On the registration and modeling of sequential medical images

Niklas Gunnarsson

UPPSALA UNIVERSITY
Department of Information Technology
On the registration and modeling of sequential medical images

Niklas Gunnarsson
niklas.gunnarsson@it.uu.se

Dec 2021

Division of Systems and Control
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

Dissertation for the degree of Licentiate of Philosophy in Electrical Engineering with specialisation in Signal Processing

© Niklas Gunnarsson 2021
ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden
Abstract

Real-time imaging can be used to monitor, analyze and control medical treatments. In this thesis, we want to explain the spatiotemporal motion and thus enable more advanced procedures, especially real-time adaptation in radiation therapy. The motion occurring between image acquisitions can be quantified by image registration, which generates a mapping between the images.

The contribution of the thesis consists of three papers, where we have used different approaches to estimate the motion between images.

In Paper I, we combine a state-of-the-art method in real-time tracking with a learned sparse-to-dense interpolation scheme. For this, we track an arbitrary number of regions in a sequence of medical images. We estimated a sparse displacement field, based on the tracking positions and used the interpolation network to achieve its dense representation.

Paper II was a contribution to a challenge in learnable image registration where we finished at 2nd place. Here we train a deep learning method to estimate the dense displacement field between two images. For this, we used a network architecture inspired by both conventional medical image registration methods and optical flow in computer vision.

For Paper III, we estimate the dynamics of spatiotemporal images by training a generative network. We use nonlinear dimensional reduction techniques and assume a linear dynamic in a low-dimensional latent space. In comparison with conventional image registration methods, we provide a method more suitable for real-world scenarios, with the possibility of imputation and extrapolation.

Although the problem is challenging and several questions are left unanswered we believe a combination of conventional, learnable, and dynamic modeling of the motion is the way forward.
Acknowledgments

With a background in team sports, I have always considered myself a team player. Although many say that Ph.D. studies are lonely and individual I disagree. Therefore there are several people how made this possible. First of all, I want to say thanks to my former industrial supervisor, present co-supervisor Jens Sjölund, who gave me the opportunity to do this Ph.D. project. I want to thank you for all your ideas and fruitful discussions. Further on I want to thanks my supervisor Thomas Schön for your excellent expertise, ability to understand domain-specific questions, and patience and availability whenever I need a sounding board. I also want to thank my current industrial supervisor Peter Kimstrand for the help and support you have given me in the later part of this project, and for your enthusiasm and ideas for the future. I am looking forward to continuing cooperation.

This research was financially supported by Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation as an industrial Ph.D. project together with Uppsala University and Elekta Instrument AB.

I also want to thank the research groups at both Uppsala University and Elekta. I have always felt welcome and inclusive wherever I have been. Thank you for a nice environment that contributes to interesting discussions and pleasant moments. And thanks to all the smart and fun people in the WASP community that I got the opportunity to know through courses, activities, and travel.

Last but not least I want to thank Malin for all her support, encouraging words, love, and kindness. And Ebba, my daughter, for all the unconditional love and happiness you give me.
List of Papers

This thesis is based on the following papers

I  Registration by tracking for sequential 2D MRI

II Learning a deformable registration pyramid

III Latent linear dynamics of spatiotemporal medical images
Contents

Notation xi

1 Introduction 1
  1.1 Background ................................................. 1
  1.2 Use case ................................................... 2
  1.3 Contribution ................................................ 2
  1.4 Outline ..................................................... 3
  1.5 Papers included in this thesis .............................. 3

2 Medical image registration 7
  2.1 An overview .................................................. 7
    2.1.1 Types of image registration problems .................. 8
  2.2 Transformation model ....................................... 10
    2.2.1 Parametric models ..................................... 10
    2.2.2 Non-parametric model .................................. 13
  2.3 Objective function ......................................... 15
    2.3.1 Similarity .............................................. 15
    2.3.2 Regularization ......................................... 19
  2.4 Optimization procedure ..................................... 20
    2.4.1 Iterative process ...................................... 20
    2.4.2 Hierarchical approach .................................. 22
  2.5 Resampling .................................................. 23
    2.5.1 Interpolation .......................................... 25
  2.6 Evaluation .................................................. 26
  2.7 Learnable approaches ....................................... 28
    2.7.1 Training process ....................................... 29
    2.7.2 Architecture ........................................... 31
    2.7.3 Implementation design ................................. 34
  2.8 Outlook ..................................................... 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Background</td>
<td>112</td>
</tr>
<tr>
<td>2.1 Variational auto-encoder</td>
<td>113</td>
</tr>
<tr>
<td>2.2 Linear Gaussian state-space model</td>
<td>113</td>
</tr>
<tr>
<td>2.3 Kalman variational auto-encoder</td>
<td>114</td>
</tr>
<tr>
<td>3 Method</td>
<td>114</td>
</tr>
<tr>
<td>3.1 Likelihood</td>
<td>116</td>
</tr>
<tr>
<td>4 Experiments</td>
<td>116</td>
</tr>
<tr>
<td>4.1 Dataset</td>
<td>116</td>
</tr>
<tr>
<td>4.2 Metrics</td>
<td>117</td>
</tr>
<tr>
<td>4.3 Implementation Details</td>
<td>117</td>
</tr>
<tr>
<td>4.4 Result</td>
<td>118</td>
</tr>
<tr>
<td>5 Discussion</td>
<td>120</td>
</tr>
</tbody>
</table>
This section provides a concise reference describing notation used throughout this thesis.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A^\top$</td>
<td>Transpose of matrix $A$</td>
</tr>
<tr>
<td>$A^+$</td>
<td>Moore-Penrose pseudoinverse of $A$</td>
</tr>
<tr>
<td>$\nabla \cdot$</td>
<td>Gradient of $\cdot$</td>
</tr>
<tr>
<td>$\nabla^2$</td>
<td>Laplace operator of $\cdot$</td>
</tr>
<tr>
<td>$\partial y/\partial x$</td>
<td>Partial derivative of $y$ with respect to $x$</td>
</tr>
<tr>
<td>$H$</td>
<td>The Hessian matrix</td>
</tr>
<tr>
<td>$B$</td>
<td>Estimate of the Hessian matrix</td>
</tr>
<tr>
<td>$\int_S f(x)dx$</td>
<td>Definite integral with respect to $x$ over the set $S$</td>
</tr>
<tr>
<td>$\text{Tr}$</td>
<td>Trace operator</td>
</tr>
<tr>
<td>$\text{det}$</td>
<td>Determinant</td>
</tr>
<tr>
<td>$*$</td>
<td>Convolutional operator</td>
</tr>
<tr>
<td>$*^l$</td>
<td>Dilated convolutional operator with dilated rate $l$</td>
</tr>
<tr>
<td>$\langle \cdot, \cdot \rangle$</td>
<td>Inner product</td>
</tr>
<tr>
<td>$| \cdot |_p$</td>
<td>$L^p$ norm of $\cdot$</td>
</tr>
<tr>
<td>$| \cdot |_2$</td>
<td>$L^2$ norm of $\cdot$</td>
</tr>
</tbody>
</table>

Probability and Information Theory:

$\text{p}(\cdot)$ | A probability distribution |
| $p_\alpha(\cdot)$ | A probability distribution given parameters $\alpha$ |
| $p(\cdot|x)$ | A probability distribution given variable $x$ |
| $\mathbb{E}_{x \sim p}[\cdot]$ or $\mathbb{E}_p[\cdot]$ | Expectation of $\cdot$ with respect to $p(x)$ |
| \text{ELBO} | Evidence lower bound |
| $D_{KL}$ | Kullback-Lieber divergence |
Image registration:

- \( J_M \): Moving image representation
- \( J_F \): Fixed image representation
- \( \mathcal{M} \): Domain of moving image
- \( \mathcal{F} \): Domain of fixed image
- \( \varphi \): Transformation function
- \( \varphi_{A \rightarrow B} \): Transformation function from domain \( A \) to \( B \)
- \( \varphi_{A} \): Affine registration
- \( J_M \circ \varphi \): Transform procedure
- \( u \): Displacement vector
- \( r \): Spatial position of an arbitrary pixel/voxel
- \( p \): Spatial position of a sparse pixel/voxel

Dynamic model:

- \( t \): Time point
- \( T \): Last time point of sequence
- \( y_t \): Observed variable at time \( t \)
- \( x_t \): Latent observed variable at time \( t \)
- \( z_t \): State variable at time \( t \)
- \( a_{1:t} \): Variables \( a \) from 1 to \( t \)
- \( a \): \( a_{1:T} \), Variables \( a \) from 1 to \( T \)
Chapter 1

Introduction

1.1 Background

Medical imaging is today an integral part of healthcare standards in many sectors. It facilitates analysis, treatment, and monitoring of medical conditions. Images from different time points and different sources are often used to analyze temporal changes and combine information between images. This requires getting the images into a shared space. The process of aligning two images of the same scene is called image registration. The goal of image registration is to find a spatial transformation that maps an image between different domains. This transformation can for example explain the motion of an image from one domain to another.

In modern imaging technology, most image processing modalities support some form of time-resolved image acquisition. Image registration of time-resolved images simply explains the dynamic of the anatomy. This is advantageous when identifying movements and temporal changes between different time points. For real-time systems, it is more likely to study motion detection than temporal changes of the subject.

Although many image registration methods perform well they are not suited for real-time scenarios. Either they are too simple, like rigid registrations which lack of flexibility, or the execution time is too long, like current commercial non-rigid deformable registration methods. Those methods are in general iterative with a slow convergence rate.

In this thesis, we want to investigate and develop motion models suitable for real-time medical imaging systems. To overcome the problem of conventional methods we have studied learnable methods using deep learning. For learnable methods, the procedure is separated into two parts, one off-line learnable part where the parameters in the model are learned, and one inference part. Since the learning is made during the off-line part the execution
CHAPTER 1. INTRODUCTION

time during inference is reduced. Deep learning methods rely on data. The learning is based on predefined data where the goal is to generalize well for unseen cases.

Furthermore, in some cases, the momentaneous movement between two different time steps in the sequence is not enough. Due to latency and missing samples, imputation and extrapolation of the motion are preferable. For this, we have studied the dynamics of high-dimensional medical image sequences, and dynamic representations is also be a part of this thesis.

1.2 Use case

An area of application of real-time image registration, or motion tracking, of particular interest to us, is real-time adaptive radiotherapy. Here, the goal is to adapt the treatment due to patient motion during treatment. Patient motion can be of several causes, e.g. respiratory motion and gastrointestinal motion. Reliable methods for motion estimation are therefore critical. With real-time image-guided radiotherapy, it would be possible to achieve treatments with higher accuracy thereby to a higher degree focusing the irradiation to the tumor and reducing side-effects in healthy tissues.

To support real-time adaptation, UMC Utrecht, Elekta AB (Stockholm, Sweden), and Philips (Best, The Netherlands) have developed the Elekta Unity [119, 146], see Figure 1.1. The Elekta Unity is an MR Linac, i.e. an integrated system consisting of an advanced linear accelerator (linac) with a 1.5T full-fledged, diagnostic quality, magnetic resonance imaging (MRI) scanner. Today, the Elekta Unity provides 2D MR imaging at 5Hz to monitor the treatment session. With this information, an overall goal is to include support for real-time adaption.

1.3 Contribution

The main contributions of this thesis are as follows:

- Paper I: an image registration method for sequential images based on a tracking framework and a sparse-to-dense interpolation scheme.

- Paper II: a learned image registration method based on a hierarchical coarse-to-fine architecture.

- Paper III: an unsupervised generative dynamic model for high dimensional medical image sequences.
CHAPTER 1. INTRODUCTION

Figure 1.1: The Elekta Unity, an integrated system of an advanced linear accelerator (linac) with a 1.5T full-fledged, diagnostic quality, magnetic resonance imaging (MRI) scanner.

1.4 Outline

This thesis starts with an introduction to image registration and dynamic modeling. Chapter 2 explains the concept of image registration. The reader will be introduced to the problem formulation, followed by the conventional iterative approach to solving the problem. The end of the chapter includes deep learning methods and shows how learnable approaches decrease the inference time in comparison with conventional methods.

Chapter 3 observes the dynamics of sequential medical images. The chapter starts with an introduction of two significant elements for dynamic modeling of high dimensional data, namely state-space modeling and dimensional reduction techniques. The last part of the chapter combines these elements and explains dynamics methods for high-dimensional dynamic systems.

Chapter 4 concludes the thesis and provide avenues for future work in the field.

1.5 Papers included in this thesis

Paper I: Registration by tracking for sequential 2D MRI


Summary: In this paper we present a tracking-based image registration method. For tracking purpose, discriminative correlation filters (DCD) are
used, and we interpolate the displacement field based on a learned sparse-to-dense interpolation network using normalized convolutional layers. The methods was evaluated on sequential 2D MRI images.

**Contribution:** The idea of using a DCF tracking framework originated from Jens Sjölund. The interpolation network, implementation, and writing were made by me with review by from Thomas Schön and Jens Sjölund.

**Paper II: Learning a deformable registration pyramid**


**Summary:** With inspiration from optical flow in computer vision we trained a coarse-to-fine network for the unsupervised/weakly supervised 3D/3D image registration problem. The network was trained on data from three different sources and we used a combined loss function of image similarities and structural information.

**Contribution:** This paper was a contribution to a learning-based image registration challenge[1]. When it comes to the architecture we used ideas from the computer vision optical flow community. The literature study, implementation, and writing were made by me, with support and review by Thomas Schön and Jens Sjölund.

**Paper III: Latent linear dynamics of spatiotemporal medical images**


**Summary:** Only based on sequential images, we present a generative method that extracts the dynamics of the underlying system. For this, we map the high-dimensional non-linear data to a lower-dimensional latent space using a conditional variational auto-encoder wherein the dynamic is modeled linearly. By reconstructing the images as a transformed version of a predefined moving image we estimate the likelihood of the displacement field. The method was evaluated on sequential cardiac ultrasound images.

**Contribution:** For this paper, we first try a native approach to generate sequential images directly, without the transformation estimate. This leads

[1]https://learn2reg.grand-challenge.org/
to poor and blurry reconstructions, but we realize potentials of the model. The literature study for the Kalman VAE was made by Jens Sjölund. The implementation, modification of the model, and writing were made by me, based on discussion and review from Thomas Schön, Jens Sjölund and Peter Kimstrand.
Chapter 2

Medical image registration

This chapter will present the basic theory of medical image registration. The chapter starts with an overview of the problem and introduces different components used to define the problem and find a solution. In the last part we will devote to how machine learning can be used to solve the problem using data.

2.1 An overview

A fundamental task in medical image processing is image registration. Medical image registration is widely used in areas such as: atlas-based segmentation [73], treatment planning and replanning [15], patient positioning [77, 147], and dose accumulation [3, 140]. The main purpose of medical image registration is to transfer functional and anatomical information from one image to another where the images can differ in time, viewpoints, image modality and/or subject.

The key idea is to define a geometric transformation \( \varphi : \mathcal{M} \rightarrow \mathbb{R}^d \) between a moving image \( J_M : \mathcal{M} \subset \mathbb{R}^d \rightarrow \mathbb{R} \), and a fixed image \( J_F : \mathcal{F} \subset \mathbb{R}^d \rightarrow \mathbb{R} \) such that, roughly speaking,

\[
J_F(r) \approx J_M(\varphi(r)). \tag{2.1}
\]

For every point \( r \in \mathcal{M} \) the geometric transformation \( \varphi(r) \) defines a mapping from the moving image to the target image. Without loss of generality this transformation can be expressed defined as

\[
\varphi(r) = r + u(r), \tag{2.2}
\]

where \( u(r) \) is a displacement vector for every point \( r \in \mathcal{M} \). Figure 2.1 shows an example of an image registration.
Figure 2.1: Before (left) and after (right) an affine registration (Section 2.2.1) of a fixed CT scan and a moving MR scan of the brain. In this example we used mutual information loss (Section 2.3.1), linear interpolation (Section 2.5.1.2) and gradient descent optimization (Section 2.4). This example was produced using Simple ITK toolbox [149].

An image registration problem is, in general, formalized as an optimization problem where the optimal displacement vector can be found by minimizing an energy functional $E$,

$$\begin{align}
\text{minimize} & \quad E(u; J_F, J_M) \\
\text{subject to} & \quad u \in \mathcal{U},
\end{align}$$

(2.3a)

(2.3b)

where $\mathcal{U}$ is the set of feasible transformations. Unfortunately, this is in general an ill-posed problem that includes implausible solutions. The aim is therefore to find a realistic transformation.

The conventional approach to solve (2.3) is to use an iterative optimization procedure, shown in Figure 2.2. The problem contains three main components [130]: the transformation model $u \in \mathcal{U}$ (Section 2.2), the objective function $E$ (Section 2.3) and the optimization procedure (Section 2.4). In the following sections, these components will be explained in detail followed by two extra components worth mentioning: resampling (Section 2.5) and evaluation metrics (Section 2.6). In the last section of this chapter (Section 2.7), we will show how deep learning, in combination with techniques from conventional methods, can be used to reduce the computational time for image registration problems. But first, we survey the most common types of image registration.

### 2.1.1 Types of image registration problems

#### Intersubject image registration

Image registration between different subjects, or patients, is called intersubject image registration. Anatomy in different subjects can differ and therefore
constitutes the biggest challenge for intersubject image registration. Examples of usage of intersubject image registration are: atlas-based segmentation, where the geometric transformation is used to map segmented regions from a labeled image to a new; and studying anatomical variability across subjects by transforming them into a common space.

**Intrasubject image registration**

Intrasubject image registration, conversely, refers to registrations between images of the same subject. Although the images are of the same subject, its state may have changed since the scans have been acquired at different time points. Some of the main causes of changes are respiratory and cardiac motions, bowel movements, and weight changes. One application for intrasubject registration is real-time image registration. This poses a major challenge as computationally efficient demands on the implementation of the method.

**Multimodal image registration**

Multimodal image registration is image registration where the images are from different modalities. The most common modalities within medical images are: computed tomography (CT), magnetic resonance imaging (MRI), positron emission tomography (PET), and ultrasound (US). Multimodal image registration is used to combine useful information from the

---

**Figure 2.2: Iterative process for solving image registration problem.**

```
input: J_F

Similarity measure (Section 2.6)

J_M \circ \varphi_n

Resampler (Section 2.5)

Optimizer (Section 2.3 & 2.4)

Model \varphi_n (Section 2.2)

n = n + 1

until convergence
```
different modalities. The major challenge for multimodal registration is to compare the tissue and organ appearance are depicted differently for different modalities [154].

Monomodal image registration

The opposite to multimodal image registration is monomodal, where the two images are given by the same image modality. Examples of monomodal image registrations are real-time tracking and atlas-based segmentation.

Image pair dimensionality

The last aspect we want to mention is the image pair dimensionality. The image pair dimension can either be 3D/3D, 2D /3D, 3D/2D, or 2D/2D. The main challenge for 2D/2D image registration problems is a limited field-of-view and difficulties in handling movements out of the 2D plane. On the other hand, 3D/3D image registration is computational heavy which results in a long execution time [154].

2.2 Transformation model

The choice of transformation model is a trade-off between computational efficiency and model flexibility. This trade-off is related to the number of parameters in the model and can vary from six, in a simple global rigid transformation, to millions [130]. Recall from equation (2.2) that the transformation model can be described by the displacement vector $u(r)$ at every point $r \in \mathcal{M}$. The model of the displacement vector is commonly categorized into parametric models and non-parametric models. The explained models in this section, except the first case of parametric models, can estimate complex deformations and are in literature defined as deformable image registration methods (DIR) [130, 144].

2.2.1 Parametric models

The general form for a parametric model is given by,

$$ u(r; \theta) = \sum_i \theta_i B_i(r), \quad (2.4) $$

where $\theta_i \in \mathbb{R}^d$ is a model parameter and $B_i : \mathcal{B} \to \mathbb{R}^d$ some basis function.

The three most popular parametric model in medical image registration: Linear registration, Thin Plate Spline (TPS) and Free Form Deformation (FFD). The first is the simplest class of image registrations where a few
numbers of parameters are used to estimate the transformation of the entire image. The last two are based on a sparse representation where spline interpolation techniques are used to estimate a dense transformation.

**Linear registration**

Linear image registration, also called affine image registration, includes translation, rotation, scaling, and skewing of the moving image. The linear registration is given by,

\[ u(r; A, b) = Ar + b, \]  

(2.5)

where \( A \) is a matrix and \( b \) is a vector. The linear parametric model is simple with few parameters. However, the transformation is uniform over the entire image and the usage is limited. Linear registration methods are mainly used in two cases: registration of rigid structures, such as registration of bones, and as preregistration in more complex cases (more of this in Section 2.4). Linear registration can also be used in combination with more advanced methods, which is something we exploit in paper II. In Example 2.2.1 we show how a 2D linear registration can be expressed as (2.4), with basis functions and parameters.

**Example 2.2.1. Affine transformation in 2D**

The affine registration in 2D is given by

\[ u(r; A, b) = Ar + b = \begin{pmatrix} a_1 & -a_2 \\ a_2 & a_1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} a_1 x - a_2 y + b_x \\ a_2 y + a_1 x + b_y \end{pmatrix}. \]  

(2.6)

The basis functions for linear registrations is therefore,

\[ B_1 = \begin{pmatrix} x \\ y \end{pmatrix}, \quad B_2 = \begin{pmatrix} -y \\ x \end{pmatrix}, \quad B_3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad B_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \]  

(2.7)

and the parameters are given by

\[ \theta_1 = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, \quad \theta_2 = \begin{pmatrix} a_2 \\ a_1 \end{pmatrix}, \quad \theta_3 = \begin{pmatrix} b_x \\ 0 \end{pmatrix}, \quad \theta_4 = \begin{pmatrix} 0 \\ b_y \end{pmatrix}. \]  

(2.8)

**Thin plate spline**

Unlike linear registration, thin plate splines (TPS) can produce non-rigid transformations. TPS is a spline method based on radial basis functions (RBF). Given a sparse sets of points, \( \{p_i\}_{i=1}^N \), called control points, and
a set of mapping coefficients, \( \{ w_i \}^N_{i=1} \), the displacement vector is estimated

\[
    u(r; p, w, A, b) = u(r; A, b) + \sum_{i=1}^{N} w_i \Phi(||r - p_i||^2), \tag{2.9}
\]

\[
    \Phi(r) = r^2 \log(r^2) \tag{2.10}
\]

is the logarithmic basis function and \( u(r; A, b) \) is a linear transformation. TPS models are robust and generates smooth displacements. The control points can be placed arbitrarily. It is therefore advantageous to place them on characteristic features in the image, called landmarks. However, TPS is both computational heavy and every sparse point has global influence. It is therefore difficult to simulate local deformations and the entire result relies on the measurement of each sparse point.

**Free Form Deformation**

Free form deformation (FFD) \[124\] is another spline-based model. In FFD the control points \( p \) are located on a grid \((n_x \times n_y \times n_z)\) in the moving image. Similar to TPS, the displacement for each point is estimated with linear registration. The dense displacement field is then estimated using 1D cubic B-spline interpolation. B-spline interpolation is a non-linear interpolation technique. In FFD, the displacement is given by

\[
    u(r; p) = \sum_{k=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_k(\nu_x)B_m(\nu_y)B_n(\nu_z)p_{i+k,j+m,l+n} \tag{2.11}
\]

where \( i = \lfloor x/n_x \rfloor - 1, j = \lfloor y/n_y \rfloor - 1, l = \lfloor z/n_z \rfloor - 1 \). The B-spline coordinates are given by

\[
    \nu_x = x/n_x - \lfloor x/n_x \rfloor, \tag{2.12a}
\]
\[
    \nu_y = y/n_y - \lfloor y/n_y \rfloor, \tag{2.12b}
\]
\[
    \nu_z = z/n_z - \lfloor z/n_z \rfloor, \tag{2.12c}
\]

and the B-spline functions are defined as

\[
    B_0(\nu) = (1 - \nu^3)/6, \tag{2.13a}
\]
\[
    B_1(\nu) = (3\nu^3 - 6\nu^2 + 4)/6, \tag{2.13b}
\]
\[
    B_2(\nu) = (-3\nu^3 + 3\nu^2 + 3\nu + 1)/6, \tag{2.13c}
\]
\[
    B_3(\nu) = \nu^3/6. \tag{2.13d}
\]

The deformation for FFD is only influenced by its nearest control points. FFD can therefore generate more complex local deformations than TPS.
2.2.2 Non-parametric model

For the non-parametric model, the geometric transformation is represented as a deformation vector \( u(r) \) for each position \( r \) and it is not limited to a sparse representation, as in the parametric framework. Since non-parametric methods estimate a vector in each position they can generate very complex deformations. However, the downside is that the estimated displacement field may lack physical significance and be unrealistic. Realistic transformations in medical image registrations are when the anatomical topology is preserved after transformation. It places certain demands on the transformation, for example, smoothness, inverse consistency, and differentiable.

Many non-parametric models are derived from physical models \[108\], such as elastic \[16\], fluid \[24, 14\], diffusion \[11\] and curvature \[40\]. The physical constraints of the model is often handled with a regularization term in the energy functional. Different regularization will be discussed in section 2.3.2.

Below we will describe two models. The first is a popular diffusion-based non-parametric model, namely the Demons algorithm \[135, 136\]. We use the Demons as a baseline in papers I and III. The second is larger deformation diffeomorphic metric mapping (LDDMM) \[11\], a topology-preserving flow-based method.

Demons

The Demons model \[135, 136\] is based on optical flow methodology and consider the images, \( J_F \) and \( J_M \), as two images from a time sequence. It assumes that the intensity of the image sequence does not change between consecutive frames, i.e.,

\[
J(r, t) = J(r + \delta r, t + \delta t).
\]  

(2.14)

This equation is called the optical flow equation. Taylor expanding the right-hand side yields

\[
J(r + \delta r, t + \delta t) = J(r, t) + \nabla_r J(r, t)^\top \delta r + J_t(r, t)\delta t + \mathcal{O}^2,
\]  

(2.15)

where \( J_t = \partial J(r, t)/\partial t \). Neglecting the higher order terms (2.14) becomes

\[
\nabla J(r, t)^\top v + J_t(r, t) = 0,
\]  

(2.16)

where \( v = \partial r/\partial t \) defines the motion. The solution to (2.16) is not unique, but one solution is given by

\[
v(r, t) = -\frac{J_t(r, t)}{||\nabla J(r, t)||^2} \nabla J(r, t).
\]  

(2.17)
This solution gives the Demons model

\[
u(r) = \frac{J_F(r) - J_M(r + u(r))}{||\nabla J_F(r)||^2} \nabla J_F(r), \tag{2.18}
\]

where \( \delta r = (r + u(r)) - r = u(r) \) and \( J_t(r) = (J_M(r + u(r)) - J_F(r))/\delta t \). Due to instability for small changes in \( J_F \) a more stable version of the Demons model is given by

\[
u(r) = \frac{J_F(r) - J_M(r + u(r))}{||\nabla J_F(r)||^2 + (J_F(r) - J_M(r + u(r)))^2} \nabla J_F(r). \tag{2.19}
\]

The Demon’s algorithm updates \( u \) iteratively by estimating the gradient \( \delta u \) in (2.19), also called demon force. Due to non-unique solutions the result is regularized with a Gaussian kernel \( g_\sigma \). The entire procedure is shown Algorithm 1. This might look different from the general procedure for iterative optimization and regularization that we will discuss later, but Pennec, Cachier and Ayache [116] shows a relationship between the Demon’s algorithm and the second-order gradient descent optimization of a SSD similarity loss with fluid regularization. More about objective functions and iterative optimization procedure are given in Section 2.3 and 2.4, respectively. Advantages of the Demon’s algorithm are that: (i) it has linear computational complexity, (ii) it is easy to implement, and (iii) suitable for parallel architectures, like GPUs. However, it is sensitive to noise and only suitable for monomodal image registration problem, due to the comparison of intensities.

**Algorithm 1** Demon’s algorithm

1: procedure DEMONS\((J_F, J_M)\)
2: repeat
3: Estimate \( \delta u \) from (2.19)
4: Update \( u \leftarrow u + \delta u \)
5: Regularize \( u \leftarrow u * g_\sigma \)
6: until convergence
7: return \( u \)
8: end procedure

**LDDMM**

For many image registration model the topology is preserved only for small deformations. One popular flow-based model that preserves the topology for larger deformations is larger deformation diffeomorphic metric mapping (LDDMM) [11]. To overcome the problem for larger deformations, LDDMM
models the transformation $\varphi(r)$ as a flow, $v_t$ of smaller diffeomorphic transformation, $\varphi_t$ over time $t \in [0, 1]$ such that $v_t = \partial \varphi_t / \partial t$. The transformation is then given by

$$\varphi = \varphi_0 + \int_0^1 v_t(\varphi) dt,$$

(2.20)

where $\varphi_0 = r$ is the initial value and the goal is to find the optimal trajectory $\varphi_t$. The trajectory is modelled as a piecewise linear, geodesic distance. LDDMM is said to be diffeomorphic. The transformation for diffeomorphic image registration models is invertible, and both the transformation and its inverse are differentiable. In diffeomorphic models the topology is preserved.

### 2.3 Objective function

The aim in image registration is to find a realistic geometric transformation $\varphi$ that maximizes similarity, or minimizing dissimilarity, between the fixed image $J_F$ and the transformed moving image, $J_M \circ \varphi$. Even though the formulation of an image registration problem is relatively simple, the problem is ill-posed by its nature and the solution is not unique. To favor probable solutions and disadvantage unrealistic ones, regularization is used. Regularization adds prior knowledge about transformation properties to the energy functional. The total energy functional $E$ becomes

$$E(u; J_F, J_M) = D(u; J_F, J_M) + \lambda R(u),$$

(2.21)

where $D$ denotes a dissimilarity between the fixed and the moving image, and $R$ is a regularization term, weighted by a scalar $\lambda$. The entire energy functional is trade-off between the intensity and structural similarities in the images and constraints of the transformation. In Section 2.3.1 we will discuss different types of similarities. The regularization term is described in Section 2.3.2.

#### 2.3.1 Similarity

The similarity depends on both intensity and structural similarities in the images. If only intensity values are present, the similarity is typically intensity-based. Including other information, such as corresponding points in the two images, or segmented regions, the similarity can be feature-based, or a combination of both. Below we define the most common intensity and feature-based similarity used for image registration problems. But first, define the $\Omega$ as the domain where the transformation obtains, $J_F, J_M : \Omega \to \mathbb{R}$, $u : \Omega \to \mathbb{R}^d$. 
CHAPTER 2. MEDICAL IMAGE REGISTRATION

Intensity-based

The intensity-based similarity makes use of the intensity distribution of the fixed and moving image to find statistical correlations between the images. We will point out four different types of intensity-based similarity loss: (i) the sum of squared difference (SSD); (ii) the normalized correlation coefficient (NCC); (iii) the mutual information (MI) \[26, 141, 131]\; and (iv) the normalized gradient field (NGF) \[59, 60]\. The first two are based on intensity differences between the images and therefore preferred for monomodal image registration tasks. The latter two, MI and NGF, works for multimodal image registration. Mutual information uses information theory to measure similarity between the images while NGF uses edge similarity of the image gradients.

1. **Sum of squared differences (SSD)**: The intensity difference between the images are measured. It assumes that the intensity distribution of the two images is identical and is therefore best suited for mono-modal image registration. The SSD is given by

\[
D_{\text{SSD}}(u; J_F, J_M) = \frac{1}{|\Omega|} \int_\Omega (J_F(r) - J_M(r + u(r)))^2 dr, \quad (2.22)
\]

where a lower value of $D_{\text{SSD}}$ indicates higher similarity.

2. **Normalized Correlation coefficient (NCC)**: Measures the intensity correlation between the images. It assumes a linear relation between the intensities of the corresponding structure in the images and are therefore best suited for mono-modal image registration, or registrations between two images of similar modality, like CT to CBCT \[90]\. Given the cross-correlation (CC)

\[
CC(u; J_F, J_M) = \langle J_F, J_M \circ \varphi \rangle = \int_\Omega J_F(r)J_M(r + u(r))dr, \quad (2.23)
\]

the NCC is given by

\[
\text{NCC}(u; J_F, J_M) = \frac{CC(u; J_F, J_M)}{\langle J_F, J_F \rangle \langle J_M \circ \varphi, J_M \circ \varphi \rangle}. \quad (2.24)
\]

Larger NCC indicates higher similarity and the corresponding similarity loss is given by

\[
D_{\text{NCC}}(u; J_F, J_M) = 1 - \text{NCC}(u; J_F, J_M)^2. \quad (2.25)
\]

3. **Mutual Information (MI)**: Mutual information measures statistical similarities between two random signals. For images, the mutual information between intensities of two images, $J_F$ and $J_M \circ \varphi$ is defined
by

\[ I(\varphi, J_F, J_M) = H(J_F) + H(J_M \circ \varphi) - H(J_F, J_M \circ \varphi), \quad (2.26) \]

where \( H(J_A) \) is the entropy of the intensity \( J_A \) and \( H(J_A, J_B) \) is the joint entropy of \( J_A \) and \( J_B \). Those entropy are defined \[126]\n
\[ H(J_A) = \int_{\mathbb{R}} p_A \log(p_A)da, \quad (2.27a) \]
\[ H(J_A, J_B) = \int_{\mathbb{R}^2} p_{A,B} \log(p_{A,B})dadb. \quad (2.27b) \]

where \( p_A \) is the marginal probability function of the intensities in image \( J_A \) and \( p_{A,B} \) is the joint PDF of the intensities in \( J_A \) and \( J_B \). Mutual information can handle non-linear intensity relationships and is preferable for multimodal image registration. Mutual information measures the similarity between the two images. The dissimilarity loss is, therefore,

\[ D_{MI}(u; J_F, J_M) = -MI(u; J_F, J_M). \quad (2.28) \]

4. **Normalized gradient field (NGF):** NGF \[59\] uses the intensity changes of the image intensity to identify similarities of the edges. Since the magnitude of gradient depends on the image modality the gradient field is normalized, and the loss is defined as

\[ D_{NFG}(u; J_F, J_M) = \int_{\Omega} 1 - (\tilde{\nabla}(J_F(r))^T \tilde{\nabla}(J_M(r + u(r)))^2 dr, \quad (2.29) \]

where

\[ \tilde{\nabla}(J_A(r)) = \frac{\nabla J_A(r)}{||\nabla J_A(r)|| \epsilon_A}, \quad (2.30) \]

and \( ||\nabla J_A(r)|| \epsilon_A = \sqrt{\nabla J_A(r)^T \nabla J_A(r) + \epsilon_A^2} \) for some \( \epsilon_A > 0 \) to handle zero division when the gradient is zero. The value of \( \epsilon \) should be large enough to limit noise from the image gradients and small enough to capture edges. An automatic choice of \( \epsilon_A \) for an image \( J_A \) is given in \[5\]

\[ \epsilon_A = \frac{\eta}{|\Omega_A|} \int_{\Omega_A} ||\nabla J_A(r)|| dr \quad (2.31) \]

where \( \eta \) is the noise level of the image \( J_A \).
CHAPTER 2. MEDICAL IMAGE REGISTRATION

Feature-based

While intensity-based similarity compares intensities directly, feature-based similarity measures correspondences between features in the images, such as points, lines, and contours. Feature-based registrations apply to both monomodal and multimodal image registration problems. Feature-based image registrations aim to minimize the distance between the corresponding features in the images. In this thesis, we will only explain similarity based on the Euclidean distance between features. Other distances exist [127, 137, 155, 17] but will not be covered here.

There are main two categories of feature-based registrations: landmark-based and surface-based. Landmark-based image registration is based on corresponding points in the moving and fixed image while surface-based matches the corresponding surfaces in the two images. Both points and surfaces can either be manually or automatically detected. Manual detection is accurate but relies on medical expertise and is time-consuming. For automatical detection, the methods require some algorithm to detect and match key points in the images. Keypoints are typically defined as landmarks or edges and corners in the images. Popular algorithms to detect and match key points in landmark-based image registrations are Scale-invariant feature transform (SIFT) [9] and Speeded up robust features (SURF) [100]. The displacement of the key points can then be obtained by minimizing the Euclidean distance between them, also called fiducial registration error (FRE). Let $p = \{p_i\}^N_{i=1}$ and $p^F = \{p_i^F\}^N_{i=1}$ be corresponding points in the moving and fixed image, respectively. The FRE$^2$ is then given by

$$D_{FRE}^2(u; p^F, p) = \frac{1}{N} \sum_{i=1}^{N} w_i^2 \| (p_i^M + u(p_i^M) - p_i^F) \|^2,$$  \hfill (2.32)

where $w_i$ is a non-negative weighting factor for each keypoint.

In surface-based image registration the keypoints are a set of points, called clouds. Let us call a cloud in the fixed image $S^F = \{p_i^F\}^N_{i=1}$ and the corresponding cloud in the moving image $S^M = \{p_i^M\}^N_{i=1}$. A similarity measurement is then given by the disparity function,

$$D_{disp}(u, S^F, S^M) = \sum_{i=1}^{N} w_i^2 \| p_i^M + u(p_i^M) - C(p_i^M, S^F) \|^2,$$  \hfill (2.33)

where $C(p_i^M, S^F)$ is a correspondence function that finds a point in $S^F$ that corresponds to $p_i^M$ in $S^M$. Two popular algorithms for the correspondence function are Iterative closest points (ICP) [12] and Random sample consensus (RANSAC) [43]. Given the sparse displacements $\{u(p_i^M)\}^N_{i=1}$ the
entire displacement field can be estimated using some of the parametric models explain in Section 2.2.1.

For sequential imaging, tracking several features can be used to estimate the sparse displacements. This approach was used in Paper I, where we tracked an arbitrary number of regions using discriminative correlation filters (DCF) [30].

2.3.2 Regularization

The ill-posedness of the image registration problem induces many local solutions to the optimization problem. To suppress unwanted local minima, a regularization term is added to the objective function. The regularisation’s role is to constraint the transformation to realistic transformation by penalize implausible ones. To encourage smoothness, many regularization terms involve the $L_2$-norm of the displacement derivatives. The first example of regularization term is diffusion, which penalizes first-order derivative of the displacement

$$R^D(u) = \frac{1}{2} \int_\Omega \sum_{i=1}^d ||\nabla u_i(r)||^2 dr.$$  \hspace{1cm} (2.34)

Another regularization is curvature, which instead penalizes the second-order derivative of the displacement,

$$R^C(u) = \frac{1}{2} \int_\Omega \sum_{i=1}^d (\nabla^2 u_i(r))^2 dr,$$  \hspace{1cm} (2.35)

where $\nabla^2$ is the Laplacian operator. In contrast to diffusion, curvature does not penalize affine deformations. Two other regularization are elastic body [16] and viscous flow regularization [23]. These two originate from physical and fluid interpretations of the problem. Both regularization terms can be written as

$$R^{E,F}(\cdot) = \frac{1}{2} \int_\Omega \mu \langle \nabla \cdot, \nabla \cdot \rangle + (\lambda + \mu)(\nabla \cdot)^2 dr$$  \hspace{1cm} (2.36)

where $\mu$ and $\lambda$ are Lamé constants and $\langle \cdot, \cdot \rangle$ the inner product. Elastic registrations operate on the displacement vector $u$ while viscous flow is applied on the velocity field $v = \delta u + u^T \nabla u$. The main difference between them are that elastic only handles small deformations while viscous fluid support larger deformations and are therefore more practical to use [23].

For LDDMM, the optimal trajectory can be finding by minimizing the energy function (2.21) with a geodesic distance-based regularizer,

$$R^{LDDMM}(v_t) = \int_0^1 ||v_t||_V^2 dt,$$  \hspace{1cm} (2.37)
where $||\cdot||_V$ is the norm of space $V$ such that $||v_t||_V = ||Lv_t||_{L2}$ for any differentiable operator $L$. However, integration over time is both computationally expensive and requires a lot of memory. It therefore places high demands on the optimization method.

### 2.4 Optimization procedure

In the previous section we defined the energy functional $E$. The goal now is to minimize the $E$ to obtain the optimal solution for the image registration problem,

$$u^* = \arg \min_u E(u; J_F, J_M).$$

This problem is nonconvex and has no closed-form solution. Therefore, the solution is often given by an iterative optimization procedure. In the first part of this section, we will cover the most common iterative optimization methods for image registration. To improve robustness and avoid local minima the image registration problem is often solved using a hierarchical approach, which is explained and discussed in the second part of this section.

#### 2.4.1 Iterative process

Depending on the nature of the variables, the iterative optimization process can be separated into two categories: continuous and discrete. In the continuous case the displacement is iteratively updated based on the gradient $\nabla E$ of the energy functional.

$$u_{i+1} = u_i - \tau f(\nabla E)$$

using step length $\tau$ and some function $f(\cdot)$. Examples of gradient based optimization methods for image registrations are: Gradient Descent (GD) \[124\], Gauss-Newton (GN) \[6\], Levenberg-Marquard (LM) \[134, 71\] and Quasi-Newton (QN) \[101\].

The simplest method of the four is gradient descent. Gradient descent is a first-order optimization method where the steps are taken in the opposite direction of the energy functional gradient, i.e.

$$f^{GD}(\nabla E) = \nabla E.$$ 

Even though gradient descent is simple, it is numerical unstable and required a small step length $\tau$. Small step length results in more iterations and longer convergence time. An example of gradient descent using SSD and Diffusion loss is shown in Example 2.4.1. The next optimization method,
Gauss-Newton, is a second-order non-linear least squares method where the Hessian is approximated from the Jacobian \( J_E \). The Gauss-Newton method is given by

\[
f_{\text{GN}}(\nabla E) = (J_E^T J_E)^{-1} \nabla E. \tag{2.41}
\]

Gauss-Newton is more numerically stable than gradient descent and can therefore use longer time steps \( \tau \). However, for computational efficiency, the Jacobian \( J_E \) has to be either sparse or small. Levenberg-Marquard is a mixture of gradient descent and Gauss-Newton. Levenberg-Marquard is given by,

\[
f_{\text{LM}}(\nabla E) = (\eta + J_E^T J_E)^{-1} \nabla E, \tag{2.42}
\]

where \( \eta \), called a damping factor, controls the impact of the two. In cases when the inverse of the Hessian is too complicated to compute the Hessian can be approximated with a positive definite matrix \( B \). This gives us the last example of continuous optimization methods, the Quasi-Newton method,

\[
f_{\text{QN}}(\nabla E) = B^{-1} \nabla E. \tag{2.43}
\]

where \( B \) is an estimate of the Hessian matrix \( H \). The Hessian matrix can be approximated using e.g., Davidon-Fletcher-Powell (DFP) or Broyden-Fletcher-Goldfard-Shanno (BFGS). Other gradient-based optimizations methods used in medical image registrations are: conjugate gradient, Powell’s conjugate directions, and stochastic gradient descent.

**Example 2.4.1.** Gradient Descent with SSD and Diffusion loss

Given

\[ E(u; J_F, J_M) = D_{\text{SSD}}(u; J_F, J_M) + \lambda R_D(u) \]

then

\[
\nabla D(u_i) = -(J_F - J_M \circ \varphi) \nabla (J_M \circ \varphi) \\
\nabla R(u_i) = -\Delta u
\]

In discrete optimization, the variables take discrete values, and the optimization is formulated as a discrete labeling problem. The optimization procedure assigns a label to each discrete variable to minimize the energy functional. The general iterative procedure for discrete optimization can be written

\[
u_{i+1} = u_i + g(E), \tag{2.44}
\]
using some function \( g(\cdot) \). Three classes of discrete optimization methods are: graph-based methods; message passing methods; and linear programming. A common approach is to formulate the problem as a Markov random field (MRF). MRF formulate to problem as an undirected graph \( \mathcal{G} = \{ \mathcal{V}, \mathcal{E} \} \) where each the node \( v \in \mathcal{V} \) can be presented as control point and the edges \( \mathcal{E} \) defines the neighborhood of each node \[51\]. The goal is then to assign a discrete label \( l_v \) to each node which corresponds to the displacement vector. We will not discuss discrete optimization methods for medical image registration but refer to Glocker \[52\] for further reading.

### 2.4.2 Hierarchical approach

The optimization procedure for complex image registration models is computationally heavy and the optimization problem includes local minima, due to the ill-posedness of the problem, so the performance relies on a good initial displacement. A common approach to simplify the procedure is the so-called hierarchical approach, which can either be model-based or multi-resolution-based. For model-based hierarchical approaches, the idea is to start with a simple model and gradually increase the complexity. The initial displacement for each model is given by the result of the previous. Multi-resolution downsamples the fixed and moving images into several lower-dimensional representations where the displacement is first estimated in the coarsest level and used as an initial guess in the finer and so on until the original resolution is reached. This is also referred to as a coarse-to-fine or pyramidal approach. One of the largest problems for image registration is estimating larger deformations. Using a pyramidal approach, large deformations can be obtained in lower resolutions and be propagated to finer levels. To avoid aliasing, i.e. sampling at a rate that is insufficient to capture changes in the images, the high frequency of the images is often reduced using low-pass smoothing. Two popular techniques for downsampling the images are using a Gaussian or Laplacian pyramid \[1\]. The Gaussian pyramid is a blur and downsamples operator. At each level, the image is convolved with a low-pass filter, e.g., a binomial filter, and then subsampled by a factor of two to reduce the dimension by half. Writing these operators in matrix form yields

\[
J_G^{(k+1)} = D^{(k)} B^{(k)} J_G^{(k)} = G^{(k)} J_G^{(k)},
\]

where \( D^{(k)} \) is the downsampling operator, \( B^{(k)} \) is the low pass filter convolution pass filter and \( G^{(k)} \) is the blur-and-downsample operator at level \( k \). We denote the representation of the blurred and downsampled image as \( J_G^{G} \). The Laplacian pyramid represents the difference of what is present between two levels of the Gaussian pyramid. This can be represented as

\[
J_L^{(k)} = J_G^{(k)} - F^{(k)} J_G^{(k+1)} = (I^{(k)} - F^{(k)} G^{(k)}) J_G^{(k)},
\]
where \( F^{(k)} = B^{(k)}U^{(k)} \) is the upsampling and blur operator of the lower dimensional Gaussian pyramid image \( J_{G}^{(k+1)} \). An illustration of a Gaussian pyramid is shown in Figure 2.3. In addition to being more robust hierarchical, approaches also converge faster regardless of several optimization processes [133].

Figure 2.3: A pyramid representation is a hierarchical representation of the images.

2.5 Resampling

Because images are represented on grids, we must be extra careful when transferring points from one image domain to another. For a transformation of a single point \( r \) there are no guarantees that the new position will be located on a grid point, so interpolation is required. The resampling procedure, therefore, plays an important role in image registration. The resampling procedure can either use forward or backward mapping. Forward mapping transfers every single point in the original image, \( J_{A} \), to its new coordinate

\[
J_{B}(r + u(r)) = J_{A}(r), \quad \forall r \in \mathcal{A}
\]  

(2.47)

to create the new image \( J_{B} \). Even though this approach is straightforward, it has two major disadvantages. The first one is gaps: it is possible that some positions in the new image are not assigned a value and so will remain unknown. The second disadvantage is overlap, which occurs when a position in the new images assigns multiple values from different positions in the original image and you must decide the value assigned in that position. An illustration of forwarding mapping, including problems with gaps and overlaps, is shown in Figure 2.4(a).

Backward mapping overcomes the problem with forwarding mapping. Backward mapping assigns values by iterating overall positions in the new
image where the value assignment is based on the inverse transformation,

\[ J_B(r) = J_A(r - u(r)), \quad \forall r \in \mathcal{B}. \]  

A backward mapping implies that all positions in the new image are assigned an intensity value such that the new image has neither gaps nor overlaps. However, since the inverse transformation likely ends up on a non-grid location, the intensity needs to be estimated using interpolation. An illustration of backward mapping is shown in Figure 2.4(b).

(a) Forward resampling has two main disadvantages: gaps (gray) and overlaps (blue).

(b) Backward resampling uses the inverse transformation and interpolation techniques to estimate the intensity in given a position.

Figure 2.4: Differences between forward resampling (a) and backward resampling (b).
2.5.1 Interpolation

Interpolation affects the images and therefore impacts the optimization procedure. In particular, the choice of interpolation method affects the smoothness of the objective function, leading to a trade-off between accuracy and speed in the optimization procedure. A common choice is to use a simple and fast interpolation technique for optimization and a more accurate for evaluation \[113\]. In general form, the interpolation can be described as a convolution of the discrete image with a continuous filter response \( h(p) \) \[95\].

\[
J(\tilde{p}) = \sum_{p \in \Omega} J[p] h_d(p - \tilde{p}), \tag{2.49}
\]

where computational complexity is reduced using symmetric and separable interpolation kernels \( h_d(p) = \prod_{i \in \{1,...,d\}} h(p^{(i)}) \). Some of the most common interpolation techniques are the first 3 order of b-spline, i.e.: (i) nearest neighbor, (ii) linear and (iii) cubic interpolation. For a more detailed analysis of interpolation methods in medical image processing, we refer to Lehmann, Gonner and Spitzer \[95\]. Before going into details, we will start with some definitions. Let \( p \) be a non-grid located point and \( \{p_i\}_{i=1}^{2d} \) be the surrounding grid located points. Then \( \Delta p \) is the distance between \( p \) and the lowest grid located point in every dimension and \( s \) is the spacing between the grid points. An 2D illustration of this is given in Figure 2.5.

![Figure 2.5: Interpolation of point \( \tilde{p} = (\tilde{x}, \tilde{y}) \) given the four surrounding grid points \( \{(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)\} \). The spacing is given by \( s = (s_x, s_y) \).](image-url)
CHAPTER 2. MEDICAL IMAGE REGISTRATION

2.5.1.1 Nearest neighbor interpolation

The simplest interpolation method is nearest neighbor interpolation, which is given by

\[ h^N(p) = \begin{cases} 
1, & 0 \leq |p| < 0.5 \\
0, & \text{otherwise} 
\end{cases}, \quad (2.50) \]

for each dimension. Although the nearest neighbor interpolation is simple, it should be avoided in most cases for resampling. Since the result is in general poor, and includes artifacts.

2.5.1.2 Linear interpolation

Another simple interpolation technique is linear interpolation, which creates a continuous curve that is better suited for image registration. Linear interpolation is based on the \(2^d\) closest grid points and its filter response is given by

\[ h^L(p) = \begin{cases} 
1 - |p|, & |p| \leq 1 \\
0, & \text{otherwise} 
\end{cases}. \quad (2.51) \]

2.5.1.3 Cubic interpolation

The last interpolation technique we want to mention is the cubic interpolation method. The cubic interpolation curve is continuous and differentiable. Cubic interpolation uses the \(4^d\) closest points to fit a piecewise cubic polynomial. The filter response for cubic interpolation is given by

\[ h^C(p) = \begin{cases} 
(a + 2)|p|^3 - (a + 3)|p|^2 + 1, & |p| < 1 \\
0, & 1 \leq |p| < 2 \\
(a + 3)|p|^3 - 5a|p|^2 + 8a|p| - 4a, & |p| \geq 2 
\end{cases}, \quad (2.52) \]

where \(a\) determines the amount of overshoot near \(\pm 1\). A common choice of \(a = -1/2\). The cubic interpolation is more accurate than both the nearest neighbor and the linear interpolation, but also more computationally demanding. Cubic resampling is therefore more practical for evaluation than during optimization. The filter responses for all three interpolation methods are shown in Figure 2.6.

2.6 Evaluation

The transformation quality can be evaluated in several ways, e.g. by evaluate the similarity of the transformed moving image and the fixed image, or by
analyzing the properties of the transformation itself. We will start with the latter. Recall the two properties mentioned in Section 2.2.2: inverse consistency and diffeomorphisms. Given two registrations, one in each direction between the fixed and the moving image, the inverse consistency can be evaluated using vector magnitude error (VME),

\[ \text{VME} = \frac{1}{|\mathcal{F}|} \sum_{r \in \mathcal{F}} \| r - \varphi_{\mathcal{F} \rightarrow \mathcal{M}} \circ \varphi_{\mathcal{M} \rightarrow \mathcal{F}}(r) \|_2 \]  \hspace{1cm} (2.53)

where a lower value indicates a higher degree of inverse consistency. For diffeomorphic transformations the displacement fields are smooth and invertible. The displacement field is diffeomorphic if the Jacobian of the displacement field are positive,

\[ \text{Jac}(u(r)) = \det(\nabla u(r)) > 0 \]  \hspace{1cm} (2.54)

for all \( r \). The last evaluation metric for the transformation model we want to mention is the End-point-error (EPE). EPE is only possible to measure if the true displacement field is given, which is not always the case. The EPE is given by

\[ \text{EPE} = \frac{1}{|\Omega|} \sum_{r \in \Omega} \| u_{GT}(r) - u(r) \|_2. \]  \hspace{1cm} (2.55)
In Section 2.3.1, we defined several similarity measurements used for the objective function. Those metrics can be used to evaluate the transformation model as well.

The evaluation depends highly on what information that is given in the problem. Given a set of point locations in the two images, similar to the fiducial registration error, the Euclidean distance can be measured. This is referred to as target registration error (TRE) \( \text{TRE} \)

\[
\text{TRE} = \frac{1}{N} \sum_{i=1}^{N} ||p_i^M + u(p_i^F) - p_i^F||_2. \quad (2.56)
\]

Global overlap of defined structures in the images, like contour regions, can be measured using the Dice similarity coefficients (DCS) \( \text{DCS} \)

\[
\text{DCS} = 2 \frac{|S_F \cap (S_M \circ \varphi)|}{|S_F| + |S_M \circ \varphi|}, \quad (2.57)
\]

where \( S_F \) and \( S_M \) are labelled masks of the contour region(s) in the fixed and moving image, respectively. The last evaluation metric we will highlight is the Hausdorff distance \( \text{H} \)

\[
H(A, B) = \max (h(A, B), h(B, A)), \quad (2.58)
\]

where

\[
h(A, B) = \sup_{p^{(A)} \in A} \inf_{p^{(B)} \in B} ||p^{(A)} - p^{(B)}||. \quad (2.59)
\]

The Hausdorff distance measures the distance between two subsets, like a contour region. It is generally recommended to use several different evaluation metrics since any single one can be misleading.

### 2.7 Learnable approaches

The previous sections explained the theory of medical image registration and showed how the problem is approached using conventional iterative methods. As noticed, the complexity of the problem leads to many iterations and long execution times. This section will explain how deep learning can be used to learn the geometric transformation between two images based on data.

Machine learning is today widely used within medical image analysis \( \text{97} \). Examples of tasks using machine learning are: (i) classification \( \text{2} \), e.g., classify a patient as healthy or sick; (ii) detection \( \text{122} \), e.g., identify position...
of an object in an image; (iii) segmentation [123], e.g., classify organ in each pixel/voxel in an image; and (iv) image registration [20, 46, 61, 97]. Conventional analysis of medical images is often based on human expertise by professionals. Extracting features in medical images is both time-consuming and requires domain-specific knowledge.

Deep learning methods have shown impressive results in feature extraction, with state-of-the-art performance for the first three tasks above [97, 55, 128]. However, this is not (yet) the case for image registration. The reason why image registration has not been as successful as the other problems with deep learning is largely due to the complexity of the problem. The output domain from image registration is a two or three-dimensional vector field. This leads to a much more complex problem than the other tasks. However, there are several benefits of using a deep-learning approach for image registrations, compared to conventional iterative methods. Many deep learning registration methods aim to learn the optimization procedure for image registration. The iterative procedure can then be replaced with a single-pass, which together with improved computational hardware, e.g., GPU, reduces the execution time for image registration. Other methods learn the similarity metric between the images [61], which can be used in the conventional framework. Metric learning is beneficial in e.g. multimodal registration where manual metrics are poor.

We will now discuss several approaches for deep learning image registration methods. Then we will review some of the most common architectures and discuss our contribution to the field.

### 2.7.1 Training process

Given data, the aim for deep learning-based image registration is to estimate proper displacement fields for unseen images. The training process can be categorized into supervised, unsupervised, and weakly supervised procedures.

**Supervised learning**

Supervised learning is the most straightforward learning process, but also the most difficult to realize. The data for supervised learning consist of the image pairs (fixed and moving images) and the true geometric transformations. The model is then trained to mimic the ground-truth transformations by e.g., minimizing the end-point-error in (2.55). Unfortunately, ground-truth data rarely exist in practice. In deep learning image registration, there are two approaches for supervised learning: (i) simulating data with synthetic transformations; and (ii) using conventional methods to generate "ground-truth" data. The synthetic approach applies randomly generated transformations
on an image to get the image pair. Transformations can be generated e.g., by uniformly sampling control points in a TPS model (Section 2.2.1) [36] or randomly generating the affine transformation matrix (Example 2.2.1) [19]. Supervised learning using synthetic transformations is limited to monomodal image registration and it relies on the generated transformations being close to reality.

In Paper I we used geodesic shooting [106] to generate synthetic transformations to train a sparse-to-dense interpolation network. In the paper we used a state-of-the-art method for real-time tracking called discriminate correlation filters (DCF) [30], based on learned feature maps from computer vision [31], to track several points in a sequential 2D MRI sequence. From each tracker, we calculated a displacement vector and used the sparse-to-dense network to interpolate between the points.

The second approach, labeling data using conventional methods, has shown promising results [148, 18, 39] where the results in some metrics are better than the conventional method. However, labeling data with conventional methods is both time-consuming, constrained by the conventional method, and sometimes results in implausible transformations [20].

Unsupervised learning

The opposite of supervised learning is unsupervised. In unsupervised deep learning image registration, the transformation target is unknown. The learning procedure is instead based on a similar objective function as conventional methods [2.21], using image similarity and regularization. One key invention for unsupervised image registration methods is the spatial transformer network (STN) [67]. The spatial transformer network is a differentiable module, hence supporting backpropagation, which transforms the moving image based on the estimated transformation parameters. This enables calculating the similarity gradient during the training process. An illustration of an STN is given in Figure 2.7. In both papers II and III, spatial transformer networks are used to transform the moving image during training.

Weakly supervised learning

The last category of the training process for deep learning-based image registration is weakly supervised learning. Weakly supervised learning uses prior information, such as segmentation or landmarks, from the images. By including metrics for these in the loss function can help the learning process. In paper II we use a weakly supervised loss including the soft Dice coefficient [107].
2.7.2 Architecture

Convolutional neural networks have made great contributions in computer vision, e.g. when estimating optical flow \cite{120, 66, 132}. These methods are today state-of-the-art in the field. The optical flow problem has many similarities with the medical image registration problem, and we saw in Section 2.2.2 that the Demon’s algorithm is based on the optical flow equation (2.14). However, there are also some dissimilarities between optical flow and image registration. First, optical flow methods estimate the displacement vector between two consecutive frames in a sequence, while medical image registration aims to find the transformation between the fixed and moving image. The image modalities may differ, and the displacement may be larger than between two frames. Another significant difference between learning deep optical flow methods and deep image registration is the lack of ground truth data for medical image registration. In optical flow there exists many ground truths annotated data \cite{103, 104}, based on both animated and realistic videos.

A common procedure for deep learning image registration is to learn lower-dimensional feature representations of the images. Lower dimensional features from high dimensional images are typically extracted using a sequence of convolutional and downsampling operators, where each convolutional operator consists of a filter kernel. In deep learning, the parameters of the kernel

---

**Figure 2.7:** The spatial transformation module consists of: (i) a localization net, which estimates the transformation parameters; (ii) a grid generator, which transforms a regular grid with the estimated transformation; and, (iii) a differentiable bilinear sampler, which transforms the feature map \( U \) and produces a sampled feature map \( V \).
are unknown beforehand. During training, the filters are learned to extract the optimal features in the images. To estimate a dense transformation, lower-dimensional features are concatenated and upsampled until their final resolution. An example of this is the U-Net \[123\]. U-Net was originally designed for image segmentation but has shown good results in deep learning-based image registration as well \[46\]. U-Net downsamples the images and extracts features at several different feature levels. Skip connections are used to concatenate features at each level during the upsampling procedure. An illustration of the U-Net architecture is given in Figure 2.8.

![U-Net architecture for image registration](image.png)

Figure 2.8: U-Net architecture for image registration. Extracted features are concatenated (gray boxes) using skip connections (dashed arrows). The convolutional layers are represented with red and green boxes. The upsampling and downsampling operators are represented with up and down arrows, respectively.

One unsupervised, and weakly supervised model, which uses the U-Net and STN architecture is the VoxelMorph model \[7, 8\]. VoxelMorph uses an SSD or NCC similarity loss in combination with a Dice loss for weakly supervised learning. For regularization a diffusion regularizer to penalize non-smooth transformations was used. VoxelMorph has later been extended to diffeomorphic and probabilistic modelling \[29, 28\]. To ensure diffeomorphic transformations the distribution of the flow field \( v_t = \partial \varphi_t / \partial t \) was modelled as a multivariate normal, and scaling and squaring \[4\] was used as numerical integration to solve the ordinary differential equation in LDDMM. These models are trained in an unsupervised manner using variational inference where the fixed image is assumed to be a noisy observation of the transformed moving image.

Another model that uses a U-Net architecture was proposed by Fan et al. \[38, 37\]. They used a GAN-based network \[53\] where the generator follows a U-Net design. Here they did not define the similarity loss. Instead, they
learn a similarity metric using a CNN-based discrimination network where the inputs are the transformed moving image and the fixed image. In their example, they evaluate the model for both monomodal and multimodal image registrations. For multimodal image registration, where similarity losses are not applicable, a learned similarity loss can be beneficial.

Cascade networks are another popular architecture for deep learning-based image registration. Cascade networks are a combination of several networks that, like the hierarchical approach for conventional learning, are combined to refine the estimated transformation. Examples of cascade networks are either a simpler affine network in combination with a more complex deformation network [153], or using a multi-resolution approach by downsampling the images and estimating a transformation at each level [68, 109]. This is also referred to as coarse-to-fine since the transformation is refined at each level. In paper II we trained a coarse-to-fine network, including a CNN-based feature pyramid that downsampled the two images to several feature levels. Illustration of the first, model-based cascade network, is shown in Figure 2.9, and the second, multi-resolution cascade network, is shown in Figure 2.10. The cascade network does not guarantee diffeomorphic transformations, but they have shown an improved performance for larger transformations [46].

Figure 2.9: An example of a model-based cascade network. The first subnetwork (Network 1) estimates a simple transformation and used it as initial guess in a more complex network (Network 2).

We have now explained the two most common and successful architectures for deep learning-based image registration networks, U-Net and Cascade networks. Other methods [96, 105, 85, 49] use, e.g., reinforcement learning (RL) where the action space is limited to a set of possible transformations. These methods are mainly focused on rigid registrations, where the action space is low. However, there exists some non-rigid methods, using free-form transformation [85] and landmark detection [49] as well. In the next section, we will discuss some useful designs for convolutional networks and how domain
Figure 2.10: Cascade network using multi-resolution approach. The fixed and moving image are downscaled to several levels. At each level, starting at the top a transformation is estimated and used as initial guess in the next level. This approach is also called coarse-to-fine.

knowledge can be included in the deep learning method.

2.7.3 Implementation design

To increase the capacity of a CNN network a common approach is to add more convolutional layers. But extra layers include more learnable parameters and an increased risk of vanishing gradients. Vanishing gradients occur when the gradients for some function within the network are vanishingly small, resulting in problems updating earlier parameters in the model. One technique to avoid vanishing gradients is to use residual connections \cite{62}. In a residual connection, the input feature map is added to the output in the network, and the gradient of the input feature map is based on the layers and itself. DenseNet \cite{64} is another technique, related to residual connections, that avoids vanishing gradients. In DenseNet, the input to each layer consists of the inputs from all previous layers. Since the input feature consists of input features from all previous layers, the channel size increases. The input features are more diversified resulting in less complex layers that can be used, i.e., layers with fewer channels. DenseNet is therefore both computational and memory efficient. Illustrations of residual connections and DenseNet connections are shown in Figure 2.11.

As mentioned earlier, the convolutional operator is used to identify feature representations in the images. Features can be edges, shapes, or some other information that is useful for the task. Large features can be obtained by stacking several CNN upon each other \cite{152}. The size of the convolutional layers defines how large region is used in the input feature map to produce its output. This region is also called receptive field. Increasing the receptive
CHAPTER 2. MEDICAL IMAGE REGISTRATION

Figure 2.11: Illustration of residual connections (a) and DenseNet connections (b). In residual connections the input feature is added, +, to the output. In DenseNet the input feature is a concatenation, c, of the current and all previous input features. The boxes represent CNN-layers.

field without increasing the number of parameters in the model can be done with dilated convolutional layers. A dilated convolutional layer is a sparse variant of the standard convolutional layer. Given a feature map \( F \) and a kernel \( k \) with kernel size \( s \) the standard 1D convolutional operator is given by

\[
(F \ast k)(p) = \sum_{i=-s/2}^{s/2} F(p + i)k(i). 
\]  

The operation for dilated 1D convolution is instead

\[
(F \ast_I k)(p) = \sum_{i=-s/2}^{s/2} F(p + li)k(i), 
\]
where \( l \) is the dilation rate. Dilated convolution can be used to obtain large features, like contextual information, in high dimensional feature maps \[151\].

One inspired work in computer vision optical flow estimation is the PWC-Net \[132\]. Instead of increasing the network complexity by adding more parameters to the model, the PWC-Net included domain knowledge in the network and manage to increase the accuracy with a lower number of parameters to learn. Here, they assume brightness constancy between features in the two images and used normalized cross correlation (Section 2.3.1) to measure the similarity between the images. They called this measure cost volume. The cost volume between two points, \( p_1 \) and \( p_2 \), in the two feature maps, \( F_1 \) and \( F_2 \), is computed

\[
\text{cv}(p_1, p_2) = \frac{1}{N} (F_1(p_1))^\top F_2(p_2). \tag{2.62}
\]

However, computing the cost volume for all points in the two images is computationally expensive. To reduce the complexity, they only compared points in a close neighborhood, called search range. This limited the transformation to small displacements. To overcome the limitation, they used two well-defined techniques. First, the used a multi-resolution cascade network, i.e. a learned pyramid (Section 2.4.2). Then, they transformed the feature maps from the second image with the initially estimated transformation at each pyramid level. This reduces the difference between the two feature maps, even for larger initial displacements, and a smaller search range is valid. The cost volume is then used to estimate the transformation at each pyramid level using a CNN network. In paper II we used a similar approach for weakly supervised learning of 3D/3D monomodal image registrations.

The last implementation detail for a convolutional-based learnable method is for irregularly sampled data. While convolutional operators are well-suited for grid sampled data, they are less for irregularly sampled data. In paper I we tracked several regions and interpolate the displacement field using a sparse-to-dense network. In this case, the input data is irregularly sampled. To interpolate the irregular grid we here trained a network with normalized convolutional layers \[35, 63\]. Normalized convolutional layers, based on normalized convolutions \[81\], use the feature maps \( F \) and a confidence measurement \( C \) and propagates those forward, aiming to minimize the data error and maximize the output confidence. The normalized convolutional layer is given by

\[
F_{i,j}^l = \left( \sum_{m,n} F_{i+m,j+n}^{l-1} C_{i+m,j+n}^{l-1} \Gamma(W_{m,n}^l) \right) \frac{1}{\sum_{m,n} C_{i+m,j+n}^{l-1} + \epsilon} + b^l, \tag{2.63a}
\]

\[
C_{i,j}^l = \frac{\sum_{m,n} C_{i+m,j+n}^{l-1} \Gamma(W_{m,n}^l) + \epsilon}{\sum_{m,n} \Gamma(W_{m,n}^l)}, \tag{2.63b}
\]

36
where $W, b$ are the kernel and bias weights, respectively, $\Gamma$ is a non-negative applicability function, e.g. the softplus function, and $\epsilon$ is a constant. In the paper, we used an initial binary confidence, $C^0$ with ones where we have a sparse displacement vector and zero elsewhere, and train the network by sample irregularly positions from generated displacement fields.

This ends the theory of conventional and learnable medical image registration. In the next section, we will discuss some reflection, current limitations, and possible future approaches to medical image registration.

### 2.8 Outlook

Image registration is an important field in medical image processing and analysis. The ill-posedness of the problem makes it complex and hard to solve. We have shown how conventional iterative methods can be used to solve the problem. These methods are computationally expensive and require a lot of time which limits clinical usage. The optimization procedure can be replaced with a deep learning approach. However, these non-iterative methods have not yet outperformed the performance of conventional methods, but we believe there is more to come.

One of the biggest challenges in deep learning medical image registration is the lack of target data. Deep learning methods require large datasets for training. Unlike, the computer vision community there is no standard annotated dataset for medical image registration. We believe that a benchmark dataset would be a great feature to the field. Even though each image registration problem is unique a benchmark dataset would make it easier to attract more researchers to the field. A benchmark dataset can also be beneficial for unique image registration problems with limited data. Using transfer learning has shown improvement in other fields \cite{114} but is not widely used for image registration.

Another challenge for image registration is validation methods for plausible transformations. Here we believe that deep metric learning \cite{88, 75} is an interested field for both regularization \cite{112} and similarity \cite{94, 38, 37}. As mentioned, similarity metrics for multimodal image registration are hard. To overcome this, one approach is to use image translation and translate one image into the intensity domain of the other, e.g., translate an MRI image to CT and then use CT/CT monomodal image registration \cite{118}.

Until now, a convolutional network has been the standard approach for computer vision and medical image analysis. A new paradigm in computer vision is visual transformers (VT) \cite{34}. Unlike CNN, visual transformers split an image into patches of fixed size, embeds them linearly, and processes them by a transformer encoder \cite{139}. VTs has outperformed some of the
classical CNN networks with a fewer number of parameters and perform well on smaller dataset \[98\]. To the best of our knowledge, VTs have not been evaluated on medical image registration, but based on its popularity it is not an overly improbable guess that it will be made in the near future.

In this chapter, we have discussed the general ideas of medical image registration. In the next chapter, we will describe the dynamics of a sequence of images and explain how the dynamics can be expressed as a time-dependent transformation.
Chapter 3

Dynamic models

Previous chapter discussed image registration and described how a displacement field transforms the pixels/voxels from a moving image domain to a fixed image domain. This section will derive the displacements fields from a moving image domain $\mathcal{M}$ to the domain of spatiotemporal images $\{\mathcal{F}_t\}_{t=1}^T$, i.e. a sequence of images. Mathematically this can be express as,

$$\varphi_t = f(\varphi_{t-1}) + v_t, \quad (3.1a)$$

$$J_t = g(\varphi_t, J_M) + e_t, \quad (3.1b)$$

where $\{J_t\}_{t=1}^T$ is the observed image sequence, $\varphi_t$ is the displacement field from the moving image $J_M$ and $J_t$, $f$ and $g$ are functions for the time-dependent displacement field and the mapping function, respectively. The model also includes noise $v_t$ and $e_t$. The above model is equivalent to a state-space model. This chapter will start with background theory of state-space models and then show how one can simplify this model using dimension reduction techniques.

3.1 State-space models

The state-space model (SSM), also referred to as Hidden Markov model (HMM) [42], describes a dynamic phenomenon of a system given observations $x_{1:T} = [x_1, \ldots, x_T]$ over time. In many cases, the observations can not completely describe the dynamics of the underlying system. Other variables, referred to as states, $z_{1:T} = [z_1, \ldots, z_T]$ are therefore included in the model to enhance performance and give a better description of the dynamics. The SSM was first introduced in 1960s [74] and it is today a standardized tool in time-series analysis. The general SSM formulation is given by,

$$z_t = f_\gamma(z_{t-1}) + v_t, \quad (3.2a)$$
\[ x_t = g_\gamma(z_t) + w_t, \quad (3.2b) \]

where \( f_\gamma \) is called state function and \( g_\gamma \) the observation function, defined by model parameters \( \gamma \). The uncertainty of the model is encoded in the noise processes, which are assumed to be independent and identically distributed and are defined by the process noise, \( v_t \), and observation noise, \( w_t \). Even though the state and observation functions are deterministic functions, the uncertainty in the noise implies uncertainties in the entire process. Hence, the state space variables \( \{z_i\}_{i=1}^T \) and the observation variables \( \{x_i\}_{i=1}^T \) are stochastic variables. The SSM can therefore be represented using probability density functions,

\[
\begin{align*}
z_t \mid z_{t-1} & \sim p_\gamma(z_t \mid z_{t-1}), \quad (3.3a) \\
x_t \mid z_t & \sim p_\gamma(x_t \mid z_t), \quad (3.3b) \\
z_1 & \sim p_\gamma(z_1), \quad (3.3c)
\end{align*}
\]

where \( p_\gamma(z_t \mid z_{t-1}) \) and \( p_\gamma(x_t \mid z_t) \) are called state distribution and emission distribution respectively, and \( p_\gamma(z_1) \) is a prior. The SSM can also be visualized as a graphical model. An illustration of this is shown in Figure 3.1.

![Figure 3.1: SMM as a graphical model. The state process (white nodes) models the dynamics. The observed process is visualized by gray nodes and represents the measurement.](image)

An important property of the SSM is that the state process is a first-order Markov chain, i.e.

\[ p(z_t \mid z_{1:t-1}) = p(z_t \mid z_{t-1}). \quad (3.4) \]

The Markov property states that once the state \( z_t \) is known, previous states \( \{z_1 \ldots z_{t-1}\} \) do not provide any additional information about the states at time \( t \), \( z_t \). It plays an important role in the SSM since it simplifies the expression of the involved probability density functions. The full probabilistic model is given by

\[ p_\gamma(z_{1:T}, x_{1:T}) = p_\gamma(z_1) \prod_{t=2}^T p_\gamma(z_t \mid z_{t-1}) \prod_{t=1}^T p_\gamma(x_t \mid z_t). \quad (3.5) \]
The deviations for the full probabilistic model and other distributions are provided in Appendix A.

A common problem for SSM to solve is the state inference problem, i.e., finding information about the states, given observed data. The state inference problem can be categorized into three categories, depending on how many observations are available. The categories are:

1. **Filtering**: The filtering problem amounts to computing the filtered distribution of the state variable given past and present observations

   \[ p_\gamma(z_t \mid x_{1:t}) \]. \hspace{1cm} (3.6)

   The filter state distributions are computed when observation arrives, and it is relevant for online scenarios.

2. **Prediction**: The prediction problem amounts to computing a future state distribution at \( k \) steps ahead given past and present observations,

   \[ p_\gamma(z_{t+k} \mid x_{1:t}), \quad k > 0 \]. \hspace{1cm} (3.7)

   The prediction problem is relevant for forecasting, and it is useful in systems with latency issues.

3. **Smoothing**: The last state inference problem is the smoothing problem. The smoothing distribution is the distribution of a state variable given past, present, and future observations,

   \[ p_\gamma(z_t \mid x_{1:t+k}), \quad k > 0 \]. \hspace{1cm} (3.8)

   Since smoothing requires observations from the future it is only relevant in an offline scenario.

In general, the posterior inference is not analytically tractable and approximations of the posterior inferences are needed. Examples of approximation methods are: sequential Monte Carlo [54, 80], variational inference [27], extended Kalman filter [143] and unscented Kalman filter [70]. We will not discuss approximation methods in this thesis but refer to e.g. [21] and [111] for further information. However, one case of SSM where the posterior inference is analytically tractable will be discussed in Section 3.1.1, namely the linear Gaussian state-space model.

So far, we have assumed that the model parameters \( \gamma \) are known. In cases where some, or all model parameters are unknown, we want to learn them from the observed data. Two commonly used parameter learning methods are
maximum likelihood and Bayesian learning. Maximum likelihood produce a point estimates of $\gamma$ by maximizing the likelihood function for the unknown deterministic parameters,

$$\gamma^* = \arg \max_{\gamma} \log p_\gamma(x_{1:T}). \quad (3.9)$$

In Bayesian learning the parameters are instead treated as a random variable and the posterior distribution of $\gamma$ conditional on the observations are given by,

$$p(\gamma \mid x_{1:T}) = \frac{p_\gamma(x_{1:T})p(\gamma)}{p(x_{1:T})}. \quad (3.10)$$

A shared distribution for the two approaches is the likelihood distribution $p_\gamma(x_{1:T})$. The likelihood can be computed either by marginalizing the full probabilistic model in (3.5) concerning the state sequence

$$p_\gamma(x_{1:T}) = \int p_\gamma(z_{1:T}, x_{1:T})dz_{1:T}, \quad (3.11)$$

or via the use of the conditional probabilities,

$$p_\gamma(x_{1:T}) = p_\gamma(x_T \mid x_{1:T-1})p_\gamma(x_{1:T-1})$$
$$= p_\gamma(x_1) \prod_{t=2}^{T} p_\gamma(x_t \mid x_{1:t-1}), \quad (3.12)$$

where the one step ahead prediction of the observations is given by marginalizing $p_\gamma(z_t, x_t \mid x_{1:t-1})$ over $z_t$,

$$p_\gamma(x_t \mid x_{1:t-1}) = \int p_\gamma(z_t, x_t \mid x_{1:t-1})dz_t$$
$$= \int p_\gamma(x_t \mid z_t)p(z_t \mid x_{1:t-1})dz_t. \quad (3.13)$$

With this information, we will now study the perhaps simplest case of SSM, the linear Gaussian state-space model.

### 3.1.1 Linear Gaussian state-space model

In cases when both the state and observed transmissions are linear operators and the stochastic processes are Gaussians (the noises are Gaussian) we get the linear Gaussian state-space model (LG-SSM),

$$z_t \mid z_{t-1} \sim \mathcal{N}(z_t \mid Az_{t-1}, Q), \quad (3.14a)$$
\[ x_t \mid z_t \sim \mathcal{N}(x_t \mid Cz_t, R), \quad (3.14b) \]
\[ z_1 \sim \mathcal{N}(z_1 \mid \mu_1, P_1), \quad (3.14c) \]

where \( Q \) and \( R \) are covariance matrices \( v_t \sim \mathcal{N}(0, Q) \) and \( w_t \sim \mathcal{N}(0, R) \), \( A \) and \( C \) are called the state and observation matrix, respectively. The parameters of the model are \( \gamma = \{ A, Q, C, R, \mu_1, P_1 \} \) and the full distribution model is given by

\[ p_{\gamma}(z_{1:T}, x_{1:T}) = \mathcal{N}(z_1 \mid \mu_1, P_1) \prod_{t=2}^{T} \mathcal{N}(z_t \mid Az_{t-1}, Q) \prod_{t=1}^{T} \mathcal{N}(x_t \mid Cz_t, R). \quad (3.15) \]

The LG-SSM is the most popular SSM due to its simplicity. As mentioned earlier the state inference problems are analytical tractable and the distributions for those remain Gaussians. Let define those distributions as

\[ p_{\gamma}(z_t \mid x_{1:t}) = \mathcal{N}(z_t \mid \hat{z}_{t|t}, P_{t|t}), \quad (3.16a) \]
\[ p_{\gamma}(z_{t+k} \mid x_{1:t}) = \mathcal{N}(z_{t+k} \mid \hat{z}_{t+k|t}, P_{t+k|t}), \quad (3.16b) \]
\[ p_{\gamma}(z_t \mid x_{1:t+k}) = \mathcal{N}(z_t \mid \hat{z}_{t+k}, P_{t+k|t}), \quad (3.16c) \]

where the filtering and prediction distributions in (3.16a, 3.16b) can be derived with Kalman filtering [74]. Kalman filtering is a two-step forward recursive algorithm for exact Bayesian filtering of LG-SSMs. The steps are prediction and measurement update. The details are provided in Algorithm 2.

**Algorithm 2 Kalman Filter**

1: procedure KALMAN_FILTER\((A, C, Q, R, \mu_1, P_1)\)
2: for \( t = 1 \) do \( T \)
3: \hspace{1cm} Prediction step:
4: \hspace{2cm} \( \hat{z}_{t+1|t} = A\hat{z}_{t|t} \) \quad \triangleright Predict state
5: \hspace{2cm} \( P_{t+1|t} = AP_{t|t}A^\top + Q \) \quad \triangleright Predict covariance
6: \hspace{1cm} Measurement update:
7: \hspace{2cm} \( K_{t+1} = P_{t+1|t}C^\top(CP_{t+1|t}C^\top + R)^{-1} \) \quad \triangleright Kalman gain
8: \hspace{2cm} \( \hat{z}_{t+1|t+1} = \hat{z}_{t+1|t} + K_{t+1}(x_{t+1} - C\hat{z}_{t+1|t}) \) \quad \triangleright Update state
9: \hspace{2cm} \( P_{t+1|t+1} = P_{t+1|t} - K_{t+1}CP_{t+1|t} \) \quad \triangleright Update covariance
10: end for
11: return \( p(z_t \mid x_{1:T}) = \mathcal{N}(z_t \mid \hat{z}_{t|T}, P_{t|T}) \)
12: end procedure

The smoothing posterior distribution can be obtained using the forward Kalman filter and then the backward recursive Rauch-Tung-Striebel (RTS)
smoother algorithm [121]. The RTS procedure is shown in Algorithm 3. The derivation of the Kalman filter and the RTS smoother is provided in for example Murphy [110] and Kailath, Sayed and Hassibi [72].

Algorithm 3 Rauch-Tung-Striebel (RTS) smoother

1: procedure RTS($A, \hat{z}_t|_t, P_t|_t, \hat{z}_{t+1}|_t, P_{t+1}|_t$)
2: for $t = T$ do
3:   $L_t = P_t|_t A^T P_{t+1}|_t^{-1}$ \Comment{Smother gain}
4:   $\hat{z}_t|_T = \hat{z}_t|_t + L_t (\hat{z}_{t+1}|_T - \hat{z}_{t+1}|_t)$ \Comment{Update state}
5:   $P_t|_T = P_t|_t + L_t (P_{t+1}|_T - P_{t+1}|_t) L_t^T$ \Comment{Update covariance}
6: end for
7: return $p(z_t \mid x_{1:T}) = \mathcal{N}(z_t \mid \hat{z}_t|_T, P_t|_T)$
8: end procedure

For the LG-SSM the log-likelihood of the observations can be expressed in closed form. Given (3.12) and the state predictions $\hat{z}_{t|t-1}$ and $P_{t|t-1}$, the log-likelihood is given by

$$
\log p_\gamma(x_{1:T}) = \log p_\gamma(x_1) + \sum_{t=2}^T \log p_\gamma(x_t \mid x_{1:t-1}) = \sum_{t=1}^T \log \mathcal{N}(x_t \mid C \hat{z}_{t|t-1}, CP_{t|t-1}C^T + R).
$$

(3.17)

Learning the parameters of an LG-SSM is possible by maximizing the log-likelihood. The function is however not convex and tends to have several local minima. Solving non-convex optimization problems can be done with gradient-based optimization, like stochastic gradient ascent. Another procedure to maximize the log-likelihood is by using expectation-maximization (EM) [32]. The procedure to use EM to learn the parameters in LG-SSMs is explained in Shumway and Stoffer [129] and Ghahramani and Hinton [48].

3.1.2 Reflection

Let now take a break for reflection. We introduced our problem in (3.1) as an SSM and we provided an analytic framework for how to work with these SSMs. Recall the Kalman filter in Algorithm 2. In the measurement update, the Kalman gain is calculated via matrix multiplications and matrix inversion. We know that our observations $J_t$ are medical images that live in higher dimensional spaces. Even though it would be possible to solve the state inference problem, using Kalman filtering and RTS smoother,
the computational complexity expands. Solving the inverse problem using Gaussian elimination has a computational complexity of $O(n^3)$ where $n = n_x \times n_y$ for 2D images and this is not feasible. Another aspect is the non-linearity of the data. It is unlikely that the frame transitions of the images are linearly dependent. In the next sections, we will discuss different methods to both reduce the dimension and linearize our problem.

3.2 Dimension reduction

This section will introduce and discuss three of the most common techniques to reduce the dimension of high dimensional data. The idea of dimension reduction is to reconstruct the data $y \in \mathbb{R}^D$ based on a latent variable $x \in \mathbb{R}^L$, where $L < D$. In order to reconstruct the data as good as possible the latent variable should include a compressed version of the data that includes the most important features. The auto-encoder (AE) is an unsupervised method where the data is reconstructed from the latent space $\hat{y} = g(x)$, based on the latent variable $x = f(y)$. Both $f : \mathbb{R}^D \rightarrow \mathbb{R}^L$ and $g : \mathbb{R}^L \rightarrow \mathbb{R}^D$ are mapping functions and can be parameterized by some unknown parameters $\phi$ and $\theta$ respectively, e.g. as neural network, $f_{\phi}$ and $g_{\theta}$. Commonly used names for the dimension reduction function $f$ and the reconstruction function $g$ are encoder and decoder, respectively, and the latent space is referred to as bottleneck. A parameterized encoder and decoder can be trained with data by minimizing a loss function, here the reconstruction loss

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{N} ||y_i - \hat{y}_i||^2 + \mathcal{R}(\varphi, \theta) \quad (3.18)$$

with some regularization term $\mathcal{R}$, e.g. $L1$ or $L2$ regularization or KL divergence (more about KL divergence in Section 3.2.2).

An illustration of an AE is provided in Figure 3.2. The encoder and decoder can be any type of function, like nonlinear deep neural networks or linear projections. Below we will discuss two techniques, namely the linear principal component analysis (PCA) and the nonlinear variational auto-encoder (VAE) that relies on variational inference.

3.2.1 Principal component analysis

One of the most popular techniques for dimension reduction is Principal Component Analysis (PCA) [115]. PCA uses an orthogonal linear projection, $v^\top : \mathbb{R}^D \rightarrow \mathbb{R}^L$ where $v^\top v = I$, to project the data into a lower orthogonal subspace, called principal subspace, such that $x_i = v^\top y_i$ and the reconstruction is $\hat{y}_i = vx_i$. For centered data, i.e., $y_i = y_i - \bar{y}$ where $\bar{y}$ is the mean
of the data samples, we can see that minimizing the reconstruction error is equivalent to maximizing the variance of the projected data,

\[
v^* = \arg \min_{v : v^Tv = I} \sum_{i=1}^{N} ||y_i - \hat{y}_i||^2 = \arg \min_{v : v^Tv = I} \sum_{i=1}^{N} ||y_i - v(v^Ty_i)||^2
\]

\[
= \arg \min_{v : v^Tv = I} \sum_{i=1}^{N} ||y_i||^2 - 2y_i^Tv^Tv_i + y_i^Tv^Tv^Tv^Tv_i = I
\]

\[
= \arg \min_{v : v^Tv = I} \sum_{i=1}^{N} ||y_i||^2 - y_i^Tv^Tv_i = \arg \max_{v : v^Tv = I} \sum_{i=1}^{N} v^Ty_iy_i^Tv
\]

To solve this we start by centering the data \( \{y_i\}_{i=1}^{N} \) and organize it as a matrix and express the sampled covariance \( \Sigma \) of the data

\[
Y = \begin{bmatrix}
y_1^T \\
\vdots \\
y_N^T
\end{bmatrix}_{N \times D}, \quad \Sigma = Y^TY. \tag{3.20}
\]

Now add the constraints \( v^Tv = I \) to the objective function using a Lagrange multiplier \( \Lambda = [\lambda_1, \ldots, \lambda_D]^T \) yields

\[
v^* = \arg \max_{v : v^Tv = I} v^T \Sigma v = \arg \max_{v} v^T \Sigma v - \Lambda(v^Tv - I). \tag{3.21}
\]

The solution is given by differentiating the expression,

\[
\Sigma v = \Lambda v. \tag{3.22}
\]

We see that the optimal \( v \) are eigenvectors of the covariance matrix \( \Sigma \). Those eigenvectors, also called principal components can be found by first decomposing the data matrix \( Y \) using the single value decomposition (SVD)

\[
Y = USV^T \tag{3.23}
\]
where both $U \in \mathbb{R}^{N \times N}$ and $V \in \mathbb{R}^{D \times D}$ are orthogonal matrices and $S \in \mathbb{R}^{N \times D}$ is a diagonal matrix with singular values $s_1 \geq s_2 \geq \cdots \geq 0$. Inserting (3.23) into the covariance $\Sigma$ gives

$$
\Sigma = Y^T Y = (USV^T)^T (USV^T) \\
= (VS^T U^T)(USV^T) = V S^T S V^T \\
= VS^2 V^T
$$

and we obtain that the eigenvectors of $\Sigma$ are $V$ with eigenvalues $S^2$ and the optimal orthogonal projection is

$$
v = V
$$

So far the transformation $V^T : \mathbb{R}^D \rightarrow \mathbb{R}^D$ does not reduce the dimension of the data. However, we know that the first principal component in $V$ is associated with the largest eigenvalue. Large principal components in $V$ have lower eigenvalues and therefore lower significance for the data representation. Dimension reduction is therefore achieved by removing the larger principal components. In Figure 3.3 we construct a PCA model based on 5000 Chest X-ray images from the NIH Clinical Center database [145].

![Figure 3.3: Dimension reduction using PCA. The top row shows the mean of 5000 chest X-ray images and the first four principal components (PC). The bottom row shows a sample and reconstruction of the data using the 5, 50, 100 and 500 first principal components.](image)

PCA linearly reduces the dimension of the data to the optimal orthogonal subspace that minimized the variance of the data. In contrast, learning a linear AE without orthogonal constraints, the spanned subspace of $v$ will be
the same as for PCA. However, there are no guarantees that the subspace for linear AEs is linearly independent, as in PCA. On the other hand, PCA can only extract linear features from the data which is not a restriction for non-linear AEs. In the next section, we will derive the concept of dimensional reduction within a probabilistic framework using variational inference. This method is called variational auto-encoder (VAE).

### 3.2.2 Variational auto-encoders

Until now, we have not discussed dimension reduction techniques within a probabilistic model framework. In a probabilistic model, the joint distribution of the latent variables and the data is given by

$$p(y, x) = p(y \mid x)p(x)$$  \hspace{1cm} (3.26)

where \(p(y \mid x)\) is the likelihood of the data given the latent variable \(x\) and \(p(x)\) is a prior. From Bayes rules, we can express the inference of the model as

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)}.$$  \hspace{1cm} (3.27)

Unfortunately, the evidence \(p(y)\) is in most cases intractable and approximations are required to continue. One approach is to approximate the posterior distribution \(q_{\lambda}(x \mid y)\) as a parametric family of distributions, defined by its parameter \(\lambda\). Then the Kullback-Leiber divergence \[89\] can be used to measure the difference between the true posterior distribution \(p(x \mid y)\) and our approximation \(q_{\lambda}(x \mid y)\)

$$D_{KL}(q_{\lambda}(x \mid y) || p(x \mid y)) = \mathbb{E}_{q_{\lambda}}[\log \frac{q_{\lambda}(x \mid y)}{p(x \mid y)}]$$

$$= \mathbb{E}_{q_{\lambda}}[\log q_{\lambda}(x \mid y)] - \mathbb{E}_{q_{\lambda}}[\log p(y, x)] + \log p(y)$$  \hspace{1cm} (3.28)

and the optimal posterior approximation is given by minimizing the dissimilarity between the approximation and true posterior,

$$\lambda^* = \arg \min_{\lambda} D_{KL}(q_{\lambda}(x \mid y) || p(x \mid y))$$  \hspace{1cm} (3.29)

Again, due to the intractable evidence \(p(y)\), (3.29) is not possible to solve. However, if we use the fact that the Kullback-Leiber divergence is greater than or equal to zero, \(D_{KL} \geq 0\), we can rewrite (3.28) as

$$\log p(y) \geq \mathbb{E}_{q_{\lambda}}[\log p_{\lambda}(y, x)] - \mathbb{E}_{q_{\lambda}}[\log q_{\lambda}(x \mid y)]$$  \hspace{1cm} (3.30)

ELBO
and see that we have a lower bound of the evidence. Maximize this lower bound, also called evidence lower bound (ELBO), is equivalent to minimizing the Kullback-Leiber divergence in (3.29),

$$\lambda^* = \arg \max_{\lambda} \text{ELBO}(q_\lambda(x \mid y))$$ (3.31)

but is now tractable. The ELBO can be rewritten using the joint distribution in (3.26),

$$\text{ELBO} = \mathbb{E}_{q_\lambda} \left[ \log p(y, x) \right] - \mathbb{E}_{q_\lambda} \left[ \log q(x \mid y) \right]$$

(3.32)

and we can see that the optimization problem maximize the expected log-likelihood $\mathbb{E}_{q_\lambda} \left[ \log p(y \mid x) \right]$. Gaussian assumptions of the likelihood are proportional to the negative reconstruction error. Maximizing the ELBO in such cases is equivalent minimize the loss function in (3.18), using the Kullback-Leiber divergence regularizer.

So far, we have derived the expression for variational inference. Variational inference maximizes the ELBO based on the variational parameters $\lambda$ given some parametric distribution family $p_\lambda(x \mid y)$. A common parametric family for the approximate posterior is a Gaussian distribution and the variational parameters are in that case $\lambda_i = \{\mu_i, \Sigma_i\}$ where $\mu_i$ and $\Sigma_i$ are the mean and covariance of the approximate posterior for a given data point $y_i$. Learning the variational parameters $\{\lambda_i\}_{i=1}^N$ based on data $\{y_i\}_{i=1}^N$ can be done by solving

$$\lambda_i^* = \arg \min_{\lambda_i} \text{ELBO}(y_i, \lambda_i)$$ (3.33)

for each data point. This method is called mean-field variational inference and assumes that the approximate posterior can be factorized as a product of independent distributions, one for each data point, i.e.

$$p(x \mid y) \approx q_\lambda(x \mid y) = \prod_{i=1}^N q_{\lambda_i}(x_i \mid y)$$ (3.34)

Mean-field variational inference does not scale well for large data sets. Instead, if the variational parameters $\lambda$ are given by a parameterized function $\lambda_i =$
CHAPTER 3. DYNAMIC MODELS

\( f_\phi(y_i) \) where the parameters \( \phi \) are shared for all data points the optimization procedure can be across all data points. This is called amortized variational inference. Furthermore let us assume that the true distribution \( p(y) \) is also given by a parametric distribution family \( \tau_i = g_\theta(x_i) \) such that

\[
p^*(y) = p_\theta(y), \quad p^*(y \mid x) = p_\theta(y \mid x), \tag{3.35}
\]

where \( \theta \) are shared across all data points. Our probabilistic model now consists of an approximate posterior parameterized with global parameters \( \phi \) and a likelihood parameterized with global parameters \( \theta \). The parametric function \( f_\phi \) and \( g_\theta \) are then like the encoder and decoder for the AE and we have now defined the Variational auto-encoder (VAE) [79]. Here the encoder and decoder are also referred to as inference - and generative networks, respectively. An illustration of the VAE using Gaussian posterior approximations is provided in Figure 3.4 and its graphical representation is shown in Figure 3.5.

When training a VAE, the expectation operator is often replaced with an unbiased Monte Carlo estimate. The ELBO for a single data point and \( M \)
Monte Carlo samples is then given by

\[ \text{ELBO} \approx \frac{1}{M} \sum_{m=1}^{M} \log p(y_i \mid x_i^{(m)}) - D_{\text{KL}}(q_\phi(x_i^{(m)} \mid y_i) \| p_\theta(x_i^{(m)})) \quad (3.36) \]

and the parameters \( \theta, \phi \) can be learned jointly by maximizing the ELBO using stochastic gradient ascent. During training the gradients of \( \phi \) need to be calculated. To calculate the gradients for \( \phi \) based on the generated samples \( x \sim p_\theta(x \mid y) \) we have to include an auxiliary variable and use the so-called reparameterization trick to obtain the gradients for \( \nabla_\phi x \). The reparameterization trick for Gaussian posterior approximations looks like

\[ x \sim p_\theta(x \mid y) = \mathcal{N}(x \mid \mu, \Sigma) \Rightarrow x = \mu + \epsilon \Sigma \quad (3.37) \]

where \( \epsilon \sim \mathcal{N}(0, I) \) and the gradients of the latent variable is \( \nabla_\phi x = \nabla_\phi \mu + \nabla_\phi \Sigma \).

The VAE assumes that the prior \( p_\theta(x) \) and the likelihood \( p_\theta(y \mid x) \) come from some parametric family of distributions. Common assumption for the prior is \( p_\theta(x) = \mathcal{N}(x \mid 0, 1) \) and a Gaussian approximate posterior. The KL-divergence between the approximate posterior and the prior can then be solved analytically,

\[ D_{\text{KL}}(\mathcal{N}(x \mid \mu, \Sigma) || \mathcal{N}(x \mid 0, 1)) = \frac{1}{2} (\text{Tr}(\Sigma) + \mu^\top \mu - L - \log \det \Sigma) \quad (3.38) \]

where \( \text{Tr}(\cdot) \) is the trace operator and \( L \) is the dimension of the latent space. The distribution assumption of the likelihood depends on the data \( y \). If the data is binary a Bernoulli distribution is practical

\[ g_\theta(x_i) = p_{i_{\text{dec}}} \Rightarrow p_\theta(y \mid x_i) = \mathcal{B}(y \mid p_{i_{\text{dec}}}). \quad (3.39) \]

For continuous data, e.g., natural images, the data is more likely given a continuous distribution. A popular parametric distribution family of the likelihood given natural images is the Gaussian distribution,

\[ g_\theta(x_i) = \{\mu_{i_{\text{dec}}}, \Sigma_{i_{\text{dec}}}) \Rightarrow p_\theta(y \mid x_i) = \mathcal{N}(y \mid \mu_{i_{\text{dec}}}, \Sigma_{i_{\text{dec}}}). \quad (3.40) \]

### 3.2.3 Reflection

We have now presented two methods for dimensional reduction, PCA and VAE. Besides, they reduce the dimension they are also generative, i.e., it is possible to generate data by sampling from lower dimensional space. The main difference is that PCA is restricted to linear mapping, while VAE can
use a non-linearity relationship between the data – and latent space. It is, therefore, possible to linearize data with non-linear dependencies with a VAE, but not with PCA. Further, VAE has no dependency restriction between the projection/inference and the reconstruction/generative, like in PCA. We can therefore design the model with more freedom, including knowledge. In paper III, our decoder first estimates a transformation, $\varphi$, from a given image, $J_A$, and reconstruction the given image, $J_B$, as a noisy observation of the transformed image $J_A \circ \varphi$.

3.3 High dimensional dynamic modeling

In previous sections, we explained dynamical systems and dimension reduction techniques. Now we will combine those techniques to describe the dynamics of high dimensional data. Learning temporal dependencies from complex and/or high dimensional sequential data is an active research area, spread over several scientific fields, e.g. fluid dynamics, audio and speech processing, and computer vision. The first dimension reduction dynamical method we will discuss is Dynamic mode decomposition (DMD) [91]. Like PCA, DMD linearly decomposes the dimension but also includes the dynamics. In Section 3.2 we saw that nonlinear methods, such as variational auto-encoders, can handle more complex representations of the data. VAEs together with dynamic modeling is in the literature referred to as dynamical variational auto-encoders (DVAE) [50]. DVAEs combine VAEs with temporal models such as state-space models (SSM), recurrent neural networks (RNN), or neural ordinary differential equations (ODEs) [150]. Most of the DVAEs are based on nonlinear state transitions [86, 87, 25, 10, 45], where the state inference problems need to be approximated with variational approximations. A comparative study of such methods is found in [47], here called deep state-space models. However, in Section 3.1.1 we saw the benefits of the LG-SSM, where we can compute the state inference problems exactly. One method that takes advantage of this is the Kalman variational auto-encoder (Kalman VAE) [44]. Kalman VAE uses a VAE for dimension reduction and assumes that the dynamics of the latent observed variables can be described with an LG-SSM. In Section 3.3.2 we present the Kalman VAE in details.

3.3.1 Dynamic mode decomposition

Dynamic Mode Decomposition (DMD) [91] is a data-driven technique to obtain linear reduced-order models for high dimensional complex systems. DMD assumes that the dynamics of the data underlies a low-dimensional behavior. The dynamic is approximated to a low-rank structure that evolves linearly in time. There exist different variants of DMD that handle various
tasks such as: sparsity-promoting DMD \cite{69}, multiresolution DMD \cite{93}, extended DMD \cite{138}, and DMD with control \cite{117}. All are based on standard DMD \cite{125}, which we will now derive. DMD decomposes the data into modes, called DMD modes and eigenvalues, that describe the underlying dominant dynamics of the system. The modes and eigenvalues are approximations of the Koopman operator \cite{82,83}, a linear infinite-dimensional operator of non-linear dynamics. Similar to PCA (Section 3.2.1), DMD linearly embeds the data into a lower dimension space. They differ in that the principal components in PCA are uncorrelated $VV^\top = I$, i.e., the temporal correlations are not modeled. In DMD the temporal behavior is associated with each mode and therefore more suitable for dynamical processes, at a similar computational cost as PCA.

For DMD, the underlying dynamics is approximated as a linear model

$$\hat{y}_k = Ay_{k-1}$$  \hspace{1cm} (3.41)

with the matrix $A \in \mathbb{R}^{D \times D}$. Given equally sampled data set $[y_0, \ldots, y_K]$, $y_k \in \mathbb{R}^D$, also referred to as snapshot, where $t = k\Delta t$, the optimal solution of $A$ that minimizes the reconstruction error is given by

$$A = \arg \min_A \sum_{k=1}^{K} \|y_k - \hat{y}_k\|^2$$ \hspace{1cm} (3.42)

If we instead collect the data into matrices such that

$$Y_1 = \begin{bmatrix} y_0 & \cdots & y_{K-1} \end{bmatrix}, \quad Y_2 = \begin{bmatrix} y_1 & \cdots & y_K \end{bmatrix}$$  \hspace{1cm} (3.43)

the optimal solution of $A$ is given by

$$A = \arg \min_A \|Y_2 - AY_1\|_F^2 = \arg \min_A \text{Tr}(Y_2 - AY_1)^H(Y_2 - AY_1)$$ \hspace{1cm} (3.44)

where $H$ is the conjugate transpose, Tr denotes the trace operator and $\| \cdot \|_F$ the Frobenius norm. Differentiating (3.44) with respect to $A$ and setting the derivative to zero,

$$\frac{\partial}{\partial A} \|Y_2 - AY_1\|_F^2 (2AY_1Y_1^H - 2Y_2Y_1)^H = 0$$  \hspace{1cm} (3.45)

yields the optimal solution

$$A = Y_2Y_1^H(Y_1Y_1^H)^{-1} = Y_2Y_1^+$$  \hspace{1cm} (3.46)
where $Y_1^+$ is the pseudoinverse of $Y_1$. Unfortunately the right hand side of (3.46) is infeasible to compute for large data sets. Instead DMD seeks for the eigenvectors, $\Psi$, and eigenvalues, $\Lambda$, of $A$ such that $\Psi^{-1}A\Psi = \Lambda$. The approximated linear model (3.41) can therefore be expressed as
\[
\hat{y}_k = Ay_{k-1} = \Psi\Lambda^k\Psi^+y_0 \quad (3.47)
\]
The eigenvectors, also called DMD modes, and eigenvalues of $A$ can be obtained by first computing the truncated single value decomposition (SVD) of $Y_1$ by only keeping the $r$ column vectors of $U_r$ and $r$ row vectors of $V_r^H$ that corresponds to the largest singular values in $S_r$. The pseudo inverse of $Y_1$ can then be expressed as
\[
Y_1^+ = V_rS_r^{-1}U_r^H \quad (3.49)
\]
using Moore-Penrose pseudoinverse. Inserting this into (3.46) gives
\[
A = Y_2^*Y_1^+ = Y_2V_rS_r^{-1}U_r^H \quad (3.50)
\]
Now define $\tilde{A} \in \mathbb{R}^{r \times r}$ as a low-dimensional representation of $A$ using orthogonal projection onto the column space of $U_r$
\[
\tilde{A} = U_r^HAU_r = U_r^HY_2V_rS_r^{-1}U_r^H \quad (3.51)
\]
Given the $L$ nonzero eigenvalues $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_L) \in \mathbb{C}^{L \times L}$ and the corresponding eigenvectors $W = [w_1, \ldots, w_L] \in \mathbb{C}^L$ of $\tilde{A}$ the modes and eigenvalues of $A$ are $\Psi = U_LW \in \mathbb{C}^L$ and $\Lambda$ respectively. We can now define the $A$ matrix with the pair of the modes $\Psi$ and eigenvalues $\Lambda$ and we can predict the future state of the dynamical system using (3.47). The DMD procedure is shown in Algorithm 4.

From the DMD modes and eigenvalues we know that $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_L) \in \mathbb{C}^{L \times L}$ and $\Psi = [\psi_1, \ldots, \psi_L] \in \mathbb{C}^L$. We can then rewrite the dynamical model (3.47) as the sum
\[
\hat{y}_k = \Psi\Lambda^k\Psi^+y_0 = \sum_{j=1}^{L} \lambda_j^k \psi_j^+\psi_j^+y_0 = \sum_{j=1}^{L} \lambda_j^k \psi_j b_j, \quad (3.52)
\]
Algorithm 4 Standard DMD

1: \textbf{procedure} DMD\((y_1, \ldots, y_K)\)
2: \hspace{1em} \textbf{Set} \(Y_1 = [y_1, \ldots, y_{K-1}], Y_2 = [y_2, \ldots, y_K]\)
3: \hspace{1em} \textbf{Compute reduced SVD of} \(Y_1: Y_1 = U_r S_r V_r^H\) \textbf{with rank} \(r\)
4: \hspace{1em} \textbf{Define} \(\tilde{\mathbf{A}}: \tilde{\mathbf{A}} = U_r^H Y_2 V_r S_r^{-1}\)
5: \hspace{1em} \textbf{Compute eigendecomposition of} \(\tilde{\mathbf{A}}: \tilde{\mathbf{A}} w_j = \lambda_j w_j\)
6: \hspace{1em} \textbf{Take non-zero eigenvalues:} \(\{\lambda_j, w_k\}_{j=1}^L\)
7: \hspace{1em} \textbf{Compute DMD modes:} \(\psi_j = U_L w_j\)
8: \hspace{1em} \textbf{return} \(\{\lambda_j, \psi_j\}_{j=1}^L\)
9: \textbf{end procedure}

where \(b_j = \psi_j^+ y_0\) is the initial amplitude of mode \(\psi_j\). In DMD each mode represents a spatial representation of the dynamics \cite{84} and the most dominant modes are those with the highest amplitude values. The eigenvalues describe the temporal behavior of the dynamics. Eigenvalues are complex conjugates \(\lambda_j = \mu_j + i\omega_j\) and can be represented as

\[
\lambda_j = \mu_j + i\omega_j = |\lambda_j| \exp(i\alpha_j), \tag{3.53}
\]

where \(|\lambda_j| = \sqrt{\mu_j^2 + \omega_j^2}\) and \(\tan(\alpha_j) = \omega_j / \mu_j\). The eigenvalues are therefore associated with the growth and frequency for each mode. Growth rates of \(|\lambda_j| > 1\) indicates unstable modes that will diverge as \(k \to \infty\). For growth rates of \(|\lambda_j| = 1\) the dynamics of these modes are harmonic oscillation and for \(|\lambda_j| < 1\) the modes are stable, and the dynamics will vanish. Modes with growth rates close to zero can be interpreted as background \cite{92}. Further on, from \cite{99} we know that the relationship between the eigenvalues of our discrete sampled system and its continuous equivalent is

\[
\lambda_j^k = \exp(\lambda_j^{(c)} t) \Rightarrow \lambda_j^{(c)} = \ln \lambda_j / \Delta t, \tag{3.54}
\]

where \(\lambda_j^{(c)}\) is an eigenvalue of the continuous system. Approximated solutions for any \(t\) are then given by

\[
x(t) = \sum_{j=1}^L \exp(\lambda_j^{(c)} t) \psi_j b_j, \tag{3.55}
\]

3.3.2 Dynamic variational auto-encoders

We now want to combine the knowledge from LG-SSM in Section 3.1.1 and the VAE in Section 3.2.2. Assume that the VAE is capable of both reducing
the dimension and linearizing the dynamics. We can then use the latent variables from the VAE as observations in an LG-SSM, wherein the dynamics are explained. A graphical model of this kind can look like Figure 3.6 and is called the Kalman variational auto-encoder (Kalman VAE) [44].

\[
\begin{aligned}
\text{y}_t \\
\text{x}_t \\
\text{z}_t \\
\end{aligned}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
The approximate posterior distribution for the Kalman VAE is given by

\[
D_{\text{KL}} \left( q_{\phi,\gamma}(z, x \mid y) \parallel p_{\theta,\gamma}(z, x \mid y) \right) = \mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log \frac{q_{\phi,\gamma}(z, x \mid y)}{p_{\theta,\gamma}(z, x \mid y)} \right]
\]

\[
= \mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log \frac{q_{\phi,\gamma}(z, x \mid y)p_{\theta,\gamma}(y)}{p_{\theta,\gamma}(z, x, y)} \right]
\]

\[
= \mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log \frac{q_{\phi,\gamma}(z, x \mid y)}{p_{\theta,\gamma}(z, x, y)} \right] + p_{\theta,\gamma}(y)
\]

which is, equivalent to VAE, still intractable due to the evidence \( p_{\theta,\gamma}(y) \). However, the evidence is lower bounded by the ELBO,

\[
\text{ELBO} = -\mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log \frac{q_{\phi,\gamma}(z, x \mid y)}{p_{\theta,\gamma}(z, x, y)} \right]
\]

\[
= \mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log \frac{p_{\theta,\gamma}(z, x, y)}{q_{\phi,\gamma}(z, x \mid y)} \right]
\]

\[
= \mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log p_{\theta,\gamma}(z, x, y) - \log q_{\phi,\gamma}(z, x \mid y) \right].
\]

If we now rewrite the full distribution model

\[
p_{\theta,\gamma}(z, x, y) = p_{\theta,\gamma}(y \mid z, x)p_{\theta,\gamma}(x, z)
\]

we obtain that the observations \( y \) are modelled by the decoder \( p_{\theta}(y \mid x) = \prod_{t=1}^{T} p_{\theta}(y_{t} \mid x_{t}) \) and the joint distribution of \( z \) and \( x \) is the full probabilistic model of the LG-SSM, i.e.

\[
p_{\theta,\gamma}(z, x, y) = p_{\theta}(y \mid x)p_{\gamma}(x, z)
\]

The approximate posterior distribution for the Kalman VAE is given by

\[
q_{\phi,\gamma}(z, x \mid y) = p_{\gamma}(z \mid x)q_{\phi}(x \mid y) = p_{\gamma}(z \mid x)\prod_{t=1}^{T} q_{\phi}(x_{t} \mid y_{t}),
\]

where \( q_{\phi}(x_{t} \mid y_{t}) = \mathcal{N}(x_{t} \mid \mu_{t}, \sigma_{t}^{2}I) \) is modelled by the encoder. Inserting this into the ELBO (3.58) gives

\[
\text{ELBO} = \mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log p_{\theta,\gamma}(z, x, y) - \log q_{\phi,\gamma}(z, x \mid y) \right]
\]

\[
= \mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log p_{\theta}(y \mid x)p_{\gamma}(z, x) - \log p_{\gamma}(z \mid x)q_{\phi}(x \mid y) \right]
\]

\[
= \mathbb{E}_{q_{\phi,\gamma}(z,x|y)} \left[ \log \frac{p_{\theta}(y \mid x)}{q_{\phi}(x \mid y)} + \log \frac{p_{\gamma}(z, x)}{p_{\gamma}(z \mid x)} \right]
\]

\[
= \mathbb{E}_{q_{\phi}(x|y)} \left[ \log \frac{p_{\theta}(y \mid x)}{q_{\phi}(x \mid y)} \right] + \mathbb{E}_{p_{\gamma}(z|x)} \left[ \log \frac{p_{\gamma}(z, x)}{p_{\gamma}(z \mid x)} \right].
\]
and we identify that the ELBO consists of one VAE part and one LG-SSM part. From Section 3.1 we know that the full probabilistic model for the SSM (3.5) is

\[
p_{\gamma}(z, x) = p_{\gamma}(z_1) \prod_{t=2}^{T} p_{\gamma}(z_t | z_{t-1}) \prod_{t=1}^{T} p_{\gamma}(x_t | z_t)
\]

\[
= \mathcal{N}(z_1 | \mu_1, P_1) \prod_{t=2}^{T} \mathcal{N}(z_t | Az_{t-1}, Q) \prod_{t=1}^{T} \mathcal{N}(x_t | Cz_t, R)
\]

(3.63)

and the conditional posterior over the LG-SSM is given by the RTS smoothing algorithm 3

\[
p_{\gamma}(z | x) = \prod_{t=1}^{T} \mathcal{N}(z_t | \hat{\mu}_{t|T}, P_{t|T})
\]

(3.64)

During training the expectations are approximated with an unbiased Monte Carlo sampler by first sampling \( \tilde{x}_t \sim q_{\phi}(x_t | y_t) \) for each \( t \) and then use the sampled values \( \tilde{x} \) to sample \( \tilde{z}_t \sim \mathcal{N}(z_t | \hat{\mu}_{t|T}(\tilde{x}), P_{t|T}(\tilde{x})) \) for each \( t \) given the RTS smoother. Then the approximate ELBO,

\[
\text{ELBO} \approx \frac{1}{M} \sum_{m=1}^{M} \log p_{\theta}(y | \tilde{x}^{(m)}) + \log \frac{p_{\gamma}(\tilde{z}^{(m)}, \tilde{x}^{(m)})}{p_{\gamma}(\tilde{z}^{(m)} | \tilde{x}^{(m)})}
\]

(3.65)

is used to jointly learn all parameters \( \{\phi, \gamma, \theta\} \) using stochastic gradient ascent. An illustration of the Kalman VAE is shown in Figure 3.7

Figure 3.7: Illustration of Kalman VAE. The latent variables \( x = [x_1, \ldots, x_T] \) from the VAE are observations in an LG-SSM.
3.3.3 Reflection

We have now seen two methods that describe the dynamics of high-dimensional dynamic systems, DMD and Kalman VAE. In our case, we believe there are several benefits of using the Kalman VAE approach in comparison with DMD. First, while DMD is restricted to linear mapping to the latent space Kalman VAE is not. We, therefore, believe the encoder/decoder can linearize the dynamic of the high-dimensional images to a linear dynamic in the latent space. Second, the decoder is more flexible than the linear reconstruction in DMD. We can therefore define the decoder after our purpose. An example of this is shown in Paper III. Here we use a conditional variation auto-encoder (CVAE) and condition the LG-SSM on the latent representation of the moving image $J_M$. For reconstruction, we first let the decoder estimate the likelihood of the transformation $\varphi_t$ for each $t$ and then estimate the image likelihood as noisy observations of the transformed moving image $J_M \circ \varphi_t$. In such a way, we derive an unsupervised generative dynamic model that estimates both the transformation and reconstructions of the images, only based on the image sequence.
Chapter 4

Concluding remarks

4.1 Conclusion

We have now introduced the problem of motion modeling and image registration in real-time systems. First, we studied the conventional iterative methods for image registration problems. We showed that the conventional approach, due to a long convergence rate, is not suited for real-time systems. Then, we introduced a deep learning approach. Here, we trained the parameters of a model based on predefined data with the goal that the model generalizes well for unseen cases. Finally, we studied the dynamics of the system. For this, we reduced the dimension of the high-dimensional images and estimated the dynamics in a lower-dimensional latent space.

Motion modeling of spatiotemporal medical images is challenging. The lack of ground truth data and the ill-posedness of the problem introduces implausible transformations. For successful modeling, we believe in combining ideas from all three approaches mentioned above. Including physical knowledge from conventional methods would simplify learning-based methods and also increase the possibility for plausible transformation. Both the cascade-network and cost volume layers explained in Section 2.7 are examples of this. Furthermore, dynamic representation enhances dependencies between time steps, which enables the possibility of imputation and extrapolating for missing data and perhaps also results in smoother transformations.

Until recently, this has been a fairly unexplored area of research that we believe will change in the near future. Successful methods in computer vision in combination with real-time image support in healthcare are two factors that accelerate this development.
4.2 Future work

There are several possible avenues for future work in the field. Both the robustness and accuracy of the presented models need to be evaluated and tested. We also want to explore our models on other data from different modalities and body sites.

Recall the use case, real-time adaptive radiotherapy, presented in Chapter I. Motion modeling is just one necessary feature for real-time adaptation in radiotherapy. Others, like dose accumulation and optimal control of plan adaptation for real-time systems, are also relevant research fields. In the presented work, except Paper II, we have studied motion fields in the two-dimensional space, however, the actual motion field is of course three dimensional. Interesting research topics would be to estimate the entire three-dimensional motion, based on sparse information, like two-dimensional images. All the proposals mentioned above are research topics that will continue to be explored during the rest of my Ph.D. project.
Appendices
Appendix A

Distributions of SSM

Using this property of a Markov chain and the conditional probabilities the following distributions can be derived:

Data distribution:

\[
p(x_{1:T} \mid z_{1:T}) = p(x_T \mid z_T)p(x_{1:T-1} \mid z_{0:T-1})
= \prod_{t=1}^{T} p(x_t \mid z_t). \tag{A.1}
\]

Prior distribution:

\[
p(z_{1:T}) = p(z_T \mid z_{1:T-1})p(z_{1:T-1})
= p(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1}). \tag{A.2}
\]

Full probabilistic model:

\[
p(x_{1:T}, z_{1:T}) = p(x_{1:T} \mid z_{1:T})p(z_{1:T})
= p(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1}) \prod_{t=1}^{T} p(x_t \mid z_t). \tag{A.3}
\]
Marginal distribution

Unfortunately, the likelihood \( p(x_{1:T}) \) is in general intractable. It can be expressed by marginalizing the full probability model,

\[
p(x_{1:T}) = \int p(x_{1:T}, z_{1:T})dz_{1:T}
\]

\[
= \int p(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1})p(x_t \mid z_t)dx_{0:T}, \tag{A.4}
\]

or marginalizing the joint PDF \( p(z_t, x_{1:t}) = p(x_t \mid z_t)p(z_t \mid x_{1:t-1}) \)

\[
p(x_{1:T}) = \prod_{t=1}^{T} p(x_t) = \prod_{t=1}^{T} \int p(x_t \mid z_t)p(z_t \mid x_{1:t-1})dx_t, \tag{A.5}
\]

and for the general case the posterior distribution can only be approximated up to some normalization constant.

Posterior distribution:

Using the Bayes’ rules we have

\[
p(z_{1:T} \mid x_{1:T}) = \frac{p(x_{1:T} \mid z_{1:T})p(z_{1:T})}{p(x_{1:T})}, \tag{A.6}
\]

which may be intractable due to the likelihood. But the posterior is also given by

\[
p(z_{1:T} \mid x_{1:T}) = p(z_1 \mid z_{2:T}, x_{1:T})p(z_{2:T} \mid x_{1:T}) = \ldots
\]

\[
= p(z_T \mid x_{1:T}) \prod_{t=1}^{T-1} p(z_t \mid z_{t+1:T}, x_{1:T}) \tag{A.7}
\]

\[
= p(z_T \mid x_{1:T}) \prod_{t=1}^{T-1} p(z_t \mid z_{t+1}, x_{1:T}).
\]
Bibliography


BIBLIOGRAPHY


Title
Registration by tracking for sequential 2D MRI

Authors
Niklas Gunnarsson, Jens Sjölund and Thomas B. Schön

Edited version of
Registration by tracking for sequential 2D MRI

Abstract

Our anatomy is in constant motion. With modern MR imaging it is possible to record this motion in real-time during an ongoing radiation therapy session. In this paper we present an image registration method that exploits the sequential nature of 2D MR images to estimate the corresponding displacement field. The method employs several discriminative correlation filters that independently track specific points. Together with a sparse-to-dense interpolation scheme we can then estimate the displacement field. The discriminative correlation filters are trained online, and our method is modality agnostic. For the interpolation scheme we use a neural network with normalized convolutions that is trained using synthetic diffeomorphic displacement fields. The method is evaluated on a segmented cardiac dataset and when compared to two conventional methods we observe an improved performance. This improvement is especially pronounced when it comes to the detection of larger motions of small objects.

1 Introduction

Image guided radiation therapy has been a key component to improve the accuracy of radiation therapy [25]. Daily imaging of the current anatomical state informs whether to modify the treatment plan or not. With integration of MRI with the treatment machines it is today possible to do real-time magnetic resonance imaging (MRI) during a treatment session [16]. Since MRI is ideally suited for imaging soft tissues, this allows identification of the shape and position of tumors and organs at risk and adjusting of the treatment accordingly.

In this paper we will describe a new method for medical image registration. Our method is suitable for 2D sequential images and it is inspired by recent progress within the domain of computer vision. We first estimate a sparse
displacement field using discriminative correlation filters (DCF) \[6\] to track a fixed number of points in the image sequence. Then, we use a trained sparse-to-dense interpolation scheme in the form of a neural network that uses normalized convolutional layers \[10\]. Normalized convolutions \[14\] use confidence and are well-suited for irregularly sampled data. Advantages of our method include i) it is modality agnostic due to the fact that the model is trained online and customized for the current image type, ii) it allows flexibility in trading off computation versus accuracy due to the freedom in selecting tracking points and iii) it is designed with real-time requirements in mind since each tracker is fast and the trackers are independent of each other. Possible disadvantages might be if a tracker loses its target and the filter is learn online to track a divergent point or if the tracking points are too sparse and movements are missed.

Even though conventional methods have shown good result they are computationally heavy as each registration problem is typically formulated as a large optimization problem that needs to be solved iteratively. Image registration methods can be categorized as physics-based methods, e.g. like Demons \[24\], interpolation-based methods or knowledge-based methods \[22\]. Our method is categorized as an interpolation-based image registration method \[22\]. The two most reputable interpolation-based techniques are B-splines \[19\] and thin-plate splines (TPS) \[4\]. Revaud et al. \[17\] proposed an interpolation-based method that uses an edge-aware geodesic distance to weight the sample points when interpolating from a sparse to dense representation.

Recently, the interest for learning-based methods has increased and the results have been impressive \[1, 5\]. By training a neural network to predict the solution, the method no longer needs to solve an optimization problem at every time step which drastically reduces the execution time. However, those methods require training data, which directly determines their applicability in the sense that the trained model is specific to the modality of the training data.

2 Method

In the following sections we describe the different steps of the method. We start with the initial location of the trackers, then continue with specific tracker techniques and finally the sparse-to-dense interpolation scheme and how this network is trained. The workflow we suggest is shown in Fig. \[\_]
PAPER I – REGISTRATION BY TRACKING FOR SEQUENTIAL 2D MRI

Figure 1: From the first image \((t = 0)\), a fixed number of trackers are initialized at specific points. When a new image \((t = 1, 2 \ldots)\) arrives the tracking positions are updated and a sparse displacement field is estimated. A sparse-to-dense interpolation scheme is used to get the desired dense representation.

### 2.1 Selection of tracking points

One factor that may have a strong influence on the performance of our method is how the set of tracking points is selected. In computer vision, there has been a range of work on automatically generating candidates such as edges and corners [20, 21, 23]. On the other hand, it is also possible to use task-based prior knowledge, such as a segmentation mask, when selecting candidates. If the aim is to estimate the deformation of a tumor or organs at risk, a task-based candidate might be preferable considering the fact that the interpolation accuracy tends to deteriorate with the distance [22]. Therefore, we suggest to sample candidates inversely proportional to the distance from a segmentation mask.

### 2.2 Tracking

In an image registration problem, the motion vector from one image to the another is referred to as the displacement field. In conventional image registration methods, the problem is most often formulated as a regression problem where the goal is to find a displacement field that minimizes some metric. A common choice is to use static points in space and estimate the displacement field by the corresponding points in the next image. These
points are often referred to as control points. We, on the other hand, define the displacement field from a sparse set of tracking points where the spatial location of these points is constantly updated. In the next section, we describe how to estimate a dense displacement field from the location of the tracking points.

We suggest discriminative correlation filters, (DCF) as trackers which have lately proven extremely useful for visual tracking, achieving state-of-the-art results on several benchmarks. The idea is to train a correlation filter \( f \) that distinguishes the tracking target from the background. A continuous score variable \( y \) is associated with each image patch \( x \) where a low score, \( y \approx 0 \), represents background while high score, \( y \approx 1 \), represents the target. The typical choice of the score is a sampled Gaussian function with its mean centered in the target position. The filter is initially trained on a set of image patches, \( \{x_i\}_{i=1,...,N} \), wherein the target and the desired output \( y_i \) are defined. In practice it is convenient to generate data from the first frame by applying different augmentation techniques, such as rotating, blurring and flipping, etc. In the simple case where the image patch has a single channel the filter is determined by solving a regularized least-squares problem of the form

\[
f^* = \arg \min_f \sum_i \|x_i * f - y_i\|^2 + \lambda\|f\|^2,
\]

but more general versions, including both multi-channel and multi-resolution representations, exist.

Once trained, the filter can be used to predict scores by convolving it with subsequent images. For speed, the convolution can be restricted to a smaller search region. For each tracking point, the displacement vector is extracted as the target location corresponding to the highest score. One advantage with DCF is that the filter can easily be updated with new training data (online) without retraining the entire model.

In this paper, we use the ECO tracker which uses deep convolutional features together with a dimensionality-reduction step that makes it faster and more robust against overfitting.

### 2.3 Sparse-to-dense interpolation

We suggest a sparse-to-dense interpolation scheme based on normalized convolutions. Traditional convolutions are well-suited for regularly sampled data, but less so if the data is irregularly sampled. Normalized convolution uses a confidence map derived from the data where a value of 0 indicates no data is given at that point and values close to 1 indicate a high certainty of the value. A dense signal can then be estimated from a sparse representation.
based on a non-negative applicability function and some basis functions. Estimating the applicability functions for several normalized convolutional filters using constant basis functions has shown good results \cite{10, 12} when interpolating dense depth scenes from LiDAR data.

We design our interpolation scheme as a neural network inspired by U-net \cite{18} with normalized convolutional layers. A detailed illustration of the network is shown in Fig. \ref{fig:2}. We propagate the sparse displacement field and the confidence map through the network by using non-negative constraints on the weights and as a loss function we use a combination of Huber loss, to minimize the estimation error, and a term that maximizes the output confidence \cite{10}.

We generate synthetic displacement fields to train our model. We assume that the displacement field is diffeomorphic and use a geodesic shooting method \cite{15} to generate synthetic displacement fields independently of any imaging data. We artificially create a sparse representation by uniformly sampling coordinates in the generated displacement field. We use a binary confidence map based on the sampled coordinates. To avoid overfitting, we generate unique displacement fields for every iteration. The model is trained offline and only executed when interpolating the sparse representation from the trackers to the dense displacement field.

Figure 2: An illustration of our sparse-to-dense interpolation network. As input we use the channels of the sparse displacement field ($u_x^0$, $u_y^0$, $c^0$) and as output we use the dense displacement field for each channel ($u_x^L$, $u_y^L$) and the propagated confidence map ($c^L$). The network consists of seven normalized convolutional layers where layer 2, 3, 4 and 5 are reused at different feature levels. Each layer is illustrated with a solid box and the numbers indicate kernel height $\times$ kernel width $\times$ number of filters. Several subsequent filters are indicated with a dashed line and the filter names around the box. $\uparrow$ and $\downarrow$ are upsampling and downsampling operators.
3 Experiments

For evaluation we used a publicly available cardiac MRI dataset\footnote{http://www.cse.yorku.ca/~mridataset/}. The dataset contains measurements from 32 subjects (one was removed due to poor image quality). The data of each subject contains an image sequence of 20 images where the ventricle starts in an expanded position, after a few time steps the ventricle has contract an then expands again to its starting position. Each time step has 8 to 15 slices of \(256 \times 256\) pixels and the slices include manual segmentation of the epicardium and endocardium. We used the manual segmentation masks to segment the myocardium (region in between the epicardium and endocardium) and the ventricle (region inside the endocardium). Since our method is 2D based we select a specific slice in the dataset. We compared our method with several others by applying the estimated displacement fields on the segmentation and calculating the Dice score \[\text{Dice} = \frac{2|S_t \cap \hat{S}_{t_0 \rightarrow t}|}{|S_t| + |\hat{S}_{t_0 \rightarrow t}|}\] (2)

where \(S_t\) is the manual segmentation at time \(t\) and \(\hat{S}_{t_0 \rightarrow t}\) is the warped segmentation using the image at time step \(t_0\) and the estimated displacement field between the image at time step \(t_0\) and the image at time step \(t\).

3.1 Setup

We initialized 150 different trackers by sampling points inversely proportional to the distance of the union of the two segmentation masks. For each tracker we used an image patch of size \(10 \times 10\) pixels and a search area of 4.5 times the patch size. The trackers were trained on the first image and the filter coefficients are updated online after every new image. For the sparse-to-dense interpolation scheme we used a binary confidence map with 1 for tracking coordinates and 0 elsewhere.

3.2 Result

For a specific image plane \((z = 4)\) the Dice scores between the warped and ground truth segmentation masks were calculated. For comparison we used the following methods i) the same tracker (ECO) but replacing the spare-to-dense interpolation scheme with thin-plate splines (TPS) \footnote{[4]}, ii) B-spline image registration \footnote{[13]} with mean square error using a grid size of \(20 \times 20\) pixels and iii) the Demons algorithm \footnote{[24]}. Table 1 shows the mean and standard deviation of the result for all image frames and subjects.
Table 1: The result for all images and subjects. The Dice mean ($\mu$) and standard deviation ($\sigma$) is shown for each method and segmentation regions (myocardium and ventricle).

<table>
<thead>
<tr>
<th>Method</th>
<th>Myocardium (Dice)</th>
<th>Ventricle (Dice)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>ECO + NCONV</td>
<td>0.810</td>
<td>0.066</td>
</tr>
<tr>
<td>ECO + TPS[4]</td>
<td>0.791</td>
<td>0.062</td>
</tr>
<tr>
<td>B-Spline[13]</td>
<td>0.730</td>
<td>0.148</td>
</tr>
<tr>
<td>Demons[24]</td>
<td>0.760</td>
<td>0.111</td>
</tr>
</tbody>
</table>

In Figure 3 we show the mean Dice score at each time step, averaged across all subjects for both the myocardium (left) and ventricle (right) segmented areas.

In Figure 3 we show the mean Dice score at each time step, averaged across all subjects. In Figure 4 we show the warped segmentation for each method between two time steps where the ventricle first is in its expanded position and then in its contracted position. This is illustrated by time steps $t = 0$ and $t = 8$ for subject 15.

4 Discussion and conclusion

Our method shows an overall better performance in terms of Dice score when compared to conventional methods. As shown in Fig. 3 and Fig. 4 our method handles larger movements better than conventional methods and it appears more clearly for smaller deformable region such as the myocardium. The method allows a trade-off between accuracy and computational capacity because of the flexibility in choosing the number and position of the tracking points. Although our current implementation is not fast enough for real-time
estimations, we remain hopeful. A single tracker can be executed in real-time but in the current implementation all trackers run sequentially. Since the trackers are independent of each other and thus the potential for significant speed improvements through parallelization exists. We also expect further gains from relating the tracking score to the confidence, and from tweaking the data augmentation for medical applications.

Acknowledgement

This research was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation. The authors would like to thank Dr. David Tilly for helpful comments.
Bibliography


Title
Learning a deformable registration pyramid

Authors
Niklas Gunnarsson, Jens Sjölund and Thomas B. Schön

Edited version of
Learning a deformable registration pyramid

Abstract

We introduce an end-to-end unsupervised (or weakly supervised) image registration method that blends conventional medical image registration with contemporary deep learning techniques from computer vision. Our method downsamples both the fixed and the moving images into multiple feature map levels where a displacement field is estimated at each level and then further refined throughout the network. We train and test our model on three different datasets. In comparison with the initial registrations we find an improved performance using our model, yet we expect it would improve further if the model was fine-tuned for each task. The implementation is publicly available[1].

1 Introduction

Image registration is a fundamental problem in medical imaging. It is widely used in applications to, for example, combine images of the same object from different modalities (multimodal registration), detect changes between images at different times (spatiotemporal registration), and map segments from a predefined image to a new image (atlas based segmentation).

The basic principle of image registration is to find a displacement field $\phi$ that maps positions in a moving image to the corresponding positions in a fixed image. Conventionally, image registration problems are often stated as optimization problems, where the aim is to minimize a complex energy function [9].

A popular heuristic for solving image registration problems is to use a coarse-to-fine approach [19] i.e. to start with a rough estimate of the displacement field and refine it in one or several steps. It is common to

downsample the fixed and moving images using a kernel based pyramid, and make a first estimate of the displacement field at the lowest resolution which is then used as an initial guess when estimating the field at the next resolution level, and so forth.

Due to the complexity of the energy function each estimate is computationally expensive and requires long execution time. Machine learning provides an alternative approach, where a model is optimized (learned) offline based on a training dataset, obviating the need for expensive optimization at test time [1]. In this paper we present an image registration method that combines the conventional coarse-to-fine approach with a convolutional neural network (CNN).

2 Method

We have developed a 3D deformable image registration method inspired by the PWC-Net [18], a 2D optical flow method popular in computer vision. Our method estimates and refines a displacement field at each level of a CNN downsampling pyramid.

2.1 Architecture

![Model architecture](image)

(a) Model architecture.

![Operations at each feature level](image)

(b) Operations at each feature level.

Figure 1: An overview of the model architecture. The moving and fixed image are downsampled into several feature maps using the pyramid [a]. Fig (b) shows operations at each feature level. Blue and white boxes represent operations with and without trainable parameters, respectively.

The pyramid downsamples the moving image $I_m$ and the fixed image $I_f$ into several feature maps $\{w_m^{(l)}, w_f^{(l)}\}_{l=1}^{L}$. At each level, starting from the top, a displacement field $\phi_d^L$ is estimated and used as an initial guess at finer levels. Fig. 1 illustrates the model architecture (6.1(a)) and operations at each level (6.1(b)). The total number of trainable parameters in our model is 8.6 million. Our model includes multiple CNN blocks. These consist of a 3D convolutional layer followed by Leaky Rely and batch normalization. All 3D
convolutional layers use a kernel size of (3,3,3). Each module of our model is explained below:

**Pyramid:** Downsamples the moving and fixed image into several feature map levels using 3D CNN layers. The same pyramid is used for the moving and the fixed images. We use a four-level pyramid ($L = 4$) where each level consists of three CNN blocks. The stride is two in the first block and one in the subsequent blocks. The number of filters at each level is 16, 32, 32, and 32, respectively.

**Warp (W):** Warps features from moving images with the estimated displacement field. This module has no trainable parameters.

**Affine (A):** A dense neural network that estimates the 12 parameters in an affine transformation. This module consists of a global average pooling followed by a dense layer.

**Cost volume (CV):** Correlation between the warped feature maps from the moving image and feature maps from the fixed image. For computational reasons the cost volume is restricted to voxel neighborhoods of size $d$. This module has no trainable parameters.

**Deform (D):** A 3D DenseNet [10] that estimates the displacement field based on its current estimate, the cost volume and the feature maps from the fixed image. This module uses 5 CNN blocks of the same type as in the Pyramid but with 64, 64, 32, 18, and 8 filters, respectively followed by a convolutional layer with 3 filters.

**Upsample (U):** Upsamples the estimated displacement field from one level to the next. Consists of an upsampling layer followed by a single 3D CNN.

### 2.2 Loss function

Our loss function combines image similarity with regularization of the displacement field. By including the intermediate estimates in the loss, we aim to gain additional control of the network. Auxiliary information, e.g. anatomical segmentations $S_m$ and $S_f$ are incorporated via an additional structural
similarity term \( \mathcal{L}_{\text{seg}} \). Our resulting loss function can be written as

\[
\mathcal{L} = \mathcal{L}_{\text{seg}} + \sum_{l=0}^{L} \left( \mathcal{L}_{\text{sim}}^{(l)} + \mathcal{L}_{\text{smooth}}^{(l)} \right). \tag{1}
\]

We use the (soft) Dice coefficient (DCS) \([14]\) for structural similarity and the normalized cross-correlation (NCC) \([16]\) for image similarity. To ensure smooth displacements we regularize the affine displacement field with the L2-loss between the estimated value and an identity displacement field \((\phi_0^{(l)})\) and the deformable field with the spatial gradient of the displacement field \([5]\),

\[
\mathcal{L}_{\text{seg}} \left( S_f, S_m, \phi_d^{(0)} \right) = \lambda(1 - \text{DCS}(S_f, S_m \circ \phi_d^{(0)})), \tag{2a}
\]

\[
\mathcal{L}_{\text{sim}}^{(l)} \left( I_f^{(l)}, I_m^{(l)}, \phi^{(l)}_d \right) = -\gamma^{(l)} \text{NCC}(I_f^{(l)}, I_m^{(l)} \circ \phi^{(l)}_d), \tag{2b}
\]

\[
\mathcal{L}_{\text{smooth}}^{(l)} \left( \phi_a^{(l)}, \phi_d^{(l)} \right) = \alpha^{(l)} ||\phi_a^{(l)} - \phi_0^{(l)}||_2^2 + \beta^{(l)} ||\nabla \phi_d^{(l)}||_2^2, \tag{2c}
\]

where \(I_m^{(l)}\) and \(I_f^{(l)}\) represent downsampled versions of the moving and fixed images at each level and \(\phi_a^{(l)}\) and \(\phi_d^{(l)}\) indicate the estimated affine and deformable registrations (for each level). The hyperparameters \(\lambda, \gamma^{(l)}, \alpha^{(l)}\) and \(\beta^{(l)}\) determine the importance of the corresponding terms.

### 3 Experiment

We evaluated the model on three different tasks from the 2020 Learn2Reg challenge \([2]\). The different tasks were: inspiration and expiration CT scans of thorax images with automatic segmented lung (Task 2) \([8]\); 3D CT abdominal images with thirteen segmented organs (Task 3); and segmented hippocampus MRI of healthy adults and adults with non-affective psychotic disorder (Task 4) \([17]\).

We trained our model on image pairs from all tasks at the same time. All images were downsampled (to a resolution of \(64 \times 64 \times 64\)) and normalized \((I_f, I_m \in [0, 1])\). The different hyperparameters were \(\lambda = 5.0, \gamma^{(l)} = 5/2^l, \alpha^{(l)} = 2^l\) and \(\beta^{(l)} = 1/2^l\) for \(l \in \{0, \ldots, 4\}\) and for cost volume search range we used \(d = 2\). The network was trained end-to-end using the Adam optimizer and a learning rate of \(10^{-4}\). To speed up training we used distributed training on three Nvidia GeForce GTX 1080 Ti graphic cards and trained the model for 100 epochs, which took approximately 24 hours.

The results are shown in Table 1. Table 2 shows examples of warping the moving image using displacement fields \(\phi_d^{(l)}\) estimated at three different levels \(l \in \{0, 2, 4\}\). Based on the total score, our approach was ranked 5th according to the public leaderboard \([2]\)
Table 1: Result on test dataset for each task.

<table>
<thead>
<tr>
<th>Task</th>
<th>Method</th>
<th>TRE&lt;sup&gt;6&lt;/sup&gt;</th>
<th>TRE&lt;sup&gt;30&lt;/sup&gt;</th>
<th>DCS&lt;sup&gt;3&lt;/sup&gt;</th>
<th>DCS&lt;sup&gt;30&lt;/sup&gt;</th>
<th>HD95&lt;sup&gt;11&lt;/sup&gt;</th>
<th>SDlogJ&lt;sup&gt;13&lt;/sup&gt;</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GPU</td>
</tr>
<tr>
<td>2</td>
<td>our</td>
<td>9.00</td>
<td>12.22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.12</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>initial</td>
<td>10.24</td>
<td>17.77</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.00</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>our</td>
<td>-</td>
<td>-</td>
<td>0.39</td>
<td>0.12</td>
<td>43.03</td>
<td>0.13</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>initial</td>
<td>-</td>
<td>-</td>
<td>0.23</td>
<td>0.01</td>
<td>46.07</td>
<td>0.00</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>our</td>
<td>-</td>
<td>-</td>
<td>0.74</td>
<td>0.67</td>
<td>2.82</td>
<td>0.16</td>
<td>0.32</td>
</tr>
<tr>
<td></td>
<td>initial</td>
<td>-</td>
<td>-</td>
<td>0.55</td>
<td>0.36</td>
<td>3.91</td>
<td>0.00</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Sample result from the validation dataset. The moving image $I_m$ (left) is warped with the estimated displacement field from several levels ($l = 4, 2, 0$). Starting from the coarsest to the finest level. The fixed image $I_f$ is shown to the right.

<table>
<thead>
<tr>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_m$</td>
<td>$I_m \circ \phi^{(4)}_d$</td>
<td>$I_m \circ \phi^{(2)}_d$</td>
</tr>
<tr>
<td>$I_m \circ \phi^{(0)}_d$</td>
<td>$I_f$</td>
<td>$I_f$</td>
</tr>
</tbody>
</table>

4 Conclusion and future work

In this paper we have shown that it is possible to include domain knowledge when developing machine learning methods for medical image registration problems. Our method operates in a coarse-to-fine manner and could be modified in many ways, e.g. by replacing the CNN pyramid with other technologies; like a Laplacian pyramid, similar to the winner of the competition [15], or modifying/removing displacement fields estimations (affine or deformable) in the levels.

In comparison with other participants in the competition our approach was to create a single general model for all tasks while other participants
used different models or different training procedures \cite{15, 7, 4} for each task. The general approach showed increased performance compared with initial registrations. In future work, we will evaluate to what extent the performance improves when fine tuning the model for each task.

During the training phase the memory usage was high (11.4 GB). In the experiments we downsampled the input images to a low resolution, using a batch size of one (at each GPU replica) and our partial cost volume had a search range of two to be able to fit the model in GPU memory (11.7 GB). We believe that an in-depth analysis of the network will reveal ways of reducing memory usage without sacrificing performance substantially, e.g. by removing superfluous layers or reducing the number of filters. One idea is to reduce the number of parameters in the DenseNet \cite{12}. Other potential improvements include: 1) training each level separately, starting from the coarsest, which will reduce the number of trainable parameters in each training process, 2) training the model on slices (2D) or thin slabs (2.5D), instead of the entire volume and iteratively estimate the entire 3D displacement field.

Acknowledgement

This research was funded by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, and the Swedish Foundation for Strategic Research grant SM19-0029.
Bibliography


[17] Amber L Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram Van Ginneken, Annette Kopp-Schneider, Bennett A Landman, Geert Litjens, Bjoern Menze et al. “A large annotated medical image dataset for the development and evaluation


Title
Latent linear dynamics in spatiotemporal medical data

Authors
Niklas Gunnarsson, Jens Sjölund, Peter Kimstrand and Thomas B. Schön
Latent linear dynamics in spatiotemporal medical data

Abstract

Spatiotemporal imaging has applications in e.g. cardiac diagnostics, surgical guidance, and radiotherapy monitoring. In this paper, we explain the temporal motion by identifying the underlying dynamics, only based on the sequential images. The model maps the inputs to a low-dimensional latent space wherein a linear relationship between a hidden state process and an observed latent process holds. For this, we use a conditional variational auto-encoder (CVAE) to nonlinearly map the higher dimensional image to a lower-dimensional space, wherein we model the dynamics with a linear Gaussian state-space model (LG-SSM). The model, a modified version of the Kalman variational auto-encoder, is end-to-end trainable, and the weights, both in the CVAE and LG-SSM, are simultaneously updated by maximizing the evidence lower bound of the marginal likelihood. In contrast to the original model, we explain the motion with a spatial transformation from one image to another. This results in sharper reconstructions and the possibility of transferring auxiliary information, such as segmentations, through the image sequence. Our experiments, on cardiac ultrasound time series, show that the dynamic model outperforms traditional image registration in execution time, to a similar performance. Further on, our model offers the possibility to impute and extrapolate for missing samples.

1 Introduction

Today, most medical imaging modalities support some form of time-resolved imaging. In some modalities, like ultrasound, it is the default mode of operation, while in others it is known under different names depending on the application, e.g. fluoroscopy [36], 4DCT [30] and 4D flow MRI [25]. It is used for analysis, in e.g. cardiac diagnostics [1] and for guidance, monitoring
In this paper we want to uncover the dynamics in a medical image time series based on nothing but the images themselves. Our aim is to learn a parameterized function, $h$, such that the image $y_t$ at time $t$ is described by the dynamical system

$$y_t = h(y_{1:t-1}) + \epsilon_t,$$

where $y_{1:t-1} = \{y_1, \ldots, y_{t-1}\}$ are the previously observed images and $\epsilon_t$ is noise. This representation makes it possible to predict $\hat{y}_t = h(y_{1:k})$ based on observations up to time $k$, i.e. to impute ($t < k$), filter ($t = k$), or extrapolate ($t > k$) images.

Temporal medical images contain high-dimensional data where the dynamics, due to cyclic and deformable motion, is nonlinear. While the linear assumption is preferable for several reasons, including tractable filtering and smoothing posteriors, it is inappropriate for the raw image series. The idea of this paper is to reduce the high-dimensional data to a low-dimensional latent space wherein the linearity assumptions are valid. For this we combine techniques from generative and dynamic modelling and train the model end-to-end. In the literature, this is referred to as a deep state-space model (DSSM) \cite{10, 17, 23}.

In the medical domain, there are many examples of motion modeling based on side information \cite{27, 35} such as from X-ray tracking of implanted markers or surface tracking. Those methods are limited to the requirement of this side information. To the best of our knowledge, the only example where the motion model is based directly and exclusively on the temporal medical images under study is the recent work by Krebs et al. \cite{21}. They model the dynamics in a low-dimensional probabilistic space using a temporal convolutional network \cite{2} and a Gaussian process prior. In their model, the prediction length is restricted based on the motion matrix. We overcome this by modeling the dynamics as a first-order Markov process.

## 2 Background

The Kalman variational auto-encoder (Kalman VAE) \cite{10} is an unsupervised model for high-dimensional sequential data that, most likely, undergoes nonlinear dynamics. Higher-dimensional observed images, $y = \{y_t\}_{t=1}^T$ are non-linearly embedded into a lower-dimensional space using a variational auto-encoder. The dynamics of the lower-dimensional features, $x = \{x_t\}_{t=1}^T$ is modeled with a linear Gaussian state-space model based on a state-space process $z = \{z_t\}_{t=1}^T$. Below we explain the variational auto-encoder and the
linear Gaussian state-space model, and then how those are combined into the framework of Kalman VAE.

2.1 Variational auto-encoder

Similar to traditional auto-encoders, variational auto-encoders (VAEs) embed the input $y$ in a lower-dimensional latent space $x$ using an encoder, $E_{\phi}$ and reconstruct the original input with a decoder, $D_{\theta}$. They differ in that VAEs are generative and reconstruct the data distribution $p_\theta(y)$ instead of a single sample $y$. In this case the true posterior $p_\theta(x \mid y)$ is intractable. By approximating the variational posterior as a multivariate Gaussian, $q_\phi(x \mid y) = \mathcal{N}(x \mid \mu_{\text{enc}}, \Sigma_{\text{enc}})$, where $\mu_{\text{enc}}$ and $\Sigma_{\text{enc}}$ are the outputs from the encoder, it is possible to sample from the variational approximation. From the KL divergence between the approximate and the true posterior we can obtain a lower bound on the true likelihood

$$\log p_\theta(y) \geq \mathbb{E}_{q_\phi(x \mid y)} \left[ \log p_\theta(y, x) - \log q_\phi(x \mid y) \right]$$

$$= \mathbb{E}_{q_\phi(x \mid y)} \left[ \log p_\theta(y \mid x) + \log \frac{p_\theta(x)}{q_\phi(x \mid y)} \right], \quad (2)$$

where the prior over the latent space is usually chosen to be a multivariate Gaussian $p(x) = \mathcal{N}(x \mid 0, I)$. This lower bound is called the evidence lower bound (ELBO) and can be estimated by sampling.

2.2 Linear Gaussian state-space model

In a VAE, each sample, $x_t \in \mathbb{R}^L$, from the approximate posterior is normally distributed with mean, $\mu_{\text{enc}}^t$, and covariance $\Sigma_{\text{enc}}^t$. If we assume Gaussian noise, it follows that the state-space vector, $z_t$, in our linear state-space model also follows a normal distribution. More precisely, we have a linear Gaussian state-space model (LG-SSM),

$$p(z_t \mid z_{t-1}) = \mathcal{N}(z_t \mid Az_{t-1}, Q),$$

$$p(x_t \mid z_t) = \mathcal{N}(x_t \mid Cz_t, R), \quad (3)$$

where $Q$ and $R$ are covariance matrices for the process and measurement noise, respectively. Given an initial guess $z_1 \sim \mathcal{N}(z_1 \mid \mu_1, P_1)$, the joint probability distribution can be expressed using the LG-SSM model (from (3)),

$$p(x, z) = p(x \mid z)p(z) = p(z_1) \prod_{t=1}^{T} p(x_t \mid z_t) \prod_{t=2}^{T} p(z_t \mid z_{t-1}). \quad (4)$$

Given observations $x$ the mean and covariance of the state-space variables is analytically tractable using a Kalman filter, $\mu_{t \mid t}, P_{t \mid t}$, and a Rauch-Tung-Striebel (RTS) smoother, $\mu_{t \mid T}, P_{t \mid T}$. 
2.3 Kalman variational auto-encoder

A Kalman VAE uses a VAE to reduce the dimension of the image time series distribution wherein the dynamics are represented linearly using an LG-SSM in the latent space. The goal is to describe the dynamics of the system in the latent space with an LG-SSM and use the decoder to reconstruct the image time series.

Similar to a regular VAE, the ELBO can be derived from the KL divergence between the approximate and true posterior. In a Kalman VAE the approximate posterior is given by

\[
q_{\phi,\gamma}(x, z \mid y) = p_{\gamma}(z \mid x)q_{\phi}(x \mid y),
\]

and the true posterior can be rewritten as

\[
p_{\theta,\gamma}(x, z, y) = p_{\theta}(y \mid x)p_{\gamma}(x, z).
\]

For a single time series \(y = \{y_t\}_{t=1}^T\) the ELBO is given by

\[
\log p_{\theta,\gamma}(y) \geq \mathbb{E}_{q_{\phi,\gamma}(x,z \mid y)}\left[ \log p_{\theta,\gamma}(x, z, y) - \log q_{\phi,\gamma}(x, z \mid y) \right]
= \mathbb{E}_{q_{\phi}(x \mid y)}\left[ \log \frac{p_{\theta}(y \mid x)}{q_{\phi}(x \mid y)} \right] + \mathbb{E}_{p_{\gamma}(z \mid x)}\left[ \log \frac{p_{\gamma}(x, z)}{p_{\gamma}(z \mid x)} \right],
\]

where \(\phi\) and \(\theta\) are the encoder and decoder parameters, respectively, and \(\gamma = \{A, C, R, Q, \mu_1, P_1\}\) are the LG-SSM parameters. It is possible to sample \((\tilde{x}, \tilde{z})\) by first sampling \(\tilde{x} \sim q_{\theta}(x \mid y)\) and then conditionally sampling \(\tilde{z} \sim p_{\gamma}(z \mid \tilde{x})\). Notice that \(p_{\gamma}(z \mid \tilde{x})\) is tractable using the Kalman smoother algorithm and the joint distribution \(p_{\gamma}(x, z)\) is given by (4). With the reparameterization trick \[19\] the model can be trained end-to-end to minimize the negative ELBO using e.g. stochastic gradient descent.

3 Method

Unlike a standard Kalman VAE, our method operates in image transformations and not image per se. More specifically, we assume that the images \(y\) are observations from the dynamical state-space model

\[
\varphi_t = f(\varphi_{t-1}) + \epsilon_t,
\]

\[
y_t = g(\varphi_t, y_M) + \eta_t,
\]

where \(y_M \in \mathcal{M}\) is a reference image, \(\varphi_t\) is the image transformation from the stationary image domain to the image domain at time \(t\) and \(\epsilon_t, \eta_t\) are process and measurement noise, respectively. We hereby describe the model as a
sequence of spatiotemporal image registrations. To be consistent with the medical image community we define $y_M$ as the moving image. An illustration of our model is provided in Figure 1.

The dynamics of our model are driven by the transformation $\varphi_t$ with respect to the spatial information in $y_M$. To include the spatial information of $y_M$ we suggest a modified Kalman VAE, where the mappings to and from the latent representation are given by a conditional variational auto-encoder (CVAE) [18], conditioned on the moving image $y_M$. The motion model in the latent space is modeled with an LG-SSM. Here, we condition the initial state prior with the latent representation, $x_M$ of the moving image, such that $p_\gamma(z_1 \mid x_M) = \mathcal{N}(z_1 \mid \mu_1(x_M), P_1(x_M))$. For this, we use a fully connected neural network to estimate the mean and variance of this distribution. The parameters of our model can be jointly updated by maximizing the ELBO, $\mathcal{L}(\phi, \theta, \gamma)$

$$
\mathcal{L}(\phi, \theta, \gamma) = \mathbb{E}_{q_\phi(x, x_M \mid y, y_M)} \left[ \log \frac{p_\theta(y \mid x, y_M)}{q_\phi(x, x_M \mid y, y_M)} + \mathbb{E}_{p_\gamma(z \mid x, x_M)} \left[ \log \frac{p_\gamma(x, z \mid x_M)}{p_\gamma(z \mid x, x_M)} \right] \right],
$$

where the conditional distributions are given by

$$
p_\theta(y \mid x, y_M) = \prod_{t=1}^T p_\theta(y_t \mid x_t, y_M),
$$

$$
q_\phi(x, x_M \mid y, y_M) = q_\phi(x_M \mid y_M) \prod_{t=1}^T q_\phi(x_t \mid y_t),
$$

Figure 1: Illustration of our model. Similar to a Kalman VAE, we learn a lower-dimensional linear-Gaussian state-space model from data, but unlike a Kalman VAE our model operates on image transformations $\varphi_t$ and not images.
\[ p_\gamma(z \mid x, x_M) = \prod_{t=1}^{T} p_\gamma(z_t \mid x, x_M), \quad (10c) \]
\[ p_\gamma(x, z \mid x_M) = p_\gamma(z_1 \mid x_M) \prod_{t=1}^{T} p_\gamma(x_t \mid z_t) \prod_{t=2}^{T} p_\gamma(z_t \mid z_{t-1}). \quad (10d) \]

### 3.1 Likelihood

In the Kalman VAE the likelihood is assumed to come from some parametric family of distribution, parameterized by the decoder network \( D_\theta(x_t) \). Instead of modeling the likelihood of the image intensity we model the motion, i.e. the likelihood of the image transformation \( \varphi_t \). Modeling the image transformations enables transferring auxiliary information, such as segmentations, from one image domain to another. From the decoder we can estimate the likelihood of the \( \varphi_t \)

\[ p_\theta(\varphi_t \mid x_t) = \mathcal{N}(\varphi_t \mid \mu_{t}^{\text{dec}}, \Sigma_{t}^{\text{dec}}), \quad (11) \]

as a multivariate Gaussian distribution, i.e. the decoder outputs \( \mu_{t}^{\text{dec}} \) and \( \Sigma_{t}^{\text{dec}} \). A common disadvantage of image-based VAEs is blurry reconstructions due to the restrictive assumption of a Gaussian likelihood \cite{9}. High frequencies, like sharp edges and fine details in the images are often missed in the reconstructions \cite{7}. In the space of image transformations, on the other hand, Gaussian assumptions are quite reasonable. Existing methods use Gaussian kernels to regularize the displacement field and avoid improbable transformations \cite{34}. We then estimate the image likelihood as a noisy observation of the transformed image, \( y_M \circ \varphi_t \), i.e.

\[ p_\theta(y_t \mid y_M, \varphi_t) = \mathcal{N}(y_t \mid y_M \circ \varphi_t, \sigma_t^2 I), \quad (12) \]

for some noise \( \sigma_t \).

### 4 Experiments

#### 4.1 Dataset

In our experiments, we used the EchoNet-Dynamic dataset \cite{29}. The dataset consists of 10030 2D ultrasound echocardiogram time series with 112 × 112 pixel frames. We fixed the time horizon to 50 time steps \((T = 50)\), which corresponded to 1 second and approximately included one cardiac cycle. We extracted one sequence per time series and its start position was selected randomly. Sequences shorter than the fixed horizon were removed (315). We allocated 7220 sequences for training, 1237 for testing and 1258 for validation. The image intensity was normalized to \([0, 1]\) for all images.
4.2 Metrics

We evaluate our model using three metrics, the Dice coefficient [8], the percentage of non-positive Jacobians (Jacobian determinants), and the execution time. The data sequence included human expert segmentation of the left ventricle at two different time points. We used the image for the first human expert segmentation as moving image $y_M$, and estimated the transformation for the sequence, including the time step for the other human expert segmented image. We can then measure the quality of transformation with the Dice coefficient, the overlap between the human expert segmentation, $S_t$ and our estimation, $S_M \circ \varphi_t$,

$$
\text{Dice} = 2 \frac{|S_t \cap (S_M \circ \varphi_t)|}{|S_t| + |S_M \circ \varphi_t|},
$$

where a perfect overlap is indicated with a Dice coefficient of 1, and 0 for no overlap. The percentage of non-positive elements in the Jacobian is an indicator of how topology-preserving the transformation is, e.g. if $|J_\varphi(p)| > 0$ for all points $p$ the transformation $\varphi$ is diffeomorphic. The Jacobian, $|J_\varphi|$ is defined by

$$
|J_\varphi| = \begin{vmatrix}
    \frac{\partial \varphi_x}{\partial x} & \frac{\partial \varphi_x}{\partial y} \\
    \frac{\partial \varphi_y}{\partial x} & \frac{\partial \varphi_y}{\partial y}
\end{vmatrix}.
$$

4.3 Implementation Details

Encoder:

The encoder consists of downsampling convolutional levels with 16, 32, 64 and 128 filters respectively, where the feature maps at each level are first downsamled using a 2-stride convolutional layer followed by a batch normalization [14] and a leaky ReLU [24] activation function. Furthermore, we use two 1-stride convolutional layers with residual connections [13] before the feature map is downsampled to the next level. The output of the last level is flattened and two dense layers are used—one for the mean, $\{\mu_t^{\text{enc}}\}_t^{T}$ and one for the variances, $\{(\sigma_t^{\text{enc}})^2\}_t^{T}$.

LG-SSM:

In the LG-SSM we use a dimension of 16 for our observations, $x_t \in \mathbb{R}^{16}$, and 32 for the state space, $z_t \in \mathbb{R}^{32}$. The mean and variance of the initial prior was estimated with a 3 layer fully connected neural network with 16, 16 and 32 units.
Decoder:

The decoder layers mirror the encoder with the same filter sizes and residual connections at each level. We let the decoder estimate $\mu_{\text{dec}}^t$, the mean of the transformation likelihood, and we used a fixed covariance $\Sigma_{\text{dec}}^t = 0.01^2 I$. When sampling from the transformation likelihood we apply a mask which corresponds to the cone-shaped field-of-view of the ultrasound image.

The mean of the image likelihood is estimated by the sample from the transformation likelihood and applying the transformation on a moving image $y_M$. For this, we use a spatial transformation module. In this example, we use the first image in the time sequence as a moving image. The variance for the image likelihood was fixed to $\sigma_t^2 = 0.01^2$.

Training:

The model is trained end-to-end using importance sampling to maximize the ELBO in (9) by jointly updating all parameters $\{\theta, \phi, \gamma\}$ in the model. We use a monotonic annealing schedule weight to the posteriors in the loss function. As an optimizer, we use Adam with exponential decay with factor 0.85 every 20 epoch and an initial learning rate of $10^{-4}$. We use a batch size of 4 image time series and train the model for 50 epochs on a single Nvidia GeForce GTX 1080 Ti graphic card ($\approx$ 17 hours training). We have implemented our method into the Tensorflow framework.

Evaluation:

When evaluating the result we use samples from the filtered distribution $x_t \sim p_\gamma(x_t \mid z_{1:t})$. We found that our initial model produced transformations where the Jacobian was frequently negative. We therefore apply regularization to the displacement field using a Gaussian kernel, $g_\sigma$

$$\hat{\varphi}_t = \varphi_t \ast g_\sigma, \tag{15}$$

as an alternative version of our model.

### 4.4 Result

We compare our model with the Demons algorithm. Demons is a popular and fast iterative method for image registration. Since it is iterative, the number of iterations is manually selected. We choose the number of iterations to 40. We also compare the result with no applied registration to verify the improvements. The average result on the validation data is shown in Table.
Table 1: We compare the average Dice, percentage of negative Jacobian determinants, and execution time with the Demons algorithm. The time is the average execution time for the entire sequence.

| Model      | info         | Avg. Dice | % $|J_\varphi| \leq 0$ | GPU (s) | CPU (s) |
|------------|--------------|-----------|-----------------|---------|---------|
| our        | filtered     | 0.81 ± 0.06 | 2.7% ± 2.7% | 0.09 ± 0.3 · 10^{-2} | 0.22 ± 0.5 · 10^{-2} |
| our reg    | filtered     | 0.81 ± 0.06 | 0.5% ± 1.0% | 0.09 ± 0.3 · 10^{-2} | 0.33 ± 1 · 10^{-2} |
| Demons     | 40 iterations| 0.81 ± 0.07 | 1.2% ± 1.0% | -       | 1.64 ± 2 · 10^{-2} |
| None       | -            | 0.74 ± 0.07 | -              | -       | -       |

4.4.1 Visualization

In Figure 2, we show the result of one sequence where we transform both the image and the human expert segmentation from one time step to the other. In this example the Dice coefficient between the other human expert segmentation at time $t = 16$ and our estimation is $0.732$, the Demons algorithm is $0.715$, and no registration is $0.583$.

Figure 2: Reconstruction and Jacobian for one time sequence using our model and the Demons algorithm. We transfer a segmented region from one time point, $t = 1$ to the other in the sequence. In this example the Dice coefficient between the human expert segmentation at $t = 16$ and our estimate is $0.732$. The Dice coefficient using Demons algorithm is $0.715$. 

119
4.4.2 Latent analysis

From the dynamic in the latent space, we can estimate the filtered, smooth, and predictive distributions. In Figure 3 we show those estimates for one of the latent space dimensions given three examples of input data: all observed values are known, imputation with every 3th sample observed and extrapolation for the last 5. We here observed a continues curve from where it is possible to impute and extrapolate for missing values.

\[ t = [1, \ldots, 50] \quad t = [1, 4, 7, \ldots, 49] \quad t = [1, \ldots, 45] \]

Figure 3: Distributions of the smooth, filtered and one-step prediction distribution given observed data for one of the 16 dimensions in the latent observation space. The blue dashed lines are the observed latent variable for the entire sequence, the distributions are represented by their average (solid line), and standard deviation (shaded region).

4.4.3 Extrapolating image sequence

The model is generative — given an moving image, \( y_M \) we can generate a displacement field by sampling from the initial prior in the latent space and propagating this sample forward in time. Figure 4 illustrates the generated result for one sample.

5 Discussion

We have described an unsupervised method for extracting latent linear dynamics directly from a medical image time series and show how we can reconstruct the displacement field from the lower-dimensional latent linear
dynamic system. With a similar performance in Dice score, our model outperformed the conventional method in speed.

We also illustrate how we can impute missing samples and extrapolate forward in time. Extrapolation support is beneficial for real-time systems when action must be taken given observed data. Good estimates of future states may be necessary to compensate for the system latency. This has e.g. been investigated for the case of real-time adaptive radio therapy where latency in the motion feedback loop has seen to cause a decrease in quality of the treatment [4].

However, we also verified implausible transformations vectors, i.e., Jacobian determinants with non-positive values. Those values mostly appear in regions of the images where the intensity is zero (67%). This is not surprising since we do not handle regularizations of the displacement field directly in the model. We showed that a low-pass Gaussian filter reduces the number of non-negative Jacobian determinants without significantly affecting performance. We estimate the transformation likelihood as an independent Gaussian with fixed variance which is probably a too strong assumption. More likely, the intensities of nearby pixels are correlated, which could be captured for instance by having a Laplacian structured precision matrix. We believe a richer explanation of the transformation likelihood could address this better. Another approach to regularize the transformation could be to estimate a sparse representation of the displacement field and use interpolation techniques [12], like B-splines [32] or thin plate splines [6] to estimate the dense
representation. Furthermore, diffeomorphic deformations can enforced by applying exponential layers \cite{22} to the network.

**Acknowledgement**

This research was supported by Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, and the Swedish Foundation for Strategic Research grant SM19-0029.
Bibliography


### Recent licentiate theses from the Department of Information Technology

<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020-006</td>
<td>David Widmann: <em>Calibration of Probabilistic Predictive Models</em></td>
</tr>
<tr>
<td>2020-005</td>
<td>Anna Wigren: <em>Exploiting Conjugacy in State-Space Models with Sequential Monte Carlo</em></td>
</tr>
<tr>
<td>2020-004</td>
<td>Muhammad Osama: <em>Machine Learning for Spatially Varying Data</em></td>
</tr>
<tr>
<td>2020-003</td>
<td>Christos Sakalis: <em>Securing the Memory Hierarchy from Speculative Side-Channel Attacks</em></td>
</tr>
<tr>
<td>2020-002</td>
<td>Ulrika Sundin: <em>Global Radial Basis Function Collocation Methods for PDEs</em></td>
</tr>
<tr>
<td>2019-006</td>
<td>Kristiina Ausmees: <em>Efficient Computational Methods for Applications in Genomics</em></td>
</tr>
<tr>
<td>2019-005</td>
<td>Carl Jidling: <em>Tailoring Gaussian Processes for Tomographic Reconstruction</em></td>
</tr>
<tr>
<td>2019-004</td>
<td>Amin Kaveh: <em>Local Measures for Probabilistic Networks</em></td>
</tr>
<tr>
<td>2019-003</td>
<td>Viktor Bro: <em>Volterra Modeling of the Human Smooth Pursuit System in Health and Disease</em></td>
</tr>
<tr>
<td>2019-001</td>
<td>Diane Golay: <em>An Invisible Burden: An Experience-Based Approach to Nurses’ Daily Work Life with Healthcare Information Technology</em></td>
</tr>
</tbody>
</table>