Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Hege Bærland

Optimal reparametrization of curves in shape analysis by introducing a deep neural network architecture

Master's thesis in Applied Physics and Mathematics
Supervisor: Elena Celledoni
June 2021

NTNU
Norwegian University of Science and Technology

Hege Bærland

Optimal reparametrization of curves in shape analysis by introducing a deep neural network architecture

Master’s thesis in Applied Physics and Mathematics
Supervisor: Elena Celledoni
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Abstract

Shapes are typically described to be what is left of a object after filtering out translation, rotation and change of size.

A big part of shape analysis consists of defining similarity and differences between shapes in order to perform different applications, such as interpolation. A distance function is then defined, and in this thesis we will use a elastic metric from [37]. Since this metric changes for curves with different parametrization, when comparing two shapes we need to optimize the distance over all reparametrizations. This is an infinite dimensional optimization problem and the main focus of this thesis, where we discretize so that in practice we end up optimizing only over a finite number of parameters.

Several iterative techniques will be tested to solve the optimization problem, such as gradient descent and Adam. Additionally, a deep neural network architecture is introduced to make the optimization more efficient, where the parametrization function is described as a single or multiple composition of an affine transformation and an activation function with a skip connection. The implementation of the discussed algorithms has been performed using Python and the tensor library for deep learning PyTorch [41].

Sammendrag

Former er typisk beskrecet som det som er igjen av et objekt uten å ta hensyn til plassering, rotasjon og størrelse.

En stor del av formanalyse består av å definere likheter og forskjeller mellom formene for å utføre ulike applikasjoner, som for eksempel interpolasjon. En avstandfunksjon er så definert, og i denne avhandlingen vil en elastisk måleenhet fra [37] bli brukt. Siden denne måleenheten endrer seg for kurver med ulik parametrisering, må vi minimere distansen over alle reparametriseringene når to kurver blir sammenlignet. Dette er et uendelig dimensjonalt optimeringsproblem og hovedfokuset i denne avhandlingen, hvor vi diskretiserer slik at i praksis så ender vi opp med å optimere bare over et endelig antall parametere.

Flere iterative teknikker blir utprøvd for å løse optimeringsproblemet, som gradient descent og Adam. I tillegg blir et nevralt nettverk introdusert for å gjøre optimiseringen mer effektiv, hvor parametriseringsfunksjonen blir brukt som aktivasjonsfunksjonen i hvert lag som genererer den nye parametriseringen og dens deriverte. Implementasjonen algoritmene diskutert i avhandlingen har blitt utført ved bruk av Python og tensorbiblioteket for dyp læring, PyTorch [41].

Preface

This thesis concludes a five-year integrated master's degree in Applied Physics and Mathematics at The Norwegian University of Science and Tehcnology (NTNU), with specialization in industrial mathematics.

I would very much like to thank my supervisor, Elena Celledoni, for the valuable guidance and good assistance through the last year.

Trondheim, June 2021
Hege Bærland

Table of Contents

Chapter 1

Introduction

1.1 Background

Shape analysis is a field much studied over the last years, and are applied to several areas. Already in 1917, a paper by D'Arcy Thompson was published, where shapes was described using mathematics and physics. Since then, a numerous of articles have been published with different approaches and methods.

One area where shape analysis is applied is when describing motions of virtual characters on TV or in computer games. They are often represented as skeleton animations, translated into curves and shapes by tracking the positions of the bones throughout the motion. Some examples of applications done related to this topic are interpolation to make the motions cyclic ([11], [14]), comparison of animations ([3]) and identifying movement ([27, 12], using signatures).

Another field where shape analysis is applied is in biomedical image analysis ([16], [15]), where for example one application is to transform one anatomical structure to the other using anatomical curve information.

Shapes are typically described as unparametrized curves in a vector space or on a manifold [4], [11]. A common approach to work with similarity between unparametrized curves is to define shapes as equivalence classes of certain mappings, where the equivalence relation is induced by reparametrization. Given two curves $c_0, c_1 : I \to M$ with $I = [a, b] \subset \mathbb{R}$ and M a vector space or a manifold, the equivalence classes $[c_0], [c_1]$ are defined as

$$c_0 \sim c_1 \iff \exists \varphi : c_0 = c_1 \circ \varphi,$$

where φ is a smooth, strictly increasing bijection[1] on I. The space of parametrized curves containing c_0 and c_1 is denoted by $\mathcal{P}$, which is an infinite dimensional manifold.

[1]A bijection is a function that is both onto and one-to-one, i.e. all elements in the image can be reached and only by one element from the codomain.

We will focus on regular or immersed curves [4], where a regular curve is defined as a curve with a non-vanishing tangent, i.e. $\dot{c}(t) \neq 0$. Hence we can define

$$\mathcal{P} := \mathrm{Imm}(I, M) := \{c \in C^{\infty}(I, M) | \dot{c}(t) \neq 0\},$$

where M is a certain space, f.ex. a manifold or $\mathbb{R}^n$. This shape space could also be defined on closed curves, where usually the interval I is replaced by the unit circle $\mathbb{S}^1$, but this can cause problems/give a more complicated approach in for instance defining a geodesic. Therefore only open curves will be examined in this paper.

The shapes, denoted $[c]$, can finally be collected in the corresponding shape space:

$$\mathcal{S} := \mathcal{P}/\sim .$$

Since we are working on similarity between curves, we need to find a way to measure this similarity, typically a distance function. Distance functions on this shape space are usually obtained from a distance function $d_{\mathcal{P}}$ on the underlying space $\mathcal{P}$ as follows:

$$d_{\mathcal{S}}([c_0], [c_1]) := \inf_{\varphi} d_{\mathcal{P}}(c_0, c_1 \circ \varphi), \tag{1.1}$$

where φ ranges over all possible curve reparametrizations. There are several methods to find the optimal reparametrization, i.e. the reparametrization giving the minimal distance between two shapes. This paper will introduce both a gradient descent and a Adam approach where an activation function depending on parameters A and b will be used to define the diffeomorphism φ, and the optimization problem will be solved with respect to these parameters. In addition, a neural network model with the same principle will be examined.

1.2 Motivation

As already stated, this theory on shape analysis is adapted into the field of motions of characters on TV or in computer games. Then the motion is defined by a skeleton of bones and joints and an animation curve. The distance between different motions are calculated with the distance function (1.1) (which will be elaborated later), and the animation curves are treated as the immerse curves defined in the space $\mathcal{P}$.

Several methods has shown good results for solving this optimization problem, among others dynamic programming and gradient descent ([14], [3]). However, the methods implemented so far have high computational cost, and it is desirable to find a method of reparametrization that minimizes this.

Chapter 2

Literature review

D'Arcy Thompson was the first to give a biophysical explanation of size and shape of organisms in *On Growth and Form* (1917). He explained the shapes of things in nature in terms of physics and mathematics. One well-known idea of his is *'the theory of transformations'*, where one species can transform into another related species by using the transformation of the standard Cartesian grid. This can especially be related to finding a similarity measure for shapes, where one approach is to measure the energy for deforming one of the shapes into the other.

2.1 Shape Analysis

There have been multiple approaches in how to define a shape, for instance by landmarks or as curves or surfaces.

In general it is common to define a shape space in two steps [44]. First a mathematical representation of curves with suitable constraints is often defined as a preshape space, and then elements of the preshape space which preserves the shape by transformations as rotation, translation, scaling and reparametrization are identified. This final quotient space is the desired shape space.

2.1.1 Describing shapes with use of landmarks

Kendall [22] defined in 1984 a shape as what is left after filtering out the effects of translation, rotation and dilatation[1], and he represented them as points in the Euclidean space which is finite dimensional. He used landmarks (later called), which is a finite set of coordinate points that determine the shape.

He defined a shape coordinate-free by saying that it is the representing point, i.e. equivalence class, in the quotient space Σ_m^k defined by

[1]change of size

$$\Sigma_m^k = \{(\mathbb{R}^m)^{k-1} \backslash 0\}/\text{Sim}, \tag{2.1}$$

where Sim is the group of similarities caused by rotations and dilatations.

If the dilatation group first is quotiented out (by replacing z with $z/\|z\|$) and then apply the rotation group SO(m), the final shape space is defined as

$$\Sigma_m^k = \text{S}_m^k/\text{SO}(m). \tag{2.2}$$

Here S_m^k is the unit sphere $\text{S}^{m(k-1)-1}$.

The shape spaces for $m = 1$ or 2 are being called shape-manifolds, and he proves that it is also a C^∞ riemannian manifold. This opens the opportunity for distance functions - a way to measure similarity between shapes.

Although this was a breakthrough in shape analysis, there are some drawbacks. The accuracy of describing a shape by using landmarks is dependent on how the marks are chosen, and it is also often difficult to automate the choice of these landmarks.

2.1.2 Describing shapes as planar curves

Kendall and his colleagues took major steps in shape analysis, but with limitation of describing shapes by using landmarks. In 1998, Younes introduced a new way of defining shapes [52], i.e. represent the boundary of the shape as a continuous, planar curve. They were seen as elements of a infinite-dimensional shape space, and adopting a Riemannian point of view gave possibilities to computation of geodesics and optimal matching between plane curves.

Further Klassen et. al. [25] computed the geodesic between closed curves using numerical algorithms by just considering arc-length parametrized planar curves. They were first to compute the geodesic directly on the space of closed curves and such that it was reparametrization invariant.

Since this, Michor og Mumford [32] and Mennuci [30], [31] among others have examined many different choices for comparing shapes using Riemannian metrics on spaces of planar curves.

2.2 Curve matching

In [42], Sebastian et. al. divided curve alignment methods into two different categories: either based on rigid transformations ([48], [28], [5]) or on nonrigid deformations ([47], [50], [21], [19]). Methods based on rigid transformations match feature points by finding the optimal rotation, translation and scaling parameters, while methods that are based on nonrigid deformations find the mapping from one curve to another which minimizes some ”stretching” and ”bending” energies. Some disadvantages of the first kind of methods are that they are sensitive to occlusion, deformations of parts and other variations in the object form. Of the second type, some disadvantages can be that the curves are treated asymmetrically, they are not rotation and scaling invariant and sensitivity to sampling of the curves and occlusion.

It was Cohen et al. [13] that introduced a deformation-based approach to curve matching. They matched high curvature points on the curves, while a smooth displacement field is maintained.

2.3 Shape space and metric

In this thesis, shapes are defined as curves. According to [6], there exists two different approaches to define the shape space. One option is to use the transitive left action of the set of spatial deformations Φ, i.e. the set of diffeomorphisms on the ambient space, on the set of shapes $\mathcal{S}$

$$\Phi \times \mathcal{S} \to \mathcal{S}, (\varphi, c) \mapsto \varphi \circ c. \tag{2.3}$$

If we equip the space of diffeomorphisms with a Φ-invariant metric, the distance between two curves c_0 and c_1 can be measured as the minimal cost of a deformation transforming one shape into the other

$$d_{\mathcal{S}}(c_0, c_1) = \inf\{d_\Phi(Id, \varphi) | \varphi \in \Phi, \varphi \circ c_0 = c_1\}. \tag{2.4}$$

This option is inconvenient because the entire ambient space is deformed when deforming a curve. Instead, it is possible to quotient out the set of temporal deformations Φ, i.e. the action of reparametrizing a curve. Then the transitive right action of Φ on the space of parametrized curves $\mathcal{M}$ is considered,

$$\mathcal{M} \times \Phi \to \mathcal{M}, c \mapsto c \circ \varphi. \tag{2.5}$$

The distance between two shapes is then measured as the distance between one curve that is fixed parametrized and one curve with optimal reparametrization,

$$d_{\mathcal{S}}(c_0, c_1) = \inf\{d_M(c_0, \varphi \circ c_1) | \varphi \in \Phi\}. \tag{2.6}$$

In this case the ambient space is not affected, therefore will this approach be the focus in this paper.

Now a metric to measure the distance is needed, where several options have been suggested throughout the years.

An important example is the reparametrization invariant L^2-metric,

$$G_c^0(w, z) = \int_0^1 \langle w(t), z(t) \rangle |c'(t)| \mathrm{dt}, w, z \in T_c\mathcal{M}, \tag{2.7}$$

where we can denote $|c'(t)|dt = dl$ as the arc length and simplify the equation. Unfortunately this choice of metric induces a vanishing geodesic distance d_S on the shape space, as was shown in [33]. Michor and Mumford [33] showed that it is possible to connect two shapes in this space with a path that can be made arbitrarily short.

This initiated studies on among others, local metrics, Sobolev metrics and elastic metrics. Sobolev metrics, which are stronger versions of G^0 where linear combinations of higher order derivatives of the tangent vectors are used in the definition of the metric. A family of these metrics for $\mathbb{R}^d$-valued curves has the form

$$G_c^n(w,z) = \int \sum_{i=0}^{n} a_i \langle D_l^i w, D_l^i z \rangle \mathrm{d}l, w, z \in T_c\mathcal{M}. \tag{2.8}$$

Here $D_l w = w'/|c'|$, i.e. the derivative with respect to the arc length, and the a_i's are constants. Sobolev metrics are further studied in among others [45], [29] and [35]. One issue discussed in [35] is completion of the space of curves for the metrics with $n = 1, 2$, including the geodesic equation. The authors of [35] also showed how the Fréchet distance is induced by the "Finsler L^∞-metric"[2].

In [35] it is shown that the geodesic equation for first-order Sobolev metrics is locally but not globally well-posed[3], but global existence of geodesics for Sobolev metrics G^n with $n \geq 2$ was found and proven in [7].

The first-order limit-case

$$G_c^{1,\infty}(w,z) = \frac{1}{L(c)} \int \langle D_l w, D_l z \rangle \mathrm{d}l, w, z \in T_c\mathcal{M}, \tag{2.9}$$

which can be seen as a descendant of [52], was studied in [34]. This is an elastic metric, and can be mapped to an L^2-metric using other coordinates (SRVT, introduced later). In [37] they look at a similar metric, where different weights are added on the tangential $D_l w^T, D_l z^T$ and normal parts $D_l w^N, D_l z^N$ of the arc-length derivatives of the tangent vectors. So the definition of a 2-parameter family of elastic metrics is

$$G_c^{a,b}(w,z) = \int a^2 \langle D_l w^N, D_l z^N \rangle + b^2 \langle D_l w^T, D_l z^T \rangle \mathrm{d}l, w, z \in T_c\mathcal{M}. \tag{2.10}$$

In [20] Joshi et al. introduced a framework convenient for getting an explicit formula for the geodesic linking two shapes in a Riemannian manifold. This framework consists of a transformation called Square Root Velocity Transformation (SRVT), and is applied on closed curves in $\mathbb{R}^n$.

This is further developed in [43], where the SRVT is applied to curves in Euclidean spaces. Then the metric will take the following form

$$d_{L^2}^2(q_0, q_1) = \int_I \|q_1(t) - q_0(t)\|^2 \mathrm{dt}, \tag{2.11}$$

and applied on the shape space $\mathcal{S}$,

$$d_{\mathcal{S}}([c_0],[c_1]) = \inf_{\varphi \in \mathrm{Diff}^+(I)} \sqrt{\int_I \|q_0(t) - q_1(\varphi(t)) \cdot \sqrt{\dot\varphi}\|^2 \mathrm{dt}}. \tag{2.12}$$

[2]the L^∞-norm of the normal projection of the tangent vector

[3]a solution exists, the solution is unique, the solution's behaviour changes continuously with the initial conditions

2.4 Reparametrization and optimization

Solving optimization problem 2.6, two things need to be determined, namely how to reparametrize the curves and which optimization method to use. Alternatively, we can find the optimal reparametrization directly, without defining it first.

There has been several different ways to define a parametrized curve, for instance angle functions, coordinate functions, curvature functions and speed functions. A concrete example is Mio et al. [36] which represents closed curves as a pair of functions, namely an angle function and a speed function.

Reparametrizations, or temporal deformations, of curves is typically defined as increasing diffeomorphisms of the interval I which the curves are defined, i.e. $\Phi = \text{Diff}^+(I)$.

In 1998, Younes [52] used numerical approximation to the curves by polygons, and then found the maximum (he had changed the problem to a maximization problem, which was not concave - hence the approximation was needed) by linear programming when the number of edges in the polygonal approximated curves were small enough, else a suboptimal steepest-descent approach can be used.

Common methods of how to find the optimal reparametrization directly is dynamic programming and gradient descent, represented in the coming subsections.

2.4.1 Dynamic programming

In [49], Ueda and Suzuki implemented a dynamic programming approach to find the optimal matching between shapes. Their main focus was shape learning and shape recognition, which we also consider as applications in this thesis, by defining a convex/concave structure of the shapes. This is being used to generalize the shapes, which is an important feature in shape learning.

First, they transform the shape samples into multiscale representations, where each discrete portion of the shape contours are observed at different viewscales.

To measure the difference/similarity between two shapes A and B, they compare them at different scales (from finest to coarsest) - the more coarse scale you need to use to get similar shapes, the bigger difference between the shapes. This is formulated as a minimization problem where total segment dissimilarity is minimized, i.e.

$$\Psi(A,B) = \min_{\{(i_w,j_w)|w=1,\dots,W\}} \sum_{w=1}^{W} \psi(\mathbf{a}(i_{w-1}+1|i_w), \mathbf{b}(j_{w-1}+1|j_w)). \tag{2.13}$$

Here ψ is the function measuring the dissimilarity between segments $\mathbf{a}(i_{w-1}+1|i_w)$ (inflection points of index $i_{w-1}+1$ to i_w of shape A) and segments $\mathbf{b}(j_{w-1}+1|j_w)$.

This minimization problem is solved using dynamic programming, making a dissimilarity table with $2N$ columns and $2M$ rows, where values $g(i_w, j_w)$ are placed in the (i_w, j_w)th element. Here N is number of inflection points to shape A, and M the same to B. The path from (i_{w-1}, j_{w-1}) to (i_w, j_w) means that $\mathbf{a}(i_{w-1}+1|i_w)$ matches $\mathbf{b}(j_{w-1}+1|j_w)$.

The elements $g(i_w, j_w)$ are calculated by the following recurrence equation,

$$g(i_w, j_w) = \min_{i_{w-1}, j_{w-1}} \{g(i_{w-1}, j_{w-1}) + \psi(\mathbf{a}(i_{w-1}+1|i_w), \mathbf{b}(j_{w-1}+1|j_w))\}. \quad (2.14)$$

This approach is also adopted to yield for other similarity measures/matching methods aswell, where the reparametrization is optimized instead [42] (adpoted to the elastic curve model in [37]). We denote Φ the set of all piecewise linear and increasing homomorphisms $\varphi : I \to I$ with vertices on the grid $I \times I$, and $\Phi_{k,l}$ as the set consisting of all $\varphi \in \Phi$ that satisfy $\varphi(k) = l$. Then we also get a table consisting of values $H(i, j)$ defined as

$$H(i, j) := \min_{\varphi \in \Phi_{i,j}} E(\varphi; 0, 0; i, j), \quad (2.15)$$

where E is a energy functional, i.e. $H(i, j)$ correpsonds to the minimal energy needed to match the curve segments $c_0|_{[0,i]}$ and $c_1|_{[0,j]}$. This energy is in other words the function we want to minimize defined in the metric 3.5, namely

$$E(k, l; i, j) = \int_k^i |q_0 - q_{k,l;i,j}|^2_{\mathbb{R}^d} \mathrm{dt} \quad (2.16)$$

where

$$q_{k,l;i,j}(t) := q_1 \circ \varphi_{k,l;i,j} \sqrt{\frac{j-l}{i-k}} \text{ and } \varphi_{k,l;i,j}(t) := l + (t-k)\frac{j-l}{i-k}. \quad (2.17)$$

To simplify, we note that this can be seen as a recursion

$$H(i, j) = \min_{k,l \in I, k<i, l<j} E(k, l; i, j) + H(k, l). \quad (2.18)$$

Finally, the optimal reparametrization is obtained by backtracking the minimizing indices and use the following formula,

$$\varphi_{i,j}(t) = \begin{cases} \varphi_{k,l;i,j}(t) & t \in [k, i] \\ \varphi_{k,l}(t) & t \in [0, k] \end{cases} \quad (2.19)$$

where (k, l) are the the solution to 2.18.

2.4.2 Gradient descent

Another common approach for solving the optimization problem of finding the optimal reparametrization is a gradient descent approach. According to [44] the computational cost is lower compared to the dynamic programming approach, which is an advantage. However, the gradient descent method has an important limitation, namely that its solution always is local for non-convex problems.

In 1995, Caselles et al. [9] and Kichenassamy et al. [23] studied how to minimize a geometric energy, a generalization of Euclidean arclength, that was defined on curves with respect to the edge-detection problem. This was done by deriving the gradient descent order flow. These articles focus on adjusting a contour/curve to fit a desirable object by minimizing an energy function, and this method can be adapted to minimizing the distance between two shapes, which is the main focus in this paper.

In [9], Caselles et al. show the relation between the energy minimization problem and geodesic computations, which is useful in computing distance between shapes.

In 2005, Sundaramoorthi et al. [45] proceeded on this theory by computing the gradient of the energy function for Sobolev metrics, discovering several drawbacks with the already existing method. The method is still restricted to the segmentation problem, i.e. minimizing a energy functional where a curve is trying to fit an object, while we want it to yield for optimizing a distance between two curves with respect to their reparametrization. This is done in among others [44].

Based on the description in [3], we get the following gradient and method.

The distance between two shapes can be interpreted as the energy needed for one of the shapes to transform into the other. The energy functional in this case is again the function which is we want to optimize (equation), i.e.

$$\mathcal{E}^{\text{op}}(\varphi) := \int_I \left| \frac{\dot{c_0}}{\sqrt{|\dot{c_0}|}} - \sqrt{\dot{\varphi}} \frac{\dot{c_1} \circ \varphi}{\sqrt{|\dot{c_1}| \circ \varphi}} \right|^2 dt = \int_I |q_0 - \sqrt{\dot{\varphi}} \cdot (q_1 \circ \varphi)|^2 dt. \tag{2.20}$$

The variation in direction $\delta\varphi$ and the resulting L^2-gradient of this energy functional is given by Lemma 12 in [3] as

$$\delta\mathcal{E}(\varphi)(\delta\varphi) = \int_I \left\langle q_0 - \sqrt{\dot{\varphi}}(q_1 \circ \varphi), \delta\dot{\varphi}\sqrt{\dot{\varphi}}(q_1 \circ \varphi) - 2\frac{d}{dt}\left(\sqrt{\dot{\varphi}}(q_1 \circ \varphi)\right)\delta\varphi \right\rangle dt \tag{2.21}$$

and

$$\nabla\mathcal{E}(\varphi) = -\left\langle q_0, \frac{\frac{d}{dt}\left(\sqrt{\dot{\varphi}}(q_1 \circ \varphi)\right)}{\dot{\varphi}} \right\rangle + \left\langle \dot{q_0}, \frac{\sqrt{\dot{\varphi}}(q_1 \circ \varphi)}{\dot{\varphi}} \right\rangle. \tag{2.22}$$

Applying this gradient multiplied with a small parameter to the diffeomorphism until the relative error is as small as desired, will give the optimal diffeomorphism and hence the minimal distance.

Chapter 3

Theoretical framework

3.1 Differential geometry

This thesis will include some theory in differential geometry, so firstly some definitions will be introduced.

First a brief definition of a manifold [40]. A manifold is roughly speaking a topological space that locally looks like an open subset of an Euclidean space, but not globally. A more precise definition is given in the appendix.

Then a differentiable manifold $(\mathcal{M}, \mathscr{F})$ conists of a topological manifold $\mathcal{M}$ which is locally an Euclidean space and a differentiable structure $\mathscr{F}$ on $\mathcal{M}$.

Definition [53] *In mathematics, a diffeomorphism $\varphi : \mathcal{M} \mapsto \mathcal{N}$ is an isomorphism of smooth manifolds $\mathcal{M}$ and $\mathcal{N}$. It is an invertible function that maps one differentiable manifold to another, such that both the function and its inverse are smooth.*

In this thesis we mostly deal with diffeomorphisms that are orientation preserving. A definition of an oriented manifold follows here.

Definition [46] *A smooth manifold $\mathcal{M}$ is called orientable if each tangent space of $\mathcal{M}$ can be oriented in a continuous way. In other words, an orientation class*[1] *can be chosen for each tangent space such that the following is satisfied: for each $p \in \mathcal{M}$ there exists an open neighbourhood U and linearly independent vector fields $X_1, .., X_n$ on U such that, for every $q \in U$, $(X_1(q), ..., X_n(q))$ belongs to the orientation class of $T_q\mathcal{M}$.*

Then according to [46], if $\mathcal{M}$ and $\mathcal{N}$ are oriented manifolds, the diffeomorphism $\varphi : \mathcal{M} \mapsto \mathcal{N}$ is orientation preserving if the derivative in p, $d\varphi_p$, preserves orientation at

[1]An orientation class: [46] Two basises of $\mathbb{R}^n$ can be continuously transformed to each other, through a family of basis, if and only if they belong to the same orientation class.

each point $p \in \mathcal{M}$. I.e., whenever $(X_1, ..., X_n)$ is in the orientation class of $T_p\mathcal{M}$, then $(d\varphi_p(X_1), ..., d\varphi_n(X_n))$ is in the orientation class of $T_{\varphi(p)}\mathcal{N}$.

3.2 A Shape Space Metric

Often in applications it is desirable to calculate the distance between two shapes, i.e. determine how different the shapes are. A short debrief of different metrics and improvements is given in the literature review. Now a further derivation of the metric relevant to this thesis will be discussed.

Shapes represent points in a shape manifold, and the distance between two shapes is given as the shortest path linking them - the geodesic. Equipping the manifold with a Riemannian metric provides a convenient framework, because then the geodesic linking two shapes, c_0 and c_1, corresponds to the optimal deformation from c_0 to c_1 [6].

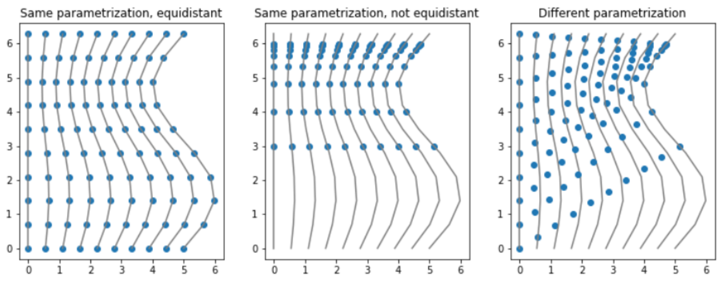


Figure 3.1: Optimal deformations for a reparametrization invariant metric. The dots represents the optimal deformation using the metric, and the lines is the actual optimal deformation.

A suggestion for simplifying the problem of finding the distance, is to require the distance function to be reparametrization invariant[2]. To do this, the group of temporal deformations $\mathcal{T}$, or reparametrizations, is quotiented out. For this we consider the transitive right action of $\mathcal{T}$ on the space of parametrized curves $\mathcal{P}$,

$$\mathcal{P} \times \mathcal{T} \to \mathcal{P}, c \mapsto c \circ \varphi.$$

This does not, however, guarantee that the distance would not change when the curves are reparametrized in different ways, as illustrated in figure 3.1. Therefore, given a $\mathcal{T}$-invariant Riemannian metric on $\mathcal{P}$, we can measure the distance between the shapes of two curves as the distance between one fixed parametrized curve and one optimally reparametrized,

$$d_{\mathcal{S}}(c_0, c_1) = \inf_{\varphi \in \mathcal{T}} d_{\mathcal{P}}(c_0, c_1 \circ \varphi), \tag{3.1}$$

where the same notation is used for the curves and their shapes. The reparametrizations of these curves are increasing diffeomorphisms of the interval I (usually $I = [0, 1]$ or $[0, 2\pi]$), i.e. $\mathcal{T} = \text{Diff}^+(I)$. Now we define the Riemannian metric on $\mathcal{P}$ by equipping each tangent space $T_c\mathcal{P}$ of $\mathcal{P}$ at point c with a scalar product, denoted by

[2]independent of reparametrization.

$$G_c : T_c\mathcal{P} \times T_c\mathcal{P} \to \mathbb{R}, (w, z) \mapsto G_c(w, z).$$

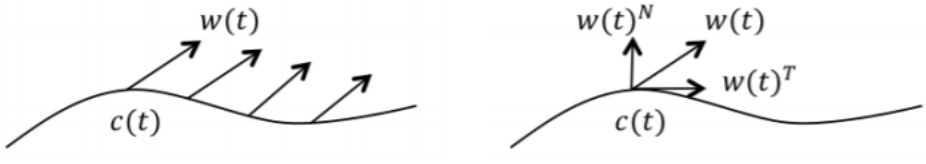


Figure 3.2: Infinitesimal deformation of a curve and its decomposition into tangential and normal parts. Figure taken from [6].

A suggestion for this metric is discussed in among others [4] and [6] (based on [37]), namely the elastic metric

$$G_c^{a,b}(w, z) = \int_M a^2 \langle D_l w^N, D_l z^N \rangle + b^2 \langle D_l w^T, D_l z^T \rangle dl, w, z \in T_c\mathcal{M}. \tag{3.2}$$

This is the most relevant metric because the arc-length derivation and integration $dl = |c'|dt$ guarantee that G is reparametrization invariant. Also because this is mapped to an L^2-metric (using SRVT, introduced later) and gives explicit expressions for the geodesics and a closed form for the geodesic distance. Here $D_l w^N, D_l z^N$ and $D_l w^T, D_l z^T$ are respectively the normal and tangential parts of the arc-length derivatives of the vector fields w and z like in figure 3.2, and a^2 and b^2 are weights.

To interpret this metric in an easier way [37], you can look at the first integrand as a measure in bending as it considers the normal components of w and z, while the second integrand can be seen as a measure of stretching as it only depends on the tangential components. The weights a and b can then be chosen to favour either stretching or bending.

To find an explicit formula, a transformation called the *Square Root Velocity Transformation* (SRVT) is introduced [20], [44]. This transform is given by

$$\mathcal{R}(c) = q := \frac{\dot{c}}{\sqrt{|\dot{c}|}},$$

with its inverse defined as

$$\mathcal{R}^{-1}(q) = \int_0^T q|q|\mathrm{dt}. \tag{3.3}$$

Then by substituting this transform for the curve and using $a = 1$ and $b = \frac{1}{2}$, the elastic distance is just the L^2-distance (shown in [44]),

$$d^2_{G^{1,\frac{1}{2}}}(c_0, c_1) = d^2_{L^2}(q_0, q_1) = \int_I \|q_1(t) - q_0(t)\|^2 \mathrm{dt}. \tag{3.4}$$

Finally, using a property of SRVT [44] stating

$$q(c \circ \varphi) = q(c) \circ \varphi \cdot \sqrt{\dot{\varphi}},$$

the geodesic distance function on the shape space $\mathcal{S} = \mathcal{P}/\text{Diff}^+(I)$ is given by

$$d_{\mathcal{S}}([c_0],[c_1]) = \inf_{\varphi \in \text{Diff}^+(I)} \sqrt{\int_I \|q_0(t) - q_1(\varphi(t)) \cdot \sqrt{\dot{\varphi}}\|^2_{\mathbb{R}^n} \text{dt}}, \tag{3.5}$$

where n is the dimension of the curve, i.e. $c \in \mathbb{R}^n$ (or a n-dimensional manifold). Note that by inducing the SRVT, the metric becomes invariant to translation since now only the derivative of the curve is considered.

The unique geodesic connecting the paths c_0 and c_1 exists if and only if there exists no $t \in [0,1]$ and no $\lambda > 0$ such that $\dot{c_0}(t) = -\lambda \dot{c_1}(t)$. Then the geodesic g is given by

$$g(\tau, t) := \mathcal{R}^{-1}((1-\tau)q_0 + \tau q_1). \tag{3.6}$$

3.3 Modelling character motions

Motions of characters are typically defined by a skeleton and an animation curve, a so-called skeletal animation, and are widely used in movie special effects, tv series and video games. A common approach for creating realistic animations is motion capturing, which is a method of recording an actor's motions and superimpose them on a virtual model.

It is possible to adapt the theory of shapes as planar curves to this area, introduced in [14]. Then the character animations are treated as points on infinite-dimensional Hilbert manifolds and motion spaces are constructed like a shape space, i.e. equivalence classes of animations under reparametrizations.

A further description of the skeletal animation is needed. The skeleton is a hierarchy of bones which are connected by joints inducing a transformation between the bones, and this hierarchy is seen as a directed, acyclic graph where each node has maximum one parent. A global transformation for every bone is achieved by traversing the graph from a root bone and perform all transformations along the graph [14].

Skeletons in computer animation are a simplification of real life skeletons, as bones and joints often are left out, grouped together or new, artificial bones are added. The bones are also given fixed lengths and with one to three degrees of rotational freedom, roll, pitch and yaw. This is called an Euler angle representation, depending on how many rotations the joint is capable of (the wrist can do all three rotations, while the knee only is capable of one). The Euler angles consists of three angles, often referred to as roll, pitch and yaw (ψ, θ, ϕ), which measure the angle with regard to the three axes x, y and z in $\mathbb{R}^3$ [2]. There are also certain constraints related to the joints, for example that the knee can't bend backwards, but this will be ignored for now. The skeleton used in the experiments of chapter 5 is shown in figure 3.3.

Collecting all bones in a set $\mathscr{B}$ and denoting a bone's degrees of freedom as dof(b) for $b \in \mathscr{B}$ allows us to define the joint space $\mathscr{J}$:

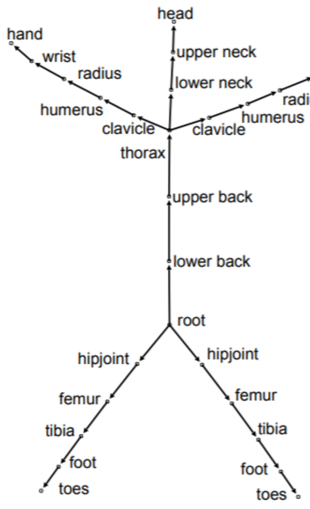


Figure 3.3: An example of the human skeleton used for computer animation. Figure taken from [14], and based on the *CMU Graphics Lab Motion Capture Database* [8].

$$\mathscr{J} := \mathbb{T}^n = \underbrace{\mathbb{S}^1 \times ... \times \mathbb{S}^1}_{n},$$

where $n = \sum_{i \in \mathscr{B}} \mathrm{dof}(i)$ is the total number of degrees of freedom in the skeleton, $\mathbb{T}^n$ denotes the n-dimensional torus and $\mathbb{S}^1$ the unit circle. Now one position of the skeleton is represented as one point in the joint space $\mathscr{J}$. Since this is an Euler angle representation, it neglects an underlying structure of animation curves, namely the Lie group structure. This extended way of representing motions is examined in [11].

The animation curve related to the motion, $\beta : [0,T] \to \mathscr{J}$, is a function that transform every bone in the skeleton for every point in a time interval.

The metric introduced earlier related to shape analysis that penalizes bending and stretching, i.e. equation (3.2), can in this case be interpreted as finding a continuous deformation of one animation into another such that we minimize the sum of changes in angular velocity and angular acceleration. This can be seen as the most energy efficient way of changing from one motion to another.

Like before, the curves are assumed to be immersions, i.e.

$$\mathrm{Imm}(M, \mathbb{R}^n) := \{q \in C^\infty(M, \mathbb{R}^n) : \dot{q}(t) \neq 0 \forall t \in M\},$$

where M is equal the interval $[0,T]$ since we are working with open curves. Since this space is not reparametrization invariant (i.e. equal motions with different reparametrization/speed are two distinct motions in this space), the space of reparametrizations Diff$([0,T])$ is again quotiened out. This leaves the final shape space/motion space to be

$$\mathscr{S} := \mathrm{Imm}([0,T], \mathbb{R}^n)/\mathrm{Diff}([0,T]). \tag{3.7}$$

Then the metric (3.5) can be used to measure the distance between two motions.

3.4 Neural networks

Neural networks are a large class of machine learning models used in many applications such as regression, classification, reinforcement learning and image generation. Machine learning is in general the study of computer algorithms that improve automatically through experience [38]. Neural networks, inspired by studies on how the neurons in the biological brain works, are characterized as a combination of simple, parametric functions between feature spaces [10]. These functions are often referred to as layers in the network. Every neural network consists of a input layer, at least one hidden layer and a output layer. To link these layers together, we use function composition.

If $\mathcal{X}^k$ denote the feature spaces for $k \in \{0, ..., K-1\}$, then a generic layer can be defined as

$$f^k : \mathcal{X}^k \times \Theta^k \to \mathcal{X}^{k+1}.$$

Here Θ^k is the set of possible parameter values of this layer. After defining this, we can define a neural network

$$\begin{aligned} \Psi : \mathcal{X} \times \Theta &\to \mathcal{Y} \\ (x, \theta) &\mapsto z^K \end{aligned} \tag{3.8}$$

as the iteration

$$\begin{aligned} z^0 &= x \\ z^{k+1} &= f^k(z^k, \theta^k), \quad k = 0, ..., K-1 \end{aligned} \tag{3.9}$$

such that $\mathcal{X}^0 = \mathcal{X}$ and $\mathcal{X}^K = \mathcal{Y}$. Here $\theta := (\theta^0, ..., \theta^{K-1}) \in \Theta^0 \times ... \times \Theta^{K-1} =: \Theta$ denotes all of the network's parameters.

A common layer type in neural networks is using an activation function, a simple and nonlinear function. An example are fully-connected layers

$$\begin{aligned} f : \mathbb{R}^M \times (\mathbb{R}^{M' \times M} \times \mathbb{R}^{M'}) &\to \mathbb{R}^{M'} \\ (z, (A, b)) &\mapsto \sigma(Az + b), \end{aligned} \tag{3.10}$$

where the *weight matrix* $A \in \mathbb{R}^{M' \times M}$ and the *bias vector* $b \in \mathbb{R}^{M'}$ are the parameters of the network. The activaion function $\sigma : \mathbb{R}^{M'} \to \mathbb{R}^{M'}$ is usually applied component-wise, e.g. the hyperbolic tangent $[\tanh(z)]_i := \tanh(z_i)$ or the sigmoid function $[\text{sigmoid}(z)]_i = \frac{1}{1+e^{-z_i}}$.

A network with more than one hidden layer is referred to as deep, and gives usually a much better result of what we want the model to learn.

To learn the parameters of a neural network model, a loss function is needed to measure the error.

3.4.1 Residual neural networks

A problem with regular neural networks is that after a number of layers, adding more layers leads to higher training error, which is undesirable. One can think that this is caused by overfitting, but this is in fact not the cause, because overfitting occurs when the training error is lower or as low as a model with fewer layers, but the test error increases. Here both the test and the training error increase. This problem is solved by introducing deep residual neural networks, see [17]. The principle behind this network is that the output of the residual layer is added to the input, i.e. it takes a shortcut over layers.

If the underlying mapping is denoted by $\mathcal{H}(x)$, we define the residual mapping as $\mathcal{F}(x) := \mathcal{H}(x) - x$. So the original mapping is recast into $\mathcal{F}(x) + x$, which is believed to be easier to optimize than the original, unreferenced mapping $\mathcal{H}(x)$ [17].

This residual $\Psi : \mathcal{X} \times \Theta \to \mathcal{X}, \Psi(x, \theta) = z^K$ is in other words defined through the iteration

$$\begin{aligned} z^0 &= x \\ z^{k+1} &= z^k + \sigma(A^k z^k + b^k), k = 0, ..., K-1. \end{aligned} \tag{3.11}$$

This requires that the input and output are of same dimension.

Chapter 4

Method

The main idea of this chapter is to define the diffeomorphism using an activation function. We will consider discrete diffeomorphisms that map the points of a grid $t_0 < t_1 < ... < t_N$ of the interval $[0, T]$ with $t_0 = 0$ and $t_N = 1$ to a new set of points in $[0, T]$, preserving the order and the boundary. The goal is to solve the optimization problem (3.5) on the space of discrete diffeomorphisms instead of $\text{Diff}^+(I)$.

Denote with $\mathbf{t} = [t_1, ..., t_{N-1}]$ the vector containing the gridpoints, the discrete diffeomorphism is then defined as

$$\varphi(\mathbf{t}) = \mathbf{t} + h\sigma(A\mathbf{t} + \mathbf{b}), \tag{4.1}$$

where $\sigma(A\mathbf{t} + \mathbf{b})$ is the activation function, h is a small parameter, A is a matrix and $\mathbf{b}$ a vector. Examples of activation functions that can be used are the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and $\sigma(x) = \tanh(x)$.

Plots of the discrete diffeomorphism when using the two different activation functions are shown in figure 4.1. Here the existence of an inverse of the diffeomorphism is verified.

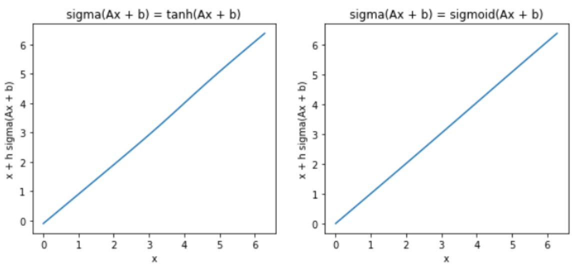


Figure 4.1: A plot of the diffeomorphism $\varphi(\mathbf{t})$ as a function of $\mathbf{t}$, for the two different activation functions. Here A is the identity matrix and $\mathbf{b}$ is a vector of 4 times negative ones, $N = 50$, $h = 0.1$.

Resulting from this definition of the diffeomorphism, the problem that will be solved in this thesis is

$$\min_{A,b} \int_I \|q_0(t) - \sqrt{\dot{\varphi}(t,A,b)}q_1(\varphi(t,A,b))\|^2 \mathrm{dt}. \tag{4.2}$$

Preservation of order and boundary

The activation functions themselves are increasing, but this does not guarantee preserving the order of the grid points. For this to be preserved, some restrictions need to be set to the matrix A and the vector b. This is illustrated in figure 4.2, where we see that the order is not preserved if A is a random generated matrix, as opposed to if A is a stochastic or the identity matrix. The vector **b** is equal $-4 \cdot \mathbb{1}$, where $\mathbb{1}$ is a vector containing only 1's. These parameters are chosen such that the initial solution for equal shapes computed in the experiments is close to the optimal. The preservation of order will be examined further in the experiments introduced in chapter 5.

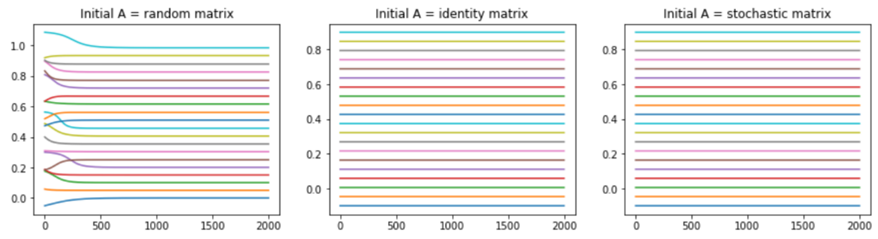


Figure 4.2: The t_i-values are plotted for $i = 1, ..., 20$ as a function of iterations. The t_i-values are within the interval $[0,1]$ as shown on the y-axis, and the number of iterations is represented on the x-axis.

In addition, we want the function $\varphi(\mathbf{t})$ to be boundary preserving (i.e. stay in the interval $[0,T]$). One option to guarantee this is to add a function f such that $f(0) = f(T) = 0$, for instance $f(x) = x(T-x)$,

$$\varphi(\mathbf{t}) = \mathbf{t} + h\mathbf{t}(T-\mathbf{t})\sigma(A\mathbf{t}+\mathbf{b}). \tag{4.3}$$

This, however, yields a bit more complicated derivative of the function, namely

$$\dot{\varphi}(\mathbf{t}) = 1 + h\mathbf{t}(T-\mathbf{t})A\dot{\sigma}(A\mathbf{t}+\mathbf{b}) + h(T-2\mathbf{t})\sigma(A\mathbf{t}+b).$$

Another option is to assume $\varphi(0) = 0$ and $\varphi(T) = T$ and do the computations just on the inner grid. This has shown to be the best approach.

Preservation of orientation

Next, the property of being orientation preserving (definition in section 3.1) is examined. Since the focus is on curves in $\mathbb{R}^n$, all that is needed is that $\dot{\varphi}(\mathbf{t}) > 0$. This follows from the fact that for $c(\mathbf{t})$ and $c(\varphi(\mathbf{t}))$ to have the same orientation, the tangent vector $\dot{c}(\varphi)\dot{\varphi}(\mathbf{t})$ needs to have same direction as $\dot{c}(\varphi)$.

The derivative of $\varphi(\mathbf{t})$ is given as

$$\dot{\varphi}(\mathbf{t}) = 1 + hA\dot{\sigma}(A\mathbf{t} + \mathbf{b}), \tag{4.4}$$

and this is bigger than 0 for $h < \min \left| \frac{1}{A\dot{\sigma}(A\mathbf{t}+\mathbf{b})} \right|$. This is usually a large number, so using $h < 1$ will normally fulfill this condition.

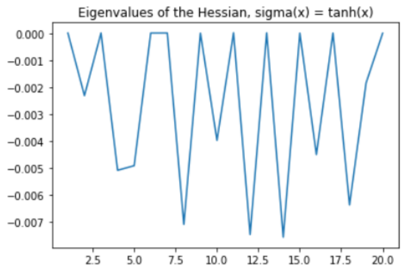
Convexity of the problem

Then the convexity of the problem is examined. There is a proposition saying that a twice differentiable function $f : \mathbb{R}^{n\times n} \times \mathbb{R}^n \mapsto \mathbb{R}$ is convex if and only if its Hessian $\nabla^2 f(x)$ is positive semi-definite for all $x \in \mathbb{R}^{n\times n} \times \mathbb{R}^n$ [18]. To know if a symmetric matrix is positive semi-definite, the eigenvalues need to be non-negative.

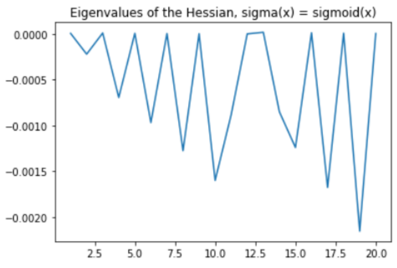
The Hessian is defined as

$$H = \begin{bmatrix} \frac{\partial^2 d}{\partial A^2} & \frac{\partial^2 d}{\partial A \partial b} \\ \\ \frac{\partial^2 d}{\partial b^2} & \frac{\partial^2 d}{\partial b \partial A} \end{bmatrix}, \tag{4.5}$$

and we can see from the plots (figure 4.3) that it has negative eigenvalues. The Hessian is calculated using the initial parameters of A and b of the experiments in chapter 5, i.e. $A = I$ and $\mathbf{b} = -4 \cdot \mathbb{1}$. Hence the Hessian is not positive semi-definite for all $x = (A, b) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n$, and we can conclude that this problem is non-convex.



(a) The eigenvalues of the Hessian matrix plotted for $\sigma(x) = \tanh(x)$.



(b) The eigenvalues of the Hessian matrix plotted for $\sigma(x) = \text{sigmoid}(x)$.

Figure 4.3

4.1 Gradient descent

To use this diffeomorphism to solve the problem (3.5), a gradient descent approach with respect to the variables A and b is introduced to find the minima of the function

$$d^2 = \int_I \|q_0(t) - \sqrt{\dot{\varphi}(t, A, b)}q_1(\varphi(t, A, b))\|^2 \text{dt}. \tag{4.6}$$

Instead of computing the gradient of this distance like discussed in section 2.4.2 in the literature review, the gradient is found by an integrated function in python called autograd. The integral in the distance formula is approximated using Simpson's rule.

A disadvantage of this method is that we need to find initial parameters close enough to the solution. This is difficult if we don't know the parametrization giving the optimal result. Since we are working with a non-convex problem, another challenge is that the solution could only be a local minimum.

4.2 Adam method

Another method for finding the minima of a function with respect to some parameters is Adam's method. An advantage of this method, is that only the first-order gradients are required, and it has few memory requirements and is computational efficient. The method is derived in [24].

Let $f(\theta)$ be a differentiable scalar function with respect to parameters θ. Then we want to minimize the expected value of this function with respect to the parameters. Denote the realizations (the value X attains when one of the possibilities did happen) of the function at timesteps $1, ..., T$ as $f_1(\theta), ..., f_T(\theta)$. Then the gradient evaluated at timestep t is denoted $g_t = \nabla_\theta f_t(\theta)$, and the algorithm is as follows

$$\begin{aligned}
&\text{while not converged:}\\
&g_t = \nabla_\theta f_t(\theta_{t-1})\\
&m^t = \frac{1}{1-\beta_1^t}(\beta_1 m^{t-1} + (1-\beta_1)g_t\\
&v^t = \frac{1}{1-\beta_2^t}(\beta_1 v^{t-1} + (1-\beta_2)g_t^2\\
&\\
&\theta^t = \theta^{t-1} - \alpha\frac{m^t}{\sqrt{v^t}+\epsilon}.
\end{aligned} \tag{4.7}$$

Here the parameters are set beforehand with $\alpha > 0$, $0 < \beta_1, \beta_2 < 1$ and $\epsilon > 0$ small. What the algorithm does is updating the exponential moving means[1] of the gradient (m_t) and its square (v_t) where β_1 and β_2 control the exponential decay rates of these means [24].

In our case, the parameters that are being optimized are $\theta = [A, b]$ and the function f is the distance function defined in (3.5), so $g_t = \nabla_\theta d_{\mathcal{P}}(c_0(t), c_1 \circ \varphi(t, \theta))$.

[1]a calculation to analyze data points by creating a series of averages of different subsets of the full data set

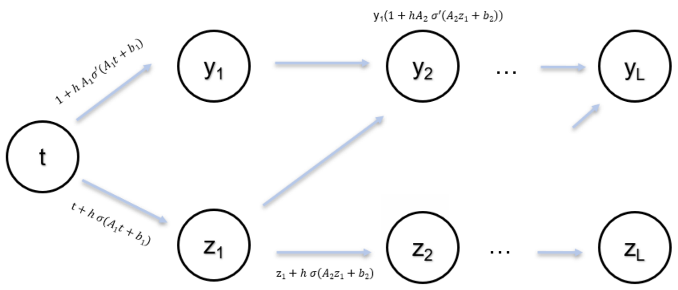

$1 + h A_1\sigma'(A_1t + b_1)$
$y_1(1 + hA_2\, \sigma'(A_2z_1 + b_2))$
$t + h\,\sigma(A_1t + b_1)$
$z_1 + h\,\sigma(A_2z_1 + b_2)$
t
y_1
y_2
y_L
z_1
z_2
z_L
…
…

Figure 4.4: A neural network model computing z and y consisting of L layers.

4.3 Neural networks

In [1], it says that we can approach an arbitrary diffeomorphism i Diff(Ω) (Ω is a compact manifold) by combining a finite number of elementary diffeomorphisms. Since the structure of residual neural networks with one layer look similar to our discrete diffeomorphism, an idea is to combine several diffeomorphisms giving the structure of a residual neural network with several layers.

Then the distance function 4.6 is used as the loss function of the model, and Stochastic Gradient Descent or Adam is the optimizer. The parameters are $\theta = (A_1, b_1, ..., A_K, b_K)$ and the feature space is $\mathcal{X} = I \times ... \times I$ K times, where K is the number of layers. I.e., $t \in \mathcal{X}$ is a vector of increasing elements, all within in the interval I, typically $I = [0, 1]$ or $I = [0, 2\pi]$.

The initial parameters A_k and b_k for $k = 1, ..., K$ are now determined by the model, and have no longer restrictions. The order seem still to be preserved by the model, showed in the experiments of chapter 5.

We also need to update the derivative in the layers, and the model becomes

$$\begin{aligned} z^{k+1} &= z^k + h\sigma(A_k z^k + b_k) \\ y^{k+1} &= y^k(1 + hA\dot{\sigma}(A_k z^k + b_k)) \end{aligned} \tag{4.8}$$

for k = 1, ..., K and the loss function is

$$\text{loss} = \int_0^1 |q_0(t) - \sqrt{y^K} q_1(z^K)|^2 \text{dt}. \tag{4.9}$$

A scetch of the model is shown in figure 4.4.

Another suggestion is to first define $\varphi'(t_j) = y_j$ and assume this to be piece-wise constant in the intervals $[t_{j-1}, t_j)$. Then this is integrated to find z. So we define

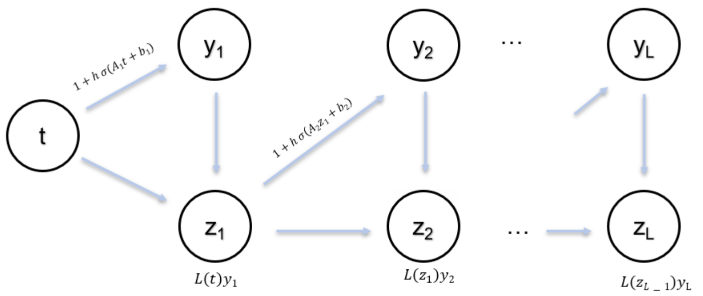


Figure 4.5: A neural network model computing z and y using L layers.

$$y = 1 + h\sigma(At + b), \tag{4.10}$$

where A and b still are the parameters we optimize on. By approximating the integral of this, we get

$$\varphi(t) = \int_0^t \varphi'(\xi)d\xi = \sum_{i=1}^{j} \varphi'(t_i)(t_i - t_{i-1}) + (t - t_j)\varphi'(t_j). \tag{4.11}$$

This is the midpoint-rule just evaluated in the last point of the interval instead of the midpoint. Setting $y_j = \varphi'(t_j)$, gives

$$z_j = \sum_{i=1}^{j} y_i(t_i - t_{i-1}), \tag{4.12}$$

and in vectorform

$$z = Ly, L = \begin{bmatrix} t_1 - t_0 & 0 & \dots & \dots & \dots \\ x_t - t_0 & t_2 - t_1 & 0 & \dots & \dots \\ t_1 - t_0 & t_2 - t_1 & t_3 - t_2 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ t_1 - t_0 & t_2 - t_1 & t_3 - t_2 & \dots & t_N - t_{N-1} \end{bmatrix} \tag{4.13}$$

This model will then look like

$$\begin{aligned} y^{k+1} = 1 + h\sigma(A_k z^k + b_k)) \\ z^{k+1} = y^{k+1}(Lz^k) \end{aligned} \tag{4.14}$$

and is illustrated in figure 4.5.

Chapter 5

Experiments and Results

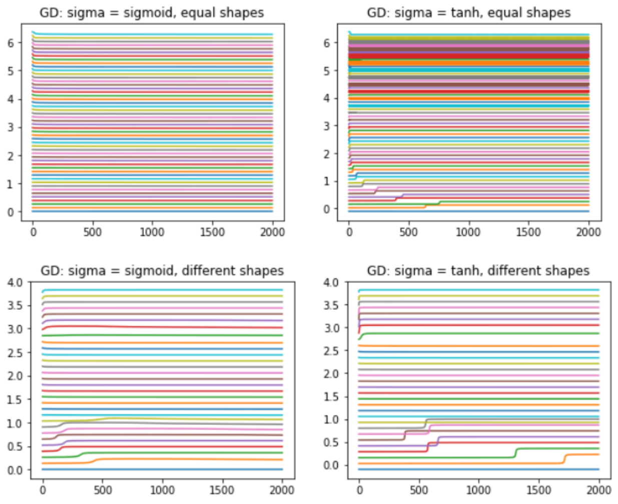
5.1 Curves in $\mathbb{R}^3$

In this section, some experiments that are performed on both equal and different shapes will be introduced. When equal shapes are considered, the curves are defined as $c_0 = c_1 = [t\text{sin}(\pi t), t\text{cos}(\pi t), t]$. For different curves, $c_0 = [t\text{sin}(\pi t), t\text{cos}(\pi t), t]$ and $c_1 = [t\text{sin}(t), t\text{cos}(t), t]$ are used. Here, $[c_0]$ and $[c_1]$ are the corresponding shapes. In all figures, the red parametrization will represent c_0 and the green will represent c_1.

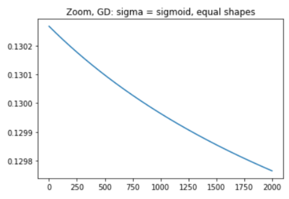
5.1.1 Gradient descent

In the following section, some experiments using the gradient descent method described in chapter 4.1 computed on both equal and different shapes are shown. The initial parametrizations used are equidistant points **t** (this is the one held constant) and $\varphi(\mathbf{t}) = \mathbf{t} + h\sigma(A\mathbf{t} + \mathbf{b})$ (this is the one being optimized) with the initial parameters $A = I$ and $b = -4 \cdot \mathbb{1}$. Here $\mathbf{v} = A\mathbf{t} + \mathbf{b}$ is a vector, and the function is performed component-wise.

From figure 5.1a it is clear that the algorithm preserves the order for only $\sigma(A\mathbf{t} + \mathbf{b}) = \text{sigmoid}(A\mathbf{t} + \mathbf{b}) = \frac{1}{1+e^{-(A\mathbf{t}+\mathbf{b})}}$, hence this is the activation function used in the experiments.



(a) The t_i-values are plotted for $i = 1, ..., N$ throughout the gradient descent algorithm, where $N = 50$ for equal shapes and $N = 30$ for different shapes. The t_i-values are within the interval $[0, 2\pi]$ as shown on the y-axis, and the number of iterations is represented on the x-axis.

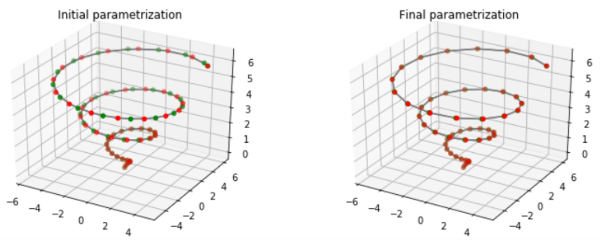


(b) Here only t_1 is plotted throughout the 2000 iterations to illustrate that the values are not constant as it seems in the left figure.

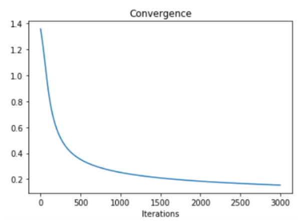
Figure 5.1

Equal shapes

To check if the method works, a convenient way to start is to compare two equal shapes with different initial parametrization. Then the optimal reparametrization is known, and it is easy to test the performance of the method.



(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \text{sigmoid}(x)$.



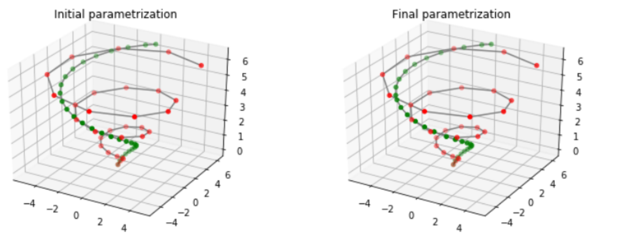
(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

Figure 5.2

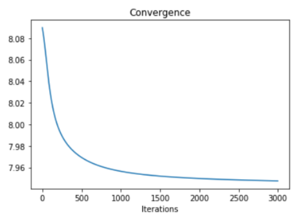
In figure 5.2, the red parametrization belong to the curve c_1 and is being optimized, while the green parametrization represents c_0 and is held constant. We see that the final solution of the red parametrization overlap the green parametrization, which was the desire in this case. Hence we can conclude that the method is working.

Different shapes

It is desirable to see if the method also works for different shapes. In this example it is difficult to know if the initial solution is close enough to the optimal solution. As known from the theory [18], we have seen also in our experiments that gradient descent works well if we start with an initial guess that is close enough to the optimal solution.



(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \text{sigmoid}(x)$.



(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

Figure 5.3

The method is converging also for different shapes, as seen in figure 5.3b. However, because of the non-convexity of the problem discussed in chapter 4, it is not guaranteed that this is in fact a global minimum. The gradient descent algorithm will in this case only find the solution closest to the initial suggestion.

5.1.2 Adam method

Again, experiments will be done on both equal and different shapes, and with both $\sigma(At + b) = \text{sigmoid}(At+b) = \frac{1}{1+e^{-(At+b)}}$ and $\sigma(At+b) = \tanh(At+b)$. The initial parametrizations used are equidistant points $\mathbf{t}$ and $\varphi(\mathbf{t}) = t + h\sigma(A\mathbf{t} + b)$, where A is a random generated diagonal matrix and $b = -\mathbb{1}$ for three of the experiments. However, for equal shapes and $\sigma(At + b) = \tanh(At + b)$ a change in initial matrix A was needed in order to converge, namely $A = -0.8 \cdot I$, where I is the identity matrix. The preservation of order is fulfilled, shown in figure 5.4.

The algorithm described in (4.7) is computed for the parameters $\alpha = 0.001$, $\beta_1 = 0.88$, $\beta_2 = 0.8$ and $\epsilon = 10e - 8$ for three of the methods, except for the experiment with equal shapes and $\sigma(At + b) = \tanh(At + b)$. Then $\alpha = 0.0009$, $\beta_1 = 0.88$, $\beta_2 = 0.93$ and $\epsilon = 10e - 8$, also in order to achieve convergence.

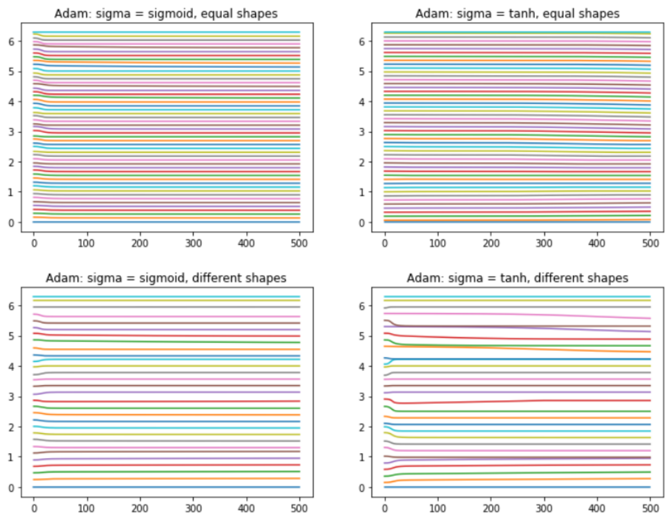
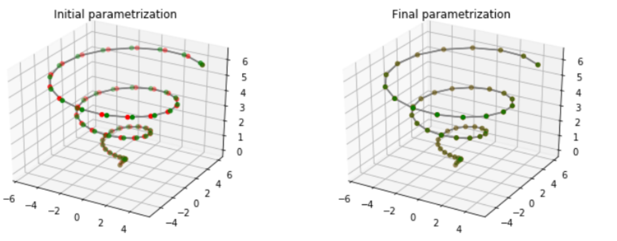
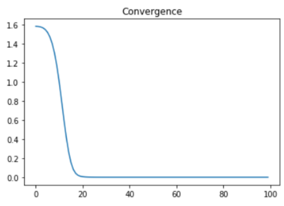


Figure 5.4: The t_i-values are plotted for $i = 1, ..., N$ throughout the Adam algorithm, where $N = 50$ for equal shapes and $N = 30$ for different shapes. The t_i-values are within the interval $[0, 2\pi]$ as shown on the y-axis, and the number of iterations is represented on the x-axis.

Equal shapes

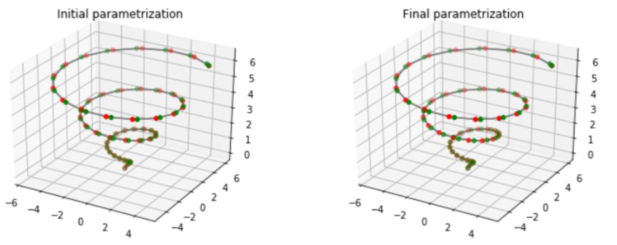


(**a**) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \text{sigmoid}(x)$

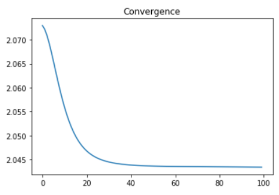


(**b**) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

Figure 5.5



(**a**) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \tanh(x)$.

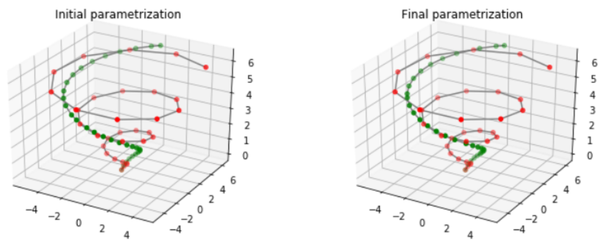


(**b**) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

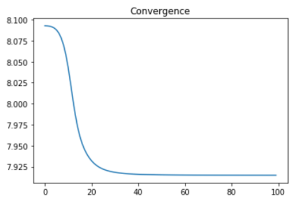
Figure 5.6

This method converges after few iterations and gives a good result for $\sigma(x) = \text{sigmoid}(x)$ in figure 5.5. For $\sigma(x) = \tanh(x)$ it also converge after few iterations, but the final distance and optimal reparametrization is not as satisfying.

Different shapes

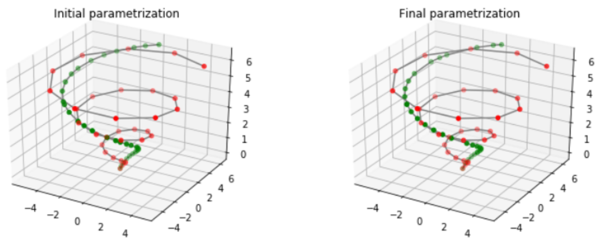


(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \text{sigmoid}(x)$.

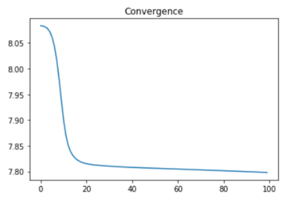


(b) A convergence plot of the method.

Figure 5.7



(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \tanh(x)$.



(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

Figure 5.8

The method with $\sigma(x) = \text{sigmoid}(x)$ is performing well also here. There is still fast convergence, but harder to know if the result is the global minimum or only a local one. Compared to gradient descent, this method attains a approximately the same minimum value of the distance function.

5.1.3 Neural networks

Now we introduce some experiments using the neural network architecture, both the residual neural network and alternative neural network described in respectively (4.8) and (4.14). All figures are computed for 5 layers and with Adam as optimizer. A confirmation for preserved order is shown in figure 5.9, 5.10 and 5.11.

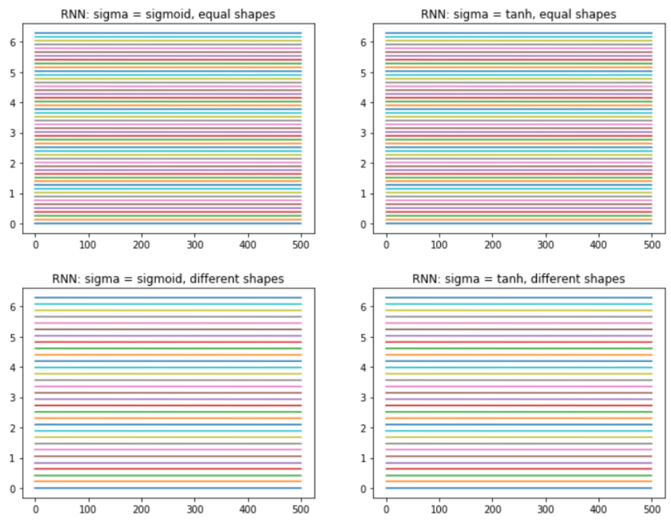
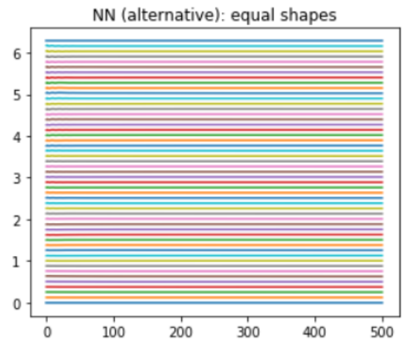
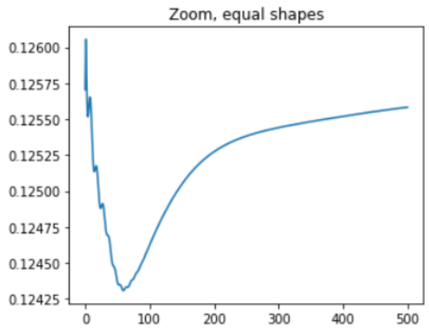


Figure 5.9: The t_i-values are plotted for $i = 1, ..., N$, where $N = 50$ for equal shapes and $N = 30$ for different shapes. The t_i-values are within the interval $[0, 2\pi]$ as shown on the y-axis, and the number of iterations is represented on the x-axis.

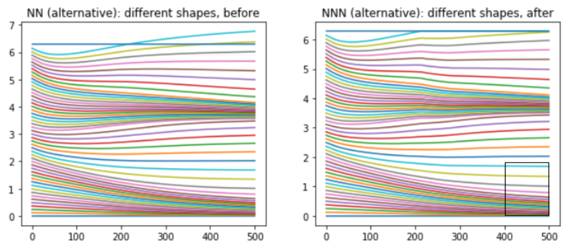


(a) The t_i-values are plotted for $i = 1, ..., N$, where $N = 50$. The t_i-values are within the interval $[0, 2\pi]$ as shown on the y-axis, and the number of iterations is represented on the x-axis.

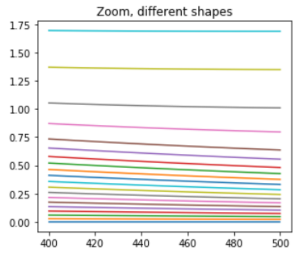


(b) Here only t_1 is plotted throughout the 2000 iterations to illustrate that the values are not constant as it seems in the left figure.

Figure 5.10



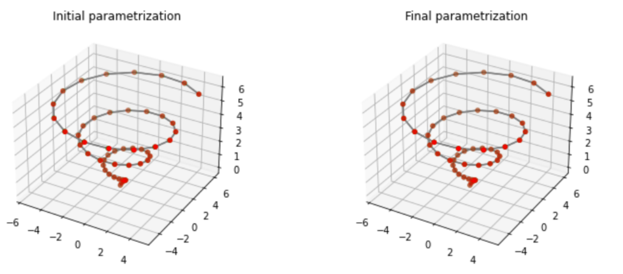
(a) The t_i-values are plotted for $i = 1, ..., N$, where $N = 30$. The t_i-values are within the interval $[0, 2\pi]$ as shown on the y-axis, and the number of iterations is represented on the x-axis.



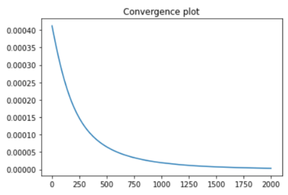
(b) Here is a small cut of the left figure (the black square) to assure the preservation is fulfilled.

Figure 5.11

Equal shapes

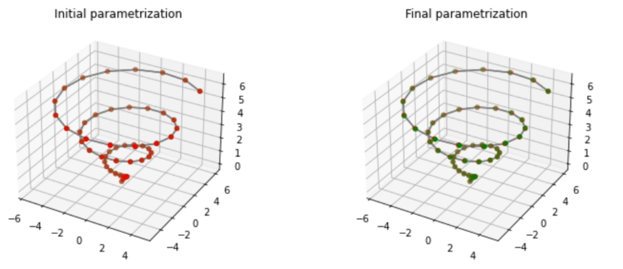


(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \text{sigmoid}(x)$.

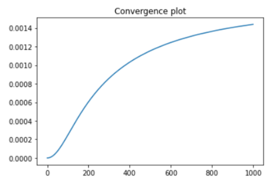


(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

Figure 5.12



(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \tanh(x)$.



(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

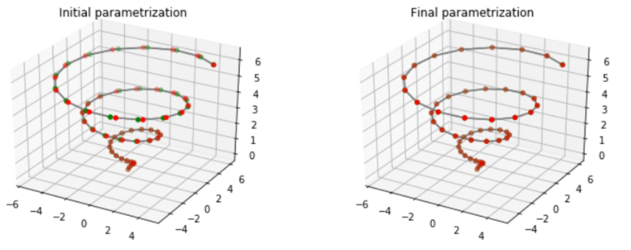
Figure 5.13

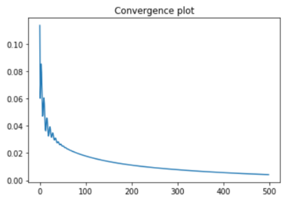
The method performed in figure 5.14 is the one with fastest convergence of the neural network approaches.

Generally we see that the neural networks converges faster than the gradient method and usually gives a better result, except for the residual neural network with $\sigma(x) = \tanh(x)$, seen in figure 5.13b.

The red and green parametrization, of respectively the curves c_0 and c_1, are overlapping already from the start, which means that the model determine initial parameters A and b close to the optimal. This can also be a reason for the good results.

The distance with respect to number of layers are also plotted for the different methods, represented in figure 5.15 and 5.16. We see that the distance has a tendency of increasing for several layers.

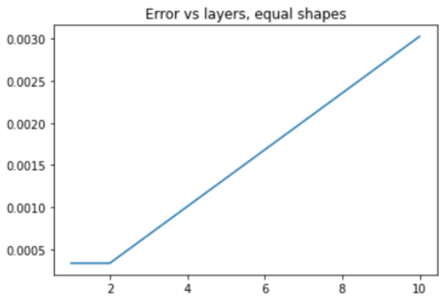


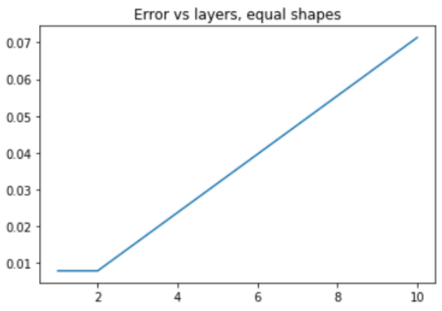


(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. This is for the alternative formulation, i.e. $\dot{\varphi}(t) = 1 + h\sigma(At + b)$, where $\sigma(x) =$ sigmoid(x).

(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

Figure 5.14





(a) A plot of the distance (y-axis) between equal shapes as a function of number of layers (x-axis). The distance is computed using 4.8 with $\sigma(x) =$ sigmoid(x).

(b) A plot of the distance (y-axis) between equal shapes as a function of number of layers (x-axis). The distance is computed using 4.8 with $\sigma(x) =$ tanh(x).

Figure 5.15

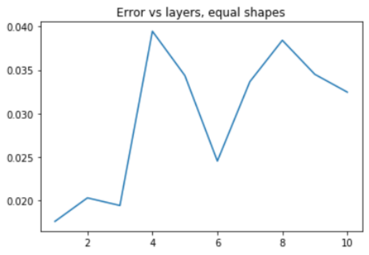
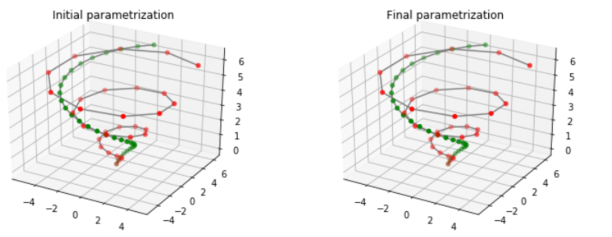
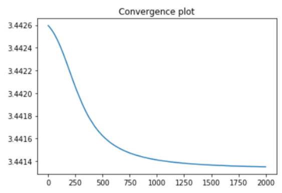


Figure 5.16: A plot of the distance (y-axis) between equal shapes as a function of number of layers (x-axis).The distance is computed using 4.14 with $\sigma(x) =$ sigmoid(x).

Different shapes

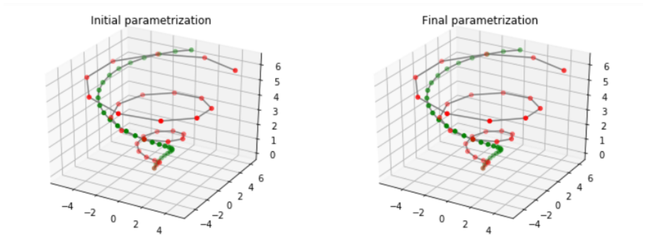


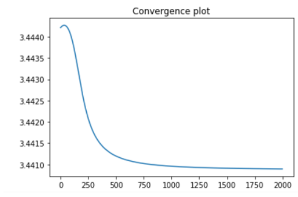


(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \text{sigmoid}(x)$, and the optimizer used is Adam.

(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

Figure 5.17





(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. Here $\varphi(t) = t + h\sigma(At + b)$ with $\sigma(x) = \tanh(x)$.

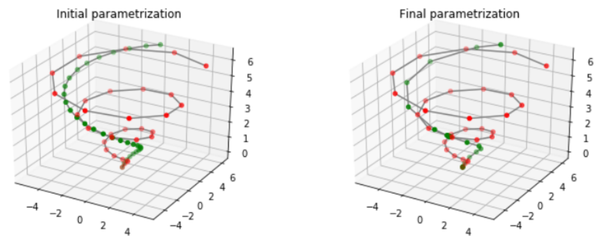
(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

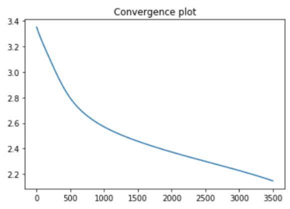
Figure 5.18

Figure 5.17 and 5.18 shows that the convergence is a bit faster when using the activation function $\sigma(x) = \tanh(x)$ than $\sigma(x) = \text{sigmoid}(x)$. The final result is approximately the same for the choice of both activation functions, and with smaller minimum distances than for the gradient descent method.

The second approach using neural networks is shown in figure 5.19, and we can see that this method gives the smallest distance between two different shapes. However, there is no clear convergence in this case.

Again, the distance with respect to number of layers are plotted for the different methods. This is represented in figure 5.20 and 5.21, and tells us that the distance decreases using more layers.

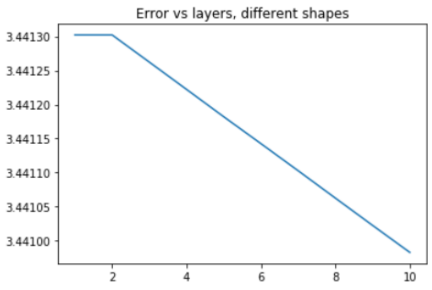




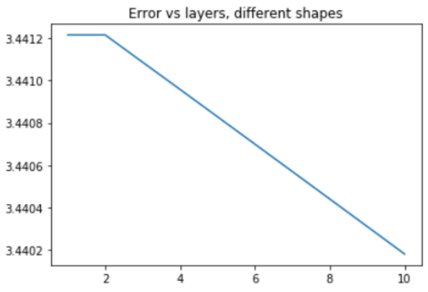
(a) The parametrization of the curves before and after the optimization. The red parametrization is constant, and the green is optimized. This is for the alternative formulation, i.e. $\dot{\varphi}(t) = 1 + h\sigma(At + b)$, where $\sigma(x) = \tanh(x)$.

(b) A convergence plot of the method. The distance are shown on the y-axis, and the iterations on the x-axis.

Figure 5.19



(a) A plot of the distance (y-axis) between different shapes as a function of number of layers (x-axis). The distance is computed using 4.8 with $\sigma(x) = \text{sigmoid}(x)$.



(b) A plot of the distance (y-axis) between different shapes as a function of number of layers (x-axis). The distance is computed using 4.8 with $\sigma(x) = \tanh(x)$.

Figure 5.20

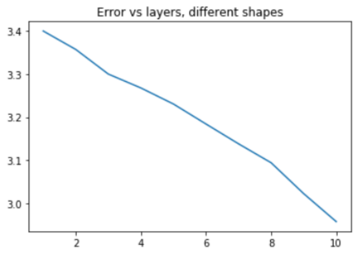
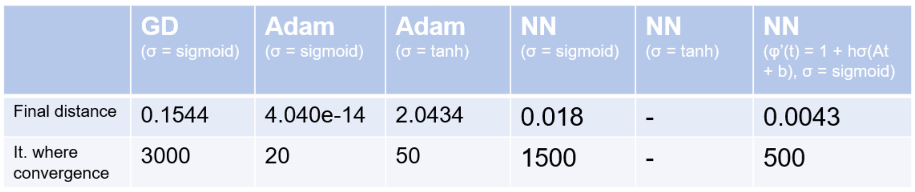


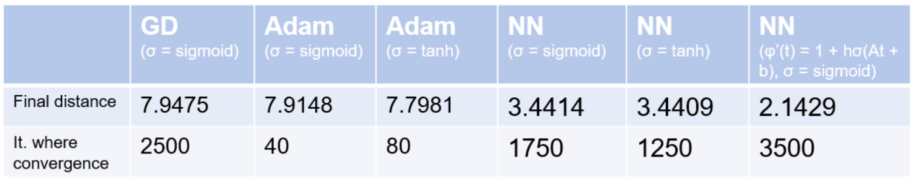
Figure 5.21: A plot of the distance (y-axis) between different shapes as a function of number of layers (x-axis). The distance is computed using 4.14 with $\sigma(x) = \text{sigmoid}(x)$.

5.1.4 Summary



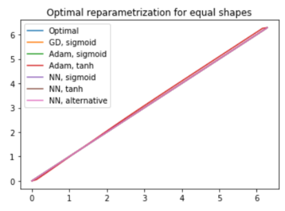
	GD (σ = sigmoid)	Adam (σ = sigmoid)	Adam (σ = tanh)	NN (σ = sigmoid)	NN (σ = tanh)	NN (φ'(t) = 1 + hσ(At + b), σ = sigmoid)
Final distance	0.1544	4.040e-14	2.0434	0.018	-	0.0043
It. where convergence	3000	20	50	1500	-	500

Figure 5.22: A table with a overview of the methods when considering equal shapes.

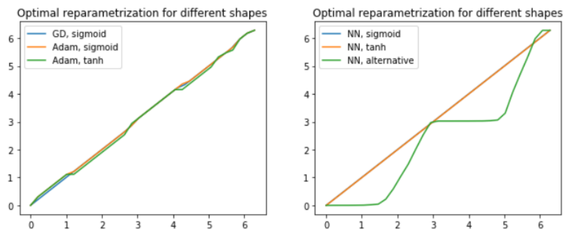


	GD (σ = sigmoid)	Adam (σ = sigmoid)	Adam (σ = tanh)	NN (σ = sigmoid)	NN (σ = tanh)	NN (φ'(t) = 1 + hσ(At + b), σ = sigmoid)
Final distance	7.9475	7.9148	7.7981	3.4414	3.4409	2.1429
It. where convergence	2500	40	80	1750	1250	3500

Figure 5.23: A table with a overview of the methods when considering different shapes.



(a) A plot of all the optimal reparametrizations for the different methods, when comparing equal shapes.



(b) A plot of all the optimal reparametrizations for the different methods, when comparing different shapes.

Figure 5.24

The tables 5.22 and 5.23 show that the Adam method with $\sigma(x) = \text{sigmoid}(x)$ is the one with fastest convergence for both cases, and has the best result for equal shapes. However, the result for the same method when considering different shapes is not optimal compared to the other methods. This can be due to the fact that the function being optimized is non-convex, hence the final result could only be a local solution.

In figure 5.24, the reparametrization giving the minimized distance between both equal and different shapes are plotted for the different methods. When comparing equal shapes, we see that most of the methods have reached the optimal parametrization. The exception is the reparametrization achieved with the Adam method and $\sigma(x) = \tanh(x)$, which have some deviation. This coincides with the value for this method in the table as well.

For different shapes there is no optimal reparametrization to compare with, but it seems that the methods achieving approximately the same minimum distance also achieves the same reparametrization. The alternative neural network approach defined in 4.14 is the reparametrization that deviate the most from the others, but also have the smallest minimum distance.

The figures that compare the distance with number of layers (5.15, 5.16, 5.20 and 5.21) show that for equal shapes the distance increases when using more layers, but decreases when comparing different shapes.

5.2 Animation curves from motion capturing

Finally, some experiments are performed on human motions, using the dataset *CMU Graphics Lab Motion Capture Database* [8]. This is a dataset consisting of 144 subjects performing various movements with labels on each movement. Each subject has one file (asf: Acclaim Skeleton File) consisting of the skeleton, i.e. length of the bones and which joints that are connected, in addition to a set of files (amc: Acclaim Motion Capture data) with one or several movements given as rotations of the joints in Euler angles over time.

The animation curves are extracted from the dataset using the code of Lystad [26], represented as 50-dimensional curves with distinct number of frames N, where N vary from 130 to approximately 420. I.e., each animation curve is a $n \times N$-matrix, where $n = 50$ is the total number of degrees of freedom in every position. The frames are translated into time points on the interval $I = [0, 1]$. The derivative of the curves is needed to transform the curves into SRVT, which we define as $\dot{c}(t_i) = \frac{c(t_{i+1}) - c(t_i)}{t_{i+1} - t_i}$. The action of applying the diffeomorphism to the SRVT of the curve, $q \circ \varphi$, is done by linear interpolation for every degree of freedom, i.e.

$$q \circ \varphi(t) = q_{i,j} + (\varphi(t_j) - t_j)\frac{q_{i,j+1} - q_{i,j}}{t_{j+1} - t_j}, \forall i = 1, ..., 50 \text{ and } j = 1, ..., N.$$

Here $q_{i,j}$ is the ij-th element of the matrix representing the animation curve. It is assumed that the new time point $\varphi(t_j) = t_j + h\sigma(\sum_{i=1}^{N} A_{ij} t_j + b_j)$ lies between t_j and t_{j+1} because of the small parameter h.

A suitable way of measuring similarity when dealing with several objects is cluster analysis. This is a method that assembles objects with similar features into groups/clusters. One way of doing this is using classical multidimensional scaling [51], where the given input is a distance $m \times m$-matrix with elements d_{ij} equal to the distance between object i and j of the total of m objects. Note that this matrix is symmetric and with zeros on the diagonal. Then a set of vectors $\mathbf{x}_i, i \in 0, ..., m-1$ with optional dimension k (usually k = 2 or 3) is found such that

$$\|\mathbf{x}_i - \mathbf{x}_j\| \approx d_{ij} \; \forall i, j \in 0, ..., m-1.$$

This is done on 9 objects in total, divided into three different movements which is ”walk”, ”run” and ”jump”.

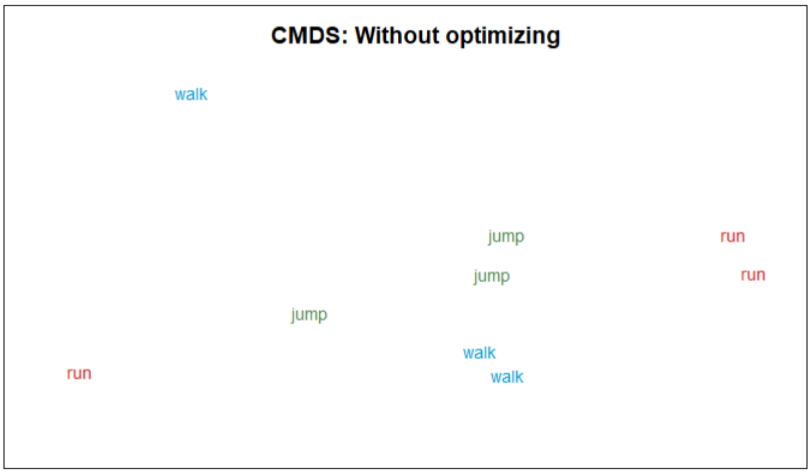


Figure 5.25: Classical multidimensional scaling plot for the distance function $d_{\mathcal{P}}$, i.e. without optimizing with respect to the parametrization.

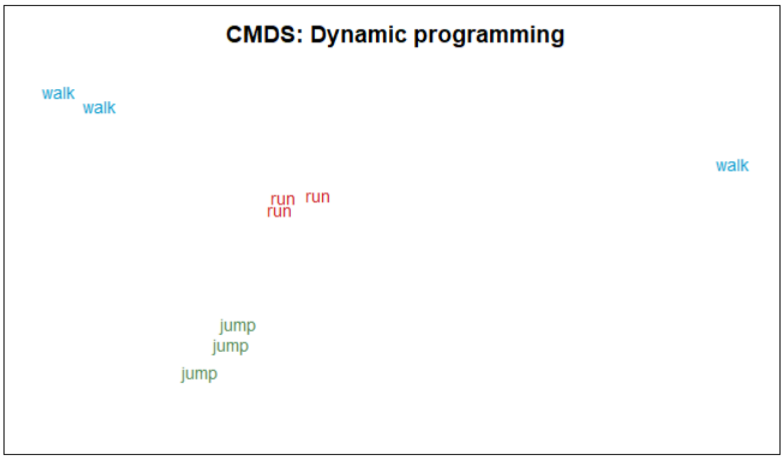
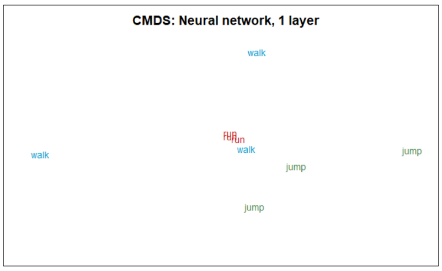
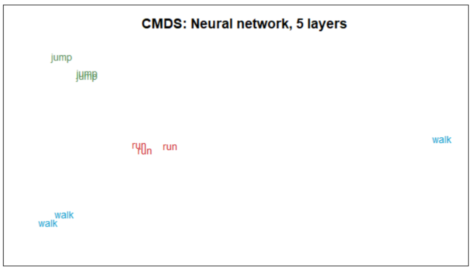


Figure 5.26: Classical multidimensional scaling plot for the dynamic programming approach.



(a) Classical multidimensional scaling plot for the method using neural networks with 1 layer.



(b) Classical multidimensional scaling plot for the method using neural networks with 5 layers.

Figure 5.27

In figure 5.27 the optimal reparametrization, and hence minimal distance, between the motions is found using the residual neural network described in (4.8). Comparing the figures 5.27a and 5.27b, it seems that the method is working better for 5 layers than 1. The difference in computational cost between this methods is nearly insignificant.

The distances between the same motions are also computed using the dynamic programming algorithm described in section 2.4.1, and the result is illustrated in figure 5.26. Then, by comparing these two methods with each other and with the non-optimized distances in 5.25, the neural network with 5 layers seems to provide a satisfying result.

Chapter 6

Conclusion and future work

This thesis has introduced a new approach for reparametrizing a curve to find the optimal distance between two curves. The parametrization function has been described as an affine transformation and an activation function with a skip connection, and the optimal distance is achieved by optimizing on this group of discrete diffeomorphisms.

First, the methods gradient descent and Adam have been computed to find the optimal distance between curves. The experiments of chapter 5 show satisfying results for both methods on curves in $\mathbb{R}^3$, especially when comparing equal shapes. When comparing different shapes, it is not guaranteed that the reached minimum is global because of the non-convexity of the problem. Additionally, these methods are known to perform well when the initial guess is close to the optimal, which is harder to determine when comparing different shapes.

A deep neural network architecture have also been introduced, now describing the parametrization as multiple compositions of the original parametrization function, and the minimum distance is found using the Adam optimizer. Experiments have been done on curves in $\mathbb{R}^3$ using two different approaches, where both seem to be suitable.

One of the neural network approaches are also induced on comparing skeletal animations, where three different motions are considered. The approach is compared with a dynamic programming approach, and according to the experiments in this thesis, they seem to have approximately the same good performance.

6.1 Future work

Something that can be examined further from this paper is to use this method on curves of the Lie group, to match skeletal animations. As we have seen, when describing animations, the motion of a character is defined by a skeleton and animation curve [14].

Every bone in the skeleton has a fixed length and one to three degrees of rotational freedom associated with it, so-called Euler angles. But describing the skeletons and animation curves using Euler angles neglects a certain structure, called the Lie group structure. An improvement to the method will therefore be to include this underlying geometric structure in the mathematical models and their numerical discretizations.

In practice this means that instead of considering a curve in $\mathbb{R}^n$, where n is the total number of joints in the skeleton, the data will consist of curves in $SO(3)^d$. Here d is the number of bones in the skeleton. This representation has shown to be robust and work well in problems of motion blending and curve closing, examined and demonstrated in [11].

Then parametrized curves in the Lie group G will also be treated as immersions, and the space of these curves will be denoted $\mathcal{P} := \text{Imm}(I, G)$. The final shape space $\mathcal{S}$ is then defined in the same way as before

$$\mathcal{S} := \mathcal{P}/\text{Diff}^+(I),$$

but using the Lie group G which is defined as

$$G = SO(3) = \{A \in M^{3\times3} | \det(A) = 1, A^T A = I\}. \tag{6.1}$$

Here the matrices contains the Euler angles for each bone.

Bibliography

[1] A.A. Agrachev and M. Caponigro. Controllability on the group of diffeomorphisms. *Annales de l'Institut Henri Poincaré C, Analyse non linéaire*, 26(6): 2503–2509, 2009. ISSN 0294-1449. doi: https://doi.org/10.1016/j.anihpc.2009.07.003. URL https://www.sciencedirect.com/science/article/pii/S0294144909000687.

[2] George B. Arfken, Hans J. Weber, and Frank E. Harris. Chapter 3 - vector analysis. In George B. Arfken, Hans J. Weber, and Frank E. Harris, editors, *Mathematical Methods for Physicists (Seventh Edition)*, pages 123–203. Academic Press, Boston, seventh edition edition, 2013. ISBN 978-0-12-384654-9. doi: https://doi.org/10.1016/B978-0-12-384654-9.00003-7. URL https://www.sciencedirect.com/science/article/pii/B9780123846549000037.

[3] M. Bauer, M. Eslitzbichler, and M. Grasmair. Landmark-guided elastic shape analysis of human character motions. pages 14–, 2018.

[4] Ma Bauer, M. Bruveris, S. Marsland, and P. W. Michor. Constructing reparameterization invariant metrics on spaces of plane curves. *Differential Geometry and its Applications*, 34:139 – 165, 2014. ISSN 0926-2245. doi: https://doi.org/10.1016/j.difgeo.2014.04.008. URL http://www.sciencedirect.com/science/article/pii/S092622451400062X.

[5] Belongie and Malik. Matching with shape contexts. In *2000 Proceedings Workshop on Content-based Access of Image and Video Libraries*, pages 20–26, 2000. doi: 10.1109/IVL.2000.853834.

[6] A. Le Brigant. Probability on the spaces of curves and the associated metric spaces via information geometry; radar applications. 2017.

[7] Martins Bruveris, Peter W. Michor, and David Mumford. Geodesic completeness for SOBOLEV metrics on the space of immersed plane curves. *Forum of Mathematics, Sigma*, 2, Jul 2014. ISSN 2050-5094. doi: 10.1017/fms.2014.19. URL http://dx.doi.org/10.1017/fms.2014.19.

[8] Carnegie-Mellon. Carnegie-mellon mocap database. `http://mocap.cs.cmu.edu/`.

[9] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22:61–79, 02 1997. doi: 10.1109/ICCV.1995.466871.

[10] E. Celledoni, M. J. Ehrhardt, C. Etmann, R. I. Mclachlan, B. Owren, C. B. Schönlieb, and F. Sherry. Structure preserving deep learning. 2009.

[11] E. Celledoni, M. Eslitzbichler, and A. Schmeding. Shape analysis on Lie groups with applications in computer animation. 2016.

[12] Elena Celledoni, Pål Erik Lystad, and Nikolas Tapia. Signatures in shape analysis: An efficient approach to motion identification. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, pages 21–30, Cham, 2019. Springer International Publishing. ISBN 978-3-030-26980-7.

[13] Isaac Cohen, Nicholas Ayache, and Patrick Sulger. Tracking points on deformable objects using curvature information. In G. Sandini, editor, *Computer Vision — ECCV'92*, pages 458–466, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-47069-4.

[14] M. Eslitzbichler. Modelling character motions on infinite-dimensional manifolds. 2014.

[15] J. Glaunes, A. Qiu, M. I. Miller, and L. Younes. Large deformation diffeomorphic metric curve mapping. 2008.

[16] U. Grenander and M. Miller. Computational anatomy: an emerging discipline. *Quarterly of Applied Mathematics*, 56:617–694, 1998.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[18] S.J. Wright J. Nocedal. *Numerical Optimization*. Springer, 2006.

[19] D.W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):583–600, 2000. doi: 10.1109/34.862197.

[20] Shantanu H Joshi, Eric Klassen, Anuj Srivastava, and Ian Jermyn. A novel representation for Riemannian analysis of elastic curves in . *Proceedings. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007(17-22 June 2007):1—7, July 2007. ISSN 1063-6919. doi: 10.1109/cvpr.2007.383185. URL `https://europepmc.org/articles/PMC3035322`.

[21] Witkin A. Terzopoulos D. Kass, M. Snakes: Active contour models. page 321–331, 1988.

[22] D. G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. 1984.

[23] Satyanad Kichenassamy, Arun Kumar, P. Olver, Allen Tannenbaum, and A. Yezzi. Gradient flows and geometric active contour models. pages 810–815, 07 1995. ISBN 0-8186-7042-8. doi: 10.1109/ICCV.1995.466855.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[25] E. Klassen, A. Srivastava, M. Mio, and S.H. Joshi. Analysis of planar shapes using geodesic paths on shape spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):372–383, 2004. doi: 10.1109/TPAMI.2004.1262333.

[26] Pål Erik Lystad. Signatures in shape analysis. `https://github.com/paalel/Signatures-in-Shape-Analysis`.

[27] Pål Erik Lystad. Signatures in shape analysis. 2019.

[28] Maurice Maes. On a cyclic string-to-string correction problem. *Information Processing Letters*, 35(2):73–78, 1990. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(90)90109-B. URL `https://www.sciencedirect.com/science/article/pii/002001909090109B`.

[29] Andrea Mennucci, A. Yezzi, and G. Sundaramoorthi. Sobolev–type metrics in the space of curves. 04 2006.

[30] Andrea C. G. Mennucci. Metrics of curves in shape optimization and analysis, 2013.

[31] Andrea C. G. Mennucci, Stefano Soatto, Ganesh Sundaramoorthi, and Anthony Yezzi. A new geometric metric in the space of curves, and applications to tracking deforming objects by prediction and filtering. *SIAM Journal on Imaging Sciences*, 4 (1), 2011. doi: 10.1137/090781139.

[32] P. W. Michor and D. Mumford. Riemannian geometries on spaces of plane curves. pages 1–48, 01 2006.

[33] Peter Michor and David Mumford. Vanishing geodesic distance on spaces of submanifolds and diffeomorphisms. *Documenta Mathematica*, 10, 10 2004.

[34] Peter Michor, David Mumford, Jayant Shah, and Laurent Younes. A metric on shape space with explicit geodesics. *Atti Accad. Naz. Lincei Cl. Sci. Fis. Mat. Natur. Rend. Lincei (9) Mat. Appl.*, 19, 07 2007. doi: 10.4171/RLM/506.

[35] Peter W. Michor and David Mumford. An overview of the Riemannian metrics on spaces of curves using the Hamiltonian approach. *Applied and Computational Harmonic Analysis*, 23(1):74–113, 2007. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2006.07.004. URL `https://www.sciencedirect.com/science/article/pii/S1063520307000243`. Special Issue on Mathematical Imaging.

[36] W. Mio and Anuj Srivastava. Elastic-string models for representation and analysis of planar shapes. volume 2, pages II–10, 01 2004. ISBN 0-7695-2158-4. doi: 10.1109/CVPR.2004.1315138.

[37] W. Mio, A. Srivastava, and S. Joshi. On shape of plane elastic curves. 2007.

[38] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[39] P. Olver. *Equivalence Invariants and symmetries*. Cambridge University Press, Cambridge, 1995.

[40] Peter J. Olver. *Geometric Foundations*, page 7–31. Cambridge University Press, 1995. doi: 10.1017/CBO9780511609565.003.

[41] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Tensors and dynamic neural networks in python with strong gpu acceleration. `https://github.com/pytorch/pytorch`.

[42] T. B. Sebastian, P. N. Klein, and B. B. Kimia. On aligning curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):116–125, 2003. doi: 10.1109/TPAMI.2003.1159951.

[43] Anuj Srivastava, Eric Klassen, Shantanu Joshi, and Ian Jermyn. Shape analysis of elastic curves in Euclidean spaces. *IEEE transactions on pattern analysis and machine intelligence*, 09 2010. doi: 10.1109/TPAMI.2010.184.

[44] Anuj Srivastava, Eric Klassen, Shantanu Joshi, and Ian Jermyn. Shape analysis of elastic curves in Euclidean spaces. *IEEE transactions on pattern analysis and machine intelligence*, 09 2010. doi: 10.1109/TPAMI.2010.184.

[45] Ganesh Sundaramoorthi, Anthony Yezzi, and Andrea Mennucci. Sobolev active contours. *International Journal of Computer Vision*, 73:345–366, 01 2005. doi: 10.1007/s11263-006-0635-2.

[46] Georgia Tech. Lecture notes 11. `https://people.math.gatech.edu/~ghomi/LectureNotes/LectureNotes11G.pdf`.

[47] Wen-Hsiang Tsai and Shiaw-Shian Yu. Attributed string matching with merging for shape recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(4):453–462, 1985. doi: 10.1109/TPAMI.1985.4767684.

[48] A. Tversky. Features of similarity. page 327–352, 1977.

[49] N. Ueda and S. Suzuki. Learning visual models from shape contours using multiscale convex/concave structure matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):337–352, 1993. doi: 10.1109/34.206954.

[50] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974. ISSN 0004-5411. doi: 10.1145/321796.321811. URL `https://doi.org/10.1145/321796.321811`.

[51] Jianzhong Wang. *Classical Multidimensional Scaling*, pages 115–129. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27497-8. doi: 10.1007/978-3-642-27497-8_6. URL `https://doi.org/10.1007/978-3-642-27497-8_6`.

[52] Laurent Younes. Computable elastic distances between shapes. *SIAM Journal on Applied Mathematics*, 58, 12 2000. doi: 10.1137/S0036139995287685.

[53] W. Zeng, M. Razib, and A. Bin Shahid. Diffeomorphism spline. page 157, 2015.

Appendix

Definitions in differential geometry

Definition [40], [39] *An m-dimensional manifold $\mathcal{M}$ is a topological space covered by a collection of open subsets $W_\alpha \in \mathcal{M}$ (coordinate charts) and maps $\mathcal{X}_\alpha : W_\alpha \mapsto v_\alpha \in \mathbb{R}^m$ one-to-one and onto (i.e. a bijection), where $\mapsto v_\alpha$ is an open, connected subset of $\mathbb{R}^m$. $(W_\alpha, \mathcal{X}_\alpha)$ define coordinates on $\mathcal{M}$. $\mathcal{M}$ is a smooth manifold if the maps $\mathcal{X}_{\alpha\beta} = \mathcal{X}_\beta \circ \mathcal{X}_\alpha^1$ are smooth where they are defined, i.e. on $\mathcal{X}_\alpha(W_\alpha \cap W_\beta)$ to $\mathcal{X}_\beta(W_\alpha \cap W_\beta)$.*