UNIVERSITÄT ZU LÜBECK
INSTITUTE OF MATHEMATICS AND
IMAGE COMPUTING

The Moving Mesh Approach for Cardiac Image Registration

Der Moving Mesh Ansatz zur Registrierung von Herzdaten

Masterarbeit

im Rahmen des Studiengangs
Mathematik in Medizin und Lebenswissenschaften
der Universität zu Lübeck

Vorgelegt von
Martje Buhr

Ausgegeben und betreut von
Prof. Dr. rer. nat. Jan Lellmann
Institute of Mathematics and Image Computing

Mit Unterstützung von
Pia Schulz, M.Sc. und Johannes Bostelmann, M.Sc.
Institute of Mathematics and Image Computing

21.08.2024

MIC

IM FOCUS DAS LEBEN

Eidesstattliche Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Quellen und Hilfsmittel angefertigt zu haben.

Lübeck,
21.08.2024

Martje Buhr

Abstract

Cardiovascular diseases are the leading cause of death worldwide. Cardiac image registration is a valuable tool for analysing cardiac health and detecting cardiac diseases. Many state-of-the-art registration algorithms use iterative optimisation between images and do not benefit from the advantages offered by machine learning.

In this thesis, we implement, extend, and validate the learning-based moving mesh approach for deformable cardiac image registration as proposed by Sheikhjafari et al. Besides the original U-Net-based architecture, we also extend the method by replacing the U-Net with an implicit neural representation. A crucial part of the moving mesh approach is the constraining of the monitor function that parameterises the deformation field. To address this, we investigate a custom activation function to enforce one of two constraints on the monitor function.

Additionally, we evaluate four methods, including a newly developed scaling method, for enforcing the second constraint on the monitor function. In order to solve the resulting Poisson equations, we implement a Fast Fourier-based solver and validate its efficiency, robustness, and accuracy. The overall method is evaluated on a benchmark data set.

Kurzfassung

Kardiovaskuläre Krankheiten stellen die weltweit führende Todesursache dar. Bildregistrierung von Aufnahmen des menschlichen Herzens ist ein wertvolles Instrument, um Herzerkrankungen zu entdecken. Viele aktuelle Registrierungsverfahren basieren auf iterativen Optimierungsverfahren und profitieren daher noch nicht von den Vorteilen des maschinellen Lernens. In dieser Arbeit wird ein von Sheikhjafari et al. vorgeschlagener lernbasierter Moving Mesh-Ansatz zur deformierbaren Registrierung kardialer Daten implementiert, erweitert und validiert.

Neben dem ursprünglichen U-Net-basierten Ansatz wird auch eine implizite neuronale Erweiterung vorgestellt. Ein zentraler Teil des Moving Mesh-Ansatzes ist das Beschränken der Monitorfunktion, die das Deformationsfeld parametrisiert. Zu diesem Zweck wird eine neue Aktivierungsfunktion entwickelt, die eine der beiden erforderlichen Bedingungen erzwingt.

Weiterhin untersuchen wir vier Ansätze – einschließlich einer neu entwickelten Skalierungsmethode – um die zweite Bedingung sicherzustellen. Um die auftretenden Poissongleichungen zu lösen, wird ein Fast Fourier-basierter Löser implementiert und hinsichtlich Effizienz, Robustheit und Genauigkeit analysiert. Das vollständige Registrierungsverfahren wird auf einem Benchmark-Datensatz evaluiert.

List of Abbreviations and Acronyms

Adam	adaptive moment estimation
ANN	artificial neural network
CNN	convolutional neural network
CT	computed tomography
det	determinant
DM	Dice metric
DICOM	Digital Imaging and Communications in Medicine
DFT	discrete Fourier transform
DST	discrete sine transform
div	divergence
div-curl	divergence-curl
ECG	electrocardiogram
ED	end-diastolic
ES	end-systolic
FFT	fast Fourier transform
HD	Hausdorff distance
INR	implicit neural representations
LDDMM	Large Deformation Diffeomorphic Metric Mapping
LV	left ventricle
MR(I)	magnetic resonance (imaging)
ODE	ordinary differential equation
PDE	partial differential equation
R	reliability
ReLU	Rectified Linear Unit
SIREN	Sinusoidal Representation Network
SAX	short-axis
SCD	Sunnybrook Cardiac Data
SGD	stochastic gradient descent

List of Symbols

$\otimes$	Kronecker product
∇	gradient
Δ	Laplace operator
$\mathcal{X}$	(cell-centred) grid
I_F	fixed image
I_M	moving image
Ω	(image) domain
$\partial\Omega$	(image) boundary
$\bar{\Omega}$	(image) boundary closure
$C^k(\Omega)$	class of k-times continuously differentiable functions in Ω
ξ	pixel location
ϕ	deformation field
$\phi_{\mathbf{f}}$	forward deformation field
$\phi_{\mathbf{b}}$	backward deformation field
J_ϕ	Jacobian determinant of ϕ
$\tau_{\mathbf{lb}}$	lower bound for Jacobian determinant
$\tau_{\mathbf{ub}}$	upper bound for Jacobian determinant
μ	monitor function
V	vector field
$\mathcal{V}_t$	velocity field at time t
γ	curl of final velocity field
$\mathcal{L}$	loss function / dissimilarity measure
σ	activation function
$\mathcal{O}$	Landau symbol

Contents

1

Introduction

1.1 Motivation

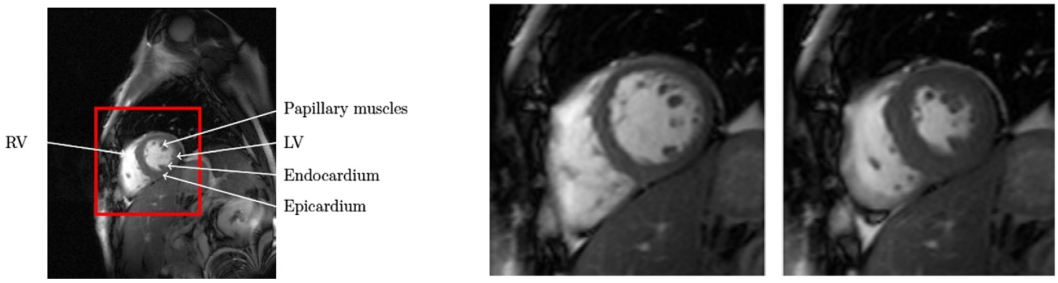
According to the World Health Organisation, cardiovascular diseases take approximately 18 million lives each year, making them the leading cause of death worldwide [28, 70, 86]. Effective diagnostics can facilitate the early detection of cardiac diseases and therefore prevent many deaths. For the detection of cardiac diseases, the precise quantification of cardiac motion and deformation is needed [28].

The heart consists of four chambers: the *right atrium*, the *left atrium*, the *right ventricle*, and the *left ventricle* (LV), which are displayed in figure 1.1a. The cardiac cycle is divided into two phases, the *systolic phase* and the *diastolic phase*. At the end of the diastolic phase (ED), the LV is entirely filled and contains the maximum volume of blood during the cardiac cycle, whereas at the end of the systolic phase (ES), it contains the minimum volume of blood. Cardiac images of the two phases are displayed in figure 1.1b, showing the variation of the LV during the cardiac cycle.

For many diagnostics, the LV is a region of particular interest [28]. Typical diagnostic indicators, such as the LV volume and the LV mass, provide insights into the global heart function and can be used to analyse the cardiac contractile function [61]. Key diagnostic indicator, such as LV volume and LV mass, can be analysed by comparing the LV during different phases of the cardiac cycle. Understanding cardiac motion is essential for an accurate analysis of the LV. The heart's motions can be divided into two main types: radial expansion and twisting [28]. Therefore, the heart's movement can be described by radial and rotational movement [7].

To analyse the previously mentioned diagnostic indicators, clinicians often compare two or more cardiac images manually, which is inaccurate and slow. As imaging tools such as *magnetic resonance imaging* (MRI) have become the reference tool for cardiac imaging [61], the amount of cardiac data to analyse is rising, increasing the need of an efficient tool to compare such images. This task can be efficiently accomplished with *image registration*.

Image registration is the task of finding coordinate transformations between two or more images such that corresponding structures are aligned. Cardiac image registration plays an important role in supporting diagnostics and monitoring the progression of cardiac diseases [50, 54]. Despite the rise of machine learning in many fields, in the field of image registration, iterative algorithms are the state-of-the-art algorithms when it comes to accuracy of the registration as well as computational performance [75]. Approaches using machine learning algorithms are typically supervised, and therefore rely on a ground truth in the training process. Although these algorithms have shown great



(a) Exemplary full size short-axis MR slice. The red box indicates the region of the heart. Some anatomical structures of the heart are shown.

(b) Example of two cardiac short-axis MR slices cut-outs of the heart at the end-diastolic phase (left) and the end-systolic phase (right). The volume variation of the left ventricle during the cardiac cycle is observable.

Figure 1.1: Examples of short-axis MR images of a heart (illustration credit [61]).

performance, in reality, data sets often do not contain the needed labels. Therefore, unsupervised methods play an important role in developing such data-driven methods in order to speed up the clinical workflow.

In 2022, Sheikhjafari et al. proposed an approach for cardiac image registration using an unsupervised machine learning approach [75], referred to as the *moving mesh approach*. A key feature of the approach is the parameterisation of the deformation field into radial and rotational movement, thus matching the motion of the heart. They applied their approach to the registration of the LV at the ED and the ES phase and achieved promising registration results.

The aim of this thesis is the validation and extension of the moving mesh approach for deformable image registration proposed by Sheikhjafari et al. in 2022 [75]:

- The first contribution is the implementation and validation of the approach by Sheikhjafari et al.
- As the original publication is missing key details on constraining the monitor function, a crucial part of the moving mesh approach, a second contribution of this thesis is to develop new approaches to satisfy these constraints, including a new activation function.
- The moving mesh approach includes solving two partial differential equations. For this, as a third contribution, a fast Fourier-based Poisson solver was implemented and tested.
- Finally, the approach is augmented with implicit neural representation.

The implementation and proposed extensions are experimentally verified on a benchmark dataset.

1.2 Related Work

Moving Mesh Generation

The moving mesh approach for cardiac image registration originates in adaptive grid generation for solving partial differential equations. In 1965, Moser studied the existence

of volume-preserving diffeomorphisms between different volume elements on the same Riemannian manifold [56]. Liao et al. proposed a new method for adaptive grid generation in 1992, based on Moser's studies, referred to as *moving mesh* generation [45, 46]. Their proposed method constructs a diffeomorphic mapping to adapt two- and three-dimensional grids. The mapping is parameterised by the *monitor function* that controls the cell volumes by describing the Jacobian determinant of the sought mapping, and a vector field with a divergence matching the monitor function. The deformation field is computed by solving an ordinary differential equation that is defined by a velocity field, which is established from the vector field. They provided a proof that a diffeomorphic deformation field exists, which has a Jacobian determinant equal to the monitor function. In 2006, Liu further proposed new methods to establish such vector field, including an approach that solves a set of Poisson equations to compute the vector field [47].

Moving Mesh-Based Image Registration

Several approaches to apply the moving mesh generation for deformable image registration have been proposed. Chen et al. applied the moving mesh generation to cardiac image registration [14]. In their method, the deformation field is parameterised with the moving mesh parameterisation using the Jacobian determinant and the velocity field. The proposed method establishes the final deformation field by iterative step-then-correct optimisation between images. They applied their approach to the registration of two-dimensional short-axis cardiac MR images to register the myocardial delineation of images from different phases of the cardiac cycle.

Their technique finds application in cardiac registration, such as the analysis of cardiac performance by Cheng et al. in 2015 and Zhang et al. in 2021 [17, 91]. Although the parameterisation is initially designed for cardiac image registration, their approach is also applied in non-medical contexts, such as the improvement of the quality of imaging of Mars rovers cameras that are used to analyse the geology and environment on Mars [21]. As the Mars rover uses two cameras, the images are aligned with the registration method proposed by Chen et al. to obtain one image.

Based on the ideas of Chen et al., Punithakumar et al. proposed similar registration approaches between 2013 and 2017. The proposed framework finds point correspondences in two-dimensional cardiac images to analyse the left ventricle and the right ventricle in MR images with step-then-correct optimisation strategy, using iterative optimisation to establish the deformation field [63, 64, 65].

Krishnaswamy et al. applied the moving mesh parameterisation to left ventricle segmentation for three-dimensional cardiac MRI and ultrasound images. The resulting optimisation problem to establish the deformation field is solved with an iterative step-then-correct algorithm [42].

In 2022, Sheikhjafari et al. adapted the registration approach by Chen et al. from an iterative approach to an unsupervised machine learning approach using a U-Net to perform the registration [75]. To the best of our knowledge, this was the first approach combining machine learning and the moving mesh method. Their approach utilises a U-Net learning a representation of the monitor function and a vector field, which are used to perform the moving mesh approach. They validated their approach with the registration of two- and three-dimensional cardiac MR images of the left ventricle.

They also applied their registration approach to the supervised segmentation of the left ventricle on two-dimensional and three-dimensional images [77]. This approach lowers

the time needed for manual segmentation by automatically computing segmentation of unseen images from one given segmentation.

The moving mesh parameterisation is also applied in other medical areas. Chu et al. applied the moving mesh parameterisation to the deformable registration of three-dimensional MR slices of breast images for breast cancer detection [18]. Similarly, Hsiao et al. modified the moving mesh approach for the non-rigid registration of the brain images [37].

Implicit Neural Representation

Implicit Neural Representation (INR) is commonly used to parameterise continuous signals, such as images, using an artificial neural network that maps spacial coordinates to corresponding intensity values [87]. Unlike conventional methods, the image's intensity values can be computed at arbitrary coordinates without using interpolation. Additionally, the limitations of discrete representations, such as limited memory, especially for large inputs, are avoided.

Notable advancements in expressiveness of INR have been achieved by Sinusoidal Representation Network (SIREN), proposed by Sitzmann et al. in 2020 [78]. By using periodic activation functions, SIREN demonstrates potential for the representation and reconstruction of three-dimensional shapes, due to their resolution independence and smooth encoding of the input features.

INR is frequently used to solve visual computing tasks, including image registration [55]. Wolterink et al. proposed INR-based image registration method in 2022, using a SIREN-based approach for intensity-based deformable image registration [85]. In their approach, for given input image coordinates, the output is the coordinate's displacement rather than the deformed image. Their approach was evaluated on four-dimensional computed tomography chest images, demonstrating better results than other tested learning-based methods.

In 2023, Byra et al. proposed another INR framework for implicitly learning the deformation fields [11]. Their approach has shown superior performance in comparison to state-of-the-art deep learning image registration frameworks for deformable image registration on three-dimensional brain MRI.

Sun et al. presented a framework for diffeomorphic image registration using INR in 2022 [81]. Their method can be used for two distinct registration approaches, allowing the modelling of displacement fields as well as velocity fields. For the latter case, the deformation field is generated by integrating the vector field using a neural ordinary differential equation solver. The framework was tested on two three-dimensional datasets, yielding comparable results to state-of-the-art registration methods.

Most INR registration methods are similar to pairwise optimisation, thus requiring individual optimising for each unseen image pair. Zimmer at al. address this by exploring generalising methods to avoid the training of an individual representation for each new image pair, trying to make INR registration more suitable for large data sets [93].

1.3 Outline

This thesis consists of seven chapters, which are structured as follows. In chapter 2, the fundamentals for machine learning based cardiac image registration are laid. First, the basics on cardiac image registration are explained. Then, the mathematical foundations

are laid, covering the two main topics vector field properties and differential equations. The next section is on image registration, covering the image registration various approaches, discretisation and interpolation methods. Finally, the machine learning basics are explained.

An overview of the moving mesh approach is provided in chapter 3. First, the origin of the moving mesh approach for diffeomorphic grid generation is discussed. Then, a proof is provided, showing that the deformations field computed with the moving mesh approach has the anticipated predefined Jacobian determinant. After that, the application of the moving mesh approach on an image registration problem is explained. Lastly, we provide details on our implementation and the augmentation of the moving mesh appraoch.

Different approaches to constrain the monitor function, a crucial part of the moving mesh approach, are provided in chapter 4. This includes the development of a new scaling method and a new activation function.

The validation of the moving mesh approach is discussed in chapter 5 and covers the following topics. First, the data sets and the evaluation metrics for the quantitative analysis of the registration are introduced. After that, the accuracy and the run time of the fast Fourier solver is analysed. Then, the registration results using a U-Net and implicit neural representation – both tested with different approaches to constrain the monitor function – are discussed.

Finally, chapter 6 summarises this work and provides an outlook.

2

Preliminaries

In this chapter, the fundamental basics of learning-based cardiac image registration are laid. This includes a brief study of the cardiac image generation, some mathematical preliminaries, image registration basics, and machine learning foundations.

As this thesis addresses the topic of cardiac image registration, we briefly introduce cardiac image generation in section 2.1.

The mathematical foundations to understand the moving mesh approach and image registration are laid in section 2.2. The section covers the main topics vector fields, differential equations and the discretisation of the Poisson equation.

Then, we introduce the image registration problem in section 2.3. This includes the concept of images and their discretisation as well as interpolation methods. Following, the image registration problem is outlined, followed by a discussion of different approaches to image registration and comparisons between them. The section concludes with an examination of regularisation, diffeomorphisms and the large deformation diffeomorphic metric mapping approach.

A this thesis focus lies on learning-based image registration, we present the fundamental concepts of machine learning in the final section 2.4. The section covers the following four concepts: an overview of artificial neural networks, followed by the basic concept of convolutional neural networks, the description of U-Nets, and an introduction to implicit neural representation. Finally, the use of machine learning in image registration is discussed.

2.1 Cardiac Image Generation

This thesis addresses the topic of cardiac image registration. To provide a foundational understanding, we offer a brief overview of cardiac image generation.

Magnetic resonance imaging (MRI) is one of the main imaging techniques for obtaining cardiac images. It can be used to discover and follow up diseases such as myocardial ischemia or infraction, as well as for planning or supporting clinicians during surgery, and to analyse the cardiac mass [29]. A benefit of MRI is that information on a patient's heart can be acquired non-invasively, making it the reference standard for evaluating cardiac function [84].

Cardiac image acquisition comes with certain challenges. The heart is an organ with a large variation in size due to the rapid motion of the heart beat. During a full cardiac cycle, the heart valvular plane that is connected to the LV moves up to 14 mm towards the apex and the myocardial walls thickness can vary up to 15 mm [68]. Obtaining MR images is a slow process, taking up to 90 minutes. Although patients are

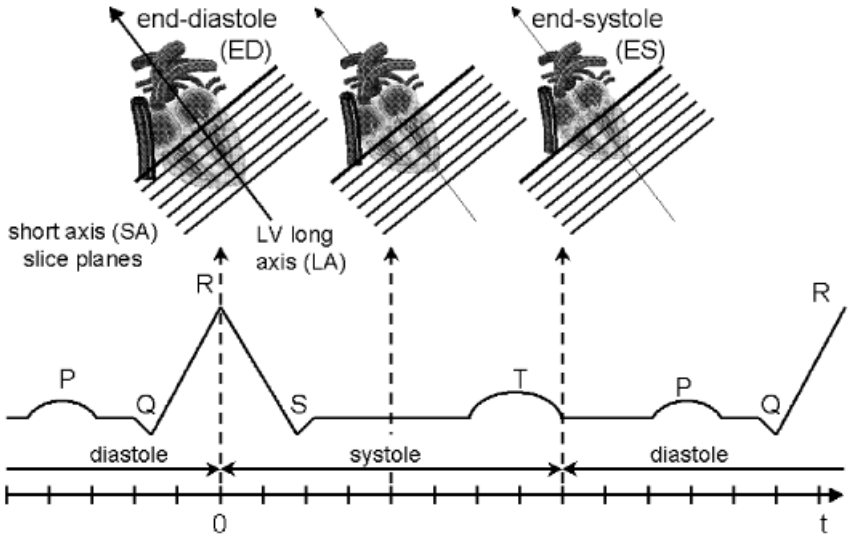


Figure 2.1: Schematic illustration of image acquisition of cine MRI [51]. The imaging process is aligned to the patient's ECG. The ECG shows distinct electric activities P, Q, R and T, for example at the end of the systolic phase (T) and at the end of the diastolic phase (R). By combining images of different heart cycles with the same ECG activity, the final image is obtained.

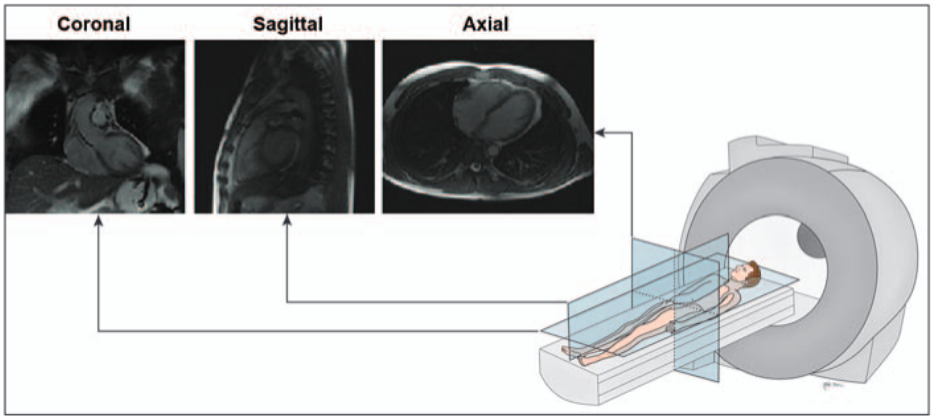
repeatedly asked to hold their breath for short periods of time for the image acquisition to slow down their heart beat, the image acquisition remains too slow, as the heart beats approximately once every second. To address this issue, the image acquisition process is split. This process is known as *cine MRI*. Cine MRI gathers only little image information from each cardiac phase and repeats this process over several cardiac cycles to reconstruct a full image [67]. To obtain accurate images of the different cardiac phases, the acquired images are aligned to the electrocardiogram (ECG) of the patient that is acquired simultaneously to the imaging process [67]. Figure 2.1 shows the process of obtaining images aligned to the ECG signals of an entire heart cycle.

Cardiac imaging is typically conducted from three different perspectives, the horizontal long axis, the vertical long axis imaging, and the short-axis (SAX). The different perspectives are displayed in figure 2.2. Each perspective highlights specific features of the heart. If the LV is the region of interest, acquiring short-axis images is optimal, as the LV is particularly prominent in these images [29].

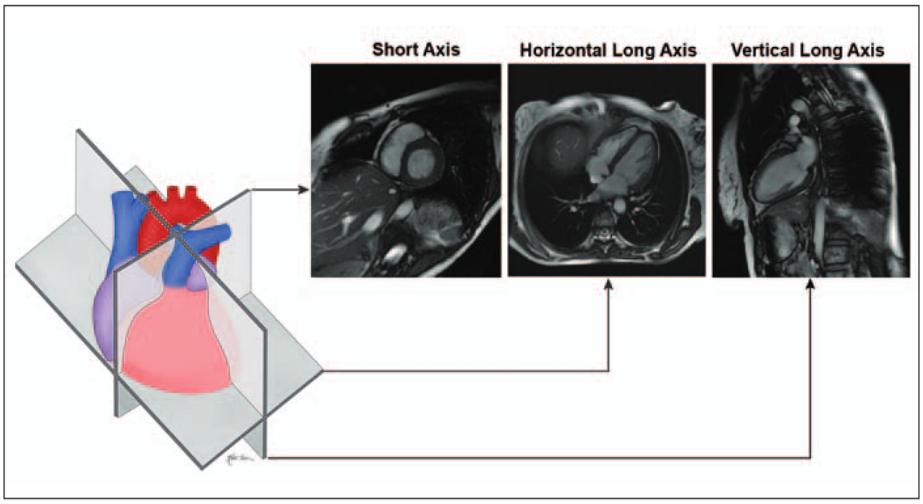
2.2 Mathematical Preliminaries and Differential Equations

In this section, we provide a review of vector field properties and differential equations. Generally, deformation fields in image registration are modelled as vector fields. In the moving mesh approach, these fields are parameterised by their *divergence* and *curl*. Thus, we begin by introducing these concepts as well as other related principles.

Additionally, we examine differential equations as they are a crucial part of the moving mesh approach, particularly focusing on solving ordinary differential equation and



(a) The three planes for cardiac MRI: the horizontal (coronal) and vertical (sagittal) long-axis planes, and the short-axis (axial) plane.



(b) Illustration of the three different planes for cardiac MR imaging. Each imaging plane shows different regions of the heart, the short-axis MR images show the left ventricle prominently.

Figure 2.2: Comparison of long- and short-axis MR images for cardiac imaging (illustration credit [29]).

Poisson equations. A main contribution of this thesis is the implementation of a fast Fourier Poisson solver, as we take a closer look at the Poisson equation, including its discretisation.

2.2.1 Properties of Vector Fields

We begin this section with the definition of vector fields:

Definition 2.1 (Vector Fields)**.** Let $\Omega \subseteq \mathbb{R}^n$ be an open subset. A *vector field* V, with $V : \Omega \to \mathbb{R}^n, \xi = (\xi_1, \dots, \xi_n) \mapsto (V_1(\xi), \dots, V_n(\xi))^\top$, is a mapping that assigns an $\mathbb{R}^n$ vector to every $\xi \in \Omega$.

Vector fields are used to describe physical phenomena, such as fluid movements. They play a crucial role in the field of differential equations. As differential equations play an important role in this thesis, certain differential operators are needed:

Definition 2.2 (Gradient of a Scalar Field)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable scalar function. The *gradient* of f at $\xi = (\xi_1, \dots, \xi_n) \in \Omega,\ \Omega \subseteq \mathbb{R}^n$, is defined as

$$\nabla f(\xi) := \begin{pmatrix} \frac{\partial f}{\partial \xi_1}(\xi) \\ \vdots \\ \frac{\partial f}{\partial \xi_n}(\xi) \end{pmatrix}, \tag{2.1}$$

where the i-th entry of ∇f contains the partial derivative of f in ξ_i direction. The resulting gradient is a vector field, $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$.

Often, functions that are differentiable and have continuous derivatives are of particular interest. These functions can be characterised as follows:

Definition 2.3 (C^k-Functions)**.** A function $f : \Omega \to \mathbb{R}$ with $\Omega \subseteq \mathbb{R}^n$ is considered to be of differentiability class $C^k(\Omega)$ if it is at least k-times continuously differentiable.

The *divergence* of a vector field indicates how much a vector field expands or shrinks at any given point.

Definition 2.4 (Divergence of a Vector Field, [9, § 40])**.** Let V be a differentiable vector field, i.e. $V : \Omega \to \mathbb{R}^n, V(\xi) = (V_1(\xi), \dots, V_n(\xi))^\top$ with $\Omega \subseteq \mathbb{R}^n$. The *divergence* of V at $\xi \in \Omega$, is defined as

$$\operatorname{div} V(\xi) := \nabla \cdot V(\xi) = \sum_{i=1}^{n} \frac{\partial}{\partial \xi_i} V_i(\xi). \tag{2.2}$$

The divergence is a scalar field. It is used to describe radial movement, as the divergence is the volume density of the outward flux. A positive divergence at every point indicates outward flux, whereas an entirely negative divergence indicates inward flux.

The *curl* – also known as the rotor – of a vector field describes the circulation density of each point of a vector field:

Definition 2.5 (Curl of a Vector Field, [9, § 40])**.** Let V be a differentiable two- or three-dimensional vector field, i.e., $V : \Omega \to \mathbb{R}^n, V(\xi) = (V_1(\xi), \dots, V_n(\xi))^\top$ for $\Omega \subseteq \mathbb{R}^n$

with $n = 2$ or $n = 3$. The *curl* of V at $\xi \in \Omega$, is defined as

$$\operatorname{curl} V(\xi) := \nabla \times V(\xi), \tag{2.3}$$

leading to the curl of a three-dimensional vector field as

$$\begin{aligned}\operatorname{curl} V(\xi) = &\left(\frac{\partial}{\partial \xi_3} V_2(\xi) - \frac{\partial}{\partial \xi_2} V_3(\xi)\right) e_1 \\ &+ \left(\frac{\partial}{\partial \xi_1} V_3(\xi) - \frac{\partial}{\partial \xi_3} V_1(\xi)\right) e_2 + \left(\frac{\partial}{\partial \xi_2} V_1(\xi) - \frac{\partial}{\partial \xi_1} V_2(\xi)\right) e_3,\end{aligned} \tag{2.4}$$

where $e_i, i = 1, \ldots, 3$, denotes the i-th unit vector.

If V is a two-dimensional vector field, all terms containing V_3 and ξ_3 vanish [47], thus the curl reduces to

$$\operatorname{curl} V(\xi) = \left(\frac{\partial}{\partial \xi_2} V_1(\xi) - \frac{\partial}{\partial \xi_1} V_2(\xi)\right). \tag{2.5}$$

For the Poisson equation that is introduced in the next section, the Laplacian is needed. The Laplacian of a function is given by the divergence of the gradient, which is the sum of all second partial derivatives:

Definition 2.6 (Laplacian)**.** Let $f : \Omega \to \mathbb{R}, \Omega \subseteq \mathbb{R}^n$, be a twice-differentiable scalar field. The *Laplacian* of f at $\xi, \xi \in \Omega$, is defined as

$$\Delta f(\xi) := \nabla \cdot \nabla f(\xi) = \operatorname{div}(\nabla f)(\xi) = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial \xi_i^2}(\xi). \tag{2.6}$$

2.2.2 Differential Equations

Differential equations show a relation of an unknown function and its derivative. They play an important role in the moving mesh approach. Therefore, this section covers ordinary differential equations and partial differential equations as well as their numerical solutions.

Ordinary Differential Equations

An *ordinary differential equation* (ODE) is an equation containing an unknown function y of one variable and its derivatives:

Definition 2.7 (Ordinary Differential Equation, c.f. [2])**.** Let $f : \Omega \to \mathbb{R}^n$ be a function with $\Omega \subseteq \mathbb{R}^{1+(k+1)n}$ and $k \geq 1$. An equation of the form

$$f(u^{(k)}(\xi), \ldots, u''(\xi), u'(\xi), u(\xi), \xi) = 0 \text{ for } \xi \in \Omega, \tag{2.7}$$

is called an ordinary differential equation of k-th order, where $u : \Omega \to \mathbb{R}$ is the unknown function and $u^{(i)}, i = 1, \ldots, k$ denotes the i-th derivative of u. If an additional condition in the form of $u(\xi_0) = u_0$ for an ODE is given, it is referred to as an initial values problem.

For the moving mesh approach, two initial value problems of the form

$$u' = f(t, u) \tag{2.8}$$

must be solved. In this thesis, the Runge-Kutta method is the preferred scheme; the reason for this is discussed in 3.4.2.

The method was initially proposed by Runge in 1895 [72]. The version commonly referred to as the Runge-Kutta method today, and the one we use in this thesis, is a slight modification of the original method introduced by Kutta in 1901. It is also known as the Runge-Kutta method with 3/8 rule [43].

Remark 2.8 (Runge-Kutta-Method with 3/8 rule [43])**.** *Given is an initial value problem of the form* (2.8) *with the initial condition* $u(t_0) = u_0$*. Let* n *be the number of time steps, let* h *be the fixed step size, let* t_i *denote the* i*-th time step. The approximate solution* u_i *at the* i*-th time step is defined iteratively by*

$$u_{i+1} = u_i + \frac{1}{8}(k_1 + 3k_3 + 3k_3 + k_4) \tag{2.9}$$

with

$$\begin{aligned} k_1 &= hf(t_i,\ u_i), \\ k_2 &= hf(t_i + \tfrac{1}{3}h,\ u_i + \tfrac{1}{3}k_1), \\ k_3 &= hf(t_i + \tfrac{2}{3}h,\ u_i + \tfrac{2}{3}k_2), \\ k_4 &= hf(t_i + h,\ u_i + k_3). \end{aligned} \tag{2.10}$$

The global accumulated error between the computed and the analytical solution is of order $\mathcal{O}(h^4)$*.*

Partial Differential Equations

A *partial differential equation* (PDE) is an equation between an unknown function with two or more variables and its partial derivatives. The formal definition is as follows:

Definition 2.9 (Partial Differential Equation, [24, Chapter 1])**.** Let Ω be an open subset of $\mathbb{R}^n$, let $k \geq 1$. An expression of the form

$$F(D^k u(\xi), D^{k-1} u(\xi), \ldots, Du(\xi), u(\xi), \xi) = 0 \text{ for } \xi \in \Omega, \tag{2.11}$$

where $D^k u$ denotes the k-th derivatives of u, is called a k-th-order partial differential equation, where

$$F : \mathbb{R}^{n^k} \times \mathbb{R}^{n^{k-1}} \times \ldots \times \mathbb{R}^n \times \mathbb{R} \times \Omega \to \mathbb{R} \tag{2.12}$$

is given, and

$$u : \Omega \to \mathbb{R} \tag{2.13}$$

is the unknown.

Usually, the behaviour of the unknown function on the domain's boundary, in this thesis denoted by $\partial\Omega$, is given. A commonly used boundary condition is *Dirichlet's* boundary condition:

Definition 2.10 (Dirichlet's Boundary Condition, c.f. [24, Chapter 6])**.** The requirement

$$u(\xi) = g(\xi) \quad \text{on} \quad \partial\Omega \tag{2.14}$$

for given g is referred to as *Dirichlet's boundary condition.* For $g(\xi) = 0$, the boundary condition is referred to as *homogeneous* Dirichlet boundary condition.

2.2.3 The Poisson Equation and Its Discretisation

The *Poisson equation* is a second-order PDE that establishes a relation of a function and the Laplacian of the multi-variable unknown function. For the application of the moving mesh approach, a set of discrete Poisson equations with homogeneous Dirichlet boundary conditions needs to be solved. In this section, we lay the focus on the discretisation of the Poisson equation and the Fourier approach for its numerical solution. In this thesis, we need to solve the two-dimensional Poisson equation, therefore, the focus lies on the two-dimensional discretisation.

To discretise the two-dimensional Poisson equation, the continuous two-dimensional Poisson equation is needed. It is defined as follows:

Definition 2.11 (The Poisson Equation [33])**.** Let Ω be an open subset of $\mathbb{R}^2$ and $u \in C^2(\Omega) \cap C^0(\overline{\Omega})$. Let $f \in C^0(\Omega)$ and $g \in C^0(\partial\Omega)$. If u fulfils

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega \\ u &= g \text{ in } \partial\Omega \end{aligned} \tag{2.15}$$

pointwise, then u is a solution of the *Poisson equation* with Dirichlet boundary conditions.

To numerically solve the Poisson equation with homogeneous Dirichlet boundary conditions, it is necessary to discretise the Poisson equation. This involves the discretisation of the Laplacian. Now, Ω is considered to be an equidistant two-dimensional grid $\mathcal{X}$ on $[-1, 1]^2$ with a grid spacing of $h = \frac{2}{d-1}$, resulting in d^2 grid points. Further details on cell-centred grids are covered in section 2.3.1. Then, u and f from definition 2.11 evaluate on the $\mathcal{X}$ as

$$\begin{aligned} u_{ij} &:= \boldsymbol{u}(-1 + ih, -1 + jh) \quad \text{for } i, j \text{ with } (ih, jh) \in \Omega \text{ (inner points)}, \\ f_{ij} &:= \boldsymbol{f}(-1 + ih, -1 + jh) \quad \text{for } i, j \text{ with } (ih, jh) \in \Omega \text{ (inner points)}, \end{aligned} \tag{2.16}$$

and satisfying the homogeneous Dirichlet boundary conditions. By applying finite differences twice, the Laplacian can be approximated by

$$\Delta u(ih, jh) \approx \frac{1}{h^2}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}) \tag{2.17}$$

for the inner points $i, j = 1, \ldots, d$ and with $u_{kl} = 0$ for $(kh, lh) \in \partial\Omega$, with second-order errors [52, Chapter 1].

Continuing with (2.17), a linear system of equations $A_h u_h = b_h$ can be established by storing the values u_{ij} in the vector u using column-major order, thus $u_h \in \mathbb{R}^{d^2}$. The vector $b_h \in \mathbb{R}^{d^2}$ stores the values $-f_{ij}$, and additional values for the boundary points. The $d^2 \times d^2$ matrix A_h represents the discrete Laplacian and is of the form

$$A_h = \frac{1}{h^2} \begin{pmatrix} M & I_d & & \\ I_d & M & \ddots & \\ & \ddots & \ddots & I_d \\ & & I_d & M \end{pmatrix} \in \mathbb{R}^{d^2 \times d^2} \tag{2.18}$$

with

$$M = \begin{pmatrix} -4 & 1 & & \\ 1 & -4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{pmatrix} \in \mathbb{R}^{d \times d} \tag{2.19}$$

and I_d being the $d \times d$ identity matrix [36].

The discretised Poisson equation

$$A_h u_h = b_h \tag{2.20}$$

can be solved using the eigenvalue decomposition of A_h. Given the eigenvalue decomposition of A_h,

$$A_h = V \Lambda V^{-1}, \tag{2.21}$$

where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_{d^2})$ is a $d^2 \times d^2$ diagonal matrix containing the eigenvalues of A_h and $V = (v_1 | \ldots | v_{d^2})$ being the $d^2 \times d^2$ matrix with the corresponding eigenvectors of A_h, the solution u_h can be computed as follows:

$$\begin{aligned} & & A_h u_h &= b_h \\ \Leftrightarrow & & V \Lambda V^{-1} u_h &= b_h \\ \Leftrightarrow & & u_h &= V^{-1} \Lambda^{-1} V b_h. \end{aligned} \tag{2.22}$$

It can be shown that V in the case of homogeneous Dirichlet boundary conditions is a symmetric matrix, implying $V^\top = V$, and orthogonal up to a scaling factor. The matrix Λ is a diagonal matrix, therefore, Λ^{-1} can be obtained by taking the reciprocal of its diagonal elements [58].

To apply this result to A_h, the eigenvalues and eigenvectors of A_h need to be determined. Due to its block structure, we can represent A_h using the Kronecker product. By exploiting the properties of Kronecker product, the eigenvalue computation can be simplified. To achieve a Kronecker representation of A_h, we use the discretisation of the one-dimensional second derivative

$$D = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{d \times d}. \tag{2.23}$$

The second derivative in two dimensions can be computed by applying the operator D to each column to obtain the second derivative in ξ_1-direction, and using the same principle again on the rows to get the second derivative in ξ_2-direction using the Kronecker product, leading to

$$\partial_{11} \approx I_d \otimes D \quad \text{and} \quad \partial_{22} \approx D \otimes I_d. \tag{2.24}$$

As the two-dimensional discrete Laplacian is the sum of the second partial derivatives, the discretisation of A can be established by

$$I_d \otimes D + D \otimes I_d = A_h \in \mathbb{R}^{d^2 \times d^2}. \tag{2.25}$$

Using this Kronecker identity of the Laplacian, we can compute the eigenvalues of A_h using the characteristics of the Kronecker product and the eigenvalues of D. Fortunately, D is a tridiagonal Toeplitz matrix, for which the following theorem holds:

Theorem 2.12 (Eigenvalues of a Toeplitz Matrix, cf. [57])**.** *Let $T \in \mathbb{C}^{d\times d}$ be a tridiagonal Toeplitz matrix, i.e.,*

$$T = \begin{pmatrix} b & c & & \\ a & b & \ddots & \\ & \ddots & \ddots & c \\ & & a & b \end{pmatrix} \in \mathbb{C}^{d\times d}, \tag{2.26}$$

for $a, b, c \in \mathbb{C}$. Then, the eigenvalues of T are given by

$$\rho_j = b + 2\sqrt{ac}\cos\left(\frac{j\pi}{d+1}\right), \quad j = 1, \dots, d, \tag{2.27}$$

with corresponding (right) eigenvectors

$$w^j = \left(\left(\frac{c}{a}\right)^{j/2} \sin\left(\frac{jk\pi}{d+1}\right)\right)_{k=1,\dots,d}. \tag{2.28}$$

Therefore, we can apply theorem 2.12 to compute the eigenvalues and eigenvectors of D, with $a = c = 1/h^2$ and $b = -2/h^2$:

Corollary 2.13. *The eigenvalues of D are given by*

$$\rho_j = \frac{1}{h^2}\left(-2 + 2\cos\left(\frac{\pi j}{d+1}\right)\right), \quad j = 1, \dots, d \tag{2.29}$$

with the corresponding eigenvectors

$$w^j = \left(\sin\left(\frac{jk\pi}{d+1}\right)\right)_{k=1,\dots,d}, \quad j = 1, \dots, d. \tag{2.30}$$

The Laplacian can be extended to two dimensions using the Kronecker product. A useful property of the Kronecker product is the following:

Theorem 2.14 (Mixed-Product Property, c.f. [48])**.** *Let M, N, O, and P be matrices with suitable dimensions such that the products MO and NP exist. Then, the equation*

$$(M \otimes N)(O \otimes P) = (MO) \otimes (NP) \tag{2.31}$$

holds.

Now, we can compute the eigenvalues and eigenvectors of A_h:

Theorem 2.15. *The eigenvalues of A_h are given by $\lambda_{ij} = \rho_i + \rho_j$ and the eigenvectors are given by $v^{ij} = w^i \otimes w^j$ for $i = 1, \dots, d$ and $j = 1, \dots, d$, where ρ_i are the eigenvalues and w^i the eigenvectors of D.*

Proof. As the products $I_d w^j, Dw^i, Dw^j$ and $I_d w^i$ exist, theorem 2.14 can be applied twice, leaving

$$\begin{aligned} A_h v^{ij} &= (I_d \otimes D + D \otimes I_d)(w^i \otimes w^j) \\ &= I_d w^j \otimes Dw^i + Dw^j \otimes I_d w^i. \end{aligned} \tag{2.32}$$

As w^i and w^j are eigenvectors of D with corresponding eigenvalues ρ_i and ρ_j and with linearity of the Kronecker product,

$$\begin{aligned} A_h v^{ij} &= w^j \otimes \rho_i w^i + \rho_j q^j \otimes w^i \\ &= (\rho_i + \rho_j)(w^i \otimes w^j) \\ &= (\rho_i + \rho_j) v^{ij} \\ &= \lambda_{ij} v^{ij}, \end{aligned} \tag{2.33}$$

holds, completing the proof for this theorem. □

As an immediate consequence of theorem 2.15, the following corollary holds:

Corollary 2.16. *The eigenvalues of the A_h are given by*

$$\begin{aligned} \lambda_{ij} &= \rho_i + \rho_j \\ &= \frac{1}{h^2}\left(-2 + 2\cos\left(\frac{i\pi}{d+1}\right) - 2 + 2\cos\left(\frac{j\pi}{d+1}\right)\right) \\ &= -\frac{4}{h^2}\left(\sin^2\left(\frac{i\pi}{2(d+1)}\right) + \sin^2\left(\frac{j\pi}{2(d+1)}\right)\right), \end{aligned} \tag{2.34}$$

with the corresponding eigenvectors

$$\begin{aligned} v^{ij} &= w^i \otimes w^j \\ &= \left(\left(\sin\left(\frac{k\pi i}{d+1}\right)\sin\left(\frac{l\pi j}{d+1}\right)\right)\right)_{k,l=1,\dots,d}, \end{aligned} \tag{2.35}$$

for $i, j = 1, \dots, d$.

With its eigenvalues and eigenvectors, we can diagonalise A_h, and the discretised Poisson equation can be solved using (2.22). Although this approach has lowered the computational costs of solving the Poisson equation, a naïve multiplication with V still requires $\mathcal{O}(m^2)$ multiplication and multiplications, with Λ^{-1} are in $\mathcal{O}(m)$ with $m := d^2$.

The solution of the Poisson equation can be obtained computationally more efficiently in Fourier space. For this, the Sylvester equation is needed:

Theorem 2.17 (Sylvester equation, [39]). *Let $M \in \mathbb{R}^{n\times n}, N \in \mathbb{R}^{l\times l}$ and $O, P \in \mathbb{R}^{n\times l}$, let I_n denote the $n \times n$ identity matrix. Let $\mathrm{vec}(P)$ denote the column-major vectorised form of the matrix P. Then*

$$(I_m \otimes M + N^\top \otimes I_n)\mathrm{vec}(O) = \mathrm{vec}(P) \;\Leftrightarrow\; MO + ON = P. \tag{2.36}$$

Now, we can apply theorem 2.17 to (2.25), therefore, the discrete Poisson equation can be rewritten to [27]

$$DU + UD = F, \tag{2.37}$$

with D as described in (2.23) and the matrix U containing the solution elements

$$U := (u_{ij})_{i,j=1,\dots,d} \in \mathbb{R}^{d\times d} \tag{2.38}$$

and the matrix F containing the source term values

$$F := (-f_{ij})_{i,j=1,\dots,d} \in \mathbb{R}^{d\times d}. \tag{2.39}$$

The matrix D can be diagonalised with the eigenvalue decomposition

$$D = S_d \Gamma S_d, \tag{2.40}$$

where the matrix $\Gamma = \mathrm{diag}(\rho_1, \dots, \rho_d)$ contains the eigenvalues of D and the matrix $S_d = (w^j)_{j=1,\dots,d}$ contains the eigenvectors of D. The matrix S_d with a scaling factor is the *discrete sine transform* (DST) of type I:

Definition 2.18 (Discrete Sine Transform (Type I), c.f. [12]). The $d \times d$ discrete sine transform matrix S_d is given by

$$S_d = \left(\sin\left(\frac{\pi ij}{d+1}\right)\right)_{i,j=1,\dots,d} \in \mathbb{R}^{d\times d}. \tag{2.41}$$

The DST of $y \in \mathbb{R}^d$ is computed by

$$\frac{1}{\sqrt{2h}}\mathrm{DST}(y) = S_d y. \tag{2.42}$$

As the DST matrix S_d is diagonal up to a scaling factor, the inverse discrete sine transform (IDST) can be computed by

$$\mathrm{IDST}(y) = \frac{1}{\sqrt{2h}}\mathrm{DST}(y). \tag{2.43}$$

The DST is denoted with $\mathrm{DST}(y) = \hat{y}$. The two-dimensional DST of a matrix $M \in \mathbb{R}^{d\times d}$ is the application if the DST on the rows of M and on the columns of M.

For enhanced readability, the discrete sine transform is displayed as $S := S_d$. Now, we define a matrix $X \in \mathbb{R}^{d\times d}$ such that

$$U = SXS \tag{2.44}$$

holds. This diagonalisation can be plugged into (2.37), leading to

$$\begin{aligned} DU + UD &= h^2 F \\ \Leftrightarrow DSXS + SXSD &= h^2 F. \end{aligned} \tag{2.45}$$

Multiplying (2.45) with S from the left and from the right leads to

$$SDSXS^2 + S^2XSDS = h^2 SFS, \tag{2.46}$$

such that the diagonalisation $DS = S\Gamma$ and $SD = \Gamma S$ can be plugged into (2.46), leading to

$$S^2\Gamma XS^2 + S^2X\Gamma S^2 = h^2 SFS. \tag{2.47}$$

As S is orthogonal, (2.47) simplifies to

$$\Gamma X + X\Gamma = h^4 SFS. \tag{2.48}$$

This leads to the component wise equation

$$\rho_i x_{ij} + x_{ij}\rho_j = h^4 \hat{f}_{ij} \tag{2.49}$$

for $i, j = 1, \dots, d$. By isolating the terms x_{ij}, we can obtain the components of X by

$$x_{ij} = h^4 \frac{\hat{f}_{ij}}{\rho_i + \rho_j} = h^4 \frac{\hat{f}_{ij}}{\lambda_{ij}}, \tag{2.50}$$

with the eigenvalues λ_{ij} of A_h as described in corollary 2.16.

As $\sin^2\left(\frac{i\pi}{2(d+1)}\right) > 0$ for $i = 1, \dots, d$, all eigenvalues $\lambda_{ij} < 0$, therefore, the division with λ_{ij} is well-defined. Re-examining (2.44), the matrix X is actually the DST transformed matrix U. Therefore, the solution of the Poisson equation U can be obtained by applying the IDST on (2.50). The DST can be computed using the *discrete Fourier transform* (DFT) as follows:

Theorem 2.19 (DST via DFT, c.f. [62, Chapter 12])**.** *Let $y \in \mathbb{R}^d$, let $x = \mathrm{DST}(y)$. Then, for the elements $x_k, k = 1, \dots, d$,*

$$x_k = -\frac{\sqrt{2d-2}}{2i}\mathrm{DFT}((0, y_1, \dots, y_d, 0, -y_d, \dots, y_1))_k \tag{2.51}$$

holds.

The computational costs can be lowered further by computing the DFT with the *fast Fourier transform* (FFT). Therefore, the solution of the discrete Poisson equation can be efficiently computed with $\mathcal{O}(m \log m)$ multiplications [12].

2.3 Image Registration

Image registration is the task of aligning two images such that their distance is minimised by aligning the images to the same coordinate system. Image registration is a crucial task in several fields, including astronomy, biology, physics, and medical image processing [54]. In a medical setting, image registration is utilised for treatment planning, disease follow-ups, such as the comparison of pre- and post-intervention images, and treatment verification during surgery [32, 50, 54]. In this thesis, we perform cardiac image registration. Therefore, the mathematical and numerical foundations for image registration are laid in the following sections.

2.3.1 Images and Image Discretisation

Images

Mathematically, an *image* I is understood as a continuous mapping

$$I : \Omega \to \mathbb{R}^k, \quad \Omega \subseteq \mathbb{R}^d, \tag{2.52}$$

from the image domain Ω to its intensity values, the real numbers $\mathbb{R}^k$ [35, 54]. Typical values are $k = 1$ and $k = 3$, where $k = 1$ indicates grey valued images and $k = 3$ indicates RGB images. As all images used in this thesis are grey valued images, we set k to 1 for the remainder of this thesis. The value d indicates the spatial dimension of the given data. In this thesis, all images are two-dimensional MRI scans, therefore, we set $d = 2$ for the remainder of this thesis. The *boundary* of the image domain is referred to as $\partial\Omega$.

The *modality* of an image refers to the image generation method. Common modalities are ultrasound, computed tomography, and MRI. If two images are acquired by the same method, they are of the same modality, referred to as *mono-modal* imaging; the converse is denoted as *multi-modal*.

Discretisation of Images

For numerical computations, discrete images are needed. Since we use medical MR images in this thesis, all images sample an object with a finite number of pixels [35]. As the given images are discrete, the discretised image registration problem are solved numerically with a *discretise-then-optimise* approach. Ideally, the main features of the image are found in its discretisation, therefore, the discretised problem can be solved with standard optimisation methods, but with lower computational costs compared to the optimise-then-discretise approaches [31, 54].

To discretise an image, we first discretise the image domain Ω. The domain Ω is now considered an n-dimensional interval $\Omega = (a_1, b_1) \times \cdots \times (a_n, b_n) \subset \mathbb{R}^n$, where n denotes the spatial dimension of Ω. A *grid* is a partitioning of a given interval into a finite number of non-overlapping cells [54]. There are different types of grids for the discretisation of an interval, in this thesis, we use *cell-centred grids*. A cell-centred grid $\mathcal{X}$ stores the images intensity values $I(\xi)$ in the centres $\xi = (\xi_1, \ldots, \xi_n) \in \mathcal{X}$ of each cell.

The grid spacing $h_j, j = 1, \ldots, n$, depends on the images size, which is determined by the number of data points d in each dimension. The grid spacing is computed by

$$h_j = \frac{b_j - a_j}{d} \quad \text{for} \quad j = 1, \ldots, n. \tag{2.53}$$

A discrete image is now considered the mapping

$$I : \mathcal{X} \to \mathbb{R}^k, \tag{2.54}$$

allocating to each cell-centre the intensity value at the corresponding image point.

From now on and for the remainder of this thesis, all images are considered to be discrete two-dimensional grey-valued images that are discretised on equidistant cell-centred grids. As the images are numerically represented by matrices, we choose the coordinate system to match the matrix indexing. An example of a two-dimensional discretised domain is displayed in figure 2.3.

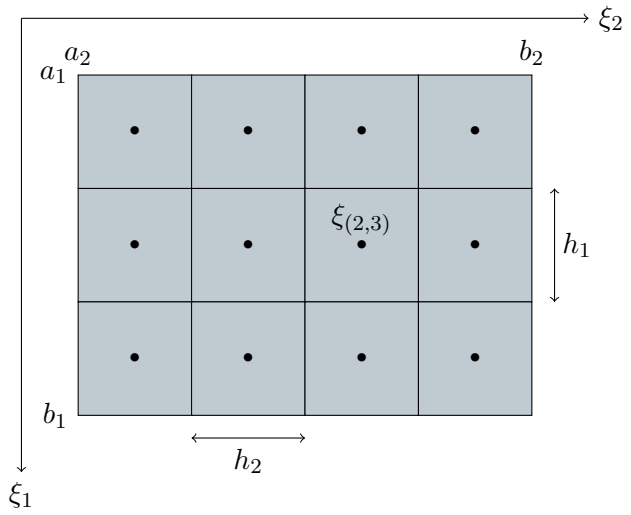


Figure 2.3: An exemplary two-dimensional equidistant cell-centred grid for $\Omega = (a_1, b_1) \times (a_2, b_2) = (0, 3) \times (0, 4)$ with a grid spacing of $h_1 = h_2 = 1$ (adapted from [54]).

Interpolation

As the images we use for this thesis are discrete, the image intensity values are only given for the cell-centres. In order to evaluate the image at points between the cell centres, the intensity values have to be interpolated. The following computations are adapted from [54, Chapter 3].

Let $\tilde{\xi} = (\tilde{\xi}_1, \tilde{\xi}_2)$ be a point between the grid points of the cell-centred grid $\mathcal{X}$, let $I : \mathcal{X} \to \mathbb{R}$ a discrete two-dimensional image. Bilinear interpolation uses the intensity values of the four closest grid points of $\tilde{\xi}$, that we denote by $(\xi_1, \xi_2), (\xi_1 + h_1, \xi_2), (\xi_1, \xi_2 + h_2)$, and $(\xi_1 + h_1, \xi_2 + h_2)$, and interpolates the intensity value of $\tilde{\xi}$ by performing linear interpolation in both dimensions. The bilinear interpolation $I^{\text{bilinear}}(\tilde{\xi})$ is computed by

$$\begin{aligned} I^{\text{bilinear}}(\tilde{\xi}) = {} & I(\xi_1, \xi_2)(1 - \zeta_1)(1 - \zeta_2) \\ & + I(\xi_1 + h_1, \xi_2)\zeta_1(1 - \zeta_2) \\ & + I(\xi_1, \xi_2 + h_2)(1 - \zeta_1)\zeta_2 \\ & + I(\xi_1 + h_1, \xi_2 + h_2)\zeta_1\zeta_2 \end{aligned} \tag{2.55}$$

with

$$\zeta_i = \frac{\tilde{\xi}_i - \xi_i}{h_i}, \quad i = 1, 2 \tag{2.56}$$

denoting the normalised distance between $\tilde{\xi}$ and ξ. Missing values outside of $\mathcal{X}$ are set to zero.

Discrete Derivatives and Integrals

Now, we introduce discrete computations of integration and derivatives that are needed for this thesis. For a discretised domain Ω and pixel locations $\xi \in \Omega$, the domain Ω is assumed to be a cell-centred grid $\mathcal{X}$ with d^2 grid points and a grid spacing of $h = \frac{2}{d-1}$.

The discrete integral of a function $\mu : \Omega \to \mathbb{R}$ can be estimated by

$$\int_\Omega \mu(\xi)\ \mathrm{d}\xi = h^d \sum_{\xi \in \mathcal{X}} \mu(\xi) + \mathcal{O}(h^2). \tag{2.57}$$

The above formula results in integration errors of order h^2 [54, Chapter 2].

In this thesis, discrete derivatives are approximated using central differences to estimate (partial) derivatives of μ with

$$\frac{\partial \mu(\xi^i)}{\partial \xi} = \frac{\mu(\xi^{i+1}) - \mu(\xi^{i-1})}{2h} + \mathcal{O}(h^2), \tag{2.58}$$

leaving errors of order h^2 [79, Chapter 1].

The described formula works for all inner points, missing values outside of Ω have to be approximated. There are several ways to compute the derivatives at the borders. In this thesis, we set missing values outside of Ω to zero.

2.3.2 The Image Registration Problem

Image registration is the task of aligning two images of the same object captured at different times, perspectives or from different devices [53]. The aim is to find a *deformation field* ϕ warping the *moving image* I_M to be as similar as possible to the *fixed image* I_F.

The mathematical problem is described as follows: For a given pair of discrete grey value images

$$I_F, I_M : \mathcal{X} \to \mathbb{R}, \tag{2.59}$$

find a deformation field $\phi : \Omega \to \Omega$ by solving the optimisation problem

$$\min_{\phi:\Omega\to\Omega} \mathcal{L}(I_F, I_M \circ \phi) \tag{2.60}$$

with $\mathcal{L}$ measuring the image dissimilarity of the transformed moving image and the fixed image.

In this thesis, we consider the deformation field to be divided into two parts, the identity and the displacement field $u : \Omega \to \mathbb{R}^2$ such that

$$\phi(\xi) = \xi + u(\xi) \tag{2.61}$$

for all $\xi \in \Omega$ [53]. Thus, the final deformation is the sum of the identity and a displacement field u, that indicates for every pixel ξ to which position it is displaced. This approach is called an Eulerian approach. Applying this deformation to the moving image leads to the transformed image $(I_M \circ \phi)(\xi) = I_M(\xi + u(\xi))$.

There are several approaches to perform image registration and the choice of approach depends on the data and the specific setting. In the following two sections, the main approaches are briefly introduced and compared. The registration approaches are roughly divided into two categories, however, there are overlaps between these categories.

2.3.3 Landmark-Based and Intensity-Based Registration

Landmark-based image registration uses pairs of distinct corresponding points, known as landmarks, from the moving and the fixed image. During the optimisation process, the total distance between all pairs of corresponding landmarks is minimised. An advantage of landmark-based image registration is the usually rather small number of features to optimise on, leading to a constrained solution space, and therefore lower computation costs [50]. Furthermore, registration of different image modalities is possible if the corresponding landmarks are found in both modalities [51].

However, data sets often do not include anatomical landmarks, and extracting landmarks is a slow and expensive task, as an expert labelling each image individually is needed. For certain organs, such as the heart, finding distinct landmarks is generally more challenging as the image resolution is usually lower compared to other organs [51]. Especially the short-axis region shows more variety in its anatomy leading to less accurate automatically placed landmarks. Although larger sets of landmarks would allow for more complex transformations [50], most data sets do not include the necessary number of anatomical landmarks. Moreover, for a small number of landmarks, much of the image information remains unused in the registration process.

Instead of relying on anatomical landmarks only, additional information such as segmentation maps can help to address the issue of a small number of landmarks. However, placing segmentations manually can take an expert up to 20 minutes per ventricle [61]. This issue can be resolved by using automatically placed anatomical landmarks [88] and segmentations [80]. Another issue of segmentation-based methods is that the segmentation accuracy must be high in order to achieve a good registration result [26, 50].

Instead of using anatomical structures for the registration, *intensity-based image registration* can be performed. Intensity-based registration operates on the image's intensity values and computes a deformation field based on a *dissimilarity metric* [26]. A dissimilarity metric compares the intensity values of two images and indicates, how similar the images are. A higher value indicates less similarity, in the case of identical images, the dissimilarity should be zero.

A common dissimilarity metric is the *mean squared error* (MSE), computing the mean quadratic error of the difference of intensity values of two images at the same position $\xi \in \mathcal{X}$. The MSE for discrete images is computed by

$$\mathcal{L}_{\text{MSE}}(I_F, I_M \circ \phi) = \frac{1}{|\Omega|} \sum_{\xi \in \Omega} \Big(I_F(\xi) - (I_M \circ \phi)(\xi) \Big)^2. \tag{2.62}$$

The MSE is computationally inexpensive, but it assumes a direct comparability of intensity values of pixels at the same position [54, Chapter 7]. Therefore, the images need to be of the same modality, ideally from the same scanner, to have matching intensity values. Other common dissimilarity metrics are the normalised cross-correlation or the mutual information, more details can be found in [54, chapter 7].

As intensity-based approaches are not in need of anatomical landmarks, they can be performed on any data set. Although intensity-based image registration has usually higher computational costs in comparison to landmark-based registration, as the optimisation is performed on the full image, it is often beneficial for the registration. Intensity-based image registration uses the underlying image information, leading to a generally more flexible and robust registration [26].

2.3.4 Parametric and Non-Parametric Image Registration

Another approach to lowering the computational costs of image registration is the parameterisation of the deformation field. *Parametric image registration* selects a deformation from a low-dimensional space by parameterising the deformation field with a finite number of parameters. In contrast, *non-parametric registration*, such as elastic or fluid registration, is not restricted to a finite-dimensional space and allows for more complex deformations [51].

An advantage of parametric image registration is the lower number of possible deformations, which lowers the computational costs for solving the associated optimisation problem. However, not all motions can be described by parametric deformations, thus, non-parametric registration can yield better registration results, especially for more complex deformations [10]. Nonetheless, parametric registration can be effective if prior knowledge about the data is available. This knowledge can be used to choose a suitable parameterisation of the deformation field, leading to reasonable registration results but lower computational costs.

Parametric deformations can be divided into two categories: *rigid* transformations and *deformable* deformations. Rigid image registration can be described as matching the same anatomical structure from both images [26]. These transformations are linear and include translations and rotations, preserving lengths, angles, and volumes of the transformed image's structures. These transformations apply the same movement to all points of an image uniformely. However, rigid transformations are often not suitable for medical registration, especially if non-rigid structures such as moving organs like the heart are involved, where changes often occur in small areas whereas other areas stay undeformed.

In contrast, deformable image registration allows for non-linear, locally differing deformations. While deformable deformations lead to higher computational costs than rigid transformations, they allow for more realistic deformations for non-rigid structures [26]. Nevertheless, rigid deformations are often used for pre-processing of the data to allow for better results in deformable image registration.

2.3.5 Regularisation

One of the main concerns of optimisation problems is to ensure that the given formulation of the problem is not ill-posed. The following formulation is the formal definition of a *well-posed* problem as stated by Hadamard:

Definition 2.20 (Well-Posedness [34])**.** A problem is called *well-posed* if the following three quantities hold:

1. the problem has a solution,
2. the solution is unique,
3. the solution depends continuously on the data.

If a problem is not well-posed, it is called *ill-posed*.

The image registration problem as stated in (2.60) is an ill-posed problem, as small changes in the input data can conduct larger changes in the solution and the solution

may not be unique [53]. This issue can be addressed by adding *regularisation*. The regularised problem can be stated as

$$\min_{\phi:\Omega\to\Omega} \mathcal{L}(I_F, I_M \circ \phi) + \mathcal{S}(\phi), \tag{2.63}$$

where $\mathcal{S}(\phi)$ is the regulariser that depends on the solution ϕ. Regularisation can for example penalise certain properties of the solution, such as physically implausible solutions like tissue folding.

The choice of regularisation is highly dependent on data characteristics. Thus, for an unknown dataset, there might not be enough information to reasonably regularise the registration, affecting well-posedness and potentially introducing unwanted bias [6]. Even with a suitable regularisation, the problem usually does not have a unique solution, causing the result to depend on the optimisation algorithm and starting point [54].

2.3.6 Diffeomorphisms

Imposing certain properties on the deformation field is particularly beneficial in medical image registration. It is often required that the resulting deformation field ϕ is invertible, with both ϕ and ϕ^{-1} satisfying smoothness conditions to preserve the topology of the registered organs [6]. Additionally, certain deformations such as the folding of tissue of anatomical structures, are implausible in a medical context. Ideally, these implausible deformations should be excluded from the set of possible solutions. Consequently, an important requirement for medical image registration is that the computed deformation field is *diffeomorphic*, as diffeomorphisms prevent the aforementioned issues. A deformation field is considered diffeomorphic if the following quantities hold:

Definition 2.21 (Diffeomorphisms [89, Definition 8.1])**.** A *homeomorphism* of Ω is a continuous bijection $\phi : \Omega \to \Omega$ such that its inverse ϕ^{-1} is continuous.

A *diffeomorphism* of Ω is a continuously differentiable homeomorphism $\phi : \Omega \to \Omega$ such that ϕ^{-1} is continuously differentiable.

In certain scenarios, both the deformation of the moving image to the fixed and the deformation of the fixed to the moving image are of interest. If the deformation field ϕ is diffeomorphic, its inverse ϕ^{-1} exists by definition. We denote the resulting two deformation fields, referred to as the forward and the backward deformation, in this thesis as $\phi_\mathrm{f} := \phi$ and $\phi_\mathrm{b} := \phi^{-1}$.

Jacobian Determinant

An efficient way to determine if a deformation field ϕ is diffeomorphic, is to analyse its *Jacobian determinant*

$$J_\phi(\xi) := \det \nabla\phi(\xi), \tag{2.64}$$

with the Jacobian matrix defined as

$$\nabla\phi(\xi) = \begin{pmatrix} \frac{\partial\phi_1}{\partial\xi_1}(\xi) & \cdots & \frac{\partial\phi_1}{\partial\xi_n}(\xi) \\ \vdots & \ddots & \vdots \\ \frac{\partial\phi_n}{\partial\xi_1}(\xi) & \cdots & \frac{\partial\phi_n}{\partial\xi_n}(\xi) \end{pmatrix} \in \mathbb{R}^{n\times n}. \tag{2.65}$$

If the deformation field ϕ is diffeomorphic, the Jacobian determinants are either all positive or negative for all $\xi \in \Omega$. This is a consequence of the implicit function theorem.

In the case of only negative determinants, the entire image is reflected, which is in a medical image setting an implausible result [49]. Therefore, enforcing a positive Jacobian determinant for ϕ results in a plausible diffeomorphic deformation field ϕ.

The Jacobian determinant of a deformation field ϕ can serve as an indicator for local volume contraction or extension. A Jacobian determinant less than 1 indicates local contraction, while a Jacobian determinant greater than 1 indicates local extension. For a Jacobian determinant of 1, the volume remains constant. Consequently, if $J_\phi(\xi) = 1$ for all $\xi \in \Omega$, the deformation is volume-preserving. As most organs are incompressible, ensuring a volume preserving registration by controlling the Jacobian determinant of the deformation field is a reasonable constraint for medical image registration.

2.3.7 Large Deformation Diffeomorphic Metric Mapping

As discussed in section 2.3.6, obtaining a diffeomorphic deformation field can be beneficial in medical image registration. However, for registrations requiring large deformations, such as the registration of the end-systolic and end-diastolic phase of the heart, ensuring a diffeomorphic deformation field can be challenging.

One approach to address this problem is the *Large Deformation Diffeomorphic Metric Mapping* (LDDMM) method. It was proposed by Beg et al. in 2005 [6]; the problem was first studied by Dupuis et al. and Trouvé et al. [23, 83]. The core concept of this approach is to decompose the final deformation field ϕ into a sequence of small diffeomorphic deformations, which together form a diffeomorphism.

A deformation field ϕ can be described as the endpoint of a flow $\phi_t(\xi) := \phi(\xi, t)$ with $\phi_t : \Omega \times [0,1] \to \mathbb{R}^n$. The flow is described by a time-dependent velocity field $\mathcal{V}_t(\xi) := \mathcal{V}(\xi, t)$, where $\mathcal{V}_t : \Omega \times [0,1] \to \mathbb{R}^n$. It is modelled by the ODE:

$$\frac{\mathrm{d}\phi_t(\xi)}{\mathrm{d}t} = \mathcal{V}_t(\phi_t(\xi)) \text{ for } t \in [0,1]. \tag{2.66}$$

The final deformation field $\phi := \phi_1$ is obtained by solving this ODE with the initial condition $\phi_0(\xi) = \xi$ for $t = 1$, as $t = 1$ represents the endpoint of the flow [90].

2.4 Machine Learning in Image Registration

With the increasing resolution of MRI scans, the amount of data requiring processing for deformable image registration is also rising. Consequently, efficient algorithms for registration tasks are essential. *Machine learning* can be utilised to address problems that are difficult to solve with traditional modelling-driven algorithms.

As this thesis focuses on medical image registration using U-Nets and implicit neural representation, this section provides an overview of machine learning basics and U-Nets. The section is divided into three parts. First, we introduce artificial neural networks. Afterwards, U-Nets are examined, followed by a brief introduction of implicit neural representation. Finally, the application of machine learning in image registration is discussed.

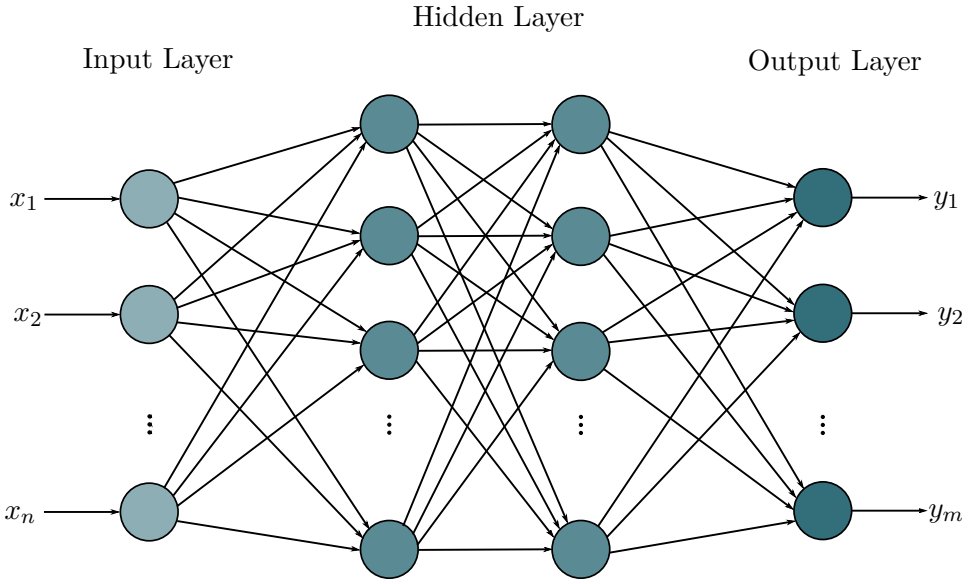


Figure 2.4: A schematic example of a an ANN, consisting of an input layer, two hidden layers, and an output layer (adapted from [59]).

2.4.1 Artificial Neural Networks

Artificial neural networks (ANNs) lay the foundations of machine learning. They are inspired by the human brain, more specific by its neurons. An *artificial neuron* is a non-linear, parametric function that builds the smallest unit of an ANN. Analogously to the neurons of the brain, an artificial neuron activates if its input exceeds a certain bias.

Mathematically, the output $y \in \mathbb{R}$ of an artificial neuron is expressed as the linear combination of the input $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ according to weights $\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$,

$$y = \sigma(\boldsymbol{x}^\top \boldsymbol{w}) = \sigma\left(\sum_{i=1}^{n} w_i x_i\right) \in \mathbb{R}, \tag{2.67}$$

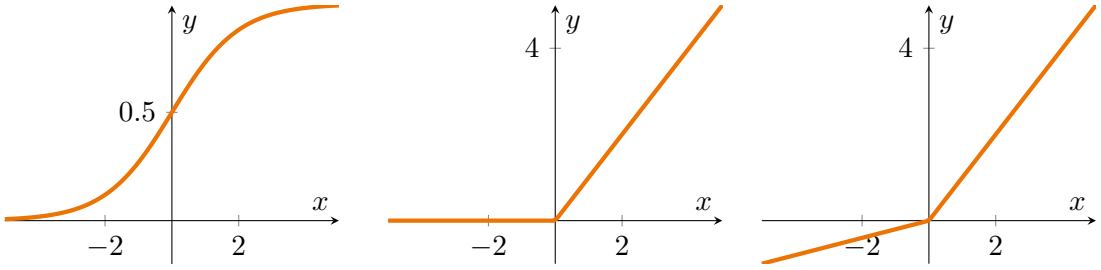
where the usually non-linear *activation function* $\sigma : \mathbb{R} \to \mathbb{R}$ determines if the neuron activates [71]. A bias can be added using an additional input element and a corresponding weight with a fixed value of 1. An ANN is the connection of several artificial neurons; the architecture of an ANN is displayed in figure 2.4.

Activation Functions

The activation functions of ANNs are usually chosen to be non-linear such that more complex relations of the input data can be found. The choice of the activation function depends on the task that the ANN is used for. Now, we briefly introduce three standard activation functions based on [74].

The Sigmoid function,

$$\text{Sigmoid}(y) = \frac{1}{1 + \mathrm{e}^{-y}} \tag{2.68}$$



(a) The Sigmoid activation function.

(b) The ReLU activation function.

(c) The leakyReLU activation function with $\alpha = 0.2$.

Figure 2.5: The three introduced activation functions. The Sigmoid function bounded by (0,1), whereas the leakyReLu function is not bounded. The ReLU function is bounded below by zero. The smoothness of the Sigmoid function in comparison to the non-smoothness of the ReLU and leakyReLu function is observable.

is an example of the class of logistic functions and is a commonly used activation function. It maps the input y to the interval $(0, 1)$. The Sigmoid function is continuously differentiable.

Another typical activation function is the Rectified Linear Unit (ReLU),

$$\mathrm{ReLU}(y) = \max\{0, y\}. \tag{2.69}$$

The ReLU function is computationally less expensive in comparison to the Sigmoid function and more efficient, as not all neurons are activated at the same time. This can also be a disadvantage, as all negative values are mapped to the same value. The ReLU function is also not continuously differentiable.

An adaptation of the ReLU function is the leakyReLU function, that maps negative values to small values:

$$\mathrm{leakyReLU}(y) = \begin{cases} y, & y \geq 0, \\ \alpha y, & \text{otherwise}, \end{cases} \tag{2.70}$$

with a usually small parameter α. Using a linear component for negative values prevents zero gradients that interfere with the learning process. The three introduced activation functions are displayed in figure 2.5.

Stochastic Gradient Descent

The learning process of an ANN is realised by updating its weights w during the training process to minimise the loss function $\mathcal{L}$. For each epoch of a training cycle, the output of the ANN is used to evaluate the loss function $\mathcal{L}$ value, using the current weights.

The weights w are updated using the backpropagation algorithm, which minimises the loss function by computing gradients with respect to the weights [71]. Computing those gradients can be a challenging task. For this thesis, we use *automatic differentiation* in backward mode with PyTorch to compute gradients. [60].

In comparison to regular gradient descent, *stochastic gradient descent* (SGD) does not compute the gradients for the entire data set, but for a small subset of training data to reduce computational costs. The update rule for SGD is given by

$$w_{\text{new}} = w_{\text{old}} - \eta_k \nabla \mathcal{L}(w_{\text{old}}, x_{i_k}), \tag{2.71}$$

where η_k denotes the positive step size, and index i_k is chosen randomly for each iteration $k \in \mathbb{N}$ [8].

The *adaptive moment estimation* (Adam) optimisation is a modified version of the SGD [41]. The algorithm contains three main features. The Adam algorithm computes the learning rates η for each iteration and each parameter individually. Additionally, Adam optimisation includes momentum by keeping an exponentially decaying average of gradients from previous iterations, leading to smoother updates and faster convergence compared to standard SGD. The optimisation in this thesis is carried out using Adam optimisation.

2.4.2 U-Nets

In 2015, Ronneberger et al. introduced the *U-Net*, a convolutional neural network for efficient image segmentation in medical image processing [69]. The U-Net architecture shares similarities with *Convolutional Neural Networks* (CNNs), which were first proposed by LeCun in 1989, and are commonly used for image processing tasks such as classification tasks [44]. Further details on CNNs can be found in [59]. Although CNNs have demonstrated good results for classification tasks, the precise localisation of the detected objects are often needed [69].

The U-Net architecture consists of three main components: the *encoder* (or *contracting path*), the *decoder* (or *expansive path*), and the *bottleneck*, which connects the encoder with the decoder. The name U-Net derives from this U-shaped architecture, which is displayed in figure 2.6.

The contracting path is similar to the architecture of a CNN and downsizes the spatial dimension of the input image by a repetition of the following steps. First, two *convolutions* are performed on the input image pair. The discrete convolution of the two-dimensional image I of size $n_1 \times n_2$ and the kernel K of size $k_1 \times k_2$ is mathematically expressed as

$$(I * K)(i,j) = \sum_{l=0}^{k_1-1} \sum_{m=0}^{k_2-1} I(l,m)K(i-l, j-m), \tag{2.72}$$

where $(I*K)(i,j)$ represents the output of the convolution at the position (i,j) [30]. For multi-channel inputs, the convolution is preformed on all channels individually, with a distinct kernel for each channel. The weights of the U-Net are the elements of the filter kernels that are shared in an entire layer, leading to a lower number of weights compared to traditional ANNs. The filter kernels are usually smaller than the input image, with their elements learned during training. Each convolutional layer is followed by a ReLU unit.

After this, the image is down-sampled using a *pooling layer*. Pooling layer reduce the images resolution by summarising entries with similar location. A common technique for this the max pooling, which selects the maximum value within a small neighbourhood of the input image. Pooling aids to achieve translation invariance and reduces the computational complexity by lowering the spatial dimension of the input [59]. Each step reduces the images spatial dimension. The combination of convolutional and pooling layer allows the U-Net to effectively detect patterns and features within the input data.

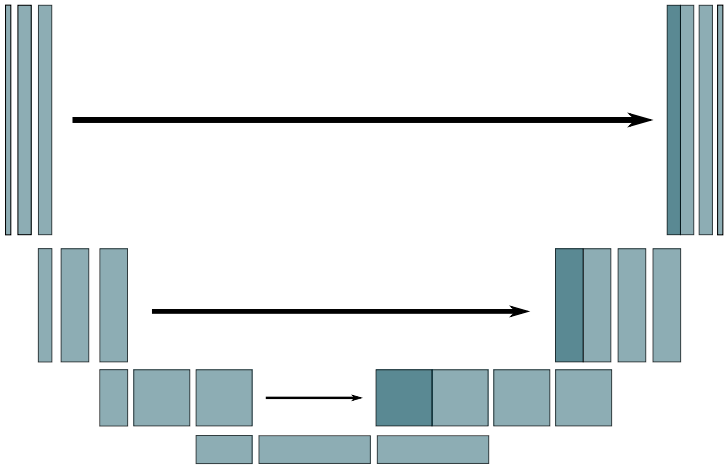


Figure 2.6: The schematic architecture of a U-Net (adapted from [69]). The image sizes at each step correspond to the sizes of the blue boxes. The black arrows represent the skip connection between the contracting and the expansive path. The dark blue boxes represent the concatenation of the information of the contracting path to the expanding path.

To connect the contracting path and the expansive path, the final layer of the contracting path performs a 1×1 convolution instead of max pooling.

The expansive path increases the spatial dimension of the input image by performing up-convolution (transposed convolution), followed by two convolutions and another ReLU unit. The up-convolution halves the number of feature channels, but increases the spatial resolution of the input. A significant feature of U-Nets are the *skip connections*. Skip connection concatenate layers of the encoder layers to the corresponding layer of the decoder, preserving the spatial information of learned features by connecting them to the upsampled features. This allows U-Nets not only to detect features, but also to localise the found objects by combining the high-resolution features detected in the contracting path with the upsampled image of the expansive path.

2.4.3 Implicit Neural Representation

Discrete representations of signals, such as discretised images, can only store details limited by the grid resolution they are discretised on. In contrast, an ideal continuous representation of a signal can have achieve an arbitrary resolution. The concept of *implicit neural representation* (INR) addresses this issue by finding a continuous representation of discrete input data, using ANNs to estimate a continuous function that represents an image or other signal implicitly [87]. The ANN is parameterised and can therefore be stored with a finite number of parameters. This allows for an arbitrarily fine resolution that can be stored with a lower number of parameters, especially for increasing input size [87].

INR utilises ANNs to parameterise continuous functions that map spatial input coordinates ξ to their corresponding data values. This is achieved by training an ANN that parameterises $\Psi : \xi \mapsto \Psi(\xi)$, where Ψ represents the object of interest, such as an image, and maps ξ to the images intensity values [78].

Compared to discrete representations, the continuous representation of a signal with INR is not limited by the grid resolution, but by the ANN architecture. Additionally, interpolation can be avoided, as the representation is continuous. As INR are designed to be differentiable, gradients and higher-order derivatives can be computed analytically and therefore leading to higher accuracy, which allows for optimisation of complex signals of typically ill-posed problems [87].

SIREN

A prominent INR approach is the Sinusoidal Representation Network (SIREN), which is used for representing three-dimensional shape representations [78]. The functions Ψ are parameterised as fully-connected networks, the parameters are optimised with gradient descent. Unlike the standard INR that uses ReLU activation function, SIREN uses periodic activation functions, more specific sine functions. The second order derivative of the ReLU activation function is zero, which limits the ability to capture fine details in the represented structures, as ReLU is less effective to model complex patterns with high frequency variation. SIREN uses sine activation functions, which can be differentiated continuously the desired number of times. This capability enables SIREN to model more complex structures. The SIREN parameterisation of Ψ is given by

$$\Psi(\xi) = \boldsymbol{W}_n(\psi_{n-1} \circ \psi_{n-2} \circ \cdots \circ \psi_0)(\xi) + \boldsymbol{b}_n, \tag{2.73}$$

with

$$\psi_i(\xi_i) = \sin(\boldsymbol{W}_i\xi_i + \boldsymbol{b}_i), \tag{2.74}$$

where $\psi_i : \mathbb{R}^{M_i} \mapsto \mathbb{R}^{N_i}$ is the i-th layer of the ANN. Within each layer, an affine transform on ξ with the weight matrix $\boldsymbol{W}_i \in \mathbb{R}^{N_i \times M_i}$ and bias $\boldsymbol{b}_i \in \mathbb{R}^{N_i}$ is then followed by a non-linear transforms in form of a sine transform.

2.4.4 Image Registration and Machine Learning

U-Nets in Image Registration

Although machine learning algorithms demonstrated promising results in various fields, many state-of-the-art image registration methods still rely on iterative optimisation between the images [75]. While iterative algorithms achieve high accuracy for the registration, they are usually computationally expensive. Furthermore, they do not use the advantages that machine learning algorithms provide, in particular, iterative algorithms do not learn features or patterns from the given data. Therefore, to receive accurate registration results, a user needs to identify such patterns and add matching regularisation terms for each new data set [75].

Applying machine learning to image registration can enable the learning of patterns, features, and generalisations from the images which could benefit the registration [75]. Recent approaches that apply machine learning to image registration have shown comparable accuracy to iterative algorithms but with lower computational costs [40].

To use U-Nets for image registration, the U-Net learns on several pairs of images that are registered. The output of the U-Net usually includes the deformation field. Although the resulting deformation fields might show good results, certain properties, such as a diffeomorphic deformation field, are often missing.

Machine learning approaches can roughly be divided into two groups: supervised and unsupervised learning. Supervised learning algorithms have shown promising results, but a ground truth is needed and the overall performance is highly dependent on the ground truth accuracy. Additionally, the ground truth to medical data is often not available, making supervised learning algorithms not applicable for medical settings [76].

Unsupervised methods avoid the need for ground truth: image registration algorithms for deformable registration using machine learning aim to solve the minimisation problem

$$\underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \, \mathcal{L}(I_F, I_M, g_\theta(I_F, I_M)), \tag{2.75}$$

where g_θ is a function modelled by the U-Net and θ are the learnable parameters of g. The loss function $\mathcal{L}$ is a measure for the dissimilarity of the registered images, c.f. section 2.3.3. The deformation field ϕ is the computed by evaluating the function, i.e.,

$$\phi = g_\theta(I_F, I_M). \tag{2.76}$$

The optimisation problem resulting from this formulation is ill-posed as the solution may not be unique. Without further constraints, the resulting deformation field is not necessarily diffeomorphic [31]. Similar to iterative registration, this issue can be addressed by adding regularisation terms, but the same issues of finding such terms occur. Regularisation is often difficult to choose and needs prior knowledge on the data that is often not available.

A different approach to achieving well-posedness is to reduce the dimension of the solution space. By parameterising the deformation field, it can described with a finite number of parameters.

INR in Image Registration

To perform image registration using INR, the INR can be utilised to generate the deformation field ϕ for a given pair of images. One of the main differences compared to registration with a U-Net is that a new parameterisation Ψ is learned for each new image pair, whereas U-Nets are trained to produce a new deformation field for a new pair of input images. The input for an INR is a set of image coordinates, and the training cycle involves updating the weights based on the loss from one image pair. Therefore, the training is performed directly on the test data and does not include a training and validation cycle with a different data set as in the U-Net training process. INR approaches have shown superior performance in comparison to state-of-the-art deep learning image registration frameworks for deformable image registration [11].

An approach for deformable image registration using a U-Net with a guaranteed diffeomorphic deformation field is the moving mesh approach proposed by Sheikhjafari et al. in 2022 [75]. For this thesis, we implement the moving mesh approach using a U-Net as proposed by Sheikhjafari et al. Additionally, we augment the moving mesh approach using an INR. In the following chapter, the moving mesh approach is introduced.

3

The Moving Mesh Approach for Deformable Image Registration

In this chapter, the moving mesh approach for deformable image registration proposed by Sheikhjafari et al. in 2022 is introduced [75]. The moving mesh method was originally developed for diffeomorphic grid generation in the context of partial differential equation. The first section 3.1 examines the origin of the moving mesh approach for grid generation.

The second section 3.2 provides the proof that the moving mesh deformation results in a diffeomorphic deformation field. The provided proof is by Liu [47]; we add further details for enhanced understanding. As Abel's lemma is needed as an auxiliary lemma for the main proof, it is introduced in this section. The section also covers how the div-curl-system, one of the main problems arising from the moving mesh approach, can be transformed into a set of Poisson equations. This is a result by [75]; we provide the computations. Then, a brief overview of the adaptations of the div-curl-system for the three-dimensional case are provided.

In section 3.3, the application of the moving mesh approach for deformable image registration is described. Afterwards, the learning-based moving mesh approach proposed by Sheikhjafari et al. is described.

Finally, we provide details on the implementation of the moving mesh approach in section 3.4. The section covers the implementation details on solving the PDE and the ODE, as well as the network architecture for the U-Net and the INR architecture.

3.1 Moving Mesh Grid Generation

Grid generation is a crucial part in the numerical solution of PDEs, as the accuracy and the computational efficiency of the solution directly depend on the discretisation [47]. Therefore, the solution field needs to be discretised – in the simple case on a grid; more generically on a *mesh*. Depending on the characteristics of the PDE, local refinement of the grid at regions of interest can benefit the accuracy and computational costs. Adaptive grid generation enhances the solution accuracy by assigning a denser grid to regions where large variation of the solution is expected and coarse grids to expected smooth regions. In the *moving mesh* approach, the overall number of grid points remains constant, as the grid points are progressively moved towards regions where higher accuracy is required. Moving the existing grid points avoids further computational costs that would emerge by adding more grid points. This approach improves the solutions accuracy, especially in regions of interest, and ensures computational efficiency of the numerical computation of the solution.

Liao et al. [45, 46] proposed the moving deformation method, an approach for moving mesh generation. Their aim is to find a deformation field describing the relocation of the mesh points, that is additionally diffeomorphic. A key feature of their approach is the *monitor function* $\mu : \Omega \to \mathbb{R}$ that is utilised to describe the movement of the grid points by describing the Jacobian determinant of the mesh at each position. The studied problem can be summarised as:

Problem 3.1 ([45])**.** Let $\Omega \subset \mathbb{R}^n$ be a bounded open domain. Let $\mu \in C^1(\bar{\Omega}), \mu > 0$, and

$$\int_\Omega \mu(\xi) \mathrm{d}\xi = |\Omega|. \tag{3.1}$$

Find a C^1 diffeomorphism ϕ from Ω onto itself such that

$$\begin{aligned} \det \nabla \phi(\xi) &= \mu(\xi) \text{ for } \xi \in \Omega, \\ \phi(\xi) &= \xi \text{ for } \xi \in \partial\Omega. \end{aligned} \tag{3.2}$$

Their approach to resolve this problem refers back to the results of Moser, who studied volume elements on Riemannian manifolds without boundaries [56]. Banyanga extended Moser's results to manifolds with boundaries [5], and Dacorongna and Tartar independently proposed an extension to bounded domains on $\mathbb{R}^n$ for $n \geq 2$ [19]. They resolved problem 3.1 by the following theorem:

Theorem 3.2 (c.f. [19])**.** *Let $k \geq 0$ be an integer and $\alpha > 0$. Let $\Omega \subset \mathbb{R}^n$ be a bounded open set with $C^{3+k,\alpha}$ boundary $\partial\Omega$. Let $\mu \in C^{k,\alpha}(\overline{\Omega})$ and*

$$\int_\Omega \mu(\xi) \mathrm{d}\xi = |\Omega|. \tag{3.3}$$

Then, there exists

$$\phi \in \mathrm{Diff}^{k+1,\alpha}(\overline{\Omega}), \tag{3.4}$$

satisfying

$$\det \nabla \phi(\xi) = \mu(\xi) \text{ for } \xi \in \Omega, \text{ and } \phi(\xi) = \xi \text{ for } \xi \in \partial\Omega. \tag{3.5}$$

Typical computational domains, such as cubes, do not fulfil the condition of a $C^{3+k,\alpha}$ boundary. Therefore, a large class of problems is not included in this result. This problem is resolved by the following theorem proposed by Liao et al. It states that for two-dimensional and three-dimensional unit volumes, a diffeomorphism with predefined Jacobian determinant can be found:

Theorem 3.3 (c.f. [45])**.** *Let $\Omega = (0,1) \times \cdots \times (0,1) \subset \mathbb{R}^n, n = 2, 3$. Let $\mu \in C^1(\Omega)$ with $\mu > 0$ in Ω and $\mu = 1$ on $\partial\Omega$, with*

$$\int_\Omega \mu(\xi) d\xi = 1. \tag{3.6}$$

Then, there exists $\phi \in \mathrm{Diff}^1(\Omega) \cap \mathrm{Diff}^0(\bar{\Omega})$ satisfying

$$\begin{aligned} \det \nabla \phi(\xi) &= \mu(\xi) \text{ for } \xi \in \Omega, \\ \phi(\xi) &= \xi \text{ for } \xi \in \partial\Omega. \end{aligned} \tag{3.7}$$

This theorem 3.3 is only stated for unit domains, but Liao et al. showed the extension to any connected bounded open domain in $\mathbb{R}^n$ with an argument involving partition of unity. Therefore, this result can be generalised to rectangular domains.

Now, we explain the construction of a deformation field satisfying problem 3.1 as proposed by Liao et al. First, they define a monitor function μ describing the mesh redistribution. Then, a vector field V is defined, satisfying $\operatorname{div} V = \mu - 1$; the construction of such a vector field is addressed in section 3.2.1. Then, a velocity field

$$\mathcal{V}_t(\xi) = \frac{V(\xi)}{t + (1-t)\mu(\xi)} \tag{3.8}$$

is constructed, using the vector field V and the monitor function μ for an artificial time $t \in [0,1]$. The deformation equation for $\phi_t : \bar{\Omega} \times [0,1] \to \mathbb{R}^n$ can be formulated to establish the final deformation field ϕ_1, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(\xi) = \mathcal{V}_t(\phi_t(\xi)) \text{ for } t \in [0,1], \tag{3.9}$$

with the initial condition $\phi_0(\xi) = \xi$. This results in an ODE for every $\xi \in \bar{\Omega}$, with a unique solution due to the boundary conditions $V = 0$ and $\phi_t(\xi) = \xi$ on $\partial\Omega$ [45]. The final deformation field satisfying (3.7) is given by ϕ_1. The monitor function μ controls the Jacobian determinant of deformation field describing the movement of the grid point, as the velocity field is scaled by the monitor function. The proof that $\det \nabla\phi_1(\xi) = \mu(\xi)$ for all $\xi \in \Omega$ is provided in the next section 3.2. The resulting velocity field ϕ_t is diffeomorphic, as its Jacobian determinant is forced positive, thus, no grid folding occurs. All steps of this approach are summarised in algorithm 1.

Before we cover how this result can be applied to deformable image registration to obtain a guaranteed diffeomorphic deformation field, in section 3.3, we provide the proof behind this concept in the following section.

3.2 Proof of Diffeomorphic Deformation Field Generation via the Moving Mesh Approach

In this section, we present the proof provided by Liu in [47], to show that the moving mesh approach is used to establish a deformation field ϕ that is guaranteed to be diffeomorphic. This is achieved by restricting its Jacobian determinant J_ϕ on the old grid to be positive. The Jacobian determinant of the deformation field is prescribed by the monitor function μ as the deformation field ϕ is parameterised by the monitor function μ and a curl of end velocity field γ. The proof also highlights why the vector field V needs to satisfy $\operatorname{div} V = \mu - 1$. For enhanced clarity and understanding, we add further details to the proof.

To get an overview of the steps of the proof, the main steps are roughly summarised:

- First, Abel's lemma that acts as an auxiliary lemma is introduced.
- The main result is to prove that $\mu(\xi) = \det \nabla\phi_1(\xi)$ holds.
- We show this by constructing a mapping H such that $H(\xi, 0) = \mu(\xi)$ and $H(\xi, 1) = \det \nabla\phi_1(\xi)$.

Algorithm 1: Moving Mesh Grid Generation [47]

Step 1: Define a continuous monitor function $\mu : \Omega \to \mathbb{R}$ that is constrained by

$$\int_\Omega \mu(\xi)\mathrm{d}\xi = |\Omega|. \tag{3.10}$$

Step 2: Find a vector field V satisfying

$$\operatorname{div} V = \mu - 1. \tag{3.11}$$

Step 3: Construct a velocity field with an artificial time $t \in [0, 1]$:

$$\mathcal{V}_t(\xi) = \frac{V(\xi)}{t + (1-t)\mu(\xi)}. \tag{3.12}$$

Step 4: Solve the ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(\xi) = \mathcal{V}_t(\phi_t(\xi)). \tag{3.13}$$

Step 5: Evaluate $\phi_t(\xi)$ for $t = 1$ to obtain the final deformation field ϕ_1.

- Then, $\frac{\partial}{\partial t}H(\xi, t)$ is computed with the help of Abel's lemma.
- Finally, we show that $\frac{\partial}{\partial t}H(\xi, t) = 0$ for $t \in [0, 1]$ and for all $\xi \in \Omega$, which results in the main result $\mu(\xi) = \det \nabla\phi_1(\xi)$.

First, we introduce Abel's Lemma:

Lemma 3.4 (Abel's Lemma)**.** *Let $M(t)$ be a $d \times d$ matrix such that each element of the matrix is differentiable on t, let $\frac{\mathrm{d}}{\mathrm{d}t}M(t)$ be the matrix with the differentiated elements of $M(t)$. If $\frac{\mathrm{d}}{\mathrm{d}t}M(t) = A(t)M(t)$, where $A(t)$ is a $d \times d$ matrix, then*

$$\frac{\mathrm{d}}{\mathrm{d}t}(\det M(t)) = (\operatorname{trace} A(t))(\det M(t)). \tag{3.14}$$

The proof of Abel's Lemma can be found in the appendix in section A.1.

Now to the main proof. Consider the following problem: for a given monitor function μ, find a diffeomorphic mapping ϕ such that

$$J_{\phi_1}(\xi) := \det \nabla\phi_1(\xi) = \mu(\xi). \tag{3.15}$$

First, we revise the assumptions. We define a monitor function $\mu : \Omega \to \mathbb{R}$, with the additional condition $\mu(\xi) > 0$ for all $\xi \in \Omega$, satisfying step 1 of algorithm 1. Then, we assume a vector field $V : \Omega \to \mathbb{R}^d$ exists such that $\operatorname{div} V = \mu - 1$, satisfying step 2 of algorithm 1. To show that (3.15) holds for the final deformation field ϕ_1, we use the vector field V as in step 3 to define a velocity field $\mathcal{V}_t : \Omega \times [0, 1] \to \mathbb{R}^d$ with $\mathcal{V}_t(\xi) := \mathcal{V}(\xi, t)$ for an artificial time $t \in [0, 1]$ such that

$$\mathcal{V}_t(\xi) = \frac{V(\xi)}{t + (1-t)\mu(\xi)}. \tag{3.16}$$

The final deformation field ϕ_1 is computed by solving the ODE of step 4

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(\xi) = \mathcal{V}_t(\phi_t(\xi)) \text{ for } t \in [0,1], \tag{3.17}$$

where $\phi_t : \Omega \times [0,1] \to \mathbb{R}^d$ with $\phi_t(\xi) := \phi(\xi, t)$ and with the initial condition $\phi_0(\xi) = \xi$.

Now, we can show that the solution ϕ_1 satisfies (3.15). To this end, we define a mapping $H : \Omega \times [0,1] \to \mathbb{R}$ by

$$\begin{aligned} H(\xi,t) &= J_{\phi_t}(\xi)\ [t + (1-t)\mu(\phi_t(\xi))] \\ &= \det \nabla \phi_t(\xi)\ [t + (1-t)\mu(\phi_t(\xi))], \end{aligned} \tag{3.18}$$

where ϕ_t is the deformation field at time step t with an artificial time $t \in [0,1]$. The mapping H is defined such that $H(\xi,t)$ at $t = 0$ evaluates to

$$H(\xi,0) = \underbrace{(\det \nabla \phi_0(\xi))}_{=1}\ \mu(\phi_0(\xi)) = \mu(\xi), \tag{3.19}$$

as $\phi_0(\xi)$ is the identity mapping, thus $\phi_0(\xi) = \xi$, and therefore its Jacobian determinant is equal to 1. For $t = 1$, $H(t,\xi)$ evaluates to the Jacobian determinant of the deformation $\phi_1(\xi)$,

$$H(\xi,1) = (\det \nabla \phi_1(\xi)) \cdot 1 = \det \nabla \phi_1(\xi). \tag{3.20}$$

By showing that

$$\frac{\partial}{\partial t} H(\xi,t) = 0 \tag{3.21}$$

for all $t \in [0,1]$ and for all $\xi \in \Omega$, implying $H(\xi,0) = H(\xi,1)$, and therefore

$$J_{\phi_1}(\xi) = \det \nabla \phi_1(\xi)) = \mu(\xi), \tag{3.22}$$

thus the Jacobian determinant $J_{\phi_1}(\xi)$ for the final deformation ϕ_1 is given by the monitor $\mu(\xi)$ for all $\xi \in \Omega$.

To aid readability, the ξ arguments are omitted in the remainder of this proof. We compute the partial derivative $\frac{\partial H}{\partial t}$ by using the product rule for differentiation on (3.18), resulting in

$$\begin{aligned} \frac{\partial H}{\partial t} &= \frac{\partial}{\partial t}[(\det \nabla \phi_t)(t + (1-t)\mu(\phi_t))] \\ &= \frac{\partial}{\partial t}[\det \nabla \phi_t][t + (1-t)\mu(\phi_t)] \\ &\quad + (\det \nabla \phi_t)\frac{\partial}{\partial t}[t + (1-t)\mu(\phi_t)]. \end{aligned} \tag{3.23}$$

In order to compute $\frac{\partial}{\partial t}(\det \nabla \phi_t)$, Abel's lemma 3.4 is applied. First, we swap the order of the gradient and the partial derivative,

$$\frac{\partial}{\partial t}(\nabla \phi_t) = \nabla \left(\frac{\partial \phi_t}{\partial t} \right). \tag{3.24}$$

Applying (3.16) to the right hand side, we obtain

$$\nabla\left(\frac{\partial\phi_t}{\partial t}\right) = \nabla(\mathcal{V}_t(\phi_t)). \tag{3.25}$$

Then, the chain rule for differentiation leads to

$$\nabla(\mathcal{V}_t(\phi_t)) = (\nabla_\phi \mathcal{V}_t(\phi_t))(\nabla\phi_t). \tag{3.26}$$

Combining the last steps shows that

$$\frac{\partial}{\partial t}(\nabla\phi_t) = (\nabla_\phi \mathcal{V}_t(\phi_t))(\nabla\phi_t) \tag{3.27}$$

holds, thus the matrices $\nabla\phi_t$ and $\nabla_\phi\mathcal{V}_t$ meet the requirements of Abel's Lemma, which we now use to compute

$$\frac{\partial}{\partial t}(\det\nabla\phi_t) = \operatorname{trace}(\nabla_\phi \mathcal{V}_t(\phi_t))(\det\nabla\phi_t). \tag{3.28}$$

As $\operatorname{trace}(\nabla_\phi\mathcal{V}_t(\phi_t))$ is the sum of the diagonal elements of the Jacobian $\nabla_\phi\mathcal{V}_t(\phi_t)$, we obtain

$$\operatorname{trace}(\nabla_\phi \mathcal{V}_t(\phi_t)) = \sum_{i=1}^{d} \frac{\partial}{\partial(\phi_t)_i}(\mathcal{V}_t(\phi_t))_i = \operatorname{div}_\phi \mathcal{V}_t(\phi_t), \tag{3.29}$$

which is the divergence of $\mathcal{V}_t$ with respect to ϕ. Combining (3.28) and (3.29) leads to

$$\frac{\partial}{\partial t}(\det\nabla\phi_t) = (\operatorname{div}_\phi \mathcal{V}_t(\phi_t))(\det\nabla\phi_t). \tag{3.30}$$

This result can be plugged into (3.23):

$$\begin{aligned}\frac{\partial H}{\partial t} &= (\operatorname{div}_\phi \mathcal{V}_t(\phi_t))(\det\nabla\phi_t)[t + (1-t)\mu(\phi_t)] \\ &\quad + (\det\nabla\phi_t)\frac{\partial}{\partial t}[t + (1-t)\mu(\phi_t)].\end{aligned} \tag{3.31}$$

In the second term, $\frac{\partial}{\partial t}[t + (1-t)\mu(\phi_t)]$ is computed by applying the chain rule for differentiation on $\frac{\partial}{\partial t}\mu(\phi_t)$ and applying (3.17):

$$\begin{aligned}\frac{\partial}{\partial t}\mu(\phi_t) &= (\nabla\mu(\phi_t))^\top\left(\frac{\partial}{\partial t}\phi_t\right) \\ &= (\nabla\mu(\phi_t))^\top\mathcal{V}_t(\phi_t).\end{aligned} \tag{3.32}$$

Therefore, with the product rule for differentiation,

$$\begin{aligned}\frac{\partial H}{\partial t} &= (\operatorname{div}_\phi \mathcal{V}_t(\phi_t))(\det\nabla\phi_t)[t + (1-t)\mu] \\ &\quad + (\det\nabla\phi_t)[1 - \mu(\phi_t) + (1-t)(\nabla\mu(\phi_t))^\top\mathcal{V}_t(\phi_t)].\end{aligned} \tag{3.33}$$

Factoring out $\det \nabla\phi$, the derivative simplifies to

$$\begin{aligned}\frac{\partial H}{\partial t} = (\det \nabla\phi_t)\Big[&(\operatorname{div}_\phi \mathcal{V}_t(\phi_t))[t + (1-t)\mu(\phi_t)] \\ &+ [1 - \mu(\phi_t) + (1-t)(\nabla\mu(\phi_t))^\top \mathcal{V}_t(\phi_t)]\Big].\end{aligned} \tag{3.34}$$

The next few steps further simplify the derivative $\frac{\partial H}{\partial t}$. By rearranging (3.16), we express V as

$$V = \mathcal{V}_t[t + (1-t)\mu], \tag{3.35}$$

which we use to compute the divergence of V with the product rule for differentiation

$$\begin{aligned}\operatorname{div} V &= \operatorname{div}(\mathcal{V}_t[t + (1-t)\mu]) \\ &= (\operatorname{div} \mathcal{V}_t)[t + (1-t)\mu] + (1-t)(\nabla\mu)^\top \mathcal{V}_t.\end{aligned} \tag{3.36}$$

This equation is rearranged to

$$(\operatorname{div} \mathcal{V}_t)[t + (1-t)\mu] = \operatorname{div} V - (1-t)(\nabla\mu)^\top \mathcal{V}_t. \tag{3.37}$$

Therefore,

$$(\operatorname{div}_\phi \mathcal{V}_t(\phi_t))[t + (1-t)\mu(\phi_t)] = \operatorname{div}_\phi V(\phi_t) - (1-t)(\nabla\mu(\phi_t))^\top \mathcal{V}_t(\phi_t), \tag{3.38}$$

which is plugged into (3.34), leaving

$$\begin{aligned}\frac{\partial H}{\partial t} &= (\det \nabla\phi_t)\Big[(\operatorname{div}_\phi \mathcal{V}_t(\phi_t))[t + (1-t)\mu(\phi_t)] + [1 - \mu(\phi_t) + (1-t)(\nabla\mu(\phi_t))^\top \mathcal{V}_t]\Big] \\ &= (\det \nabla\phi_t)\Big[\operatorname{div} V(\phi_t) - (1-t)(\nabla\mu(\phi_t))^\top \mathcal{V}_t(\phi_t) + 1 - \mu(\phi_t) + (1-t)(\nabla\mu(\phi_t))^\top \mathcal{V}_t\Big] \\ &= (\det \nabla\phi_t)\Big[\operatorname{div}_\phi V(\phi_t) + 1 - \mu(\phi_t)\Big].\end{aligned} \tag{3.39}$$

As V is constructed such that $\operatorname{div} V = \mu - 1$, this leads to

$$\operatorname{div}_\phi V(\phi_t) = \mu(\phi_t) - 1, \tag{3.40}$$

resulting in

$$\begin{aligned}\frac{\partial H}{\partial t} &= (\det \nabla\phi_t)(\operatorname{div}_\phi V(\phi_t) + 1 - \mu(\phi_t)) \\ &= (\det \nabla\phi_t)(\mu(\phi_t) - 1 + 1 - \mu(\phi_t)) \\ &= 0.\end{aligned} \tag{3.41}$$

Thus, together with (3.19) and (3.20), the main result

$$J_{\phi_1}(\xi) = \det \nabla\phi_1(\xi) = \mu(\xi) \text{ for all } \xi \in \Omega \tag{3.42}$$

is proven. Consequently, the Jacobian determinant of the deformation field ϕ computed by the moving mesh approach is defined by the monitor function μ. By choosing a positive monitor function, the moving mesh approach for grid generation results in a diffeomorphic deformation field ϕ. The remaining problem is to find a vector field V satisfying $\operatorname{div} V(\xi) = \mu(\xi) - 1$ for all $\xi \in \Omega$. We address this issue in the following section.

3.2.1 The Div-Curl System

One of the main components in the moving mesh approach is the construction of a vector field V satisfying step 2 of algorithm 1. There are different approaches to compute such vector field [47]. One approach is based on solving Poisson equations. The condition $\operatorname{div} V = \mu - 1$ alone does not ensure a unique solution for V, thus more constraints are needed. As there are the two components V_1 and V_2 for the vector field V in the two-dimensional case, at least two equations are necessary to obtain a unique solution. A way to achieve this is by adding a constraint on the curl of the vector field with Dirichlet boundary conditions [92]. The deformation field can be represented by the divergence and curl (div-curl) system [16], leading to the set of equations

$$\begin{cases} \operatorname{div} V = \dfrac{\partial}{\partial \xi_1} V_1 + \dfrac{\partial}{\partial \xi_2} V_2 = \mu - 1, \\ \operatorname{curl} V = \dfrac{\partial}{\partial \xi_1} V_2 - \dfrac{\partial}{\partial \xi_2} V_1 = \gamma, \end{cases} \tag{3.43}$$

where $\gamma : \mathbb{R}^2 \to \mathbb{R}$ is the curl of the final velocity field $\mathcal{V}_1$, and since

$$\mathcal{V}_1(\xi) = \frac{V(\xi)}{1 + (1-1)\mu(\xi)} = V(\xi), \tag{3.44}$$

γ is defined as

$$\gamma := \operatorname{curl}(V). \tag{3.45}$$

Therefore, the curl of V remains the same, whereas its divergence is adapted according to the monitor function.

To solve the div-curl system for V, it can be transformed into a set of Poisson equations, where V_1 and V_2 are the components of the vector field V and therefore the target variables. This result is described in [75]; we now present our computations for this result. Again, for enhanced readability, the ξ arguments are omitted.

To obtain the Poisson equations, the Laplacians ΔV_1 and ΔV_2 are needed. Therefore, the second order derivatives needed for ΔV_1 and ΔV_2 are computed using (3.43). The first step to achieve a set of Poisson equations is to rearrange (3.43) to eliminate the derivative of V_1 in ξ_1-direction

$$\frac{\partial}{\partial \xi_1} V_1 = \mu - 1 - \frac{\partial}{\partial \xi_2} V_2. \tag{3.46}$$

To get the second order derivative, this result is differentiated

$$\begin{aligned} \frac{\partial^2}{\partial \xi_1^2} V_1 &= \frac{\partial}{\partial \xi_1} \left(\mu - 1 - \frac{\partial}{\partial \xi_2} V_2 \right) \\ &= \frac{\partial}{\partial \xi_1} \mu - \frac{\partial^2}{\partial \xi_1 \partial \xi_2} V_2. \end{aligned} \tag{3.47}$$

Analogously, the derivative of V_2 with respect to ξ_2 can be expressed as

$$\frac{\partial}{\partial \xi_2} V_2 = \mu - 1 - \frac{\partial}{\partial \xi_1} V_1. \tag{3.48}$$

Differentiating this result, we obtain the second order derivative

$$\begin{aligned}\frac{\partial^2}{\partial\xi_2^2}V_2 &= \frac{\partial}{\partial\xi_2}\left(\mu - 1 - \frac{\partial}{\partial\xi_1}V_1\right)\\ &= \frac{\partial}{\partial\xi_2}\mu - \frac{\partial^2}{\partial\xi_1\partial\xi_2}V_1.\end{aligned} \tag{3.49}$$

Now, we compute the partial derivative of the curl

$$\begin{aligned}\frac{\partial}{\partial\xi_1}\operatorname{curl} V &= \frac{\partial}{\partial\xi_1}\left(\frac{\partial}{\partial\xi_1}V_2 - \frac{\partial}{\partial\xi_2}V_1\right)\\ &= \frac{\partial^2}{\partial\xi_1^2}V_2 - \frac{\partial^2}{\partial\xi_1\partial\xi_2}V_1 = \frac{\partial}{\partial\xi_1}\gamma.\end{aligned} \tag{3.50}$$

The second-order partial derivative of V_1 with respect to ξ_2 is isolated

$$\frac{\partial^2}{\partial\xi_1^2}V_2 = \frac{\partial}{\partial\xi_1}\gamma + \frac{\partial^2}{\partial\xi_1\partial\xi_2}V_1. \tag{3.51}$$

Analogously, the second order partial derivative of ξ_1 is obtained by differentiating the curl in ξ_2-direction,

$$\begin{aligned}\frac{\partial}{\partial\xi_2}\operatorname{curl} V &= \frac{\partial}{\partial\xi_2}\left(\frac{\partial}{\partial\xi_1}V_2 - \frac{\partial}{\partial\xi_2}V_1\right)\\ &= \frac{\partial^2}{\partial\xi_2\partial\xi_1}V_2 - \frac{\partial^2}{\partial\xi_2^2}V_1 = \frac{\partial}{\partial\xi_2}\gamma,\end{aligned} \tag{3.52}$$

and rearranging the result to isolate the second-order derivative

$$\frac{\partial^2}{\partial\xi_2^2}V_1 = \frac{\partial^2}{\partial\xi_1\partial\xi_2}V_2 - \frac{\partial}{\partial\xi_2}\gamma. \tag{3.53}$$

Combining (3.47) and (3.53), the Laplacian ΔV_1 can be computed:

$$\begin{aligned}\Delta V_1 &= \frac{\partial^2}{\partial\xi_1^2}V_1 + \frac{\partial^2}{\partial\xi_2^2}V_1\\ &= \frac{\partial}{\partial\xi_1}\mu - \frac{\partial^2}{\partial\xi_1\partial\xi_2}V_2 + \frac{\partial^2}{\partial\xi_1\partial\xi_2}V_2 - \frac{\partial}{\partial\xi_2}\gamma\\ &= \frac{\partial}{\partial\xi_1}\mu - \frac{\partial}{\partial\xi_2}\gamma.\end{aligned} \tag{3.54}$$

For V_2, combining (3.49) and (3.51) leads to

$$\begin{aligned}\Delta V_2 &= \frac{\partial^2}{\partial\xi_1^2}V_2 + \frac{\partial^2}{\partial\xi_2^2}V_2\\ &= \frac{\partial}{\partial\xi_1}\gamma + \frac{\partial^2}{\partial\xi_1\partial\xi_2}V_1 + \frac{\partial}{\partial\xi_2}\mu - \frac{\partial^2}{\partial\xi_1\partial\xi_2}V_1\\ &= \frac{\partial}{\partial\xi_2}\gamma + \frac{\partial}{\partial\xi_1}\mu.\end{aligned} \tag{3.55}$$

Together, (3.54) and (3.55) lead to the set of Poisson equations

$$\begin{cases} \Delta V_1 = \dfrac{\partial}{\partial \xi_1}\mu - \dfrac{\partial}{\partial \xi_2}\gamma, \\ \Delta V_2 = \dfrac{\partial}{\partial \xi_2}\mu + \dfrac{\partial}{\partial \xi_1}\gamma, \end{cases} \tag{3.56}$$

with target variables V_1 and V_2 building the vector field $V = (V_1, V_2)$. The vector field V obtained by solving the set of Poisson equations with Dirichlet boundary conditions satisfies $\operatorname{div} V = \mu - 1$.

3.2.2 Adaptation to Three Dimensions

The previous section covered the div-curl-system for the two-dimensional case. However, the approach can be adapted to three-dimensional data, which we briefly explain. One of the main changes is the div-curl system to compute the – now three-dimensional – vector field $V = (V_1, V_2, V_3)$. As the curl of three-dimensional end velocity field γ is three-dimensional as well, the div-curl system leads to four equations for three variables V_1, V_2, and V_3, leaving an overdetermined system of equations. Therefore, a dummy variable θ is introduced with $\theta \equiv 0$ in Ω and $\theta = 0$ on $\partial\Omega$ to solve the system; more details can be found in [13, 47]. The corresponding div-curl system is

$$\begin{cases} \operatorname{div} V &= \dfrac{\partial V_1}{\partial \xi_1} + \dfrac{\partial V_2}{\partial \xi_2} + \dfrac{\partial V_3}{\partial \xi_3} = f^1, \\ \operatorname{curl}_1 V &= \dfrac{\partial \theta}{\partial \xi_1} + \dfrac{\partial V_3}{\partial \xi_2} - \dfrac{\partial V_2}{\partial \xi_3} = f^2, \\ \operatorname{curl}_2 V &= \dfrac{\partial \theta}{\partial \xi_2} + \dfrac{\partial V_1}{\partial \xi_3} - \dfrac{\partial V_3}{\partial \xi_1} = f^3, \\ \operatorname{curl}_3 V &= \dfrac{\partial \theta}{\partial \xi_3} + \dfrac{\partial V_2}{\partial \xi_1} - \dfrac{\partial V_1}{\partial \xi_2} = f^4. \end{cases} \tag{3.57}$$

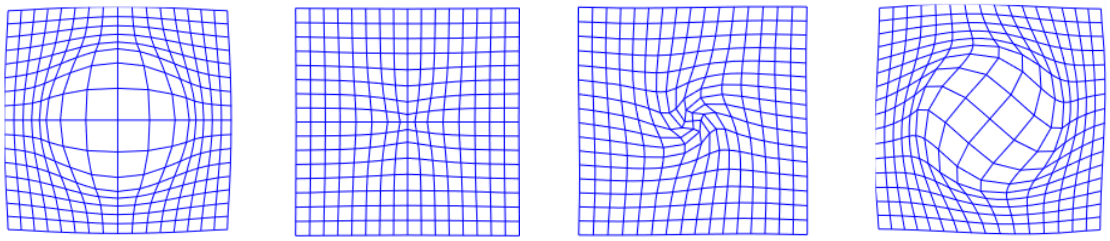
Similarly to the two-dimensional case, the system can be converted to a set of three Poisson equations

$$\begin{cases} \Delta V_1 = \dfrac{\partial f^1}{\partial \xi_1} + \dfrac{\partial f^3}{\partial \xi_3} - \dfrac{\partial f^4}{\partial \xi_2} = F^1, \\ \Delta V_2 = \dfrac{\partial f^1}{\partial \xi_2} + \dfrac{\partial f^4}{\partial \xi_1} - \dfrac{\partial f^2}{\partial \xi_3} = F^2, \\ \Delta V_3 = \dfrac{\partial f^1}{\partial \xi_3} + \dfrac{\partial f^2}{\partial \xi_2} - \dfrac{\partial f^3}{\partial \xi_1} = F^3, \end{cases} \tag{3.58}$$

which can be solved with Dirichlet boundary conditions to obtain the vector field V.

3.3 The Moving Mesh Approach in Image Registration

In this section, we examine how the moving mesh grid generation is applied to image registration. Therefore, we explain how the moving mesh approach is used to parameterise the deformation field. After that, the learning-based moving mesh approach for deformable image registration by Sheikhjafari et al. that we implement and augment for this thesis is presented.



(a) $(\mu, \gamma) = (10, 0)$ (b) $(\mu, \gamma) = (0.1, 0)$ (c) $(\mu, \gamma) = (0.1, 1)$ (d) $(\mu, \gamma) = (10, -2)$

Figure 3.1: Geometrical interpretation of two-dimensional deformation fields using different parameters for the Jacobian determinant μ and the curl of the final velocity field γ. The relation of μ to the radial component (extension or shrinkage of the grid cells at the centre) and of γ to the rotational component (twisting of the grid) of the deformation field are observable (illustration credit: [14]).

3.3.1 Moving Mesh-Based Image Registration

Parameterisation of the Deformation Field

The moving mesh based image registration is categorised as parametric image registration, as the deformation field is parameterised as follows. By applying the moving mesh grid generation to the image registration problem, the deformation field ϕ can be entirely described by its Jacobian determinant μ and a curl of end velocity field γ. As $\operatorname{div} V = \mu - 1$ holds, the deformation field is characterised by the divergence and the curl, which are directly related to radial and rotational motion. Therefore, the resulting deformation fields are ideal to model the cardiac motion, as the movement of the heart can be divided into radial and rotational movements. A geometrical interpretation of two-dimensional deformation fields for different values of μ and γ are displayed in figure 3.1.

The corresponding image registration problem can be stated as a constrained optimisation problem: for a given pair of discrete grey-valued images $I_F, I_M : \mathcal{X} \to \mathbb{R}$, find μ and γ parameterising ϕ, such that ϕ is the solution of

$$\min_{\phi:\Omega\to\mathbb{R}^n} \mathcal{L}(I_F, I_M \circ \phi) \tag{3.59}$$

under the constraints

$$\begin{cases} \displaystyle\int_\Omega \mu(\xi)\, d\xi = |\Omega|, \\ \tau_{\text{ub}} \geq \mu(\xi) \geq \tau_{\text{lb}} > 0. \end{cases} \tag{3.60}$$

The first constraint ensures that the domain remains the same after the deformation, while the parameters τ_{lb} and τ_{ub} of the second constraint represent the lower and the upper bound of the Jacobian determinant and therefore determine the range of allowed deformation [64]. By forcing τ_{lb} to be positive, the deformation field is guaranteed to be diffeomorphic, making it suitable for medical image registration, as tissue folding is prevented [47]. In addition, constraining the Jacobian determinant preserves topology and as the Jacobian determinant is related to cell volumes, the incompressibility of the registered object is approximately ensured if τ_{ub} and τ_{lb} are chosen close to 1 [14]. The

smoothness of the deformation field is implicitly given by the parameterisation, leading to plausible registration results.

Forward and Backward Deformation Field

For certain image registration problems it is important not only to find the deformation field registering the moving image I_M to the fixed image I_F, but also the fixed to the moving image. Since the deformation field computed by the moving mesh approach is diffeomorphic and thus invertible, both deformation fields can be computed. As introduced in section 2.3.6, the forward and the backward deformation field are denoted as $\phi_\mathrm{f} := \phi_1$ and $\phi_\mathrm{b} := \phi_1^{-1}$.

The moving mesh approach has the advantage that it allows for a simultaneous computation of both deformation fields. The forward deformation field ϕ_f comes from the ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(\xi) = \mathcal{V}_t(\phi_t(\xi)), \tag{3.61}$$

with the initial condition $\phi_0(\xi) = \xi$. The backward deformation field ϕ_b is computed by solving the ODE with the negative, time-inverted velocity field $-\mathcal{V}_{(1-t)}$, using the same initial condition as for ϕ_f.

3.3.2 Learning-Based Deformable Cardiac Image Registration with the Moving Mesh Parameterisation

Several applications of the moving mesh parameterisation to image registration problems have been proposed [14, 64, 42], as also discussed in section 1.2. Although these approaches have shown promising registration results, the deformation fields are established iteratively and therefore do not benefit from the advantages of machine learning, as discussed in section 2.4.4. To the best of our knowledge, the first approach to apply machine learning to the moving mesh parameterisation was proposed in [75]. The authors proposed an end-to-end unsupervised U-Net based approach for cardiac image registration using the moving mesh parameterisation of the deformation field. This approach is a parametric deformable intensity-based image registration method with a guaranteed diffeomorphic deformation field. As the method is unsupervised, ground truth for the registration is not required.

Using the moving mesh parameterisation, the image registration problem can be stated as the constrained optimisation problem (3.59). Sheikhjafari et al. solve this problem using a U-Net, that learns the parameters μ and V and therefore implicitly γ of the moving mesh parameterisation. The U-Net input are the fixed image I_F and the moving image I_M. The output of the U-Net has three channels: one output channel represents the now discrete monitor function μ, the remaining two represent the discrete vector field V on the grid $\mathcal{X}$. The problem is stated as

$$\min_{\theta \in \mathbb{R}^n} \mathcal{L}(I_F, I_M, g_\theta(I_F, I_M)), \tag{3.62}$$

where the U-Net models the function g_θ with learnable parameters θ. The U-Net output then is computed as $(\mu, V) = g_\theta(I_F, I_M)$.

As the monitor function needs to satisfy the conditions (3.60), the constraints are now forced on the monitor function. If the boundaries of the Jacobian determinant τ_lb and τ_ub

Algorithm 2: Moving Mesh Based Deformable Image Registration [75]

Input: I_F and I_M: image pair to be registered,
$\tau_{\text{lb}}, \tau_{\text{ub}}$: boundaries for Jacobian determinant
Output: $\phi_{\text{f}}, \phi_{\text{b}}$: forward and backward deformation fields

Step 1: pass the input to the U-Net to compute μ and V
Step 2: moving mesh approach:
  Step 2.1: force constraints (3.60) on μ
  Step 2.2: compute the curl of the velocity field V to obtain Poisson equations
  Step 2.3: solve Poisson equations to obtain V, compute velocity field $\mathcal{V}_t$
  Step 2.4: solve ODEs to compute ϕ_{f} and ϕ_{b}
Step 4: compute the loss function
Step 5: update μ and V using back propagation

are positive, the resulting deformation field ϕ is diffeomorphic. Constraining the monitor function is such a crucial aspect of the moving mesh approach that we dedicate chapter 4 to this problem.

To obtain ϕ, the constrained monitor function μ_{new} and the vector field V are used to perform the steps 2 to 5 in algorithm 1. The vector field satisfying step 2 is obtained by solving the div-curl system as described in section 3.2.1. As described in the previous section 3.3.1, the ODE of step 4 is solved twice to establish the forward deformation field ϕ_{f} and the backward deformation field ϕ_{b}.

The next step is to warp the images; the results are used to evaluate the bidirectional loss function

$$\mathcal{L}_{\text{total}}(I_F, I_M, \phi_{\text{f}}, \phi_{\text{b}}) = \frac{1}{2} \cdot \mathcal{L}(I_F, I_M \circ \phi_{\text{f}}) \; + \; \frac{1}{2} \cdot \mathcal{L}(I_M, I_F \circ \phi_{\text{b}}), \tag{3.63}$$

where $\mathcal{L}$ denotes a dissimilarity metric. The loss is then used to update μ and V via backpropagation. Optimisation of μ and γ is performed with SGD.

A new image pair can be registered by evaluating the U-Net with the previously learned parameters, computing μ and V for the image pair. By applying steps 2 to 5 of algorithm 1, the forward and the backward deformation fields are established. Thus, to register a new image pair, no additional optimisation is required.

An overview of the network architecture is displayed in figure 3.2; a summary of all steps of the algorithm can be found in algorithm 2. The algorithm can be used for the registration of two-dimensional and three-dimensional images, the main difference lies in the div-curl system. The algorithm is designed for the registration of cardiac images of the ES phase and the ED phase.

For this thesis, we implement the moving mesh approach as described by Sheikhjafari et al. to validate the approach. Additionally, we augment their approach using an implicit neural representation. The following section covers implementation details and the used network architecture.

3.4 Implementation Details and Network Architecture

This section covers the implementation details and network architecture for our implementation of the moving mesh approach. First, we outline the implementation strate-

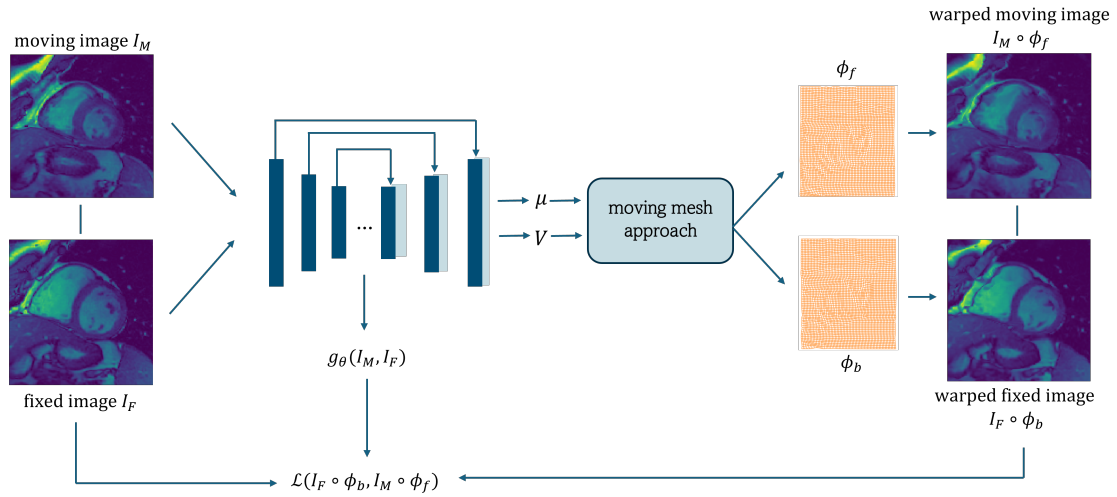


Figure 3.2: End-to-end unsupervised network architecture. The U-Net g_θ takes the images I_M and I_F as input. Its output is the monitor function μ, describing the Jacobian determinant of the deformation ϕ and the vector field V. Then, the moving mesh approach is applied to the output. With the resulting forward and backward deformation fields ϕ_f and ϕ_b, the registration is performed to compute the loss $\mathcal{L}$ to update μ and V using back propagation (adapted from [75]).

gies used for solving the ODEs and PDEs within the moving mesh approach. Then, the network architecture using a U-Net is described, followed by our augmentation using implicit neural representation. For our implementation, we use Python 3.7.6 with PyTorch 1.13.1 [60].

3.4.1 Numerically Solving the PDEs

To obtain the vector field V satisfying $\operatorname{div} V = \mu - 1$, the vector field returned by the U-Net is adjusted as described in section 3.2.1. To solve the system of Poisson equations numerically, the Poisson equations are discretised as described in section 2.2.3. Therefore, the first step is the computation of the two-dimensional DST of the right-hand side of the Poisson equations. The DST is computed using the DFT for the modified input as described in theorem 2.19. A faster implementation of the DFT can be performed with the FFT, which is implemented by `torch.fft.rfft`. The Dirichlet boundary conditions are implicitly enforced by using the DST.

In Fourier space, the solution is obtained by dividing by the diagonal entries of the – now diagonal – operator. The solution in the real space of the Poisson is then obtained by applying the two-dimensional inverse DFT.

3.4.2 Numerically Solving the ODEs

For the moving mesh approach, two ODEs need to be solved in order to obtain the deformation fields ϕ_f and ϕ_b. Although Sheikhjafari et al. used the Euler method with arbitrary time steps to integrate the vector field, in this thesis we use the fourth-order

Runge-Kutta method with 3/8 rule [43, 72]. Due to the stiffness of the ODE and resulting numerical issues of zero flow in the integration, the use of the Euler method did not lead to usable results. The initial value problems for the ODEs are solved with the `odeint` function from the `torchfiffeq` package using the method `rk4` [15].

The forward deformation field ϕ_f is computed by solving the ODE with the velocity field $\mathcal{V}_t$. Analogously, the backward deformation field ϕ_b is computed by solving the ODE with the negative time-inverted velocity field $-\mathcal{V}_{(1-t)}$, as described in 3.3.

3.4.3 Discretisation and Interpolation

Interpolation is needed to compute the deformed images as well as to evaluate the monitor function and the velocity field in between grid points. The bilinear interpolation relies on the PyTorch function `nn.functional.grid_sample`. The padding mode is chosen as "zeros" to match the Dirichlet boundary conditions. Only for the interpolation of μ, the padding mode is set to "border", as the values of μ should not be zero to prevent grid folding. As the function expects normalised values between -1 and 1, the computational domain is considered a cell-centred grid on $[-1, 1]^2$.

3.4.4 U-Net and INR Architecture

One part of the moving mesh approach proposed by Sheikhjafari et al. is the U-Net that generates the monitor function μ and the vector field V before applying the moving mesh method. The U-Net used in this thesis is based on the VoxelMorph U-Net. Now, the VoxelMorph U-Net architecture is briefly described. Additionally, the used network parameters are provided.

For the U-Net implementation, we utilise the PyTorch backend of the VoxelMorph framework, a U-Net framework for deformable parametric image registration [3, 4]. The code is provided in [20]. The input to the framework are the grey-valued moving and fixed images I_M and I_F, which are concatenated to form a two-channel input image. To apply the moving mesh approach, a three-channel output is needed, consisting of one channel for the monitor function μ and two channels for the vector field V.

We achieve this by choosing the same U-Net architecture as Sheikhjafari et al. [75]: the encoder contains a convolutional layer with 16 kernels, followed by down-convolution layers with 32, 64, and 64 kernels. The decoder performs up-convolutions with 64, 64, 32, 32, 32, 16, and 3 kernels. All kernels are of size 3×3. The optimisation of the kernel elements is performed with SGD and Adam optimisation, as all steps are designed to be differentiable [41]. The function parameters are shared, and therefore optimised globally for all training images. The used loss function is

$$\mathcal{L}_\mathrm{total}(I_F, I_M, \phi_\mathrm{f}, \phi_\mathrm{b}) = \frac{1}{2} \cdot \mathcal{L}(I_F, I_M \circ \phi_\mathrm{f}) \; + \; \frac{1}{2} \cdot \mathcal{L}(I_M, I_F \circ \phi_\mathrm{b}), \tag{3.64}$$

where we use the MSE as the dissimilarity metric $\mathcal{L}$. In the last U-Net layer, we replace the leakyReLU activation function in the channel representing the monitor function by our new activation function that bounds the values of μ to $[\tau_\mathrm{lb}, \tau_\mathrm{ub}]$. We introduce the new activation function in section 4.2.1.

The VoxelMorph input generator randomly chooses two input images for the training. For this thesis, images of the ED and the ES of the same patient and the same cardiac

cycle are registered. Therefore, the data generator is modified to match a randomly chosen images from the training set with its corresponding ED image, if the image is an ES image, or vice versa.

Although the VoxelMorph framework is designed for two- and three-dimensional input data, some adaptations are made to use the framework for two-dimensional images. Issues of the gradient function with two-dimensional inputs are resolved by using [1].

3.4.5 Moving Mesh Approach Using INR

As an additional validation tool for the moving mesh approach, we use INR, as it is more robust and – compared to U-Net – its registration accuracy does not depend on the training data set as the optimisation is performed for each image pair individually. For the implementation, we replace U-Net with a SIREN network. The SIREN network learns a continuous representation of the monitor function μ and the vector field V that are then processed with the moving mesh approach. The used SIREN network consists of five 512-dimensional layers.

Instead of directly using the images, the input to the SIREN network is a coordinate representation of the images. The image coordinates are forwarded in column-major representation. The activation functions for SIREN are mainly sine functions, where the frequency of the sine of the first layer `w0-initial` is set to 10. For the final activation function, we choose the identity function instead, as the final output should not be restricted to the range $[-1, 1]$. Additionally, to constrain the values of μ to $[\tau_{\text{lb}}, \tau_{\text{ub}}]$, we apply our new activation function to the monitor function. This activation function as well as the constraining methods for the monitor function are discussed in the next chapter.

4

Constraining the Monitor Function

The monitor function plays a crucial role in the moving mesh-based image registration, as it parameterises the deformation field. An important part of the moving mesh approach is the adaptation of the monitor function. As discussed in 3.3.1, in order to receive a diffeomorphic deformation field, the monitor function μ needs to satisfy the conditions

$$\begin{cases} \int_\Omega \mu(\xi)\,\mathrm{d}\xi = |\Omega|, \\ \tau_{\mathrm{ub}} \geq \mu(\xi) \geq \tau_{\mathrm{lb}} > 0. \end{cases} \tag{4.1}$$

As no further details on satisfying these constraints are provided by Sheikhjafari et al., three different approaches for constraining the monitor function are proposed in this chapter. In section 4.1, two methods based on prior work using clamping and simple scaling are described. Additionally, we develop a new approach to satisfy the conditions in section 4.2. We build a new activation function based on the Sigmoid function to avoid the use of the clamp function in 4.2.1. Additionally, we design a new approach to scaling of the monitor function in 4.2.2.

4.1 Constraining by Cropping and Scaling

4.1.1 Cropping and Scaling Once

As described by Chen et al. [14], the first approach consists of clamping and scaling of the monitor function. The clamping is performed by

$$\mu_{\mathrm{c}}(\xi) := \mathrm{clamp}(\mu(\xi)) = \min(\max(\mu(\xi), \tau_{\mathrm{lb}}), \tau_{\mathrm{ub}}) \text{ for all } \xi \in \Omega, \tag{4.2}$$

followed by scaling the clamped monitor function μ_{c} with

$$\mu_{\mathrm{new}}(\xi) = \mu_{\mathrm{c}}(\xi) \frac{|\Omega|}{\sum_{\xi' \in \Omega} \mu_{\mathrm{c}}(\xi')} \text{ for all } \xi \in \Omega. \tag{4.3}$$

With this approach, the integration constraint is satisfied, but the scaling can change the monitor function such that some of its values might exceed $[\tau_{\mathrm{lb}}, \tau_{\mathrm{ub}}]$. Hence, this approach suggested in the literature does in fact not guarantee that both constraints are satisfied.

4.1.2 Constraining by Repeated Cropping and Scaling

As the previous approach does not necessarily satisfy both conditions, we introduce a slight extension of the previous approach. Therefore, we repeatedly clamp and scale the monitor function until either the monitor function satisfies

$$\left| \int_{\Omega} \mu(\xi) \, \mathrm{d}\xi - |\Omega| \right| \leq 10^{-8} \tag{4.4}$$

after clamping, or a maximum of 25 iteration is performed. Although the altered monitor function should satisfy the constraint (3.60) in more cases than the previous approach, it is still not guaranteed that both constraints are satisfied. For a high number of iteration, the repeated clamping leads to larger constant regions of μ, which results in vanishing gradients in this area. The vanishing gradients can interfere with optimisation performed with gradient descent.

4.2 New Approaches to Constrain the Monitor Function

Both of the previous approaches do not necessarily satisfy both conditions at the same time. To avoid this problem, we introduce a new approach to avoid the use of the clamping function in 4.2.1 and a new approach for the scaling of the monitor function in 4.2.2.

4.2.1 Avoiding Clamping by Using a New Activation Function

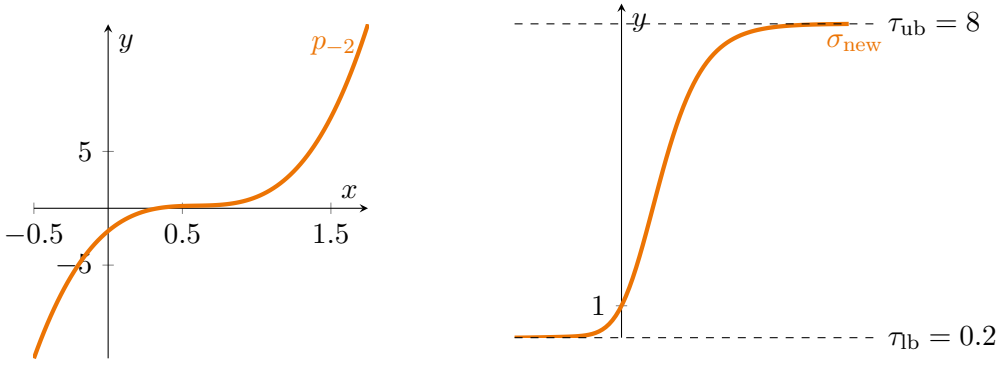
So far, the clamping function is used to satisfy the second condition of (3.60). As the simple clamping might change the behaviour of the monitor function, we develop a new approach to satisfy the second condition of (3.60). For this approach, we apply a new activation function $\sigma_{\text{new}} : \mathbb{R} \to \mathbb{R}$ on the last layer of the U-Net on the channel representing the monitor function to ensure μ is bounded by τ_{lb} and τ_{ub}, avoiding the use of the clamping function. To this end, an activation function based on the Sigmoid activation function that bounds the values of μ to $[\tau_{\text{lb}}, \tau_{\text{ub}}]$ is employed.

The Sigmoid function evaluates to values in $(0, 1)$, and the value 0 is mapped to 0.5. To satisfy the integration constraint, the monitor function needs to have a mean value of 1. Therefore, a constant of 0.5 is added to the Sigmoid function σ, such that $\sigma(0) = 1$. Due to the added constant, the Sigmoid function is bounded by $[0.5, 1.5]$ for all $x \in \mathbb{R}$. Now, the aim is to find a function mapping the minimum value of the Sigmoid function to τ_{lb} and the maximum value of the sigmoid function to τ_{ub}. As τ_{lb} and τ_{ub} are later fixed to $\tau_{\text{lb}} = 0.2$ and $\tau_{\text{ub}} = 8.0$, the activation function is specifically computed for these values.

For that reason, we compute the coefficients of a third degree polynomial $p : \mathbb{R} \to \mathbb{R}$, $p(x) = p_3 x^3 + p_2 x^2 + p_1 x + p_0$, satisfying

$$\begin{aligned} p(1.0) &= 1.0, \\ p(0.5) &= \tau_{\text{lb}} = 0.2, \\ p(1.5) &= \tau_{\text{ub}} = 8.0. \end{aligned} \tag{4.5}$$

The degree of the polynomial is chosen one degree higher than the number of equations such that the remaining degree of freedom can be adapted to ensure p is monotonically



(a) The polynomial p_{-2}.

(b) The new activation function σ_{new}.

Figure 4.1: The monotonically increasing polynomial p_{-2} (left) that is plugged into the shifted Sigmoid function to establish the new activation function σ_{new} (right). The new activation function does not exceed τ_{lb} and τ_{ub} and preserves the monotony of the Sigmoid function.

increasing in the interval $[0.5, 1.5]$. Otherwise, the monotony of the Sigmoid function might not be preserved.

This results in an underdetermined system of three linear equations

$$\begin{aligned} p(1.0) = \quad & p_3 + p_2 + p_1 + p_0 \quad &&= 1.0, \\ p(0.5) = \quad & \tfrac{1}{8}p_3 + \tfrac{1}{4}p_2 + \tfrac{1}{2}p_1 + p_0 \quad &&= 0.2, \\ p(1.5) = \quad & \tfrac{27}{8}p_3 + \tfrac{9}{4}p_2 + \tfrac{3}{2}p_1 + p_0 \quad &&= 8.0. \end{aligned} \tag{4.6}$$

The polynomials

$$p_r(x) = \left(\frac{112}{15} - \frac{4}{3}r\right)x^3 + \left(-10 + 4r\right)x^2 + \left(\frac{53}{15} - \frac{11}{3}r\right)x + r \tag{4.7}$$

for $r \in \mathbb{R}$ satisfy (4.6). For $r = -2$, the derivative

$$p_r'(x) = \left(\frac{112}{5} - 4r\right)x^2 + \left(-20 + 8r\right)x + \left(\frac{53}{15} - \frac{11}{3}r\right) \tag{4.8}$$

is positive for all $x \in [0.5, 1.5]$, resulting in a monotonically increasing p_r. Using the polynomial

$$p_{-2}(x) = \frac{152}{15}x^3 - 18x^2 + \frac{163}{15}x - 2 \tag{4.9}$$

leads to the new activation function

$$\begin{aligned} \sigma_{\text{new}}(x) &= p_{-2}(\sigma(x) + 0.5) \\ &= \frac{152}{15}(\sigma(x) + 0.5)^3 - 18(\sigma(x) + 0.5)^2 + \frac{163}{15}(\sigma(x) + 0.5) - 2, \end{aligned} \tag{4.10}$$

satisfying (4.5), since

$$\begin{aligned} \sigma_{\text{new}}(0) &= 1, \\ \lim_{x\to\infty} \sigma_{\text{new}}(x) &= 8, \\ \lim_{x\to-\infty} \sigma_{\text{new}}(x) &= 0.2. \end{aligned} \tag{4.11}$$

The polynomial p_{-2} is displayed in figure 4.1a, the new activation function σ_{new} is displayed in figure 4.1b.

4.2.2 New Approach to Scaling

To avoid the afore mentioned issues arising from iteratively constraining μ, we develop a new approach to scale the monitor function, ensuring that the conditions (3.60) are satisfied without iterating. Due to the application of the new activation function, the monitor function derived from the U-Net satisfies the condition that it is bounded by $[\tau_{\text{lb}}, \tau_{\text{ub}}]$. Therefore, this approach focuses on scaling the monitor function to force the integration constraint on μ.

In order to satisfy

$$\int_\Omega \mu(\xi)\ \mathrm{d}\xi = |\Omega|, \tag{4.12}$$

the mean value of μ should be 1. This approach also matches the idea of volume preserving registration, as a mean of 1 of the Jacobian determinant forces the overall volume of the registered object to remain the same, as the monitor function controls the Jacobian determinant of the deformation field.

The approach distinguishes between the two possible cases after clamping μ:

$$\int_\Omega \mu(\xi)\ \mathrm{d}\xi \geq |\Omega| \tag{4.13}$$

and

$$\int_\Omega \mu(\xi)\ \mathrm{d}\xi < |\Omega|. \tag{4.14}$$

For the first case (4.13), the values of μ that are greater than 1 are scaled down, whereas for the second case, the values of μ less than 1 are scaled up. Accordingly, the domain Ω is divided into two disjoint subsets, defined as

$$\begin{aligned} \Omega^+ &:= \left\{ \xi \in \Omega \;\middle|\; \mu(\xi) \geq 1 \right\}, \\ \Omega^- &:= \left\{ \xi \in \Omega \;\middle|\; \mu(\xi) < 1 \right\}, \end{aligned} \tag{4.15}$$

with $\Omega = \Omega^+ \cup \Omega^-$ and $\Omega^+ \cap \Omega^- = \emptyset$. We choose the scaling factors such that the first condition of (3.60) is satisfied after scaling μ, while ensuring that the scaling does not interfere with the second condition. A visualisation of the approach for the first case is displayed in figure 4.2. The scaling factors are computed as follows:

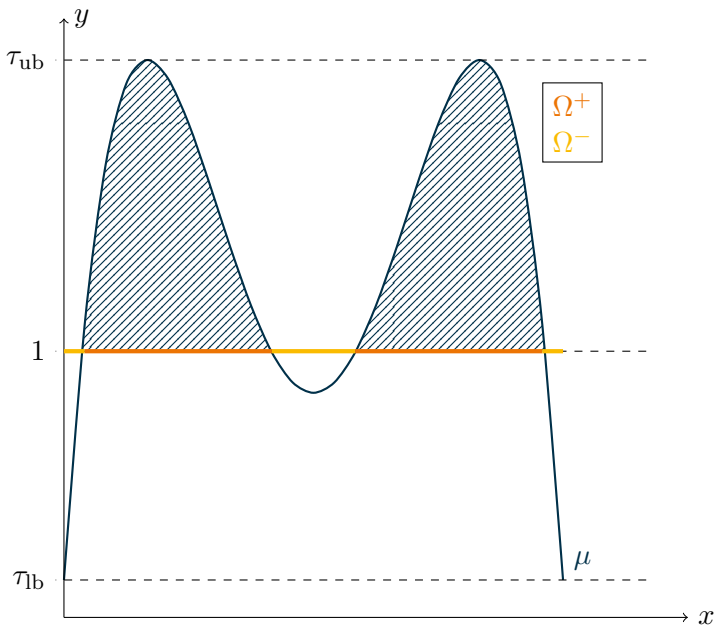


Figure 4.2: Visualisation of the new approach to scale the monitor function μ in the case (4.13). As the integral over μ is too big, the values of μ greater than 1 (hatched area) are scaled down such that the conditions (3.60) are satisfied. The subset Ω^+ is visualised in orange, Ω^- in yellow.

Proposition 4.1 (Scaling the Monitor Function, First Case)**.** *Assume that* (4.13) *holds. Define the scaled monitor function* μ_{new} *as*

$$\mu_{new}(\xi) := \begin{cases} \alpha\,(\mu(\xi) - 1) + 1 & \text{for } \xi \in \Omega^+ \\ \mu(\xi) & \text{for } \xi \in \Omega^- \end{cases} \tag{4.16}$$

with scaling factor

$$\alpha = \frac{|\Omega| - \int_{\Omega^-} \mu(\xi)\ \mathrm{d}\xi - \int_{\Omega^+} 1\ \mathrm{d}\xi}{\int_{\Omega^+} \mu(\xi) - 1\ \mathrm{d}\xi}. \tag{4.17}$$

Then, μ_{new} *satisfies the first condition of* (3.60)*, i.e.,*

$$\int_{\Omega} \mu_{new}(\xi)\ \mathrm{d}\xi = \int_{\Omega^+} \alpha\,(\mu(\xi) - 1) + 1\ \mathrm{d}\xi + \int_{\Omega^-} \mu(\xi)\ \mathrm{d}\xi = |\Omega|. \tag{4.18}$$

Proof. First, we introduce the following definitions to aid the readability:

$$\begin{aligned} \mu^+ &:= \int_{\Omega^+} \mu(\xi)\ \mathrm{d}\xi, \quad & \mu^- &:= \int_{\Omega^-} \mu(\xi)\ \mathrm{d}\xi, \\ |\Omega^+| &:= \int_{\Omega^+} 1\ \mathrm{d}\xi, & |\Omega^-| &:= \int_{\Omega^-} 1\ \mathrm{d}\xi. \end{aligned} \tag{4.19}$$

The factor α is simplified by plugging in (4.19), leading to

$$\alpha = \frac{|\Omega| - \int_{\Omega^-} \mu(\xi)\ \mathrm{d}\xi - \int_{\Omega^+} 1\ \mathrm{d}\xi}{\int_{\Omega^+} \mu(\xi) - 1\ \mathrm{d}\xi}$$

$$= \frac{|\Omega| - \mu^- - |\Omega^+|}{\mu^+ - |\Omega^+|}. \tag{4.20}$$

As the current case (4.13) implies that the mean of μ is greater than 1 for $\xi \in \Omega^+$, the denominator of (4.20) is strictly positive.

Now, the integral of the adapted monitor function μ_{new} is computed. First, we use the linearity of integration, leading to

$$\begin{aligned}\int_\Omega \mu_{\text{new}}(\xi)\ \mathrm{d}\xi &= \int_{\Omega^+} \alpha\left(\mu(\xi) - 1\right) + 1\ \mathrm{d}\xi + \int_{\Omega^-} \mu(\xi)\ \mathrm{d}\xi \\ &= \alpha\left(\int_{\Omega^+} \mu(\xi)\ \mathrm{d}\xi - \int_{\Omega^+} 1\ \mathrm{d}\xi\right) + \int_{\Omega^+} 1\ \mathrm{d}\xi + \int_{\Omega^-} \mu(\xi)\ \mathrm{d}\xi. \end{aligned} \tag{4.21}$$

The terms can be simplified using (4.19),

$$\int_\Omega \mu_{\text{new}}(\xi)\ \mathrm{d}\xi = \alpha\left(\mu^+ - |\Omega^+|\right) + |\Omega^+| + \mu^-. \tag{4.22}$$

By plugging (4.20) into (4.22), the integral evaluates to

$$\begin{aligned}\int_\Omega \mu_{\text{new}}(\xi)\ \mathrm{d}\xi &= \frac{|\Omega| - \mu^- - |\Omega^+|}{\mu^+ - |\Omega^+|}\left(\mu^+ - |\Omega^+|\right) + |\Omega^+| + \mu^- \\ &= |\Omega| - \mu^- - |\Omega^+| + |\Omega^+| + \mu^- \\ &= |\Omega|, \end{aligned} \tag{4.23}$$

which completes the proof. □

Analogously, the monitor function is scaled in the second case:

Proposition 4.2 (Scaling the Monitor Function, Second Case)**.** *Assume that*(4.14) *holds. The scaled monitor function* μ_{new} *is defined as*

$$\mu_{new}(\xi) := \begin{cases} \beta\left(\mu(\xi) - 1\right) + 1 & \text{for } \xi \in \Omega^- \\ \mu(\xi) & \text{for } \xi \in \Omega^+ \end{cases} \tag{4.24}$$

with scaling factor

$$\beta = \frac{|\Omega| - \int_{\Omega^+} \mu(\xi)\ \mathrm{d}\xi - \int_{\Omega^-} 1\ \mathrm{d}\xi}{\int_{\Omega^-} \mu(\xi) - 1\ \mathrm{d}\xi}. \tag{4.25}$$

Then, μ_{new} *satisfies the first condition of* (3.60)*, i.e.,*

$$\int_\Omega \mu_{new}(\xi)\ \mathrm{d}\xi = \int_{\Omega^-} \beta\left(\mu(\xi) - 1\right) + 1\ \mathrm{d}\xi + \int_{\Omega^+} \mu(\xi)\ \mathrm{d}\xi = |\Omega|. \tag{4.26}$$

The proof is completely symmetric to the proof for the first case and therefore omitted.

5

Numerical Results

In this chapter, the we present the numerical results of our implementation. First, we introduce the Sunnybrook Cardiac Data used for analysing the implemented framework in section 5.1. The section also covers the challenges that arise when using the data.

Following, we describe the evaluation metrics that are used for the quantitative analysis of the registration results in section 5.2.

Next, we analyse the accuracy and basic functionality of the FFT-based Poisson solver through three test cases along three different resolutions in section 5.3.

Then, the results of our implementation of the moving approach for image registration using different approaches to constrain the monitor function are presented and discussed in section 5.4: Firstly, the registration setup is briefly described in 5.4.1. Secondly, the registration results of the U-Net implementation are provided and analysed in 5.4.2. Thirdly, the registration results and analysis of the results for the moving mesh approach using INR are provided in 5.4.3. Finally, a comparison between the moving mesh-based registration with a U-Net and INR is provided in 5.4.4.

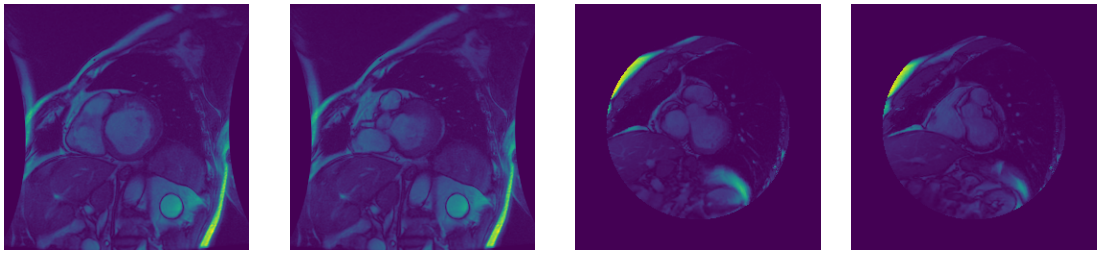
5.1 Data Set

In this thesis, the moving mesh approach is tested using the Sunnybrook Cardiac Data. The same data set has been used by Sheikhjafari et al. and is publicly available [75].

The Sunnybrook Cardiac Data (SCD) consists of cine MR images of 45 patients, categorised into the following four pathological groups: heart failure with infarction, heart failure without infarction, left ventricular hypertrophy, and healthy controls [66]. Of the 45 patients, nine belong to the group of healthy patients, the other pathological groups consist of twelve patients each.

For each patient, both short-axis and long-axis MR images are provided that were acquired by the same 1.5 T MRI scanner. For the experiments in this thesis, we use the SAX images as the focus lies on the registration of the LV. In addition to the images, the data set includes hand-drawn contours of the LV for the ED and ES images. The dataset of each patient consists of 20 images covering an entire cardiac cycle, beginning with the ED phase [73].

The images are in the DICOM (Digital Imaging and Communications in Medicine) format. They have a resolution of 256×256 pixels, with a slice thickness of 8 mm and gaps of 8 mm between slices as well as a pixel spacing of 1.25 mm. The images exhibit a high variability, as they show different slices, with a particularly high variation in the background. Four exemplary images of the SCD are displayed in figure 5.1, where two different background types are observable. The entire data set consists of 395 pairs of



(a) ED patient 1. (b) ES patient 1. (c) ED patient 2. (d) ES patient 2.

Figure 5.1: Exemplary images of the SCD at ED and ES phase of two patients [66]. The high variation of the background of the images as well as the high variation of the LV volume between the ED and the ES phase are observable.

ED and ES images.

Preparation of the SCD

As the VoxelMorph framework does not support DICOM images for training, the image are converted to `npz` files before usage. Following the description of Sheikhjafari et al., the data is divided into a training, a validation, and a test set, each consisting of the images of 15 patients. In order to avoid bias, an equal distribution of all pathological groups across the training, validation, and test data set is ensured. Therefore, each of the sets contains data of four patients from the pathological groups – heart failure with infarct, heart failure without infarct, and hypertrophy – and three healthy patients. This distribution is achieved by assigning every third patient into the training set, the following patient into the validation set, and the remainder into the test set. This distribution results in a training set of 127 image pairs, a validation set of 139 pairs, and a test set of 129 pairs.

5.2 Evaluation Metrics

As the main focus of this thesis is the validation of the moving mesh approach proposed by Sheikhjafari et al., we choose the same evaluation metrics to quantitatively analyse the registration results [75]. The four metrics are now described.

5.2.1 Dice Metric

The *Dice metric* computes the overlap or the similarity of two regions [22]. The Dice metric DM of two regions A and B can be computed by

$$\mathrm{DM}(A, B) = \frac{2|A \cap B|}{|A| + |B|}. \tag{5.1}$$

The Dice metric yields values in $[0, 1]$, where a value of 1 indicates a complete overlap of the two regions, whereas a values of 0 indicates no similarity of the two regions.

For the experiments in this thesis, the given regions are the segmentation masks of the LV of the fixed image and the deformed moving image and vice versa. As the data sets come with coordinates of the contours of the left ventricle, a segmentation mask is

generated of the given contours coordinates using the `cv2.fillPoly` function. The Dice metric is computed using the Dice Metric from the VoxelMorph framework. The segmentation masks are then registered to compute the DM of the registered segmentation of the moving image to the segmentation of the fixed image.

Although the images of the SCD are cropped to a resolution 128×128, the Dice metrics remain comparable to those obtained by Sheikhjafari et al., as the segmentation region is entirely contained in this region. The larger images consist of more background that remains mainly undeformed by the registration and therefore does not effect the Dice metric.

5.2.2 Hausdorff Distance

The *Hausdorff distance* measures the maximum distance between the contours of two objects [38]. In practise, we obtain the directed Hausdorff distance $\widetilde{\mathrm{HD}}$ of two objects A and B by computing the maximum distance of a list of their corresponding contour points C_A and C_B, i.e.

$$\widetilde{\mathrm{HD}}(A, B) = \max_{p \in C_A} \Big\{ \min_{q \in C_B} d(p, q) \Big\}, \tag{5.2}$$

where $d(\cdot)$ denotes the Euclidean distance. The undirected Hausdorff distance is then computed by

$$\mathrm{HD}(A, B) = \max\Big\{ \widetilde{\mathrm{HD}}(A, B),\ \widetilde{\mathrm{HD}}(B, A) \Big\}. \tag{5.3}$$

To evaluate the accuracy of cardiac registration, regions A and B are the segmentations of the LV. The directed Hausdorff distance is implemented using the `scipy` function `spatial.distance.directed_hausdorff` [82]. The undirected Hausdorff distance is then computed by taking the maximum of the two directed Hausdorff distances. As the `scipy.spatial.distance.directed_hausdorff` function needs the coordinates of the segmentation masks, the coordinates of deformed images are computed using `skimage.measure.find_contours`.

With the same explanation as provided for the Dice metric, the Hausdorff distance is not affected by the cropping of the images in our setup.

5.2.3 Reliability

The *Reliability* compares the Dice metrics of all image pairs of a data set and computes the (statistical) probability of obtaining a Dice metric greater than δ:

$$\begin{aligned} \mathrm{R}(\delta) &= P_r(\mathrm{DM} > \delta) \\ &= \frac{\#\text{ images segmented with a DM higher than } \delta}{\#\text{ images}}, \end{aligned} \tag{5.4}$$

with $\delta \in [0, 1]$, as the Dice metric is restricted by $[0, 1]$ [75]. The Reliability shows how reliable an algorithm performs for a given accuracy δ. Following Sheikhjafari et al., we set δ to 0.75.

5.2.4 Jacobian Determinant of the Deformation Field

As described in 2.3.6, the Jacobian determinant can be used to examine whether a deformation field is diffeomorphic and therefore, the appearance of mesh folding can be examined. As the moving mesh approach should generate a diffeomorphic deformation field, we validate this by computing its Jacobian determinate, using `torch.gradient` and `torch.det` for the computation.

5.3 Examination of the FFT Poisson Solver

The FFT-based Poisson solver plays a critical role in the moving mesh approach to adjust the vector field V according to the description in section 3.2.1. To ensure the accuracy and efficiency of the implemented solver, we evaluate the solver independently from the registration, using three distinct test functions across three different grid resolutions.

FFT Solver Test Setup

For our tests, the computational domain is modelled as a discrete equidistant grid of size $d \times d$, where d represents the grid resolution. The tests are performed with resolutions $d = 64$, $d = 128$, and $d = 256$. These varying grid sizes allow the analysis of the solver's performance and robustness, and provide insight into its computational efficiency and accuracy with increasing resolution. The solver is tested using Dirichlet boundary conditions that are implicitly applied through the DST, as described in 2.2.3.

We evaluate the FFT Poisson solver by solving the Poisson equations $\Delta u = F_i$ for three distinct right-hand sides F_i, $i = 1, \dots, 3$, where each test is designed to examine different aspects of the FFT solver's performance:

1. The first test is the simple case of a constant field

$$F_1(x, y) \equiv 1. \tag{5.5}$$

   This scenario serves as a baseline test to verify the basic functionality of the FFT solver. It is particularly useful for identifying larger errors that may arise from discretisation.

2. The second test case is the sinusoidal field

$$F_2(x, y) = -2 \left(\frac{\pi}{d}\right)^2 \sin\left(\frac{\pi x}{d}\right) \sin\left(\frac{\pi y}{d}\right), \tag{5.6}$$

   which is more complex than the first case. It is designed to test the solver's accuracy, particularly near the boundaries, and helps to evaluate how well the solver maintains accuracy as the grid resolution increases.

3. The third test is to analyse the performance of the FFT solver for a discontinuous field: we choose the step function

$$F_3(x, y) = \begin{cases} 1, & x, y > \frac{d}{2}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

Table 5.1: Solution accuracy of the FFT Poisson solver. The accuracy is the absolute value of the Laplacian of the computed solution and the initial right-hand side of the Poisson equation. The Poisson equations were solved 10 000 times for each test case, the reported results are the mean accuracy across these experiments.

	$d = 64$	$d = 128$	$d = 256$
$\lvert F_1 - \Delta\tilde{F}_1\rvert$	5.55×10^{-5}	2.27×10^{-4}	1.21×10^{-3}
$\lvert F_2 - \Delta\tilde{F}_2\rvert$	1.80×10^{-7}	1.70×10^{-7}	2.20×10^{-7}
$\lvert F_3 - \Delta\tilde{F}_3\rvert$	1.57×10^{-5}	6.42×10^{-5}	3.22×10^{-4}

Table 5.2: Run times t_{F_i} of the FFT Poisson solver in milliseconds for the three test function F_i, $i = 1, \ldots, 3$, across the three grid resolutions $d = 56, d = 128$, and $d = 256$. The run times are the average run time over 10 000 solved Poisson equations.

	$d = 64$	$d = 128$	$d = 256$
t_{F_1}	1.10 ms	1.66 ms	3.63 ms
t_{F_2}	1.11 ms	1.67 ms	3.63 ms
t_{F_3}	1.15 ms	1.69 ms	3.74 ms

The discontinuity of the step function can be challenging for the solver, as such abrupt changes can lead to unwanted numerical artefacts. As discontinuities frequently occur in images such as the later used Sunnybrook Cardiac Data, it is essential to test the solver's robustness for these scenarios.

To analyse the run time of the FFT solver, we execute each experiment 10 000 times, testing all combinations of test functions and resolutions.

FFT Solver Results

Table 5.1 presents the mean accuracies $|F_i - \Delta\tilde{F}_i|$ for the FFT Poisson solver for the different test cases $i = 1, \ldots, 3$. The mean accuracies for F_1 range between 5.55×10^{-5} for the lowest resolution and 1.21×10^{-3} for the highest resolution. For the second test case, the mean accuracies show similar results around 2×10^{-7} for all resolutions. The third test case shows mean accuracies between 1.57×10^{-5} and 3.22×10^{-4}.

In table 5.2, the average run times required to solve a single Poisson equation for the three different test functions across the three grid resolutions are presented. The run times range from 1.10 ms to 1.15 ms for the resolution $d = 64$, from 1.66 ms to 1.69 ms for $d = 128$, and from 3.63 ms to 3.74 ms for the highest resolution $d = 256$.

The outcomes of the Poisson equation for the constant field F_1 is visualised in figure 5.2, the results of the sinusoidal field F_2 are displayed in figure 5.3, and the results for the step function F_3 are shown in figure 5.4.

Discussion of the FFT Solver Results

For constant field F_1, the expected solution is a quadratic function with a positive curvature. The computed solutions $\tilde{F}_1$ align with these expectations, as shown in the second column of figure 5.2. The homogeneous Dirichlet boundary conditions ensure that the solutions is zero at the boundaries, which is accurately reflected in the results. Additionally, the Laplacian of the solution should yield a constant field, which is observable

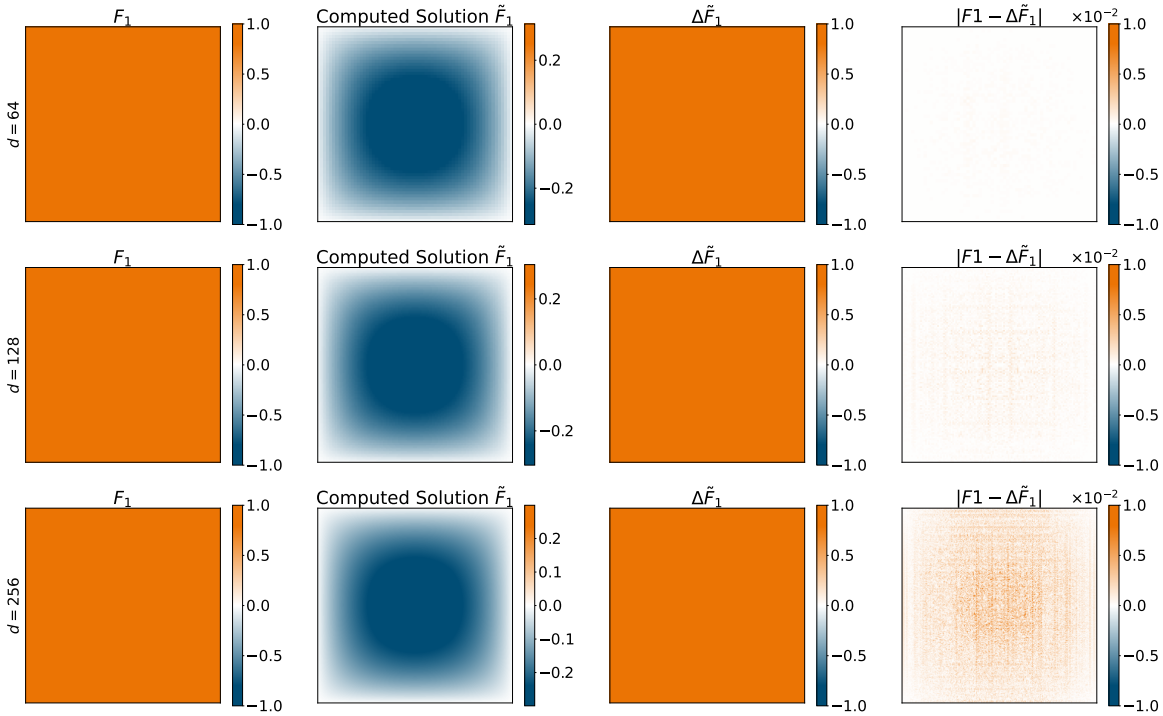


Figure 5.2: Solutions of the Poisson equation using the FFT solver for the constant right-hand side F_1. Each row corresponds to a different grid resolution with resolutions of $d = 64$, $d = 128$, and $d = 256$. In the first column, the constant field F_1 is displayed, the second column shows the computed solution $\tilde{F}_1$. The third column presents the Laplacian of the computed solution, the final column shows the absolute difference of F_1 and the Laplacian of the computed solution $\tilde{F}_1$. The solution shows the expected negative quadratic behaviour and tends to zero at the boundary. The Laplacians of the solution show the expected behaviour, being a constant field as F_1. The absolute error $|F_1 - \Delta\tilde{F}_1|$ increases with increasing resolution, showing slightly larger errors around the centre.

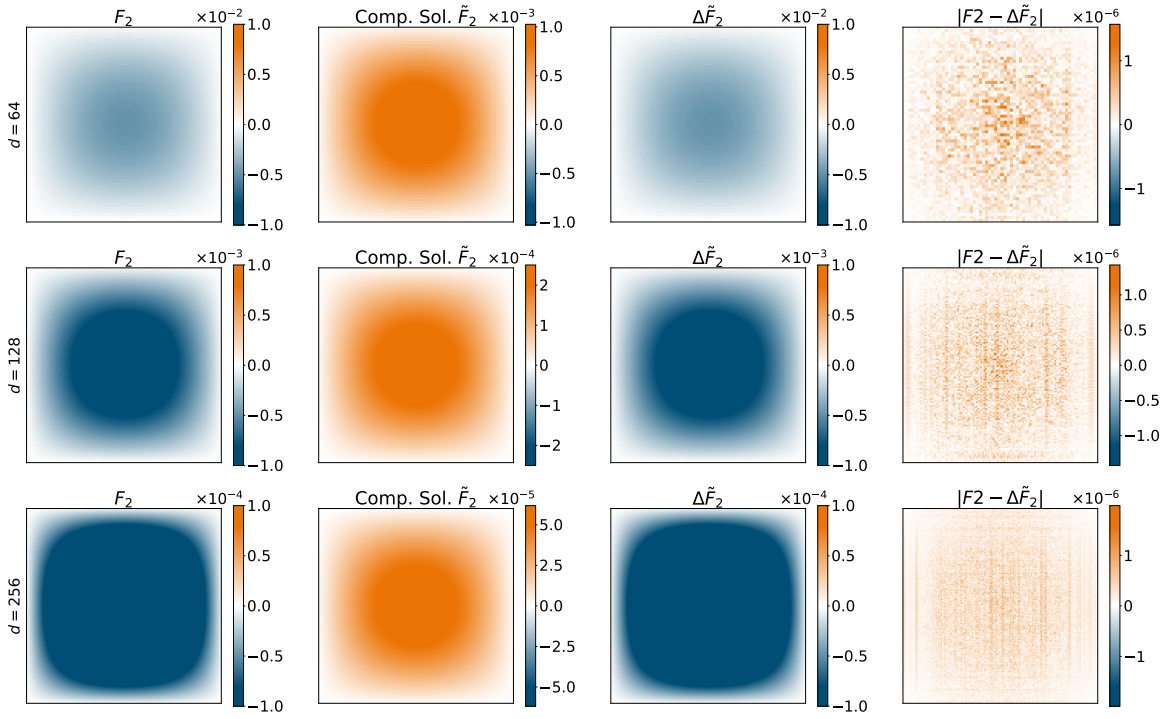


Figure 5.3: Visualised results of the FFT solver for the right-hand side F_2, compare figure 5.2. The computed solution shows the expected positive sine function and tends to zero at the boundary. The absolute error $|F_2 - \Delta\tilde{F}_2|$ look about evenly distributed over the entire domain, showing its maximum around the centre.

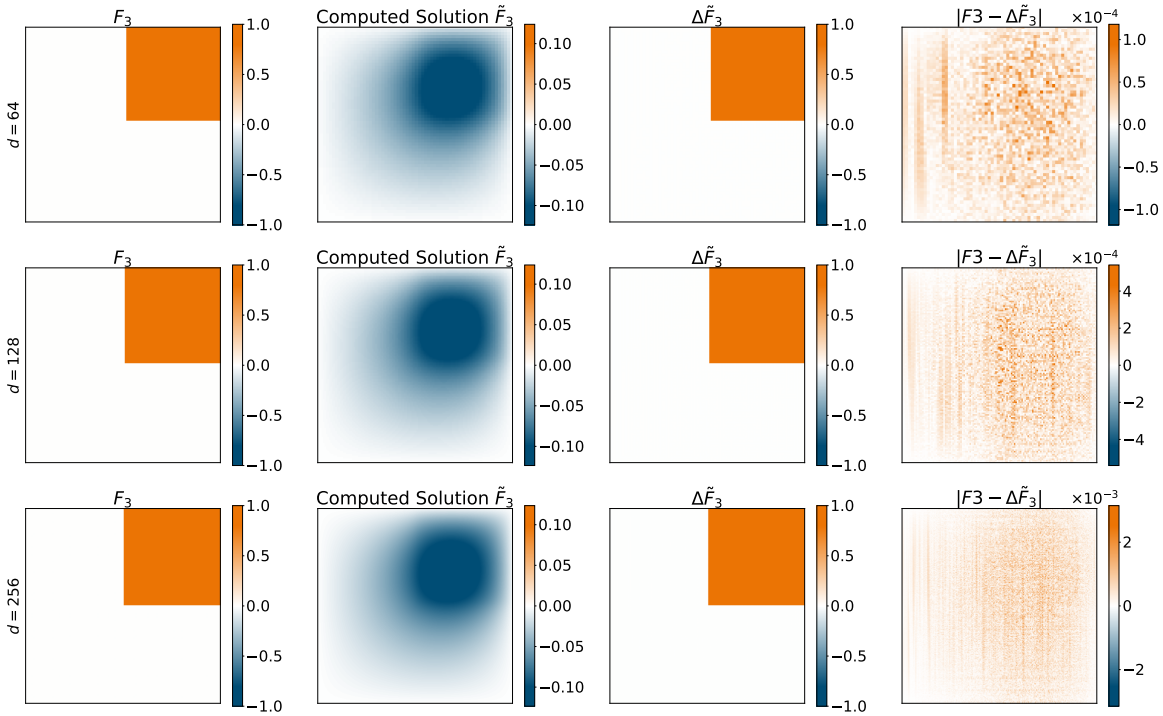


Figure 5.4: Visualised results of the FFT solver for right-hand side F_3, compare figure 5.2. In the first column, the step function is visualised. The solutions show a steep slope close to the step. The Laplacians $\Delta\tilde{F}_3$ show the expected step. The absolute error $|F_3 - \Delta\tilde{F}_3|$ increases with increasing resolution and is slightly higher in the top right corner.

in the third column in figure 5.2. The absolute errors of F_1 and $\Delta\tilde{F}_1$ remain approximately constant on the entire domain, showing slightly higher values in the centre of the domain, which is visible in the last column of figure 5.2. The absolute errors are approximately doubled for doubled resolution.

The function F_2 is one negative sinusoidal wave, therefore, the expected solution is a symmetric, positive sinusoidal wave with a peak in the centre of the domain, and fading to zero close to the boundary. This behaviour is observable in the second column of figure 5.3. Additionally, the solution satisfies the Dirichlet boundary conditions as it reaches zero at the boundary. The total errors for this test function are very similar for all three resolutions, as observable by table 5.1. The highest accuracy errors occur at the minimum of the sine waves, which is observable in the last column of figure 5.3, but the total errors remain small, being around 2×10^{-7}. Due to the smooth and periodic characteristics of the sine function, the solution should not show larger artefacts, which is satisfied.

Although the step function F_3 is discontinuous, the solution of the corresponding Poisson equation is expected to be smooth on the domain. Due to the step, the solution is expected to show a steep slope in the area of the step. This behaviour is clearly observable for the computed solutions that are displayed in the second column of figure 5.4. Again, the homogeneous Dirichlet boundary conditions are satisfied. The absolute errors of F_3 and $\Delta\tilde{F}_3$ approximately double with increasing resolution.

Overall, the solver shows the highest accuracy for the sinusoidal wave function F_2, with mean errors in the order of 2×10^{-7}. The constant field F_1 produces slightly larger errors compared to the step function F_3, however, they show the same behaviour of increasing errors for increasing resolution. The difference in solution accuracy can be attributed to the nature of the DST underlying the FFT-based solver. As a constant field does not have any oscillating characteristics, it is challenging for the FFT solver to compute the optimal coefficients of the sine frequencies composing the solution. In contrast, the sinusoidal wave can be naturally described by sine waves, leading to a more accurate computation of the coefficients. Despite the sharp discontinuity in F_3, the FFT solver manages to produces relatively accurate results for F_3. The step can be decomposed using a series of sine waves with varying frequencies, explaining the slightly better performance of the FFT solver on the step function than on the constant field. For F_1 and F_3, it is observable that the absolute errors of the Laplacian of the computed solution are roughly doubled when the input dimension is doubled, whereas the errors of F_2 remain relatively stable. This trend indicates the expected decrease of the accuracy with increasing resolution; the consistent accuracy observed in the sinusoidal field can be attributed to its alignment with the underlying DST properties.

The run times show a similar trend for all three test cases. For the same resolution, the solver takes about the same time, which is the expected outcome. The time increases with increasing resolution, however, the solver is efficiently fast for our use.

In summary, the FFT solver shows the expected behaviour for all of our test cases, demonstrating both robust performance and accuracy across different scenarios while maintaining computational efficiency for increasing resolution. Therefore, it is suitable for the application to the moving mesh image registration.

5.4 Examination of the Moving Mesh-Based Image Registration

5.4.1 Setup

To validate the moving mesh approach by Sheikhjafari et al., the following experiments are performed. We perform image registration to test the moving mesh approach using a U-Net and an INR with different methods to constrain the monitor function, as described in chapter 4. The following methods to constrain the clamped monitor functions are tested and evaluated:

1. scaling once,
2. repeated clamping and scaling,
3. the new scaling approach,
4. no further scaling.

The clamping is performed using our new activation function; the parameters to clamp the monitor function are set to $\tau_{\text{lb}} = 0.2$ and $\tau_{\text{ub}} = 8.0$. The proposed methods are evaluated on the test set of Sunnybrook Cardiac data, focusing on the registration of images from the ES and the ED phase.

Although Sheikhjafari et al. performed the registration on the entire images, in this thesis, the images are cropped to a resolution of 128×128 pixels to lay the focus of the registration on the LV region, as the training set is relatively small. Additionally, this approach enhances the speed of the training process.

As Sheikhjafari et al. did not provide any information on the number of steps they used to solve the ODE, we test two step sizes: 10 time steps and 20 time steps as proposed by Chen et al. [14]. The tests are performed on an NVIDIA A100 GPU with 80GB memory using CUDA 12.0.

5.4.2 Registration Using a U-Net

U-Net Setup

To validate the moving mesh approach, the U-Nets are trained using the following settings. Deviating from the proposed learning rate of 5×10^{-4}, we choose a learning rate of 1×10^{-4}, as pre-tests have shown better results for this learning rate. The U-Net is trained using the training set with 500 epochs with 100 steps per epoch. Every ten epochs, the model is evaluated on the validation set. The model performing best on the validation set is selected to perform the registration on the test set.

U-Net Registration Results

The registration using the moving mesh approach with a U-Net shows the following results. In figure 5.5, the validation loss for of the U-Net training process for the scaling once method and 10 ODE steps is displayed. The displayed loss function is exemplary for all U-Net losses as they show similar behaviours. The models showing minimal loss – thus performing best – on the validation set is selected to perform the registration on the test set.

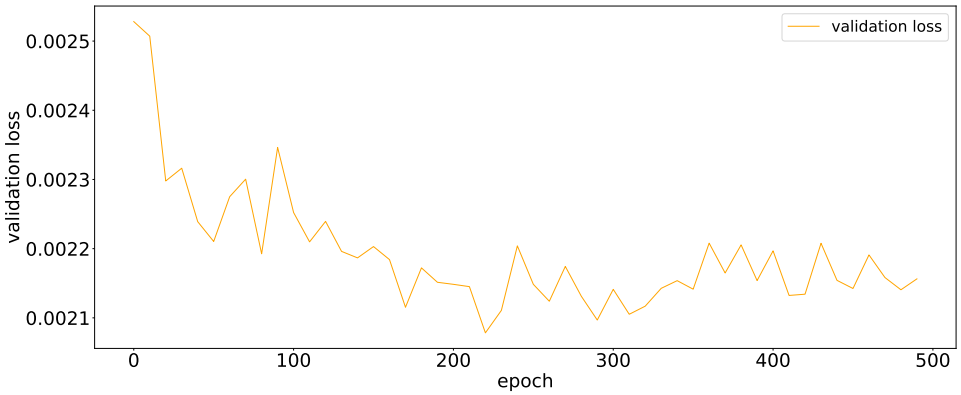


Figure 5.5: Validation loss of the U-Net training process for the scaling once method and 10 ODE steps. The horizontal axis represents the training epoch, the vertical axis represents the loss that the model achieves on the validation data. The loss is computed using the same loss function as used for training. This particular approach shows the lowest validation loss in epoch 220, therefore, this model is chosen to evaluate the registration. The validation losses for other constraining methods show similar trends.

Figure 5.6 shows the registration outcomes for the different constraining methods on the same image pair, while figure 5.7 shows exemplary registration results for the new constraining method on four image pairs. Additionally, table 5.3 presents the numerical registration results. The average MSE for the undeformed test data is 0.0054 ± 0.007. All methods demonstrate lower MSE values compared to the input data, with the scaling once approach yielding an average MSE of 0.0038, and the new scaling method achieving an average MSE of 0.0041. No relevant difference in the average MSE between the forward deformations and the backward deformations is observable.

For the input images, the DM is 0.72 ± 0.18. The scaling once and the new scaling method both achieve an average DM of 0.75 for the forward deformations and the backward deformations. The iterative and the no scaling approach achieve slightly lower average DM values of 0.73 and 0.74 for the forward and the backward deformation fields.

The average HD for the input images is 8.79 ± 4.03. Most approaches show an increase in the HD, with the backward deformation field of the iterative constraining method reaching 9.92. Notably, only the scaling once method shows a decrease in the HD, achieving 8.61 in the forward deformation and 8.34 in the backward deformation.

The reliability across all approaches varies between 0.50 and 0.59, with stronger differences between the forward and the backward deformation for the scaling once and the new scaling method. The input images have a reliability of 0.5. The percentage of negative Jacobian determinants varies between 0.00% and 0.19%, with higher values occurring in the forward deformations.

Figure 5.8 shows the number of iterations required by the iterative method to meet the stopping criteria. The average number of iterations performed for the U-Net model with 10 ODE steps is 1.43. Notably, the maximum number of iterations observed is 10.

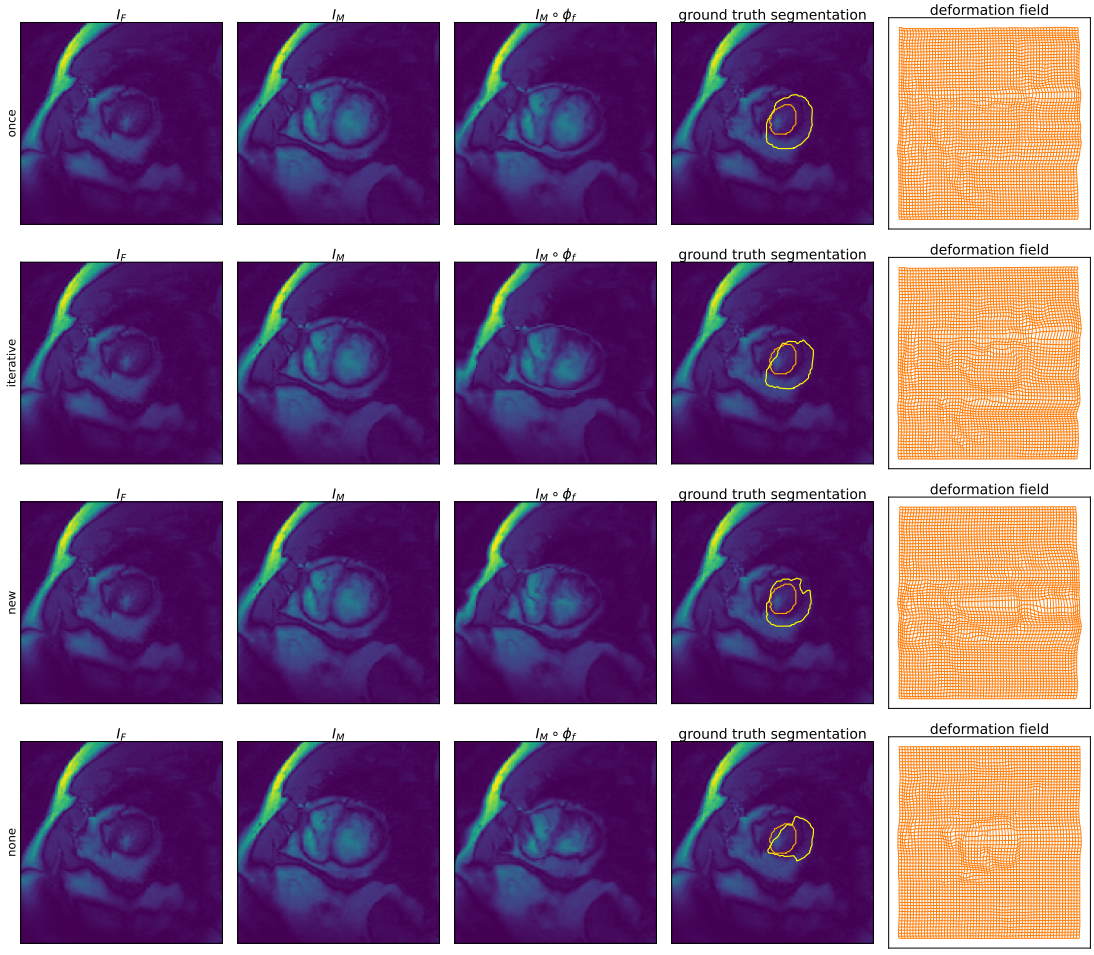


Figure 5.6: Registration results for the Sunnybrook Cardiac Data with a U-Net using the four different methods for constraining the monitor function and 10 ODE steps. Each row represents one constraining method. Top to bottom: scaling once, constraining iteratively, new scaling method, no constraining. The first column shows the fixed image, the second column the moving image. In the third column, the deformed moving image after registration is displayed. The fourth column shows the segmentation masks of the deformed moving image (yellow) and the segmentation of the fixed image (orange), which should ideally agree. The deformation fields are displayed in the final column. The four different methods lead to differing registration results. While the scaling once and the iterative method show similar results, the other two methods show varying segmentations. The no constraining method shows as expected less smooth results.

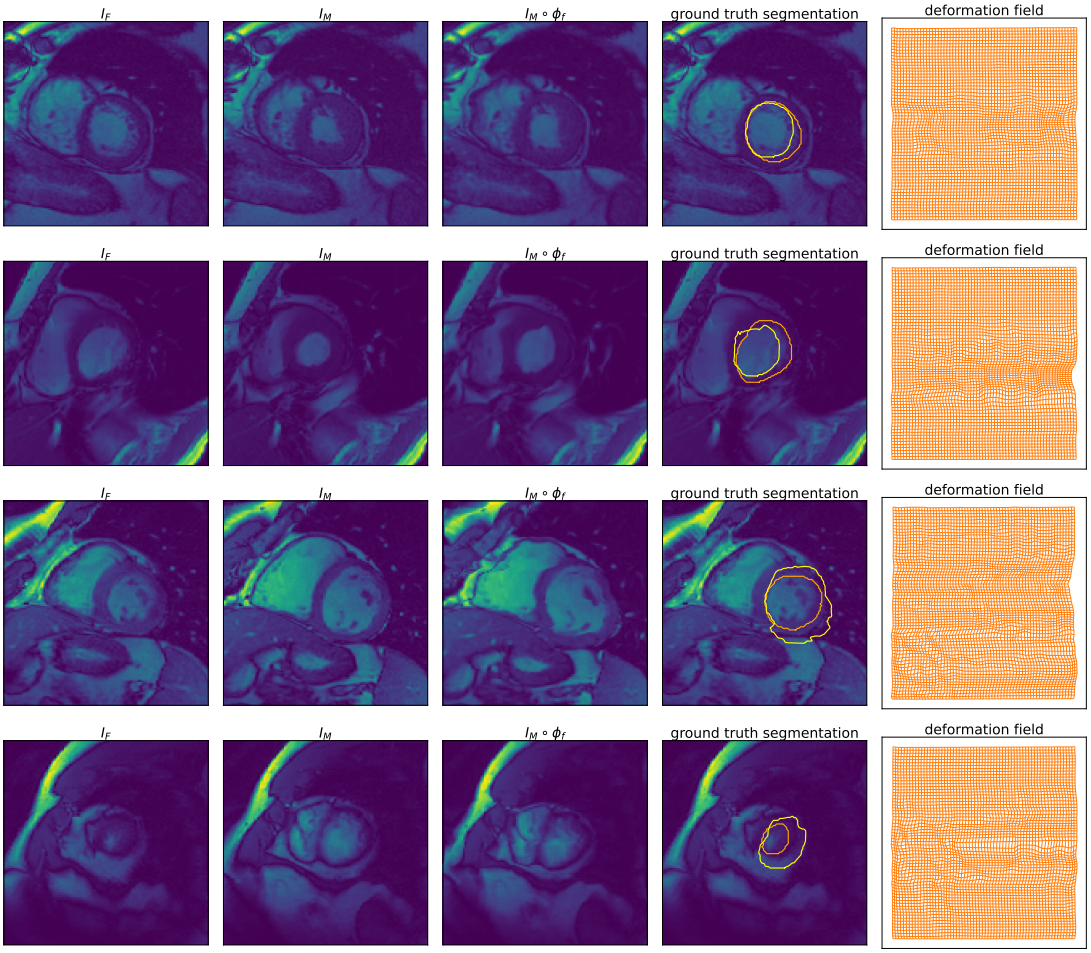


Figure 5.7: Four registration results for the Sunnybrook Cardiac Data with a U-Net registration using the new scaling method to constrain the monitor function and 10 ODE steps. The figure uses the same layout as figure 5.6. The first two rows show two better registrations, with improved DM, while the last two rows, with decreased DM, showing unwanted distortions of the left ventricle. The deformation fields of the two last rows contain negative Jacobian determinants, indicating grid folding.

Table 5.3: Quantitative evaluation of the registration results for the moving mesh-based registration on the Sunnybrook Cardiac Data using a U-Net with 10 time steps in the ODE. The models performing best on the validation set are evaluated on the test set, the evaluation epoch of the chosen model as well as the constraining approach for the monitor function are given in the first column.
The following metrics are used: the mean squared error (MSE), the Dice Metric (DM), the Hausdorff distance (HD), with mean and standard deviation, the reliability R(0.75) and the percentage of negative Jacobian determinants $\%|J| < 0$. A higher DM and a lower MSE and HD indicate more accurate registration results. The higher the reliability R(0.75), the more images have a DM of 0.75 or higher. A lower percentage of negative Jacobian determinant indicates less mesh folding.
The models are evaluated for the forward deformation field registering the moving image and the backward deformation field registering the fixed image. The best performance for the forward and the backward deformation are marked in bold for each metric.

constrain method model	**evaluation metric**	**forward**	**backward**
scaling once model 220	MSE	$\mathbf{0.0038 \pm 0.004}$	$\mathbf{0.0039 \pm 0.004}$
	DM	$\mathbf{0.75 \pm 0.17}$	$\mathbf{0.75 \pm 0.18}$
	HD	$\mathbf{8.61 \pm 3.85}$	$\mathbf{8.34 \pm 3.58}$
	R	**0.59**	**0.57**
	$\%\|J\| < 0$	**0.02**	**0.00**
iterative constraining model 480	MSE	0.0041 ± 0.004	0.0043 ± 0.004
	DM	0.73 ± 0.18	0.74 ± 0.18
	HD	9.71 ± 4.35	9.92 ± 4.98
	R	0.50	0.51
	$\%\|J\| < 0$	0.19	0.03
new scaling model 290	MSE	0.0039 ± 0.004	$\mathbf{0.0039 \pm 0.004}$
	DM	$\mathbf{0.75 \pm 0.18}$	$\mathbf{0.75 \pm 0.18}$
	HD	8.94 ± 4.16	8.80 ± 4.04
	R	0.57	**0.57**
	$\%\|J\| < 0$	0.04	**0.0**
no scaling model 90	MSE	0.0041 ± 0.004	0.0040 ± 0.005
	DM	0.74 ± 0.18	0.74 ± 0.18
	HD	8.95 ± 4.58	8.71 ± 4.36
	R	0.54	0.56
	$\%\|J\| < 0$	**0.02**	**0.0**

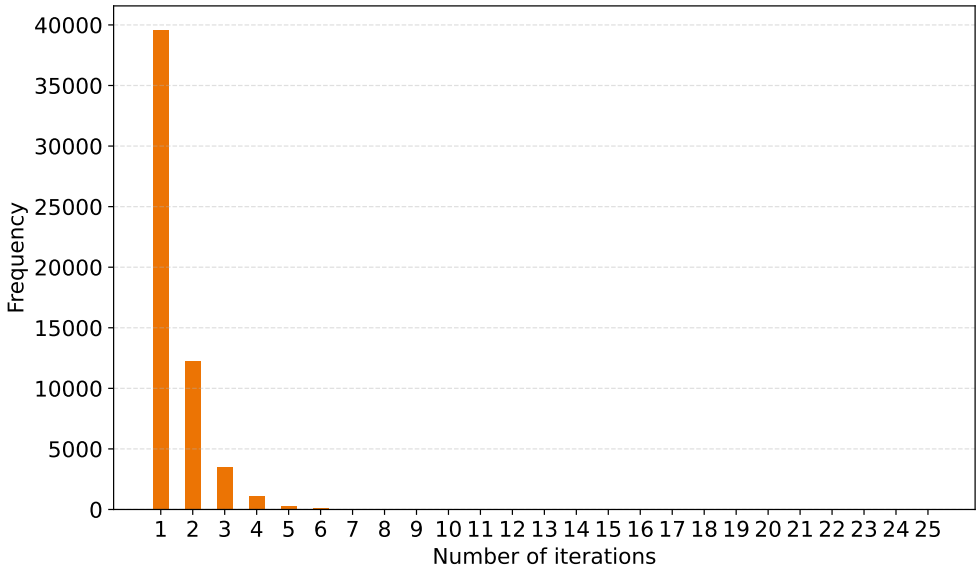


Figure 5.8: This bar plot illustrates the distribution of iterations for iteratively constraining the monitor function using the U-Net model with 10 ODE steps. The horizontal represents the number of iterations that are performed, whereas the vertical shows the frequencies of these iterations. The algorithm stops if either the constraints (3.59) are satisfied or the maximum of 25 iterations is performed. A cluster around one and three iterations is observable. The average number of iterations is 1.43.

Discussion of U-Net Registration Results

Among the tested methods, the scaling once method yields the best results across all evaluation metrics, making it the most effective approach for U-Net registration. However, the new scaling method produces comparable results, particularly for the backward deformation, although it performs slightly worse for the forward deformation. The iterative constraining method stands out with an overall poorer performance, especially in reliability, as it has not improved in comparison to the input images. The no scaling method outperforms the iterative methods, but yields poorer results than the scaling once and the new scaling method.

The different scaling strategies lead to differing registration outcomes, as observable in figure 5.6. This figure illustrates an example of a particularly challenging registration, as the LV and other surrounding structures show strong variation. The variation in the deformation fields highlights that, depending on the constraining method, different deformations are learned. While the scaling once method does not lead to much deformation for this example, the new constraining method and the no scaling method lead to less smooth deformations, as they would be expected for a plausible registration result.

Although the iterative approach and the scaling once method are expected to show similar results – especially with the knowledge that the average number of iterations is 1.43, with a maximum of 10 performed iterations – the iterative approach demonstrates poorer performance. This may be attributed to the repeated clamping, which potentially leads to a flattened monitor function and a slower learning process, thus yielding inferior registration results.

The distribution shown in figure 5.8 can be seen as an indicator of how often the scaling once method would violate the first condition of (3.60). For this example, both conditions are satisfied after one iteration in around 70% of the cases.

Our new constraining method does not yield improved results, although the constraints (3.60) for the monitor function are satisfied after its application. As observable in figure 5.7, the approach struggles with images showing a strong variation of the LV. This is particularly observable in the third and the fourth row of the figure, where the segmentations of the LV of the registered image (yellow) show uneven and irregular results.

We are unable to validate the results reported by Sheikhjafari et al. for the moving mesh-based image registration. The authors reported results of an average DM of 0.88, a reliability of 0.90, and an average HD of 5.25. Several factors might explain our inferior results:

- The authors do not specify how they constrain the monitor function, which plays a crucial role in the moving mesh approach, making it challenging to achieve similar results.

- Details regarding the data distribution are lacking. Their test image pairs have an average DM of 0.62, whereas our test images have an average DM of 0.72, leading to results that are not directly comparable. This also holds for the HD, where their input images have a HD of 16.02 compared to our images with a HD of 8.79.

- Our training set is relatively small. Following their method, the SCD is divided into three sets, consisting of 15 patients each. However, it is unclear whether their

training included only the images of the ED and ES phase or if images of all cardiac phases were used.

- Key details about the U-Net architecture and training are missing. This includes the number of epochs and steps per epoch for the training process.

One of the key features of the moving mesh-based image registration is the guaranteed diffeomorphic – and therefore folding-free – deformation field. However, for the U-Net training, we cannot verify the folding-free registration, as nearly all methods yield deformation fields containing at least some points with negative Jacobian. Among our tested methods, the iterative method results in the highest percentage of points with negative Jacobian. This behaviour is to some degree expected for all but the new approach, as the constraints (3.60) are not necessarily satisfied after constraining. As the new scaling approach should always satisfy both constraints, it can be assumed that the issue arises due to small numerical errors. This assumption is supported by the fact that the percentages are relatively low.

Notably, for the backward deformation fields only the iterative method shows points with negative Jacobian. The reason for this behaviour remains unclear but could potentially be related to the ODE solver. Another possible explanation is that the forward deformation always registers the ED to the ES, while the backward deformation works in the opposite direction. This could introduce different challenges for the registration, explaining the generally slightly variation between the results for the forward and the backward deformations.

The presence of negative Jacobian determinants indicates that some deformation fields are not diffeomorphic. Moreover, the deformation fields displayed in figure 5.6 do clearly not satisfy the required self-to-self-mapping, regardless of the choice of constraining method. This issue could arise from implementation errors or numerical inaccuracies, rather than being an issue of the moving mesh approach itself.

5.4.3 Registration Using Implicit Neural Representation

INR Registration Setup

The INR approach trains an individual model for each image pair of the test set. Each model is trained with 500 optimisation steps and a learning rate of 5×10^{-4}, we choose the final model after 500 steps to evaluate the registration.

INR Registration Results

Table 5.4 presents the registration results using INR with the four different approaches to constrain the monitor function, with 10 time steps for the ODE. All four approaches yield similar results, with an average DM ranging between 0.77 and 0.78. The mean MSE values range from 0.0024 to 0.0027, with lower values observed for the forward deformation. The reliability ranges from 0.62 to 0.66, showing higher values in the backward deformation across all four methods.

All approaches exhibit a small percentage of negative Jacobian determinants for the forward deformation fields, with values of 0.1% and 0.02%. No points with negative Jacobians occur in backward deformation fields.

The scaling once method achieves the lowest average HD for the forward deformations, with a value of 9.08, while the new scaling method yields the lowest HD of 9.52 for

Table 5.4: Quantitative evaluation of the registration results for the moving mesh-based registration on the Sunnybrook Cardiac Data using an INR with 10 time steps in the ODE, compare table 5.3.

method	evaluation metric	forward	backward
scaling once	MSE	$\mathbf{0.0024 \pm 0.002}$	$\mathbf{0.0026 \pm 0.003}$
	DM	0.77 ± 0.17	0.77 ± 0.18
	HD	$\mathbf{9.08 \pm 4.72}$	9.67 ± 6.65
	R	0.60	0.64
	$\%\lvert J\rvert < 0$	**0.01**	**0.00**
iterative constraining	MSE	0.0026 ± 0.003	0.0027 ± 0.003
	DM	0.77 ± 0.18	0.77 ± 0.18
	HD	9.31 ± 5.11	9.86 ± 6.68
	R	0.60	0.64
	$\%\lvert J\rvert < 0$	0.02	**0.00**
new scaling	MSE	0.0025 ± 0.003	0.0027 ± 0.003
	DM	$\mathbf{0.78 \pm 0.17}$	$\mathbf{0.78 \pm 0.18}$
	HD	9.27 ± 5.21	$\mathbf{9.52 \pm 6.62}$
	R	0.62	0.64
	$\%\lvert J\rvert < 0$	**0.01**	**0.00**
no scaling	MSE	0.0025 ± 0.003	$\mathbf{0.0026 \pm 0.003}$
	DM	$\mathbf{0.78 \pm 0.18}$	$\mathbf{0.78 \pm 0.18}$
	HD	9.33 ± 5.06	9.61 ± 6.75
	R	**0.64**	**0.65**
	$\%\lvert J\rvert < 0$	0.02	**0.00**

the backward deformation fields. All four approaches yield higher HD values for the backward deformations, where the iterative approach reaches the highest HD of 9.86.

Figure 5.9 shows the numbers of iterations for the iterative approach. A cluster around one and three is observable. The average number of iterations is 1.42, the highest number of performed iterations is 11. Visual results for the four different constraining approaches are displayed in figure 5.10 and in figure 5.11, visual results of the new constraining method are shown in figure 5.12.

Discussion of INR Registration Results

Among the four tested approaches, the no scaling method shows the best performance in the DM and the reliability, while the new scaling method yields comparable results for the DM, but with a lower reliability. Overall, the four constraining methods yield similar results for all metrics, with the most notable variation observed in the HD. This indicates that the choice of constraining method for registration using INR has a rather small impact on the registration. This can be observed in figure 5.10, showing exemplary the comparable registration results of the different constraining methods.

The exemplary results in figure 5.12 show the differing performance of the new scaling method. While the first two rows show registrations with high overlap of the segmentations, the last two rows show implausible results with strong contortion of the segmentation, leading to high HD.

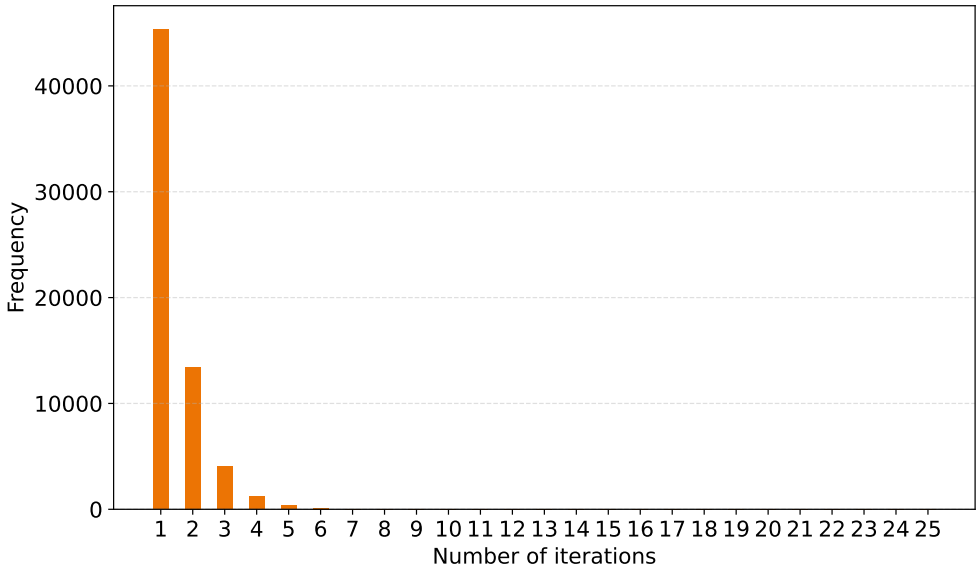


Figure 5.9: This bar plot illustrates the distribution of iterations for iteratively constraining the monitor function using an INR with 10 ODE steps, compare table 5.8. A cluster for one to three iterations is observable. The average number of iterations is 1.42.

It might seem counter-intuitive that INR shows good results in the DM, while performing worse in the HD, even though both metrics evaluate the segmentations. This can be explained by the HD's sensitivity to outliers and boundary accuracy. For example, the results displayed in figure 5.11 illustrate this: the segmentations show a relatively high overlap, resulting in a high DM, but significant outliers in the top right corner of all segmentations contribute to a high HD.

This can potentially be explained by the choice of the parameter w_0, which is crucial for the performance of the INR method, and could probably be improved through further parameter testing. Ideally, the parameter is adapted for each new image pair. In figure 5.11, the deformed images show contortions that might be caused by a poor choice of the parameter w_0. The outliers of the segmentation appearing in the same images might also be caused by a poor choice of w_0 for this specific image pair.

Again, we cannot confirm the diffeomorphic deformation fields of the moving mesh approach, as clearly visible in figure 5.11. The deformation fields hurt the diffeomorphic property especially in the bottom right corner. This fact is supported by the small percentage of negative Jacobian determinants, where the forward deformation field shows higher values for all four constraining methods than the backward deformations.

Regarding the iterative approach, most cases require in most cases only one and – less frequent – two iterations. This might explain the similar results of the scaling once and the iterative approach, as the approaches are equal if only one iteration is performed. Nevertheless, the scaling once method outperforms the iterative method in the HD.

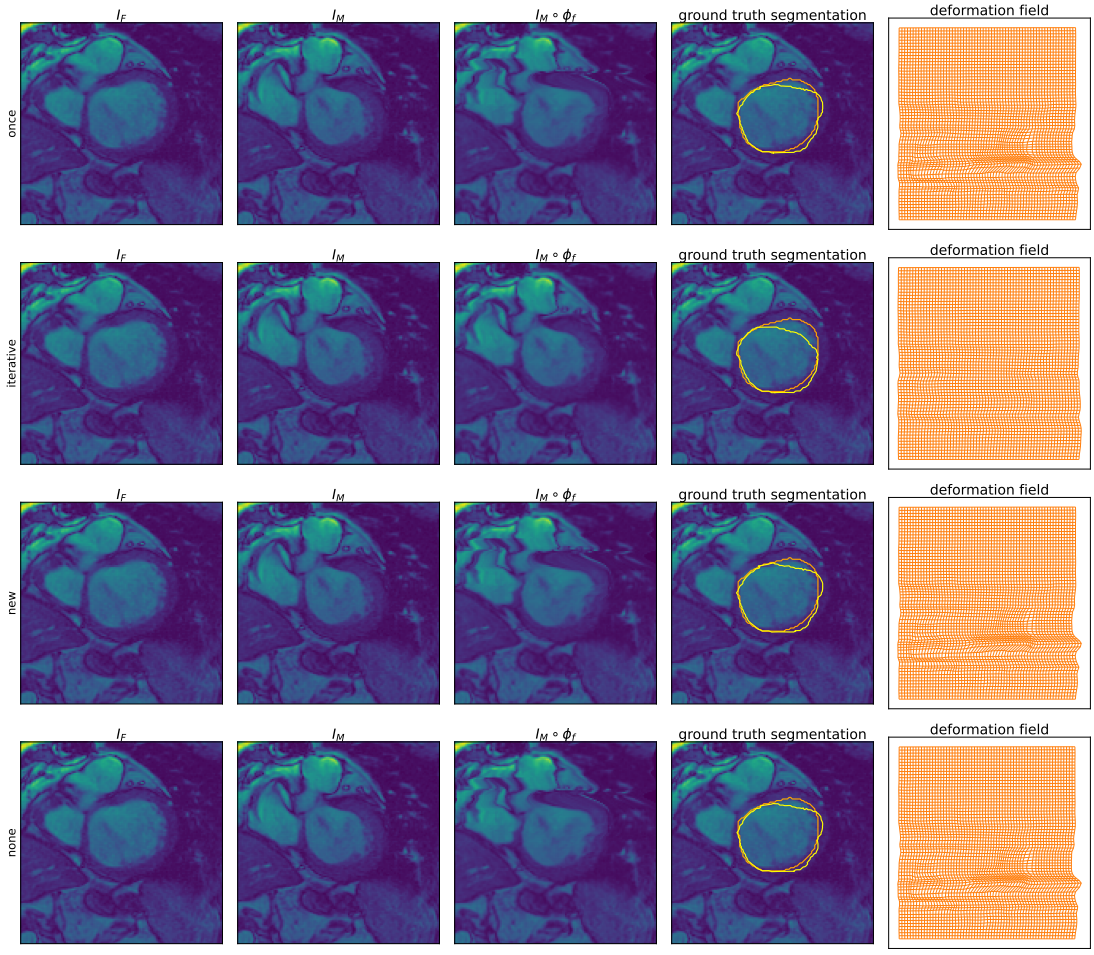


Figure 5.10: Registration results for the Sunnybrook Cardiac Data with an INR using the four different methods to constrain the monitor function and 10 ODE steps, compare figure 5.6. Despite the use of four different methods to constrain the monitor function, similar registration results are achieved across all approaches, with the exception of the iterative method, which produces a slightly different deformation field. Overall, these displayed results are an example of a better registration result.

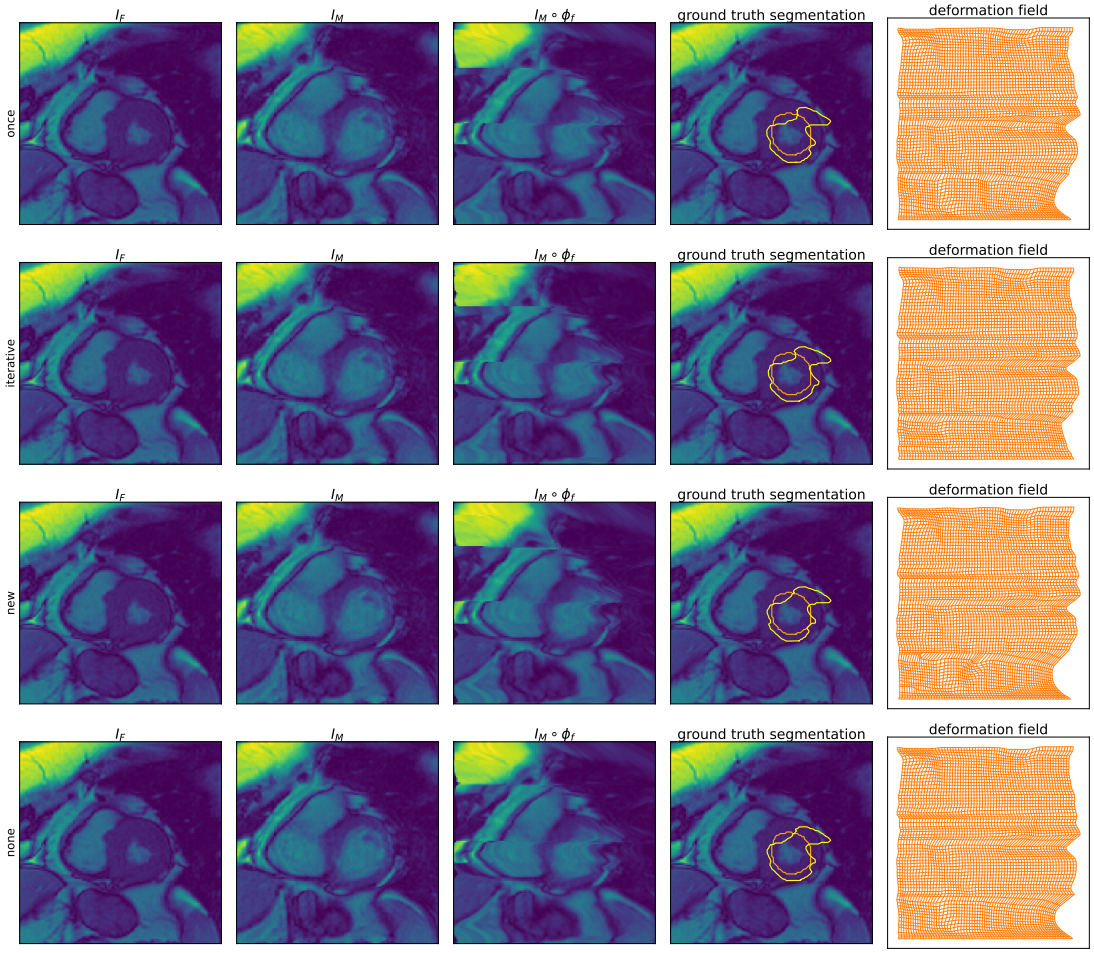


Figure 5.11: Registration results for the Sunnybrook Cardiac Data with implicit neural representation using the four different method to constrain the monitor function and 10 ODE steps, compare figure 5.6. This figure displays an example of poor registration results, as all of the registrations show implausible distortions of the left ventricle. Additionally, this registration demonstrates a high DM but a lower HD. While the segmentations show an overall high overlap, the segmentations have strong outliers in the top right corner.

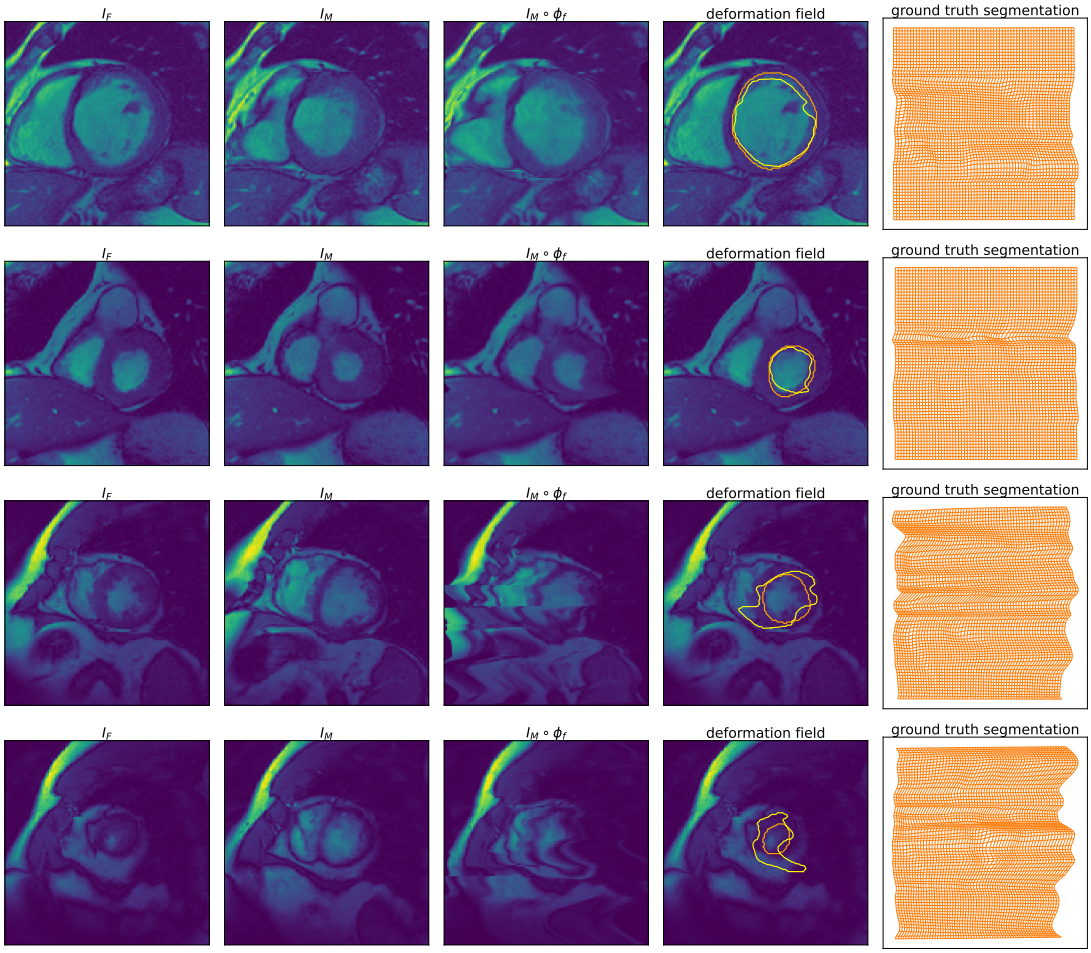


Figure 5.12: Registration results for the Sunnybrook Cardiac Data with INR using the new constraining method for the monitor function and 10 ODE steps, compare figure 5.6. The first two rows show examples of good registration results, while the last two rows show examples where the method performed worse. The deformation fields of the last two rows are clearly not diffeomorphic. They show strong irregularities from the expected rather round shape of the LV, leading to implausible results.

5.4.4 Comparison of the Registration Results Using a U-Net and INR

Now we compare the performance of the U-Net- and INR-based approaches. Overall, the INR approach shows better registration results compared to the U-Net in terms of the DM, the reliability, and the MSE, yet it fails to achieve the results proposed by Sheikhjafari et al. While INR avoids many uncertainties associated with U-Net training and achieves values in the DM up to 0.79, it still does not meet the reported DM of 0.88. Although the registration utilising INR demonstrates a higher DM than the U-Net approach, the HD values are distinctly higher for all except the iterative approach. This could be due to the outliers mentioned in the previous section.

The method used to constrain the monitor function appears less important for INR than for the U-Net. This might be caused by the overall better performance of the INR in comparison to U-Nets. As the INR directly trains on the individual image pair, it is less dependent on the used data. The training set for the U-Net is relatively small, which might explain its inferior results. In comparison to the INR, the U-Net performs best on images that are similar to the training images, making INR more versatile for a broader variety of images.

As the iterative approach requires the most time among our tested constraining methods and is outperformed in both, the U-Net and INR, it is probably the least useful constraining method. The other three approaches yield similar results, with a slightly better performance of the scaling once method.

Comparing the registration times of U-Net and INR is challenging due to their fundamentally different approaches. While the U-Net trains one model prior to the registration, allowing the registration of a new image pair with a single forward pass, the INR trains a new model for each new image pair individually. In our setup, the U-Net trains a model in approximately one hour and 15 minutes for 10 ODE steps, an unseen image pair is then registered within approximately 0.04 seconds. Using INR, the registration of an individual image pair includes the training time for a new model, taking up to 35 seconds for a new image pair. Therefore, the U-Net is more suitable for scenarios requiring real-time registration, while INR is preferable when registration time is less critical or the number of registrations is limited.

For the results with 20 ODE steps, we refer the reader to the appendix; the results are displayed in table A.1 and table A.2, showing similar results to the results for 10 ODE steps. Although we expected that more ODE steps would lead to more accurate ODE solutions, our experiments do not show an improvement in the registration results for the increased number of steps.

Table 5.5: Training and registration times for U-Net and INR for 10 ODE steps and the different constraining methods. All times are in seconds. For reference: 4500 seconds are one hour and 15 minutes.

ODE steps	**constrain method**	**U-Net training**	**U-Net registration**	**INR registration**
10	once	4490.4696 s	0.0434 s	33.4592 s
	iterative	4690.5610 s	0.0437 s	35.8274 s
	new	4450.3578 s	0.0435 s	33.7923 s
	none	4879.9498 s	0.0435 s	34.1079 s

6

Conclusion

The aim of this thesis was the validation and extension of the moving mesh algorithm for deformable cardiac image registration proposed by Sheikhjafari et al. [75]. Our work includes the implementation of an FFT-based Poisson solver and an evaluation of its accuracy and run times. We tested the solver using three different functions across three grid resolutions and could successfully verify the functionality, robustness, and efficiency of the solver, which showed the expected results on all tests.

A critical part of the moving mesh approach is the constraining of the monitor function, which involves clamping and scaling of the monitor function. To address this, we developed a new activation function that clamps the values of the monitor function to the required range. The new activation function avoids the use of simply clamping the values, and thus preserves the behaviour of the monitor function. Additionally, we explored four different methods to scale the monitor function. Two approaches perform a simple global scaling, where one of these approaches performs the clamping and scaling iteratively until both constraints are satisfied or a maximum number of iterations is performed. We also introduced a new approach that adaptively scales the monitor function depending on whether the integral is too large or too small: scaling it down in the areas where it exceeds the prescribed mean value of one, or scaling it up in areas where it falls below. To understand the importance of the scaling, we additionally tested the registration without any scaling.

We performed the registration using a U-Net framework as proposed by Sheikhjafari et al., and augmented this approach by replacing the U-Net with an implicit neural representation. The registration results were validated on a benchmark data set. In our implementation, the INR outperformed the U-Net in all metrics except the Hausdorff distance. Our best results were achieved using an INR, with a Dice Metric if 0.78, a reliability of 0.65 and a Hausdorff distance of 9.33. However, our implementation did not match the registration results reported by Sheikhjafari et al., who achieved a Dice metric of 0.88, a reliability of 0.90, and a Hausdorff distance of 5.25. Possible reasons for this were discussed in section 5.4.4, including missing key information on their U-Net set up and on the constraining of the monitor function, as well as the limited size of the training data set. Additionally, the choice of the parameter w_0 for the INR framework might have contributed to worse performance. Moreover, the test data set distribution was not comparable, leading to incomparable registration results.

While our new activation function successfully clamps the monitor function to the desired range, further testing and comparison to the simple clamping function could examine the influence of the clamping on the outcome of the registration.

Although a major advantage of the moving mesh-based registration is that the regis-

tration can be performed without explicit regularisation, it could still potentially benefit from regularisation. In the case of the registration of the left ventricle, penalising non-smooth outlines could be beneficial, as we observed strong outliers from an expected round shape in some cases. As the contours are provided for our data set, they could be incorporated for the U-Net training process. For the INR approach, which optimises directly on the individual image pair, in practise the segmentations are usually unavailable, and the regularisation could force smoothness of the deformation field.

Further testing of our implementation on a larger training set could help to determine whether the observed errors are due to implementation issues or the limited size of the data set. As the method proposed by Sheikhjafari et al. is also applicable to the registration of three-dimensional images, extending our implementation to three dimensions could be a further next step.

References

[1] N-dimensional gradient loss for pytorch. `https://github.com/voxelmorph/voxelmorph/pull/333/commits/2500ee34647c844675cba3dace45879c30049889`. Accessed: 2024-03-20.

[2] Bernd Aulbach. *Gewöhnliche Differentialgleichungen*, volume 2. Spektrum, Akad. Verl., 1997.

[3] Guha Balakrishnan, Amy Zhao, Mert Sabuncu, John Guttag, and Adrian V. Dalca. An unsupervised learning model for deformable medical image registration. *CVPR: Computer Vision and Pattern Recognition*, pages 9252–9260, 2018.

[4] Guha Balakrishnan, Amy Zhao, Mert Sabuncu, John Guttag, and Adrian V. Dalca. Voxelmorph: A learning framework for deformable medical image registration. *IEEE TMI: Transactions on Medical Imaging*, 38:1788–1800, 2019.

[5] Augustin Banyaga. Formes-volume sur les variétésa bord. *Enseignement Math*, 20 (2):127–131, 1974.

[6] Mariza Faisal Beg, Michael I. Miller, Alain Trouvé, and Laurent Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International journal of computer vision*, 61:139–157, 2005.

[7] Bart Bijnens, M Cikes, C Butakoff, Marta Sitges, and Fatima Crispi. Myocardial motion and deformation: What does it tell us and how does it relate to function? *Fetal diagnosis and therapy*, 32(1-2):5–16, 2012.

[8] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

[9] Louis Brand. *Vector analysis*. Courier Corporation, 2012.

[10] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM computing surveys (CSUR)*, 24(4):325–376, 1992.

[11] Michal Byra, Charissa Poon, Muhammad Febrian Rachmadi, Matthias Schlachter, and Henrik Skibbe. Exploring the performance of implicit neural representations for brain image registration. *Scientific Reports*, 13(1):17334, 2023.

[12] Raymond H. Chan, Michael K. Ng, and CK Wong. Sine transform based preconditioners for symmetric Toeplitz systems. *Linear algebra and its applications*, 232: 237–259, 1996.

[13] Ching-Lung Chang and Max D. Gunzburger. A finite element method for first order elliptic systems in three dimensions. *Applied Mathematics and Computation*, 23(2): 171–184, 1987.

[14] Hua-mei Chen, Aashish Goela, Gregory J. Garvin, and Shuo Li. A parameterization of deformation fields for diffeomorphic image registration and its application to myocardial delineation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 340–348. Springer, 2010.

[15] Ricky T. Q. Chen. torchdiffeq, 2018. URL `https://github.com/rtqichen/torchdiffeq`. PyTorch Implementation of Differentiable ODE Solvers.

[16] David Keun Cheng et al. *Field and wave electromagnetics*. Pearson Education India, 1989.

[17] June Cheng-Baron, Kelvin Chow, Joseph J. Pagano, Kumaradevan Punithakumar, D. Ian Paterson, Gavin Y. Oudit, and Richard B. Thompson. Quantification of circumferential, longitudinal, and radial global fractional shortening using steady-state free precession cines: a comparison with tissue-tracking strain and application in fabry disease. *Magnetic Resonance in Medicine*, 73(2):586–596, 2015.

[18] Mei-Yi Chu, Hua-Mei Chen, Chih-Yao Hsieh, Ting-Hung Lin, Hsi-Yue Hsiao, Guo-jun Liao, and Qi Peng. Adaptive grid generation based non-rigid image registration using mutual information for breast MRI. *Journal of signal processing systems*, 54: 45–63, 2009.

[19] Bernard Dacorogna and Jürgen Moser. On a partial differential equation involving the jacobian determinant. In *Annales de l'Institut Henri Poincaré C, Analyse non linéaire*, volume 7, pages 1–26. Elsevier, 1990. doi: https://doi.org/10.1016/S0294-1449(16)30307-9.

[20] Adrian V. Dalca, Guha Balakrishnan, et al. VoxelMorph: learning-based image registration. `https://github.com/voxelmorph/voxelmorph`. Accessed: 2024-01-12.

[21] Minh Dao, Chiman Kwan, Bulent Ayhan, and James F. Bell. Enhancing mastcam images for mars rover mission. In *Advances in Neural Networks-ISNN 2017: 14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, June 21–26, 2017, Proceedings, Part II 14*, pages 197–206. Springer, 2017.

[22] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[23] Paul Dupuis, Ulf Grenander, and Michael I. Miller. Variational problems on flows of diffeomorphisms for image matching. *Quarterly of applied mathematics*, pages 587–600, 1998.

[24] Lawrence C. Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[25] Gerd Fischer. *Lineare Algebra*. Vieweg+Teubner Verlag Wiesbaden, 16th edition, 2010.

[26] Clinton B. Fookes and Mohammed Bennamoun. *Rigid and non-rigid image registration and its association with mutual information: A review*. 2002.

[27] Daniel Fortunato and Alex Townsend. Fast Poisson solvers for spectral methods. *IMA Journal of Numerical Analysis*, 40(3):1994–2018, 2020.

[28] Mireille Garreau, Antoine Simon, Dominique Boulmier, Jean-Louis Coatrieux, and Hervé Le Breton. Assessment of left ventricular function in cardiac MSCT imaging by a 4D hierarchical surface-volume matching process. *International Journal of Biomedical Imaging*, 2006(1):037607, 2006.

[29] Daniel T. Ginat, Michael W. Fong, David J. Tuttle, Susan K. Hobbs, and Rajashree C. Vyas. Cardiac imaging: Part 1, MR pulse sequences, imaging planes, and basic anatomy. *American Journal of Roentgenology*, 197(4):808–815, 2011.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[31] Eldad Haber and Jan Modersitzki. Numerical methods for volume preserving image registration. *Inverse problems*, 20(5):1621, 2004.

[32] Eldad Haber and Jan Modersitzki. Image registration with guaranteed displacement regularity. *International journal of computer vision*, 71(3):361–372, 2007.

[33] Wolfgang Hackbusch. Die Poisson-Gleichung. *Theorie und Numerik elliptischer Differentialgleichungen*, pages 27–38, 2017.

[34] Jacques Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, pages 49–52, 1902.

[35] Derek LG Hill, Philipp G. Batchelor, Mark Holden, and David J. Hawkes. Medical image registration. *Physics in medicine & biology*, 46(3):R1, 2001.

[36] Roger W. Hockney. A fast direct solution of Poisson's equation using fourier analysis. *Journal of the ACM (JACM)*, 12(1):95–113, 1965.

[37] Hsi-Yue Hsiao, Chih-Yao Hsieh, Xi Chen, Yongyi Gong, Xiaonan Luo, and Guojun Liao. New development of nonrigid registration. *The ANZIAM Journal*, 55(3): 289–297, 2014.

[38] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[39] Antony Jameson. Solution of the equation ax+xb=c by inversion of an m*m or n*n matrix. *SIAM Journal on Applied Mathematics*, 16(5):1020–1023, 1968.

[40] Xi Jia, Joseph Bartlett, Tianyang Zhang, Wenqi Lu, Zhaowen Qiu, and Jinming Duan. U-net vs transformer: Is U-net outdated in medical image registration? In *International Workshop on Machine Learning in Medical Imaging*, pages 151–160. Springer, 2022.

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

[42] Deepa Krishnaswamy, Michelle Noga, Harald Becher, Pierre Boulanger, and Kumaradevan Punithakumar. A novel 3D-to-3D diffeomorphic registration algorithm with applications to left ventricle segmentation in MR and ultrasound sequences. *IEEE Access*, 11:3144–3159, 2023.

[43] Wilhelm Kutta. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*. Teubner, 1901.

[44] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

[45] Guojun Liao and Dale Anderson. A new approach to grid generation. *Applicable analysis*, 44(3-4):285–298, 1992.

[46] Guojun Liao and J. Su. Grid generation via deformation. *Applied mathematics letters*, 5(3):27–29, 1992.

[47] Jie Liu. New development of the deformation method. *The University of Texas at Arlington*, 2006.

[48] Shuangzhe Liu, Götz Trenkler, Tõnu Kollo, et al. Professor Heinz Neudecker and matrix differential calculus. *Statistical Papers*, 65:2605–2639, 2024. doi: 10.1007/s00362-023-01499-w. URL `https://doi.org/10.1007/s00362-023-01499-w`.

[49] Yihao Liu, Junyu Chen, Shuwen Wei, Aaron Carass, and Jerry Prince. On finite difference jacobian computation in deformable image registration. *International Journal of Computer Vision*, pages 1–11, 2024.

[50] JB Antoine Maintz and Max A. Viergever. A survey of medical image registration. *Medical image analysis*, 2(1):1–36, 1998.

[51] Timo Makela, Patrick Clarysse, Outi Sipila, Nicoleta Pauna, Quoc Cuong Pham, Toivo Katila, and Isabelle E. Magnin. A review of cardiac image registration methods. *IEEE Transactions on medical imaging*, 21(9):1011–1021, 2002.

[52] Andreas Meister. *Numerik linearer Gleichungssysteme*, volume 4. Springer, 2011.

[53] Jan Modersitzki. *Numerical Methods for Image Registration*. Oxford University Press, 2004.

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

[55] Amirali Molaei, Amirhossein Aminimehr, Armin Tavakoli, Amirhossein Kazerouni, Bobby Azad, Reza Azad, and Dorit Merhof. Implicit neural representation in medical imaging: A comparative survey. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2381–2391, 2023.

[56] Jürgen Moser. On the volume elements on a manifold. *Transactions of the American Mathematical Society*, 120(2):286–294, 1965.

[57] Silvia Noschese, Lionello Pasquini, and Lothar Reichel. Tridiagonal Toeplitz matrices: properties and novel applications. *Numerical linear algebra with applications*, 20(2):302–326, 2013.

[58] Susan T. O'Donnel, Peter Geiger, and Martin H. Schultz. Solving the Poisson equation on the FPS-164. Technical report, YALEU/DCS/RR-293, Research Center for Scientific Computing, Dept. of Computer Science, Yale University, 1983.

[59] Keiron O'shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[61] Caroline Petitjean and Jean-Nicolas Dacher. A review of segmentation methods in short axis cardiac MR images. *Medical image analysis*, 15(2):169–184, 2011.

[62] William H. Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[63] Kumaradevan Punithakumar, Ismail Ben Ayed, Ali Islam, Aashish Goela, Ian G. Ross, Jaron Chong, and Shuo Li. Regional heart motion abnormality detection: An information theoretic approach. *Medical image analysis*, 17(3):311–324, 2013.

[64] Kumaradevan Punithakumar, Michelle Noga, Ismail Ben Ayed, and Pierre Boulanger. Right ventricular segmentation in cardiac MRI with moving mesh correspondences. *Computerized Medical Imaging and Graphics*, 43:15–25, 2015.

[65] Kumaradevan Punithakumar, Pierre Boulanger, and Michelle Noga. A GPU-accelerated deformable image registration algorithm with applications to right ventricular segmentation. *IEEE Access*, 5:20374–20382, 2017.

[66] Perry Radau, Yingli Lu, Kim Connelly, Gideon Paul, Alexander J. Dick, and Graham A. Wright. Evaluation framework for algorithms segmenting short axis cardiac MRI. *The MIDAS Journal*, 2009.

[67] Prabhakar Shantha Rajiah, Christopher J. François, and Tim Leiner. Cardiac MRI: state of the art. *Radiology*, 307(3):e223008, 2023.

[68] Walter J. Rogers Jr, Edward P. Shapiro, James L. Weiss, Maurice B. Buchalter, Frank E. Rademakers, Myron L. Weisfeldt, and Elias A. Zerhouni. Quantification of and correction for left ventricular systolic long-axis shortening by magnetic resonance tissue tagging and slice isolation. *Circulation*, 84(2):721–731, 1991.

[69] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[70] Max Roser. Causes of death globally: what do people die from? *Our World in Data*, 2021. https://ourworldindata.org/causes-of-death-treemap, accessed: 2024-06-03.

[71] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[72] Carl Runge. Über die numerische Auflösung von Differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, 1895.

[73] Bosung Seo, Daniel Mariano, John Beckfield, Vinay Madenur, Yuming Hu, Tony Reina, Marcus Bobar, Mai H Nguyen, and Ilkay Altintas. Cardiac MRI image segmentation for left ventricle and right ventricle using deep learning. *arXiv preprint arXiv:1909.08028*, 2019.

[74] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[75] Ameneh Sheikhjafari, Deepa Krishnaswamy, Michelle Noga, Nilanjan Ray, and Kumaradevan Punithakumar. Unsupervised diffeomorphic cardiac image registration using parameterization of the deformation field. *arXiv preprint arXiv:2208.13275*, 2022.

[76] Ameneh Sheikhjafari, Michelle Noga, Kumaradevan Punithakumar, and Nilanjan Ray. A training-free recursive multiresolution framework for diffeomorphic deformable image registration. *Applied Intelligence*, 52(11):12546–12555, 2022.

[77] Ameneh Sheikhjafari, Deepa Krishnaswamy, Michelle Noga, Nilanjan Ray, and Kumaradevan Punithakumar. Deep learning based parameterization of diffeomorphic image registration for cardiac image segmentation. *IEEE Transactions on NanoBioscience*, 2023.

[78] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[79] Gordon D. Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985.

[80] Yucheng Song, Shengbing Ren, Yu Lu, Xianghua Fu, and Kelvin KL Wong. Deep learning-based automatic segmentation of images in cardiac radiography: a promising challenge. *Computer Methods and Programs in Biomedicine*, 220:106821, 2022.

[81] Shanlin Sun, Kun Han, Hao Tang, Deying Kong, Junayed Naushad, Xiangyi Yan, and Xiaohui Xie. Medical image registration via neural fields. *arXiv preprint arXiv:2206.03111*, 2022.

[82] Abdel Aziz Taha and Allan Hanbury. An efficient algorithm for calculating the exact Hausdorff distance. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2153–2163, 2015.

[83] Alain Trouvé. An infinite dimensional group approach for physics based models. *technical report (electronically)*, 1995.

[84] Moises Vasquez and Eike Nagel. Clinical indications for cardiovascular magnetic resonance. *Heart*, 105(22):1755–1762, 2019.

[85] Jelmer M. Wolterink, Jesse C. Zwienenberg, and Christoph Brune. Implicit neural representations for deformable image registration. In *International Conference on Medical Imaging with Deep Learning*, pages 1349–1359. PMLR, 2022.

[86] World Health Organization. Cardiovascular diseases (CVDs). `https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)`, 2021. Accessed: 2024-06-03.

[87] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

[88] Hui Xue, Jessica Artico, Marianna Fontana, James C Moon, Rhodri H Davies, and Peter Kellman. Landmark detection in cardiac MRI by using a convolutional neural network. *Radiology: Artificial Intelligence*, 3(5):e200197, 2021.

[89] Laurent Younes. *Shapes and diffeomorphisms*, volume 171. Springer, 2010.

[90] Laurent Younes, Felipe Arrate, and Michael I. Miller. Evolutions equations in computational anatomy. *NeuroImage*, 45(1):S40–S50, 2009.

[91] Jianping Zhang and Yanyan Li. Diffeomorphic image registration with an optimal control relaxation and its implementation. *SIAM Journal on Imaging Sciences*, 14 (4):1890–1931, 2021.

[92] Xingling Zhou. On uniqueness theorem of a vector function. *Progress in Electromagnetics Research*, 65:93–102, 2006.

[93] Veronika A. Zimmer, Kerstin Hammernik, Vasiliki Sideri-Lampretsa, Wenqi Huang, Anna Reithmeir, Daniel Rueckert, and Julia A. Schnabel. Towards generalised neural implicit representations for image registration. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 45–55. Springer, 2023.

A

Appendix

A.1 Proof of Abel's Lemma

Here, we show the proof of Abel's lemma given by Liu [47], to which we add further details.

Lemma A.1 (Abel's Lemma)**.** *Let $M(t)$ be a $d \times d$ matrix such that each element of the matrix is differentiable on t, let $\frac{\mathrm{d}}{\mathrm{d}t}M(t)$ be the matrix with the differentiated elements of $M(t)$. If $\frac{\mathrm{d}}{\mathrm{d}t}M(t) = A(t)M(t)$, where $A(t)$ is a $d \times d$ matrix, then*

$$\frac{\mathrm{d}}{\mathrm{d}t}(\det M(t)) = (\operatorname{trace} A(t))(\det M(t)).$$

Proof. To aid readability, the t arguments are omitted for the remainder of this proof. Let $M = (m_{ij})_{i,j=1}^{d} \in \mathbb{R}^{d\times d}$ be an arbitrary matrix such that a matrix $A \in \mathbb{R}^{d\times d}$ exists satisfying

$$\frac{\mathrm{d}}{\mathrm{d}t}M = AM,$$

where the elements of $\frac{\mathrm{d}}{\mathrm{d}t}M$ are denoted by $\frac{\mathrm{d}}{\mathrm{d}t}M = (m'_{ij})_{i,j=1}^{d}$ and the elements of A are denoted by $A = (a_{ij})_{i,j=1}^{d}$.

A general representation of the determinant is given by the Leibniz formula for determinants [25, Chapter 3],

$$\det(M) = \sum_{s\in\mathcal{S}_d} \left(\operatorname{sgn}(s) \prod_{i=1}^{d} m_{i,s(i)} \right),$$

where $\mathcal{S}_d$ denotes the symmetric group of degree d and $s \in \mathcal{S}_d$ is a permutation with

$$\operatorname{sgn}(s) = \begin{cases} +1, & \text{if } s \text{ is an even permutation,} \\ -1, & \text{if } s \text{ is an odd permutation.} \end{cases}$$

In order to to prove Abel's lemma, $\det(M)$ is represented by the Leibniz formula

$$\frac{\mathrm{d}}{\mathrm{d}t}\det(M) = \frac{\mathrm{d}}{\mathrm{d}t} \sum_{s\in\mathcal{S}_d} \left(\operatorname{sgn}(s) \prod_{i=1}^{d} m_{i,s(i)} \right).$$

As the derivative of a sum is the sum of the derivatives and $\operatorname{sgn}(s)$ does not depend on t, the term can be rearranged by moving the derivative into the sum

$$\frac{\mathrm{d}}{\mathrm{d}t}\det(M) = \sum_{s\in\mathcal{S}_d}\left(\operatorname{sgn}(s)\frac{\mathrm{d}}{\mathrm{d}t}\prod_{i=1}^{d} m_{i,s(i)}\right). \tag{A.1}$$

Now, the product rule for differentiation is applied. Using the product rule on a product of d functions $f_i : \mathbb{R} \to \mathbb{R}$ with $i = 1, \ldots, d$, results in

$$\frac{\mathrm{d}}{\mathrm{d}t}\prod_{i=1}^{d} f_i(t) = \sum_{j=1}^{d}\left(\frac{\mathrm{d}}{\mathrm{d}t}f_j(t)\prod_{\substack{i=1\\ i\neq j}}^{d} f_i(t)\right).$$

Applying this to (A.1) leads to

$$\frac{\mathrm{d}}{\mathrm{d}t}\det(M) = \sum_{s\in\mathcal{S}_d}\left(\operatorname{sgn}(s)\sum_{j=1}^{d} m'_{j,s(j)}\prod_{\substack{i=1\\ i\neq j}}^{d} m_{i,s(i)}\right).$$

Since $\operatorname{sgn}(s)$ is not depending on j and the sums are finite, the sums can be swapped

$$\frac{\mathrm{d}}{\mathrm{d}t}\det(M) = \sum_{j=1}^{d}\left(\sum_{s\in\mathcal{S}_d}\operatorname{sgn}(s)\ m'_{j,s(j)}\prod_{\substack{i=1\\ i\neq j}}^{d} m_{i,s(i)}\right),$$

where the term in the parentheses is the Leibniz formula for the matrix M with one row consisting of the differentiated elements of M. Therefore, the derivative of $\det(M)$ results in a sum of d determinants of modified versions of M, where in i-th row the elements of M are swapped to their derivatives, leaving

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\det(M) = &\begin{vmatrix} m'_{11} & m'_{12} & \ldots & m'_{1d}\\ m_{21} & m_{22} & \ldots & m_{2d}\\ \vdots & \vdots & \ddots & \vdots\\ m_{d1} & m_{d2} & \ldots & m_{dd}\end{vmatrix} + \begin{vmatrix} m_{11} & m_{12} & \ldots & m_{1d}\\ m'_{21} & m'_{22} & \ldots & m'_{2d}\\ \vdots & \vdots & \ddots & \vdots\\ m_{d1} & m_{d2} & \ldots & m_{dd}\end{vmatrix}\\ &+\cdots+\begin{vmatrix} m_{11} & m_{12} & \ldots & m_{1d}\\ m_{21} & m_{22} & \ldots & m_{2d}\\ \vdots & \vdots & \ddots & \vdots\\ m'_{d1} & m'_{d2} & \ldots & m'_{dd}\end{vmatrix}.\end{aligned} \tag{A.2}$$

Now, under the assumption $\frac{\mathrm{d}}{\mathrm{d}t}M = AM$, the elements m'_{ij} in (A.2) can be replaced by $m'_{ij} = \sum_{k=1}^{d} a_{ik}m_{kj}$. Then, for the i-th determinant of (A.2), $i = 1, \ldots, d$, the i-th row can be simplified by subtracting a_{ik} times the k-th row, $k = 1, \ldots, d, k \neq i$, of the i-th row. These operations do not change the determinants, as they are row equivalent

operations, leaving the simplified determinants as

$$\frac{\mathrm{d}}{\mathrm{d}t}\det(M) = \begin{vmatrix} a_{11}m_{11} & a_{11}m_{12} & \dots & a_{11}m_{1d} \\ m_{21} & m_{22} & \dots & m_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{d1} & m_{d2} & \dots & m_{dd} \end{vmatrix} + \begin{vmatrix} m_{11} & m_{12} & \dots & m_{1d} \\ a_{22}m_{21} & a_{22}m_{22} & \dots & a_{22}m_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{d1} & m_{d2} & \dots & m_{dd} \end{vmatrix} + \dots + \begin{vmatrix} m_{11} & m_{12} & \dots & m_{1d} \\ m_{21} & m_{22} & \dots & m_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{dd}m_{d1} & a_{dd}m_{d2} & \dots & a_{dd}m_{dd} \end{vmatrix}.$$

The i-th determinant has the factor a_{kk} in the k-th row. Now using the multilinearity of determinants, the terms reduce to

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\det(M) &= a_{11}\det(M) + a_{22}\det(M) + \dots + a_{dd}\det(M) \\ &= (a_{11} + a_{22} + \dots + a_{dd})\det(M) \\ &= (\operatorname{trace} A)(\det M), \end{aligned}$$

which completes the proof for Abel's lemma. □

A.2 Registration Results for 20 ODE Steps

Table A.1: Quantitative evaluation of the registration results for the moving mesh-based registration on the Sunnybrook Cardiac Data using a U-Net with 20 time step in the ODE, compare table 5.3.

constrain method model	**evaluation metric**	**forward**	**backward**
scaling once model 490	MSE	0.0039 ± 0.004	0.0039 ± 0.004
	DM	0.74 ± 0.18	0.75 ± 0.19
	HD	9.05 ± 4.50	$\mathbf{8.54 \pm 4.05}$
	R	0.51	0.53
	$\%\lvert J\rvert < 0$	0.02	0.01
iterative constraining model 410	MSE	0.0043 ± 0.005	0.0044 ± 0.005
	DM	0.74 ± 0.18	0.74 ± 0.18
	HD	$\mathbf{8.99 \pm 4.37}$	8.75 ± 3.99
	R	0.55	**0.55**
	$\%\lvert J\rvert < 0$	**0.00**	**0.00**
new scaling model 270	MSE	$\mathbf{0.0038 \pm 0.004}$	0.0039 ± 0.004
	DM	$\mathbf{0.75 \pm 0.18}$	$\mathbf{0.76 \pm 0.18}$
	HD	9.12 ± 5.03	9.03 ± 4.59
	R	**0.57**	**0.55**
	$\%\lvert J\rvert < 0$	0.01	**0.00**
no scaling model 280	MSE	$\mathbf{0.0038 \pm 0.004}$	$\mathbf{0.0038 \pm 0.004}$
	DM	$\mathbf{0.75 \pm 0.17}$	$\mathbf{0.76 \pm 0.19}$
	HD	9.16 ± 4.17	9.35 ± 5.92
	R	**0.57**	**0.55**
	$\%\lvert J\rvert < 0$	0.07	0.02

Table A.2: Quantitative evaluation of the registration results for the moving mesh-based registration on the Sunnybrook Cardiac Data using implicit neural representation with 20 time step in the ODE, compare table 5.3.

constrain method	**evaluation metric**	**forward**	**backward**
scaling once	MSE	0.0025 ± 0.003	0.0027 ± 0.003
	DM	$\mathbf{0.78 \pm 0.18}$	$\mathbf{0.78 \pm 0.18}$
	HD	9.14 ± 4.83	9.68 ± 7.22
	R	0.62	**0.64**
	$\%\lvert J\rvert < 0$	0.02	**0.00**
iterative constraining	MSE	0.0026 ± 0.003	0.0028 ± 0.003
	DM	0.77 ± 0.18	0.77 ± 0.18
	HD	$\mathbf{8.86 \pm 4.57}$	$\mathbf{9.44 \pm 6.35}$
	R	0.60	0.61
	$\%\lvert J\rvert < 0$	0.02	0.01
new scaling	MSE	$\mathbf{0.0024 \pm 0.002}$	$\mathbf{0.0026 \pm 0.003}$
	DM	$\mathbf{0.78 \pm 0.18}$	$\mathbf{0.78 \pm 0.18}$
	HD	9.25 ± 4.94	10.05 ± 7.30
	R	**0.64**	**0.64**
	$\%\lvert J\rvert < 0$	**0.01**	0.01
no scaling	MSE	$\mathbf{0.0024 \pm 0.003}$	$\mathbf{0.0026 \pm 0.003}$
	DM	$\mathbf{0.78 \pm 0.17}$	$\mathbf{0.78 \pm 0.18}$
	HD	9.43 ± 5.87	9.62 ± 6.58
	R	**0.64**	**0.64**
	$\%\lvert J\rvert < 0$	**0.01**	0.01

Table A.3: Training and registration times for U-Net and INR for 20 ODE steps and the different constraining methods. All times are in seconds. For reference: 8100 seconds are two hours and 15 minutes.

ODE steps	**constrain method**	**U-Net training**	**U-Net registration**	**INR registration**
20	once	7729.9066 s	0.0681 s	61.4333 s
	iterative	8211.6507 s	0.0679 s	64.3033 s
	new	7794.6702 s	0.0682 s	62.3587 s
	none	7765.3996 s	0.0681 s	61.2617 s

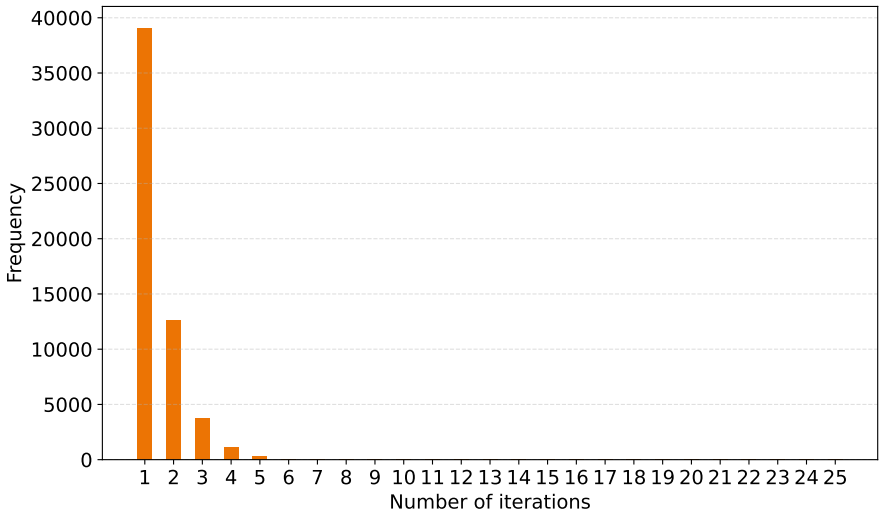


Figure A.1: This bar plot illustrates the distribution of iterations for iteratively constraining the monitor function using a U-Net with 20 ODE steps, compare table 5.8. A cluster for one to three iterations is observable. The average number of iterations is 1.44.

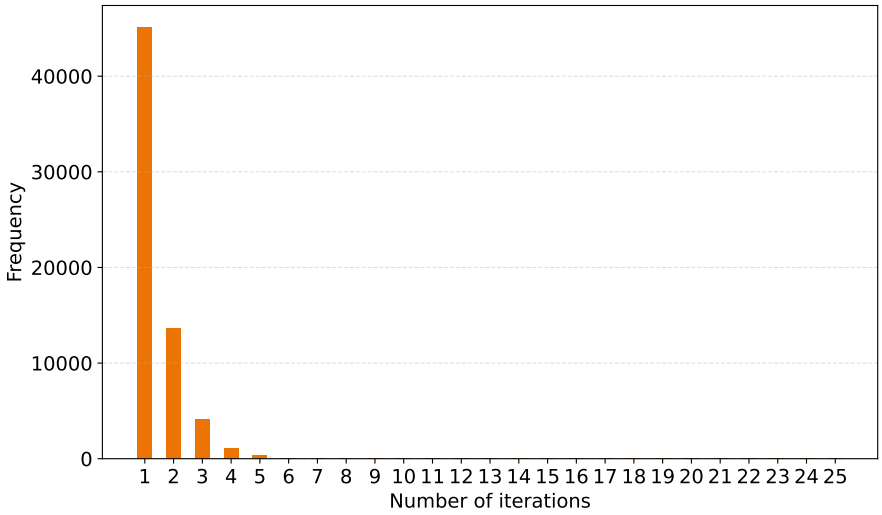


Figure A.2: This bar plot illustrates the distribution of iterations for iteratively constraining the monitor function using an INR with 20 ODE steps, compare table 5.8. A cluster for one to three iterations is observable. The average number of iterations is 1.42.