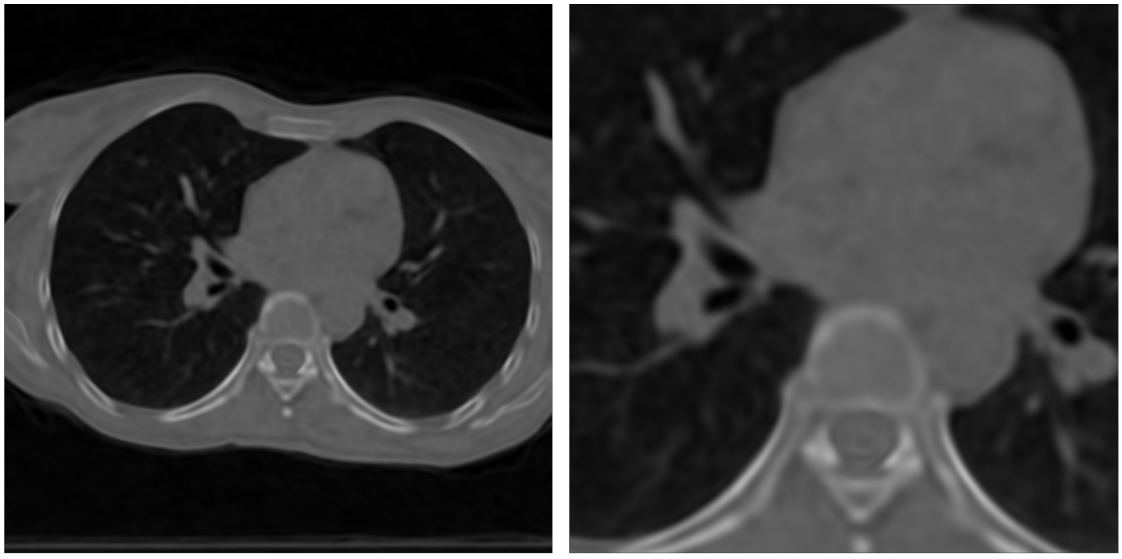
(b) Fixed image.

Figure A.2: Evolution of the PSNR during the optimization of the neural implicit representations, for both the fixed and the moving image from 4DCT1.



(a) Original image.

(b) Original image, zoomed in.



(c) Neural implicit representation of the image.

(d) Neural implicit representation of the image, zoomed in.

Figure A.3: First image of 4DCT1 from the DIR-Lab dataset compared to the neural implicit representation of this image that is used for the experiments in chapter 5. Especially in the zoomed-in views it becomes apparent that there is no perfect correspondence between the images and their neural implicit representations within our implementation. The PSNR of this implicit representation is 39.1. The image contains 6,160,384 voxels, whereas the neural implicit representation is described by 526,336 weights.

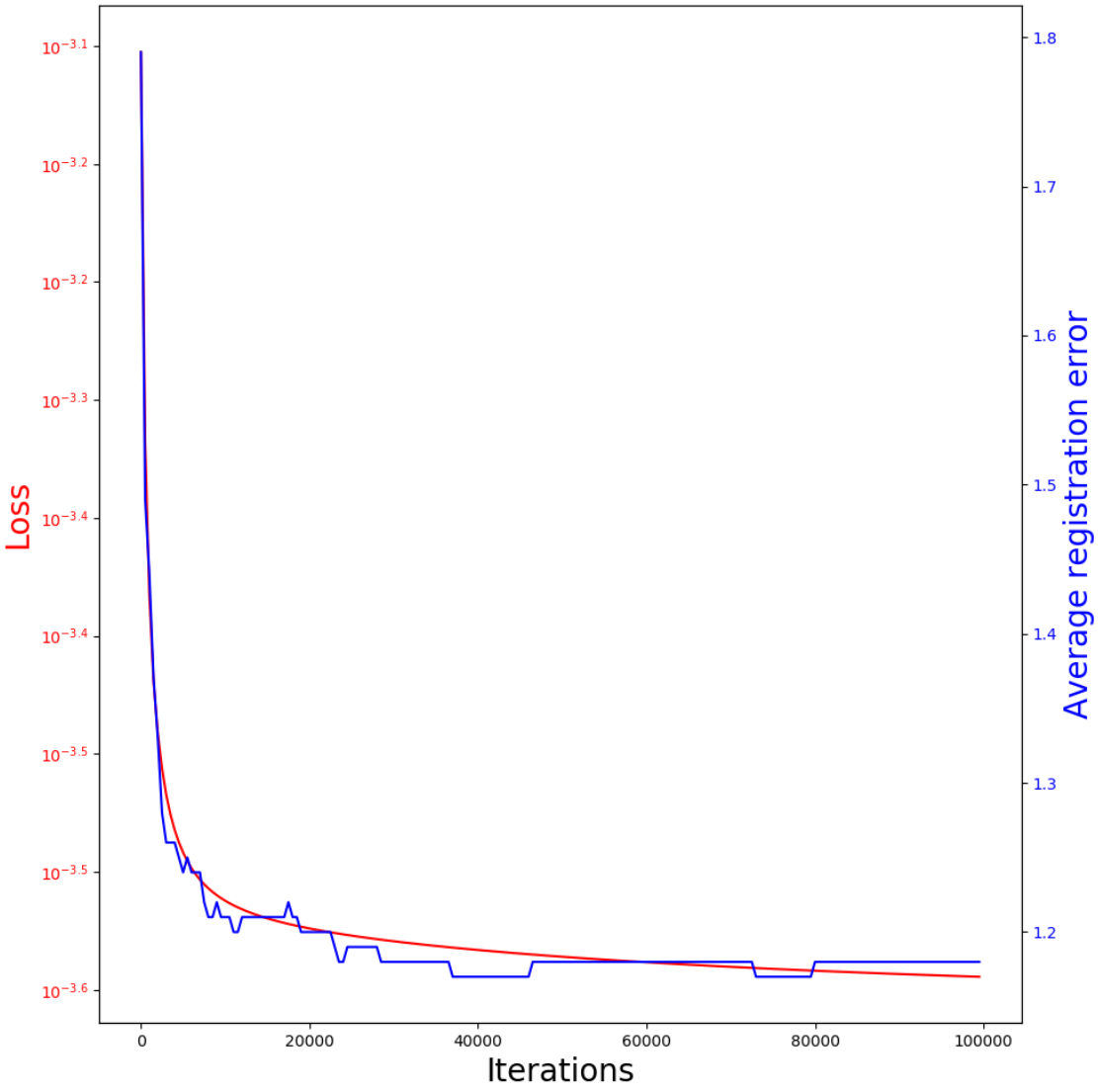
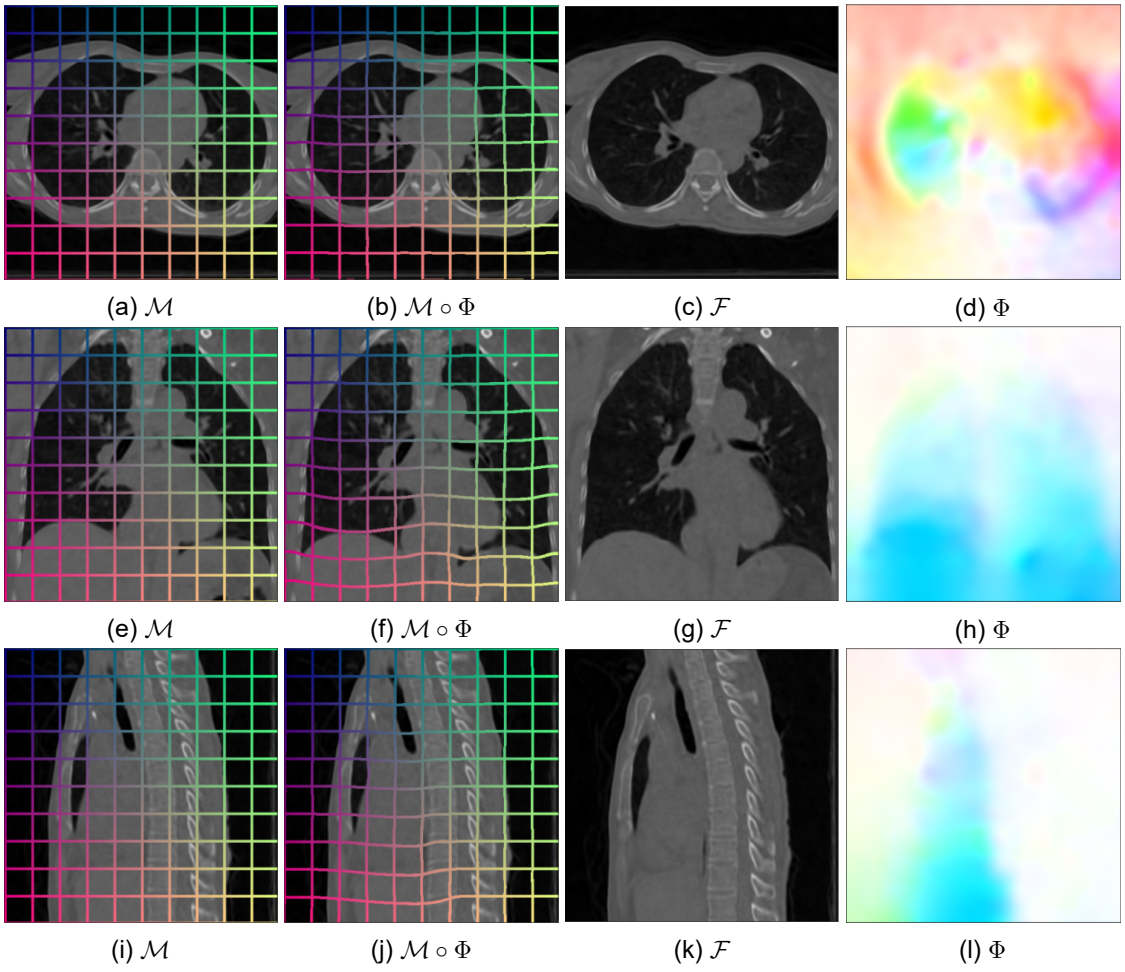


Figure A.4: Evolution of both the loss function and the average registration error during optimization for model $\mathcal{V}$, optimized for predicting the deformation between the images of 4DCT1. The loss is plotted on a logarithmic scale.

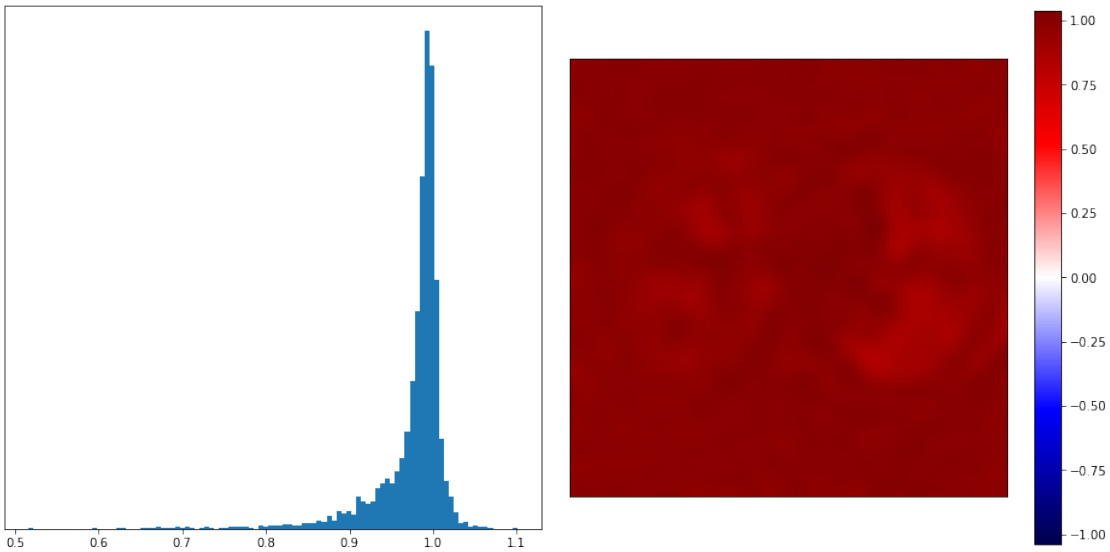


(a) $\mathcal{M}$ (b) $\mathcal{M} \circ \Phi$ (c) $\mathcal{F}$ (d) Φ

(e) $\mathcal{M}$ (f) $\mathcal{M} \circ \Phi$ (g) $\mathcal{F}$ (h) Φ

(i) $\mathcal{M}$ (j) $\mathcal{M} \circ \Phi$ (k) $\mathcal{F}$ (l) Φ

Figure A.5: Images of Φ from model $\mathcal{V}$ from different angles. (a) - (d) Axial, (e) - (h) coronal, (i) - (l) sagittal.



(a) Distribution of the determinant of the Jacobian.

(b) Visualization of the determinant of the Jacobian for an axial plane.

Figure A.6: Images of the determinant of the Jacobian of Φ from model $\mathcal{V}$.