Shape Analysis on Lie Groups: Reparameterization and Classification Techniques

Jørgen Norvik Bakken

June 6, 2024

Abstract

In this thesis, we explore two reparameterization techniques: a fully discretized method employing the Square Root Velocity Transform (SRVT) with dynamic programming, and a geodesic interpolation method using Sequential Least Squares Programming (SLSQP). Both methods were evaluated for their effectiveness in achieving optimal reparametrizations and for discretizing curves based on their geometric shapes through their shape space distance. Additionally, we utilize the logarithmic signature to compute shape space distances for distinguishing geometric forms, as an alternative to the reparameterization methods. This study extends these methods from the special orthogonal group ($SO(3)^n$), previously researched, to the special Euclidean group ($SE(3)^n$), with synthetic data validating their potential utility.

We further extend our understanding of these methods on ($SO(3)^n$) by incorporating post-processing steps such as dimensionality reduction through Principal Component Analysis (PCA) and Classical Multi-Dimensional Scaling (cMDS), as well as clustering to enhance the data's ability to distinguish between different geometric forms. These techniques were applied to motion capture data, yielding promising results.

Sammendrag

I denne avhandlingen utforsker vi to reparametriseringsteknikker: en fullstendig diskretisert metode som bruker kvadratrot-hastighetstransformasjon med dynamisk programmering, og geodetisk interpolasjonsmetode ved bruk av sekvensiell minstekvadraters programmering. Begge metodene ble evaluert for deres effektivitet i å oppnå optimale reparametriseringer og for å skille kurver basert på deres geometriske former gjennom deres formromsavstand. I tillegg bruker forskningen den logaritmiske signaturen for å beregne formromsavstander for å skille geometriske former, som et alternativ til reparametriseringsmetodene. Denne studien utvider disse metodene fra den spesielle ortogonale gruppen ($SO(3)^n$), som tidligere har vært forsket på, til den spesielle euklidske gruppen ($SE(3)^n$), med syntetiske data som validerer deres potensielle nytteverdi.

Vi utvider også vår forståelse av disse metodene for ($SO(3)^n$), ved å innlemme etterbehandlingssteg som dimensjonsreduksjon gjennom hovedkomponentanalyse og klassisk multidimensjonal skalering, samt klynging for å forbedre dataens evne til å skille mellom forskjellige geometriske former. Disse teknikkene ble anvendt på bevegelsesfangstdata, og ga lovende resultater.

Preface

This master's thesis marks the end of my five-year journey towards a Master of Science in Applied Physics and Mathematics, with a specialization in industrial mathematics, at the Norwegian University of Science and Technology (NTNU).

I am grateful for the guidance and inspiration provided by my supervisor, Professor Elena Celledoni. Her mentorship and dedication have been invaluable.

In addition, I would like to thank Carnegie Mellon University. The data used in this project was obtained from mocap.cs.cmu.edu, a database created with funding from NSF EIA-0196217.

Jørgen Norvik Bakken
Trondheim
June 2024

Contents

Figures

Tables

List of Algorithms

Chapter 1

Introduction

Humans begin learning geometry early by interacting with their environment and recognizing different shapes. This natural understanding evolves as they grow and may turn into formal mathematical knowledge. Over time, this intuitive sense of shapes and spatial relationships may be quantified, structured, and expanded into a comprehensive understanding of geometry.

The field of shape analysis traces its origins to D'Arcy Thompson [1], who studied the shapes of plants and animals, demonstrating how non-linear transformations could standardize the shapes of different organisms. Since then, various approaches have been developed to tackle shape analysis. One significant contribution to the field is Kendall's work in 1984 [2], where he articulated the concept of shape as the equivalence class in a quotient space and informally defined it as "what remains after accounting for variations due to translations, rotations, and dilatations". Additionally, the differential geometric perspective, outlined by Srivastava et al. [3], has provided a comprehensive framework for shape analysis.

Historically, shape comparison involved identifying specific feature points or landmarks along the boundaries of objects [2]. Lately more literature regarding analysis of shapes as elements of infinite-dimensional Riemannian manifolds [4], an area initially explored by Younes [5], has grown. For a greater overview, see [6, 7]. Today shape analysis can be use for a wide range of applications, such as protein structure comparison, medical image diagnosis, plant leaf classification, inverse obstacle scattering and more [8–11].

Signatures, introduced by K.-T. Chen in the context of smooth paths [12] and later expanded by Lyons into the framework of geometric rough paths [13], have proven essential in diverse areas such as analyzing solutions to controlled differential equations, tackling classification challenges in time series, enhancing approaches in machine learning [14], and in topological data analysis [15]. Consequently, the theory of rough paths has developed into a comprehensive set of methods, serving both theoretical mathematical investigations and practical applications.

Shape analysis can address various problems in computer animation, such as periodicity, looping of movement, interpolation between movements, and enhan-

cing motion recognition [16–20]. One technique for activity identification from video or motion capture data involves extracting silhouettes from video frames and applying shape comparison techniques, augmented by time-series modeling for analyzing sequences in shape spaces. Motion capture data, often provided in Acclaim Skeleton File (ASF) for skeleton structure and Acclaim Motion Capture (AMC) formats for motion data, captures an actor's movements onto a virtual skeleton for animating a 3D environment [21, 22].

A method for modeling these animations, represented by Eslitzbichler, involves treating them as curves on an n-torus, with each point on the curve corresponding to a pose in $\mathbb{R}^3$ [16]. This approach facilitates movement comparison utilizes the Square Root Velocity Transform (SRVT) introduced in [4, 23]. Celledoni et al. expanded on this concept by applying SRVT to curves valued in Lie groups, specifically modeling animations within $SO(3)^n$ and later generalizing it to homogeneous spaces [17, 24]. Further, refining curve parametrization techniques, Bauer et al. introduced the use of gradient descent and dynamic programming algorithms to achieve optimal reparametrizations for comparing character animations [25]. Building upon these methodologies, Celledoni et al. demonstrated the potential for classifying motion capture data using the framework from Bauer et al., alongside employing the logarithmic signature method [18]. Additionally, Bærland et al. explored the use of deep neural networks for classification [26].

Srivastava et al. (2012) describe shape analysis through differential-geometric approaches as focusing on several primary objectives: quantifying shape differences, creating templates for specific shape classes, modeling shape variations through statistical models, and clustering, classifying, and estimating shapes [3]. This thesis aims to address the quantification of shape differences, as well as the clustering, classification, and estimation of shapes, utilizing methodologies within Lie groups. This work explores reparameterization techniques, focusing on two methods: SRVT (Square Root Velocity Transform) combined with dynamic programming, and geodesic interpolation. These methods are utilized to achieve optimal curve alignment for comparing curves in Lie groups. Additionally, this work investigates the logarithmic signature method as an alternative approach to shape comparison in Lie groups. The comparison is based on assessing the geometric shape distance between curves. This work builds upon and expands the methodologies presented in [18].

The thesis begins by laying the groundwork in Chapter 2. This chapter introduces the fundamental theories essential for understanding the methods discussed later. Next, Chapter 3 delves into reparameterization. It examines two approaches: the Square Root Velocity Transform (SRVT) for adjusting curves in Lie groups through dynamic programming, and a geodesic interpolation method. Chapter 4 explores the logarithmic signature method as an alternative for comparing shapes within Lie groups. This method offers a less computationally expensive approach. Moving to real-world applications, Chapter 5 applies these methods to motion capture data. This chapter also discusses various ways to refine this data. Finally, Chapter 6 concludes the thesis by summarizing the key findings and suggesting

directions for future research.

The research conducted in this thesis has the potential to contribute to several of the United Nations Sustainable Development Goals (SDGs) [27]. The techniques developed are particularly relevant to SDG 3 (Good Health and Well-Being), as they facilitate the classification of human movements, which can enhance the development of personalized rehabilitation programs and medical treatments. Additionally, this research can advance SDG 9 (Industry, Innovation, and Infrastructure) by improving robotic movement classifications, thereby enhancing industrial efficiency, reducing waste, and improving safety across manufacturing sectors. Moreover, the precise analysis of animal movements contributes to SDG 14 (Life Below Water) and SDG 15 (Life on Land) by offering better conservation strategies and management practices through deeper insights into the behaviors and interactions of both marine and terrestrial species. By providing a comprehensive understanding of shape analysis, this research aligns with the UN's broader goals for sustainable development.

Note to the Reader

This thesis builds upon the foundational work presented in my project thesis [28] from fall 2023, which marked my initial exploration into the fields of topology and algebra. The implementation and code can be found in the authors GitHub repository[1].

In this document, various methodologies are employed to address analogous problems, with redundancy minimized by introducing synthetic data in the initial chapter and referencing it throughout. While expanding from individual elements to collections within Lie groups, we revisit key theoretical concepts to maintain structural coherence without unnecessary complexity. The hat map notation "$\wedge$" for both SO(3) and SE(3) is used, as it is standard in the literature, despite its potential for confusion. I hope readers will understand these choices.

[1] https://github.com/JorgenBakken/MastersThesis

Chapter 2

Theoretical Framework

In this chapter, we lay the theoretical groundwork for the methodologies explored later in this thesis. We introduce fundamental concepts for Lie Groups, shape space, and tools for analyzing curves in SO(3) and SE(3). Additionally, we consider the direct products $\mathrm{SO}(3)^n$ and $\mathrm{SE}(3)^n$, which facilitate the analysis of multiple rotations or rigid body motions, aiding in the study of complex shape dynamics in higher dimensions.

2.1 Fundamental Concepts

We refer to [29] for the conventions and notations used in this thesis. A Lie group G is a differentiable manifold with a smooth group structure, ensuring smooth multiplication and inversion operations. The corresponding Lie algebra, $\mathfrak{g}$, is the tangent space at the identity element of G and features a Lie bracket $[\cdot,\cdot]:\mathfrak{g}\times\mathfrak{g}\to\mathfrak{g}$, which is bilinear, antisymmetric, and satisfies the Jacobi identity.

For a Lie group G and elements $g,h\in G$, left and right translations are denoted as $L_g(h)=gh$ and $R_g(h)=hg$. The differential of the right translation by h, T_gR_h, maps from the tangent space at g, T_gG, to the tangent space at gh, $T_{gh}G$.

A Lie algebra-valued one-form on a Lie group G assigns, at each point $g\in G$, a linear functional from the tangent space at g, T_gG, to the Lie algebra $\mathfrak{g}$. Specifically, for any tangent vector $X_g\in T_gG$, there exists a one-form ω_g in the cotangent space T_g^*G that maps X_g to an element of $\mathfrak{g}$.

The Cartesian product $G\times G$ consists of all pairs (g_1,g_2) with $g_1,g_2\in G$, operating under component-wise group actions. The corresponding Lie algebra is $\mathfrak{g}\oplus\mathfrak{g}$, with the Lie bracket on $G\times G$ defined as:

$$[(g_1,g_2),(h_1,h_2)]=([g_1,h_1],[g_2,h_2]), \tag{2.1}$$

where $[g_1,h_1]$ and $[g_2,h_2]$ are the Lie brackets within $\mathfrak{g}$.

2.2 The Maurer–Cartan Form and The Right Logarithmic Derivative

The -Cartan form, denoted by ω_g, is a fundamental one-form $\omega_g : T_gG \to \mathfrak{g}$ on the Lie group G as detailed in [30, p. 373]. At any point $g \in G$, it is defined via right translation by g^{-1}, denoted as $R^*_{g^{-1}}$, and is given by:

$$\begin{aligned} \omega_g : T_gG \to \mathfrak{g}, \\ \omega_g = T_g(R_{g^{-1}}) = (R_{g^{-1}})^*, \end{aligned} \tag{2.2}$$

acting on any tangent vector $X \in T_gG$ as:

$$\omega_g(X) = (R_{g^{-1}})^* X \in \mathfrak{g}. \tag{2.3}$$

This form is right-invariant, linking the Lie algebra $\mathfrak{g}$ to the group's structure, as noted in [31, p. 71].

For a smooth curve $c : I \to G$ with $I \subset \mathbb{R}$ and $c \in C^\infty$, the right logarithmic derivative $\delta^r c$ is defined by [30] as:

$$\delta^r c(t) = (\mathrm{R}_{c(t)^{-1}})^* \dot{c}(t), \tag{2.4}$$

where $\dot{c}(t)$ is the time derivative of the curve c.

In the general linear group GL(n), with matrix multiplication as the group operation, the Cartan form is defined as follows:

$$(R_g)_*(\dot{c}(t)) = \frac{d}{dt}(R_g \circ c(t)) = \frac{d}{dt}(c(t) \cdot g) = \dot{c}(t) \cdot g, \tag{2.5}$$

where $\cdot$ denotes matrix multiplication. Therefore, the right logarithmic derivative in GL(n) is given by:

$$\delta^r c(t) = (R_{c(t)^{-1}})_*(\dot{c}(t)) = \dot{c}(t) \cdot c(t)^{-1}. \tag{2.6}$$

This effectively maps the tangent vector $\dot{c}(t)$ at $c(t)$ to the tangent space at the identity of GL(n).

2.3 Shape Analysis on Lie Groups

Shape analysis is crucial for understanding the structure and diversity of geometric objects. Within Lie groups, it extends mathematical analysis to dynamic shapes. Shape spaces capture intrinsic geometry, focusing on essential features rather than specific parameterizations.

2.3.1 The Shape Space

Consider two curves $c_1, c_2 \in \mathrm{Imm}(I, G)$, where $\mathrm{Imm}(I, G)$ represents the space of immersions from the interval $I = [0, 1] \subset \mathbb{R}$ into a Lie group G. The group of all orientation-preserving diffeomorphism of the interval I, denoted $\mathrm{Diff}^+(I)$, is defined as follows, detailed in [18]:

$$\mathrm{Diff}^+(I) := \{\varphi \in C^\infty(I, I) \mid \varphi'(t) > 0 \,\forall\, t \in I, \varphi(0) = 0, \varphi(1) = 1\}. \tag{2.7}$$

Following [17], curves c_1 and c_2 are considered equivalent, belonging to the same shape space, if there exists an orientation-preserving diffeomorphism $\varphi \in \mathrm{Diff}^+(I)$ such that $c_1 = c_2 \circ \varphi$. This equivalence establishes an equivalence class under the action of $\mathrm{Diff}^+(I)$, thus defining the shape space. Formally, the shape space is the quotient space:

$$\mathcal{S} := \mathcal{P}/\mathrm{Diff}^+(I), \tag{2.8}$$

where the space $\mathcal{P}$ includes the immersions, encompassing all curves with non-vanishing first derivatives, and is defined by:

$$\mathcal{P} := \mathrm{Imm}(I, G), \tag{2.9}$$

This construction implies that the shape space $\mathcal{S}$ captures all curves up to orientation-preserving reparameterizations, highlighting the geometric properties of shapes over their specific parameterizations.

2.3.2 Geodesic Distance in Shape Space

Consider two curves, c_1 and c_2, which are elements of the space of immersions $\mathrm{Imm}(I, G)$, where G is a finite-dimensional Lie group and I is the interval $[0, 1]$. To measure distances between such immersions, we employ a metric $d_{\mathcal{P}}$ defined on the space $\mathcal{P}$ (2.9). It is crucial for $d_{\mathcal{P}}$ to reflect the intrinsic geometric properties of the curves, independent of their parameterization. This requirement is encapsulated by the property of reparameterization invariance:

$$d_{\mathcal{P}}(c_1 \circ \varphi, c_2 \circ \varphi) = d_{\mathcal{P}}(c_1, c_2), \quad \forall\, \varphi \in \mathrm{Diff}^+(I). \tag{2.10}$$

The shape space metric $d_{\mathcal{S}}$ is then defined by minimizing over all reparameterizations:

$$d_{\mathcal{S}}(c_1, c_2) := \inf_{\varphi \in \mathrm{Diff}^+(I)} d_{\mathcal{P}}(c_1, c_2 \circ \varphi). \tag{2.11}$$

This definition effectively removes the influence of parameterization, focusing purely on the geometric shape of the curves.

By asserting that the metric $d_{\mathcal{P}}$ is reparameterization invariant for any two curves and for any orientation-preserving diffeomorphism $\varphi \in \mathrm{Diff}^+(I)$, the metric $d_{\mathcal{S}}$ also inherits this invariance [17, Lemma 3.4].

Utilization of the Euclidean norm might lead to vanishing distances for non-identical curves, as highlighted by Michor and Mumford [32]. To address this limitation, a more robust choice involves Sobolev-type metrics based on the arc length derivative [33], which incorporate derivatives of curves to capture more subtle geometric differences.

2.3.3 Square Root Velocity Transform (SRVT)

Introduced by Srivastava et al. [4], the Square Root Velocity Transform (SRVT) is a pivotal tool in shape analysis [6, 17, 18, 24, 25, 34–36]. The SRVT transforms curves into a space where standard linear operations are meaningful, simplifying the complex problems of shape analysis and comparison.

The transformation for curves within a Lie group is defined as follows [17]:

$$\begin{aligned} \mathcal{R} : \operatorname{Imm}(I, G) &\to \{q \in C^\infty(I, \mathfrak{g}) \mid q(t) \neq 0 \,\forall\, t \in I\}, \\ q(t) = \mathcal{R}(c)(t) &:= \frac{R^{-1}_{c(t)_*}(\dot{c}(t))}{\sqrt{\|R^{-1}_{c(t)_*}(\dot{c}(t))\|}}, \end{aligned} \tag{2.12}$$

where $\dot{c}(t)$ denotes the tangent vector of the curve at point $c(t)$, and $R^{-1}_{c(t)_*}$ is the differential of the inverse right translation by $c(t)$. This operation transforms the tangent vector into the Lie algebra $\mathfrak{g}$, and the division by the square root of its norm ensures the result is a unit vector, thereby reparameterizing the curve by its arc length.

As noted in [17], SRVT is equivariant with respect to reparameterization, satisfying $\mathcal{R}(c \circ \varphi) = \mathcal{R}(c) \circ \varphi \cdot \sqrt{\dot{\varphi}}$. Additionally, SRVT is translation invariant, meaning $\mathcal{R}(R_g \cdot c) = \mathcal{R}(c)$. This implies that the transformed representation of a curve remains unchanged under translation by an element g in the Lie group.

The translation invariance property implies that SRVT does not retain information about the initial position of the curve, leading to its non-injectivity. This characteristic highlights a fundamental aspect of SRVT: while it effectively captures the geometric essence of a curve, it abstracts away certain specific details like the starting point and orientation in space.

2.3.4 Pseudometric Based on SRVT

The pseudometric $d_{\mathcal{P}}$ on the space of immersions $\mathcal{P}$ is defined following the framework in [17, Definition 3.7]:

$$d_{\mathcal{P}}(c_0, c_1) := \sqrt{\int_I \|q_0(t) - q_1(t)\|^2 \, dt} = d_{L^2}(\mathcal{R}(c_0), \mathcal{R}(c_1)), \tag{2.13}$$

where $q_i := \mathcal{R}(c_i)$ for $i = 0, 1$. This pseudometric, $d_{\mathcal{P}}$, is invariant under reparameterization, as shown in [17, Proposition 3.8].

The subspace $\mathcal{P}_*$ is defined as a closed submanifold of $\mathcal{P}$:

$$\mathcal{P}_* := \{c \in \mathrm{Imm}(I, G) : c(0) = e\}, \tag{2.14}$$

where e denotes the identity element of the Lie group G. This subset represents immersions that start at the identity, formally expressed as $\mathcal{P} \cap C^\infty(I, G)$. To adjust any smooth curve $c : I \to G$ to start at the identity, the curve is right-translated by applying the inverse of its initial point $c(0)^{-1}$ to every point along the curve:

$$c(t) \mapsto c(t) \cdot c(0)^{-1} \quad \forall\, t \in [0, 1]. \tag{2.15}$$

The pseudometric $d_{\mathcal{P}_*}$ on $\mathcal{P}_*$ utilizes SRVT to induce a metric from the L^2-metric on the tangent space $C^\infty(I, \mathfrak{g} \setminus \{0\})$:

$$d_{\mathcal{P}_*}(c_0, c_1) := d_{L^2}(\mathcal{R}(c_0 \cdot c_0(0)^{-1}), \mathcal{R}(c_1 \cdot c_1(0)^{-1})), \tag{2.16}$$

where $q_i = \mathcal{R}(c_i)$ for $i = 0, 1$.

According to [17, Definition 3.10], we can define the shape space $\mathcal{S}_*$ as

$$\mathcal{S}_* := \mathcal{P}_* / \mathrm{Diff}^+(I). \tag{2.17}$$

Finally, the pseudometric $d_{\mathcal{S}_*}$ on $\mathcal{S}_*$ is given by

$$d_{\mathcal{S}_*}(c_0, c_1) = \inf_{\varphi \in \mathrm{Diff}^+(I)} d_{\mathcal{P}_*}(\mathcal{R}(c_0), \mathcal{R}(c_1 \circ \varphi)), \tag{2.18}$$

providing a geodesic distance on $\mathcal{S}_*$, as established in [37].

2.3.5 Geodesic Interpolation

Geodesic interpolation within a Lie group G allows for smooth transitions between two elements $c_0, c_1 \in G$. This process involves mapping the elements to the associated Lie algebra using the logarithmic map, performing linear interpolation within the Lie algebra, and then mapping the interpolated elements back to the Lie group via the exponential map [38, 39].

For two elements c_0 and c_1 in G, the geodesic interpolation path $\zeta(t)$, where $t \in [0, 1]$, is defined as:

$$\zeta(t) = \exp(t \cdot \log(c_1 c_0^{-1}))c_0, \tag{2.19}$$

where $\zeta(t)$ represents the shortest path between c_0 and c_1 in the Lie group. This approach ensures that the interpolation path is smooth and respects the intrinsic geometric structure of the Lie group.

2.4 Curves in SO(3)

The special orthogonal group SO(3) consists of all 3×3 rotation matrices, rigorously defined by the conditions:

$$\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times3} \mid R^T R = I_3,\ \det(R) = 1\}, \tag{2.20}$$

where I_3 represents the 3×3 identity matrix and R^T denotes the transpose of R. Matrices in SO(3) characterize all proper rotations in $\mathbb{R}^3$, defined by their orthogonality, unity determinant, and orientation preservation [40].

Following [41, 42], the Lie algebra associated with SO(3), denoted by $\mathfrak{so}(3)$, is the tangent space at the identity matrix I_3. It contains all skew-symmetric matrices infinitesimally close to I_3.

To establish an isomorphism between $\mathfrak{so}(3)$ and $\mathbb{R}^3$, the hat map is introduced. For a vector $\omega = [\omega_1, \omega_2, \omega_3]^T \in \mathbb{R}^3$, we associate a skew-symmetric matrix in $\mathfrak{so}(3)$ via the hat map:

$$\begin{aligned} &\wedge : \mathbb{R}^3 \to \mathfrak{so}(3), \\ &\hat{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \end{aligned} \tag{2.21}$$

The vee map performs the inverse operation:

$$\begin{aligned} &\vee : \mathfrak{so}(3) \to \mathbb{R}^3, \\ &\omega = \hat{\omega}^{\vee} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}. \end{aligned} \tag{2.22}$$

Consequently, for any $\omega \in \mathbb{R}^3$, the identity $\hat{\omega}^{\vee} = \omega$ holds true, affirming that the composition of the hat map followed by the vee map constitutes the identity mapping on $\mathbb{R}^3$.

Rodrigues' rotation formula connects the matrix exponential map from the Lie algebra $\mathfrak{so}(3)$ to the Lie group SO(3) [41, 43, 44]:

$$\begin{aligned} &\exp : \mathfrak{so}(3) \to \mathrm{SO}(3), \\ &R = \exp(\hat{\omega}) = I_3 + \frac{\sin(\theta)}{\theta}\hat{\omega} + \frac{1-\cos(\theta)}{\theta^2}\hat{\omega}^2, \end{aligned} \tag{2.23}$$

where $\theta = \|\omega\|$ is the norm of the vector ω, representing the magnitude of rotation. This formula provides a direct method to convert a skew-symmetric matrix to a rotation matrix.

For a rotation matrix $R \in \mathrm{SO}(3)$, the rotation angle θ can be retrieved by:

$$\theta = \arccos\left(\frac{\mathrm{trace}(R)-1}{2}\right), \tag{2.24}$$

which is used in the computation of the matrix logarithm map [38, 41]:

$$\begin{aligned} \log &: \mathrm{SO}(3) \to \mathfrak{so}(3), \\ \hat{\omega} = \log(R) &= \frac{\theta}{2\sin(\theta)}(R - R^T), \end{aligned} \tag{2.25}$$

where we assume that θ is not a multiple of π to avoid a singularity in the denominator.

The Frobenius norm is used to measure the magnitude of elements in $\mathfrak{so}(3)$, allowing direct comparison with the Euclidean norm in $\mathbb{R}^3$. The norms are related by:

$$\frac{1}{2}\|\hat{\omega}\|_F^2 = \frac{1}{2}\mathrm{trace}(\hat{\omega}^T\hat{\omega}) = \omega_1^2 + \omega_2^2 + \omega_3^2 = \|\omega\|^2. \tag{2.26}$$

This framework can be extended to $\mathrm{SO}(3)^d$, the Cartesian product of d copies of SO(3), defined as:

$$\mathrm{SO}(3)^d = \{(R_1, R_2, \ldots, R_d) \mid R_i \in \mathrm{SO}(3),\ i = 1, \ldots, d\}, \tag{2.27}$$

where each R_i is a 3×3 rotation matrix. This definition ensures that each component R_i is an independent 3×3 rotation matrix, operating within its own three-dimensional space. The collection of these matrices forms a product group that encapsulates rotational motions in d independent three-dimensional spaces.

The tangent space of the Cartesian product $G \times G$ at the identity is isomorphic to the direct sum of the tangent spaces of G at each identity [29]:

$$T_{(e,e)}(G \times G) \cong T_eG \oplus T_eG. \tag{2.28}$$

Since $\mathrm{SO}(3)^d$ is a Cartesian product of d copies of SO(3), its tangent space at the identity element, $(e, e, \ldots, e)$, where e denotes the 3×3 identity matrix, is isomorphic to the direct sum of the tangent spaces of each SO(3) at the identity:

$$T_{(e,e,\ldots,e)}(\mathrm{SO}(3)^d) \cong \bigoplus_{i=1}^{d} T_e\mathrm{SO}(3). \tag{2.29}$$

The Lie algebra of a Lie group is isomorphic to its tangent space at the identity. Therefore, for $\mathrm{SO}(3)^d$, the corresponding Lie algebra, denoted $\mathfrak{so}(3)^d$, is the direct sum of d copies of $\mathfrak{so}(3)$:

$$\mathfrak{so}(3)^d := \bigoplus_{i=1}^{d} \mathfrak{so}(3). \tag{2.30}$$

Operations within $\mathfrak{so}(3)^d$ are performed component-wise, allowing for the independent manipulation of each component. This extension enables the application of the matrix exponential and logarithm maps to $\mathfrak{so}(3)^d$ by applying these maps to each component separately. This enables us to extend the matrix exponential map

and the matrix logarithmic map to $\mathfrak{so}(3)^d$ by applying them to each component of the input matrix.

For an element $\hat{\omega}^d = (\hat{\omega}_1, \hat{\omega}_2, \ldots, \hat{\omega}_d) \in \mathfrak{so}(3)^d$, the matrix exponential map is defined as:

$$\begin{aligned} \exp : \mathfrak{so}(3)^d \to \mathrm{SO}(3)^d, \\ R^d = \exp(\hat{\omega}^d) = (\exp(\hat{\omega}_1), \exp(\hat{\omega}_2), \ldots, \exp(\hat{\omega}_d)), \end{aligned} \tag{2.31}$$

where $\exp(\hat{\omega}_i)$ is the matrix exponential of the i-th component of $\hat{\omega}^d$. The matrix logarithmic map is defined similarly:

$$\begin{aligned} \log : \mathrm{SO}(3)^d \to \mathfrak{so}(3)^d, \\ \hat{\omega}^d = \log(R^d) = (\log(R_1), \log(R_2), \ldots, \log(R_d)), \end{aligned} \tag{2.32}$$

where $\log(R_i)$ is the matrix logarithm of the i-th component of R^d. These maps provide a direct method to convert skew-symmetric matrices to rotation matrices in $\mathfrak{so}(3)^d$ and vice versa, facilitating independent operations on each component in a multi-body system.

It is possible to extend the hat and vee maps, as defined in (2.21) and (2.22), to $\mathfrak{so}(3)^d$. The hat map is defined as:

$$\begin{aligned} \wedge : \mathbb{R}^{3d} \to \mathfrak{so}(3)^d, \\ \hat{\omega}^d = \begin{bmatrix} \hat{\omega}_1 & 0 & \ldots & 0 \\ 0 & \hat{\omega}_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \hat{\omega}_d \end{bmatrix}, \end{aligned} \tag{2.33}$$

where $\hat{\omega}_i$ is the skew-symmetric matrix associated with the i-th component of ω^d. The vee map is defined as:

$$\begin{aligned} \vee : \mathfrak{so}(3)^d \to \mathbb{R}^{3d}, \\ \omega^d = \begin{bmatrix} \omega_{1,1} \\ \omega_{1,2} \\ \omega_{1,3} \\ \vdots \\ \omega_{d,1} \\ \omega_{d,2} \\ \omega_{d,3} \end{bmatrix}, \end{aligned} \tag{2.34}$$

where ω_i is the vector associated with the i-th component of $\hat{\omega}^d$. These maps allow for the conversion of vectors in $\mathbb{R}^{3d}$ to skew-symmetric matrices in $\mathfrak{so}(3)^d$ and vice versa. This facilitates the manipulation of multi-body systems in a vectorized form, enhancing computational efficiency and providing clarity in handling complex rotational dynamics.

For the Cartesian product, we extend the Frobenius norm to $\mathfrak{so}(3)^d$ by summing the squared Frobenius norms of each component:

We define the norm on $\hat{\omega}^d$ as:

$$\|\hat{\omega}^d\|_F^2 := \sum_{i=1}^{d} \|\hat{\omega}_i\|_F^2, \tag{2.35}$$

providing the following relation:

$$\frac{1}{2}\|\hat{\omega}^d\|_F^2 = \frac{1}{2}\sum_{i=1}^{d} \text{trace}(\hat{\omega}_i^T \hat{\omega}_i) = \sum_{i=1}^{d} (\omega_{i,1}^2 + \omega_{i,2}^2 + \omega_{i,3}^2) = \|\omega^d\|_2^2, \tag{2.36}$$

which relates the Frobenius norm of a skew-symmetric matrix in $\mathfrak{so}(3)^d$ to the Euclidean norm of the corresponding vector in $\mathbb{R}^{3d}$.

2.5 Curves in SE(3)

The special Euclidean group SE(3) encapsulates all possible configurations of a rigid body in three-dimensional space, encompassing both rotations and translations. This group consists of 4×4 matrices representing rigid transformations, defined as:

$$\text{SE}(3) = \left\{ T \in \mathbb{R}^{4\times 4} \mid T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}, R \in \text{SO}(3), t \in \mathbb{R}^3 \right\}, \tag{2.37}$$

where R is a rotation matrix from the special orthogonal group SO(3), and t is a translation vector in $\mathbb{R}^3$. This ensures transformations preserve distances and angles, maintaining the rigid body's geometric integrity [45, 46].

The Lie algebra of SE(3), denoted $\mathfrak{se}(3)$, represents the tangent space at the identity element I_4. It comprises 4×4 matrices that are infinitesimally close to the identity, describing small motions of a rigid body [45, 46]:

$$\mathfrak{se}(3) = \left\{ \hat{\xi} \in \mathbb{R}^{4\times 4} \mid \hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix}, \omega \in \mathbb{R}^3, v \in \mathbb{R}^3 \right\}, \tag{2.38}$$

where $\hat{\omega}$ is the skew-symmetric matrix of the angular velocity vector ω, and v represents the infinitesimal translations.

The hat ($\wedge$) and vee ($\vee$) maps facilitate the correspondence between $\mathbb{R}^6$ and $\mathfrak{se}(3)$ [45, 46], allowing transitions between vector and matrix representations:

$$\wedge : \mathbb{R}^6 \to \mathfrak{se}(3),$$
$$\hat{\xi} = \begin{bmatrix} \omega \\ v \end{bmatrix}^\wedge = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.39}$$

This operation converts a twist vector from $\mathbb{R}^6$ to its matrix representation in $\mathfrak{se}(3)$, capturing combined rotational and translational velocities. Conversely, the vee operation reverts a matrix in $\mathfrak{se}(3)$ to its vector form:

$$\begin{aligned} \vee : \mathfrak{se}(3) &\to \mathbb{R}^6, \\ \xi = \hat{\xi}^{\vee} = \begin{bmatrix} \omega \\ v \end{bmatrix} &= \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}. \end{aligned} \tag{2.40}$$

The relationship between the Lie algebra $\mathfrak{se}(3)$ and the Lie group SE(3) is mediated by the exponential and logarithmic maps.

The exponential map transfers elements from the Lie algebra $\mathfrak{se}(3)$ to the Lie group SE(3), using exponential coordinates $\xi = (\omega, v)$ [45]

$$\begin{aligned} \exp : \mathfrak{se}(3) &\to \mathrm{SE}(3), \\ T = \exp(\hat{\xi}) = \exp\left(\begin{bmatrix} \omega \\ v \end{bmatrix}^{\wedge}\right) &= \begin{bmatrix} \exp(\hat{\omega}) & J_l(\omega)v \\ 0 & 1 \end{bmatrix}, \end{aligned} \tag{2.41}$$

where $\hat{\xi}$ is the twist element, and $J_l(\omega)$ is the left Jacobian matrix, given by [45]:

$$J_l(\omega) = I_3 + \frac{1-\cos(\|\omega\|)}{\|\omega\|^2}\hat{\omega} + \frac{\|\omega\| - \sin(\|\omega\|)}{\|\omega\|^3}\hat{\omega}^2, \tag{2.42}$$

The left Jacobian accounts for the nonlinearity in the rotational component of the motion. Conversely, the logarithmic map reverts elements from the Lie group SE(3) back to the Lie algebra $\mathfrak{se}(3)$, which is essential for analytical and computational purposes [45]:

$$\begin{aligned} \log : \mathrm{SE}(3) &\to \mathfrak{se}(3), \\ \hat{\xi} = \log(T) = \log\left(\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}\right) &= \begin{bmatrix} \log(R) & J_l^{-1}(\omega)t \\ 0 & 0 \end{bmatrix}, \end{aligned} \tag{2.43}$$

where $J_l^{-1}(\omega)$ is the inverse of the left Jacobian matrix, formulated as [45]:

$$J_l^{-1}(\omega) = I_3 - \frac{1}{2}\hat{\omega} + \left(\frac{1}{\|\omega\|^2} - \frac{1+\cos(\|\omega\|)}{2\|\omega\|\sin(\|\omega\|)}\right)\hat{\omega}^2, \tag{2.44}$$

When working with SE(3), it is important to understand how the norm of the hat map corresponds to the vector. We can examine this through the Frobenius and Euclidean norms.

$$\|\hat{\xi}\|_F^2 = \mathrm{trace}(\hat{\xi}^T\hat{\xi}) = \|\hat{\omega}\|_F^2 + \|v\|_F^2 = 2\|\omega\|_2^2 + \|v\|_2^2, \tag{2.45}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and $\|\cdot\|_2$ denotes the Euclidean norm. The trace$(\cdot)$ operator takes the trace of a matrix. This equation shows that it is not

possible to directly convert the Frobenius norm of $\hat{\xi}$ to the Euclidean norm of ξ. While in SO(3), there is a direct relationship between the Frobenius norm of $\hat{\omega}$ and the Euclidean norm of ω, specifically $\frac{1}{2}\|\hat{\omega}\|_F^2 = \|\omega\|_2^2$ as seen in (2.26). This means that we need to scale the rotational error by a factor of 2, when working with the vector form, to equalize the contributions of the rotational and translational errors.

The concept of the special Euclidean group SE(3) can be generalized to higher dimensions by considering $\mathrm{SE}(3)^d$, the Cartesian product of d copies of SE(3). This extended group is defined as:

$$\mathrm{SE}(3)^d = \{(T_1, T_2, \ldots, T_d) \mid T_i \in \mathrm{SE}(3),\, i = 1, \ldots, d\}, \tag{2.46}$$

where each T_i is an independent 4×4 rigid body transformation matrix. This configuration represents motions in multiple, independent three-dimensional spaces.

The tangent space of the Cartesian product $G \times G$ at the identity is isomorphic to the direct sum of the tangent spaces of G at each identity, as seen in (2.28). Thus, for $\mathrm{SE}(3)^d$, the tangent space at the identity element $(e, e, \ldots, e)$, where e denotes the identity element of SE(3), follows this principle:

$$T_{(e,e,\ldots,e)}(\mathrm{SE}(3)^d) \cong \bigoplus_{i=1}^{d} T_e\mathrm{SE}(3). \tag{2.47}$$

The Lie algebra of a Lie group is isomorphic to its tangent space at the identity. Therefore, for $\mathrm{SE}(3)^d$, the corresponding Lie algebra, denoted $\mathfrak{se}(3)^d$, is the direct sum of d copies of $\mathfrak{se}(3)$

$$\mathfrak{se}(3)^d := \bigoplus_{i=1}^{d} \mathfrak{se}(3), \tag{2.48}$$

indicating that the Lie algebra $\mathfrak{se}(3)^d$ is isomorphic to the direct sum of the tangent spaces of SE(3) at each identity element.

Within $\mathfrak{se}(3)^d$, operations are performed component-wise, enabling independent manipulation of each transformation matrix. This framework extends the matrix exponential and logarithmic maps to $\mathfrak{se}(3)^d$ by applying these operations to each matrix component separately.

For an element $\hat{\xi}^d = (\hat{\xi}_1, \hat{\xi}_2, \ldots, \hat{\xi}_d)$ in $\mathfrak{se}(3)^d$, the matrix exponential map is expressed as:

$$\begin{aligned} \exp : \mathfrak{se}(3)^d &\to \mathrm{SE}(3)^d, \\ T^d = \exp(\hat{\xi}^d) &= \left(\exp(\hat{\xi}_1), \exp(\hat{\xi}_2), \ldots, \exp(\hat{\xi}_d)\right), \end{aligned} \tag{2.49}$$

where $\exp(\hat{\xi}_i)$ is the matrix exponential of the i-th component. Similarly, the matrix logarithmic map is defined as:

$$\begin{aligned} \log : \mathrm{SE}(3)^d &\to \mathfrak{se}(3)^d, \\ \hat{\xi}^d = \log(T^d) &= (\log(T_1), \log(T_2), \ldots, \log(T_d)), \end{aligned} \tag{2.50}$$

where $\log(T_i)$ is the matrix logarithm of the i-th component.

The hat and vee maps, fundamental in representing twists and their corresponding matrices in SE(3), can be efficiently extended to $\mathrm{SE}(3)^d$.

The hat map extension maps vectors from $\mathbb{R}^{6d}$ into matrices within $\mathfrak{se}(3)^d$, as follows:

$$\begin{aligned} \wedge : \mathbb{R}^{6d} &\to \mathfrak{se}(3)^d, \\ \hat{\xi}^d &= (\hat{\xi}_1, \hat{\xi}_2, \ldots, \hat{\xi}_d), \end{aligned} \tag{2.51}$$

where $\hat{\xi}_i$ is the matrix form of the i-th component of the twist vector ξ^d. This conversion expresses the collective rotational and translational velocities of multiple bodies as a single entity. Conversely, the vee map converts matrices from $\mathfrak{se}(3)^d$ back into vectors in $\mathbb{R}^{6d}$:

$$\begin{aligned} \vee : \mathfrak{se}(3)^d &\to \mathbb{R}^{6d}, \\ \xi^d &= (\xi_1, \xi_2, \ldots, \xi_d), \end{aligned} \tag{2.52}$$

where ξ_i is the vector form of the i-th matrix in $\hat{\xi}^d$.

These extended maps facilitate the manipulation of high-dimensional transformations in a vectorized form, enhancing computational efficiency and clarity.

For the Cartesian product $\mathrm{SE}(3)^d$, the Frobenius norm in $\mathfrak{se}(3)^d$ is defined as:

$$\|\hat{\xi}^d\|_F^2 = \sum_{i=1}^{d} \|\hat{\xi}_i\|_F^2. \tag{2.53}$$

This corresponds to the Euclidean norm in $\mathbb{R}^{6d}$ as follows:

$$\|\hat{\xi}^d\|_F^2 = \sum_{i=1}^{d} \|\xi_i\|_2^2 = \sum_{i=1}^{d} \left(2\|\omega_i\|_2^2 + \|v_i\|_2^2\right) = 2\|\omega^d\|_2^2 + \|v^d\|_2^2, \tag{2.54}$$

where $\xi_i \in \mathbb{R}^6$ is decomposed into angular velocity $\omega_i \in \mathbb{R}^3$ and linear velocity $v_i \in \mathbb{R}^3$. This indicates that, in vector form, the rotational error is scaled by a factor of 2 compared to the translational error. To equalize their contributions to the overall error, one would need to either double the weight of the rotational error or halve the weight of the translational error.

Chapter 3

Optimal Reparameterization of Parametric Curves

Reparameterization lets us rewrite curves, capturing their inherent shape independent of how they're traced. This chapter explores methods for finding optimal reparameterizations to calculate the distance between shapes. We'll delve into these techniques, how they distinguish shapes, and analyze their properties.

3.1 Foundations for Discrete Reparameterization

Discrete reparameterization is a key technique in numerical analysis and applied mathematics, especially for interpolating curves and trajectories. This section covers the basics of discrete reparameterization, focusing on interpolating sequences of discrete points along a curve. We start by detailing the interpolation method, showing how discrete points are handled and how a continuous approximation is created.

3.1.1 Geodesic Interpolation

The interpolation method in equation (2.19) can be modified to handle a sequence of n discrete points along a curve. We define $\{\bar{c}_k := c(s_k)\}_{k=0}^{n}$ as the sequence of elements, where $\bar{c}_k$ corresponds to the timestamp s_k, with $0 = s_0 < s_1 < \cdots < s_n = 1$, where the interpolation function is given by:

$$\tilde{c}(t) = \sum_{k=0}^{n-1} \chi_{[s_k, s_{k+1})}(t) \exp\left(\frac{t - s_k}{s_{k+1} - s_k} \log\left(\bar{c}_{k+1}\bar{c}_k^{-1}\right)\right)\bar{c}_k, \tag{3.1}$$

and $\chi_{[s_k, s_{k+1})}(t)$ is an activation function defined by:

$$\chi_{[s_k, s_{k+1})}(t) = \begin{cases} 1 & \text{if } t \in [s_k, s_{k+1}), \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

The function χ ensures that each segment κ of the curve $\bar{c}$ contributes only within its designated interval, providing a continuous approximation of $\bar{c}$ through $\tilde{c}(t)$. For each segment κ on the interval $[s_k, s_{k+1})$, the interpolation is detailed as:

$$\kappa(t) = \exp((t-s_k)\eta_k)\,\bar{c}_k, \tag{3.3}$$

where η_k is computed by:

$$\eta_k = \frac{\log(\bar{c}_{k+1}\bar{c}_k^{-1})}{s_{k+1}-s_k}. \tag{3.4}$$

For a curve in a matrix group, the right logarithmic derivative $\delta^r(\kappa(t))$, derived from the Maurer-Cartan form equation (2.6), is:

$$\begin{aligned}\delta^r(\kappa(t)) &= \dot{\kappa}(t)\kappa(t)^{-1} \\ &= \eta_k \exp((t-s_k)\eta_k)\,\bar{c}_k\,(\eta_k \exp((t-s_k)\eta_k)\,\bar{c}_k)^{-1} \\ &= \eta_k \exp((t-s_k)\eta_k)\exp((t-s_k)\eta_k)^{-1} \\ &= \eta_k,\end{aligned} \tag{3.5}$$

which shows that the right logarithmic derivative remains constant over each interval $[s_k, s_{k+1})$.

3.1.2 Curve Reparameterization

Consider a discrete curve $\bar{c}$ defined over the interval I. The reparameterization of $\bar{c}$ involves applying a diffeomorphism $\varphi \in \mathrm{Diff}^+(I)$. This process results in a new curve, denoted $\bar{c}_\varphi$.

For each parameter s_i corresponding to the points of $\bar{c}$, the diffeomorphism φ transforms these parameters according to:

$$\hat{s}_i = \varphi(s_i), \quad i = 0, 1, \ldots, n. \tag{3.6}$$

The reparameterized curve $\bar{c}_\varphi$ is constructed by evaluating the continuous approximation of $\bar{c}$, denoted $\tilde{c}$, at the transformed parameters $\hat{s}_i$. This is in accordance with the interpolation equation (3.1), leading to:

$$\bar{c}_{\varphi,i} = \tilde{c}(\hat{s}_i), \quad i = 0, 1, \ldots, n. \tag{3.7}$$

3.1.3 SRVT

Given a curve segment κ as in equation (3.3), its SRVT, denoted $\bar{q}_k$, is computed using (3.4) and (3.5). The SRVT for a curve segment is defined as:

$$\bar{q}_k = \frac{\delta^r(\kappa)}{\sqrt{\|\delta^r(\kappa)\|}} = \frac{\eta_k}{\sqrt{\|\eta_k\|}}. \tag{3.8}$$

The SRVT for the entire curve $\bar{c}$ at time t is given by:

$$\mathcal{R}(\bar{c})(t) := \bar{q}(t) = \sum_{k=0}^{n-1} \chi_{[s_k, s_{k+1})}(t)\bar{q}_k, \tag{3.9}$$

where $\chi_{[s_k,s_{k+1})}(t)$ is the activation function defined in equation (3.2).

As established in Lemma 3.9 [17], the inverse SRVT is crucial for reconstructing a curve from its SRVT representation. Specifically, the reconstruction of the curve's point $\bar{c}_{k+1}$ from $\bar{c}_k$ is facilitated by:

$$\bar{c}_{k+1} = \exp\left(\|\bar{q}_k\|\bar{q}_k\right)\bar{c}_k, \tag{3.10}$$

where $\bar{c}_0 = e$, and e denotes the identity element of the group.

The step-by-step reconstruction of each segment k, where $1 \leq k < n-1$, unfolds as follows:

$$\begin{aligned}
\bar{c}_{k+1} &= \exp\left((s_{k+1}-s_k)\|\bar{q}_k\|\bar{q}_k\right)\bar{c}_k && (3.11)\\
&= \exp\left((s_{k+1}-s_k)\left\|\frac{\eta_k}{\sqrt{\|\eta_k\|}}\right\|\frac{\eta_k}{\sqrt{\|\eta_k\|}}\right)\bar{c}_k && (3.12)\\
&= \exp\left(\frac{s_{k+1}-s_k}{s_{k+1}-s_k}\log(\bar{c}_{k+1}\bar{c}_k^{-1})\right)\bar{c}_k && (3.13)\\
&= \bar{c}_{k+1}. && (3.14)
\end{aligned}$$

This iterative approach effectively demonstrates how the inverse SRVT enables the sequential recovery of $\bar{c}_{k+1}$ from $\bar{c}_k$, thereby enabling the comprehensive reconstruction of $\bar{c}$ from $\bar{q}$.

3.2 Analysis of SRVT

Inspired by the work of Martin Bruveris [47], we expand on the analysis of the Square Root Velocity Transform (SRVT) for curves in Lie matrix groups. Given a curve $c : I \to G$, where G is a Lie matrix group (specifically $SO(3)$ or $SE(3)$), we consider its transformation under a specific mapping $\mathcal{R}$, defined as follows:

$$\mathcal{R} : G \to \mathbb{R}^n, \quad \mathcal{R}(c) = \frac{c'}{\|c'\|}, \tag{3.15}$$

where c' denotes the derivative, of c with respect to its parameter. The SRVT is represented as a composition of two mappings:

$$\mathcal{R} = V \circ \mathcal{S}, \tag{3.16}$$

where

$$\begin{aligned}
&V : \mathbb{R}^n \to \mathbb{R}^n,\\
&V(x) = \frac{x}{\sqrt{\|x\|}} = x\|x\|^{\frac{1}{2}-1},\ \forall\, x \in \mathbb{R}^n,
\end{aligned} \tag{3.17}$$

and

$$\begin{aligned}
&\mathcal{S} : G \to \mathbb{R}^n,\\
&\mathcal{S}(c) = c',\ \forall\, c \in G.
\end{aligned} \tag{3.18}$$

As established in A.3, V is a $\frac{1}{2}$-Hölder continuous function, implying it adheres to the property:

$$\|V(x)-V(y)\| \leq C\|x-y\|^{\frac{1}{2}}, \forall\, x, y \in \mathbb{R}^n, \tag{3.19}$$

where $C \in \mathbb{R}^+$ is a constant. Consequently, the SRVT's sensitivity to changes in the curve can be quantified as follows:

$$\|\mathcal{R}(x)-\mathcal{R}(y)\|_{L^2} = \left\|V(x')-V(y')\right\|_{L^2} \leq C\left\|x'-y'\right\|_{L^2}^{\frac{1}{2}}, \tag{3.20}$$

illustrating that small variations in the input curve result in bounded changes in the transformed curve, as per the SRVT's $\frac{1}{2}$-Hölder continuity.

Assume that $c : I \to G$ is a continuously differentiable function, denoted as $c \in C^1(I, G)$. This implies that its derivative c' is continuous across the interval I, or $c' \in C^0(I, G)$. Furthermore, if c' is Lipschitz continuous with a Lipschitz constant L, then for all $t_1, t_2 \in I$, the following inequality is satisfied:

$$\|c'(t_1)-c'(t_2)\|_{L^2} \leq L\|t_1-t_2\|_{L^2}, \tag{3.21}$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^n$. This condition asserts that the rate of change of the function c, as indicated by its derivative c', varies in a controlled manner across I, with the magnitude of change in c' bounded linearly by the distance between any two points within I.

This implies

$$\|\mathcal{R}(c)(t_1)-\mathcal{R}(c)(t_2)\|_{L^2} \leq C_L\|t_1-t_2\|_{L^2}^{\frac{1}{2}}, \tag{3.22}$$

where $C_L = C \cdot L$ is a constant. This inequality demonstrates that the SRVT's sensitivity to reparameterization is bounded by the reparameterization itself. In other words, the SRVT's response to changes in the curve is controlled by the rate of change of the curve.

This follows directly from the results proven in A.3, where we established the α-Hölder continuity of the mapping V. Specifically, the $\frac{1}{2}$-Hölder continuity of V ensures that:

$$\|\mathcal{R}(c)(t_1)-\mathcal{R}(c)(t_2)\|_{L^2} \leq C_L\|t_1-t_2\|_{L^2}^{\frac{1}{2}}, \tag{3.23}$$

thereby validating the SRVT's controlled sensitivity to variations in the input curve.

3.3 Dynamic Programming Using a Fully Discretized Method

In this section, we present a fully discretized method for solving the optimal reparameterization problem using dynamic programming. We apply the Square Root Velocity Transform (SRVT) to a discretized curve and then use dynamic programming techniques to find the optimal solution. We will detail the implementation

process, discuss the search depth, and provide numerical results to demonstrate the effectiveness of our approach.

3.3.1 Overview of Methodology

To determine the shape distance $d_{\mathcal{S}_*}$ as in Equation (2.18), we use the fully discretized approach by Bauer et al. [25] rather than the semidiscretized method by Wøien [48]. This method converts the continuous diffeomorphism problem into a finite-dimensional optimization problem. The interval $I = [0,1]$ is discretized into $\mathcal{I} = [s_1, s_2, \ldots, s_M]$, establishing a series of linear transformations.

Each transformation $\varphi_{k,l;i,j}$ maps the subinterval $[k,i]$ to the target interval $[l,j]$, defined as:

$$\varphi_{k,l;i,j}(t) = l + (t-k)\frac{j-l}{i-k}, \tag{3.24}$$

with the derivative's square root:

$$\sqrt{\dot{\varphi}_{k,l;i,j}} = \sqrt{\frac{j-l}{i-k}}. \tag{3.25}$$

The optimal reparameterization $\varphi_{k,l;i,j}$ is found by minimizing a local energy functional $\mathcal{E}(k,l,i,j)$, which quantifies the difference between the curves q_1 and q_2 reparameterized via $\varphi_{k,l;i,j}$:

$$\mathcal{E}(k,l,i,j) := \left\| q_1(t) - q_2(\varphi_{k,l;i,j}(t))\sqrt{\dot{\varphi}_{k,l;i,j}} \right\|_{L^2} + \lambda \cdot \Psi, \tag{3.26}$$

where λ is a positive regularization parameter and Ψ is a regularization term.

For the SE(3) group, which includes both rotational and positional components, the local energy functional requires adjustment. Specifically, the first three of the six elements, corresponding to the rotational components, are scaled by a factor of 2 due to the norm scaling in SE(3), see 2.5. The modified local energy functional for SE(3) is:

$$\mathcal{E}_{\mathrm{SE3}}(k,l,i,j) := \left\| \begin{pmatrix} 2 \cdot \left(q_1(t)_{\mathrm{rot}} - q_2(\varphi_{k,l;i,j}(t))_{\mathrm{rot}} \right) \\ q_1(t)_{\mathrm{tra}} - q_2(\varphi_{k,l;i,j}(t))_{\mathrm{tra}} \end{pmatrix} \sqrt{\dot{\varphi}_{k,l;i,j}} \right\|_{L^2} + \lambda \cdot \Psi, \tag{3.27}$$

where $q_1(t)_{\mathrm{rot}}$ and $q_2(\varphi_{k,l;i,j}(t))_{\mathrm{rot}}$ are the rotational components, and $q_1(t)_{\mathrm{tra}}$ and $q_2(\varphi_{k,l;i,j}(t))_{\mathrm{tra}}$ are the translational components.

As we use dynamic programming on fully discretized intervals, we achieve an optimal solution, but not necessarily a unique one. To ensure uniqueness and control the solution's properties, we utilize a regularization term as in [25], penalizing deviation from the original parameterization:

$$\sum_{k < s_m \leq i} \left| \varphi_{k,l;i,j}(s_m) - s_m \right|^2. \tag{3.28}$$

The functional constructs a cost matrix A of size $M \times M$:

$$A_{i,j} = \min_{k,l \in \mathcal{I}, k<i, l<j} \big(\mathcal{E}(k,l,i,j) + A_{k,l} \big). \tag{3.29}$$

This matrix is calculated using dynamic programming to minimize the cost of transforming subintervals $[k,i]$ to $[l,j]$. The dictionary P records the optimal indices (k,l) that yield the minimum, storing them as $P(i,j) = (k,l)$. This allows backtracking from the final state A_{S_m,S_m} to the start, reconstructing the sequence of transformations $\varphi_{k,l,i,j}$, constructing a piecewise linear approximation of the diffeomorphism φ, as illustrated in Fig. 3.1.

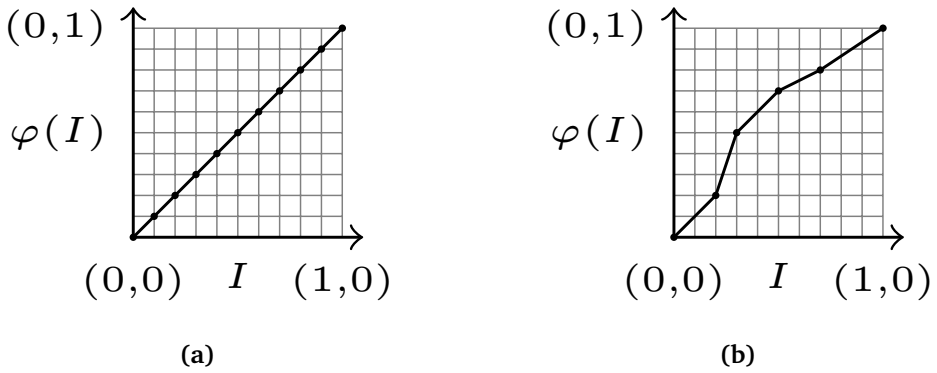


Figure 3.1: Reparameterization using full discretization. (a) the identity transformation (no change to the original parameterization), and (b) a reparameterized path.

The condition $k < i$ and $l < j$ ensures the transformation proceeds positively, avoiding loops and maintaining the orientation of q_2.

3.3.2 Neighborhood Search Strategy

Despite its straightforward implementation, this method presents significant computational challenges. For a grid comprising $n \times n$ points, the process involves solving n^2 subproblems, each requiring $\mathcal{O}(n^2)$ operations. This leads to a total computational complexity of $\mathcal{O}(n^4)$, making the method computationally intensive, especially for large-scale applications.

To mitigate these computational demands, a strategic limitation of the search area for each subproblem is proposed, as illustrated in Figure 3.2. The size of this area must be adaptable based on the grid's dimensions; a fixed size may excessively restrict path movement in smaller grids. Employing this neighborhood search strategy significantly reduces computational cost to $\mathcal{O}(n^2|\mathcal{N}(n)|)$, where $|\mathcal{N}(n)|$ indicates the search area's size.

The limited search area is defined as follows:

$$\mathcal{N}_{(i,j)}(n) = \{(k,l) \mid i-m \le k < i,\ j-m < l < j,\ \gcd(i-k, j-l) = 1\}, \tag{3.30}$$

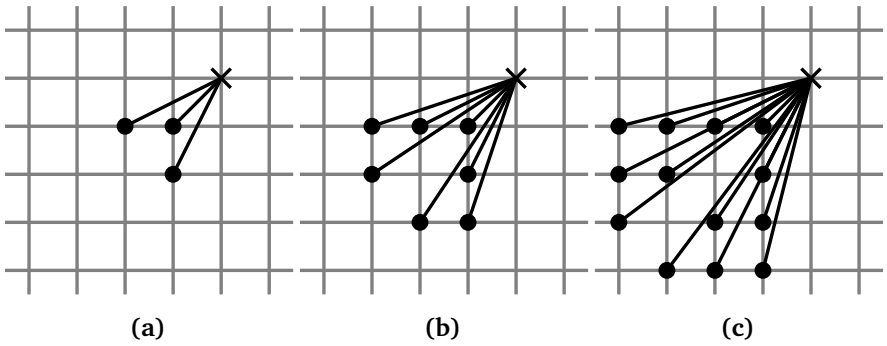


Figure 3.2: Visualization of limited search areas for different values of search depth m: (a) $m = 2$, (b) $m = 3$, and (c) $m = 4$. These figures demonstrate how increasing search depths expand the area being considered, thereby influencing the computational cost and efficiency of the reparameterization process.

where $m = f(n) \in \mathbb{N}^+$ is a function of the grid size n that determines the extent of the search area. The condition $\gcd(i-k, j-l) = 1$ ensures that the path does not traverse any intermediate grid points, maintaining a direct route between the specified points and improving the computational efficiency of the process.

To illustrate the implementation of this strategy, we present the pseudocode for the optimal reparameterization process using dynamic programming in Algorithm 1. This algorithm aligns two curves by optimizing the parameterizations of the second curve, reducing the computational complexity through the limited search area defined earlier.

Algorithm 1 Optimal Reparameterization via Dynamic Programming

Require: c_1, c_2: Curves, $I_1, I_2 \in \mathbb{R}$: Initial parameterizations

function OPTIMIZEREPARAM(c_1, c_2, I_1, I_2)
  $(I, c_1', c_2') \leftarrow \text{AlignCurves}(c_1, c_2, I_1, I_2)$
  $(c_{1,e}', c_{2,e}') \leftarrow \text{MoveStartToIdentity}(c_1', c_2')$ ▷ (2.6)
  $(q_1, q_2) \leftarrow \text{vee}(\text{SRVT}(c_{1,e}', c_{2,e}'))$ ▷ (3.8), (2.22), (2.40)
  $A, P \leftarrow \text{ComputeCostMatrix}(q_1, q_2)$ ▷ (3.29)
  $\text{path} \leftarrow \text{TraceOptimalPath}(P)$
  $I_{2,opt} \leftarrow \text{LinearInterpolate}(\text{path}_x, \text{path}_y)(I_2)$
  $c_{2,opt} \leftarrow \text{Interpolate}(I_2, I_{2,opt})(c_2)$ ▷ (3.1)
  return $(c_{2,opt}, I_{2,opt})$
end function

The algorithm detailed in Algorithm 2 calculates the cost and parent matrices used in the dynamic programming approach. By leveraging the limited search area, it ensures efficient computation of the optimal path.

Algorithm 2 Calculate Cost and Parent Matrices for Dynamic Programming

Require: $I \in \mathbb{R}^n$: Grid points
Require: $q_0, q_1 \in \mathbb{R}^{n-1}$: Values defined on the domain of I
Require: depth $\in \mathbb{N}^+$: Depth of the search area

function CALCULATECOSTMATRIXANDPARENT(I, q_0, q_1, depth)
  $n \leftarrow$ length(I)
  $A \leftarrow$ matrix_of_infinity(n, n) ▷ Initialize cost matrix to a large value
  $P \leftarrow$ empty matrix(n, n) ▷ Initialize parent matrix
  $A[0,0] \leftarrow 0$ ▷ Set initial cost to zero
  $P[0,0] \leftarrow$ None ▷ No predecessor for initial condition
  for $i \leftarrow 0$ **to** $n-1$ **do**
    for $j \leftarrow 0$ **to** $n-1$ **do**
      preds $\leftarrow$ findPreds(i, j, depth) ▷ (3.30)
      for pred $\in$ preds **do**
        cost_to_pred $\leftarrow$ localCost(i, j, pred, q_0, q_1) ▷ (3.26), (3.27)
        cost $\leftarrow$ cost_to_pred $+ A$[pred] ▷ (3.29)
        if cost $< A[i,j]$ **then**
          $A[i,j] \leftarrow$ cost
          $P[i,j] \leftarrow$ pred
        end if
      end for
    end for
  end for
  return P, A
end function

3.3.3 Synthetic Data Generation

In this subsection, we outline the process of generating synthetic data for numerical analysis. This synthetic data is critical for evaluating our reparameterization framework. We generate data by solving differential equations with different parameterizations. The synthetic data includes trajectories, parameterizations, and curves for different groups such as SO(3), SE(3), $SO(3)^3$, and $SE(3)^3$. These elements are used later in our numerical analysis.

Generating Synthetic Trajectories

To generate synthetic trajectories, we model the system using the differential equation:

$$\frac{d}{dt}g(t) = g(t)\hat{u}(t), \quad t \in [0,1], \tag{3.31}$$

where $g(t) \in SO(3)$ or SE(3), and $\hat{u}(t)$ is the hat map of $u(t)$. We assume an initial value $g(0) = e$ (the identity element). By varying $\hat{u}(t)$ with different parameterizations, we synthesize diverse datasets. The trajectories are approximated using the fourth-order Runge-Kutta (RK4) method [49, 50] over discrete time intervals.

For SO(3), we set $u(t) = \omega_i(t) \in \mathbb{R}^3$. For SE(3), we set $u(t) = \xi_i(t) = [\omega_i(t), v_i(t)]^T \in \mathbb{R}^6$, which is mapped to the corresponding Lie algebra via (2.21) and (2.39).

To demonstrate the generation of synthetic data for Cartesian curves, $n = 3$ is chosen. For either $SO(3)^3$ or $SE(3)^3$, we define three different functions $u(t)$. Thus, solving (3.31) generates three synthetic trajectories over discrete time intervals.

We define nine $\omega_i(t)$ and nine $v_i(t)$ to generate the rotational and translational components, respectively, used for $u(t)$. The rotational functions are defined in (3.32), and the translational functions are defined in (3.33).

$$\begin{aligned}
\omega_1(t) &= [2\sin(3t)\exp(t), 4\cos(3t), 3t\sin(t)\cos(t)]^T \\
\omega_2(t) &= [3t, 4\sin(10t), 3t\sin(t)\cos(t)]^T \\
\omega_3(t) &= [4t^2, 5\sin(4t)\sin(6t), 3t\cos(t)]^T \\
\omega_4(t) &= [3\sin(2t)\exp(-t), 5\cos(2t), 4t\sin(t)\cos(t)]^T \\
\omega_5(t) &= [t^3, 5\sin(5t), 2t\sin(t)\sin(t)]^T \\
\omega_6(t) &= [2t^2, 3\sin(6t)\cos(t), 5t\cos(t)\cos(t)]^T \\
\omega_7(t) &= [-1, 6\sin(3t), 3t\sin(t)]^T \\
\omega_8(t) &= [t^2, 4\cos(3t), t\sin(t)]^T \\
\omega_9(t) &= [3t, 5\sin(2t), 2t\cos(t)]^T
\end{aligned} \tag{3.32}$$

$$
\begin{aligned}
v_1(t) &= [3t\sin(t)\cos(t), t, \cos(t), \sin(3t)]^T \\
v_2(t) &= [\sin(2t), \cos(3t), \exp(t)]^T \\
v_3(t) &= [\cos(t)\cos(3t), \sin(4t), \cos(2t)]^T \\
v_4(t) &= [t^2, \sin(t^2), \exp(-t)]^T \\
v_5(t) &= [\cos(2t), \log(t+1), t]^T \\
v_6(t) &= [\sin(t)\cos(t), t^3, \sin(t^3)]^T \\
v_7(t) &= [\cos(5t), \exp(t/2), t\sin(t)]^T \\
v_8(t) &= [\sin(2t), t\exp(-t), \cos(t^2)]^T \\
v_9(t) &= [t^2\cos(t), \sin(t^2), -0.5]^T
\end{aligned}
\tag{3.33}
$$

By solving (3.31) with $u(t) = \omega_i(t)$, for $i = 1, 2, \ldots, 9$ over equidistant points on $[0, 1]$, we obtain the rotational trajectories depicted in Figure 3.3. Similarly, by solving (3.31) with $u(t) = \xi_i(t)$, for $i = 1, 2, \ldots, 9$ over equidistant points on $[0, 1]$, and extracting the positions, we generate the translation plot shown in Figure 3.4. These plots provide a clear visual representation of the varied rotational and translational paths used later in the analysis. The rotations are plotted on spheres, representing pure rotational movements.

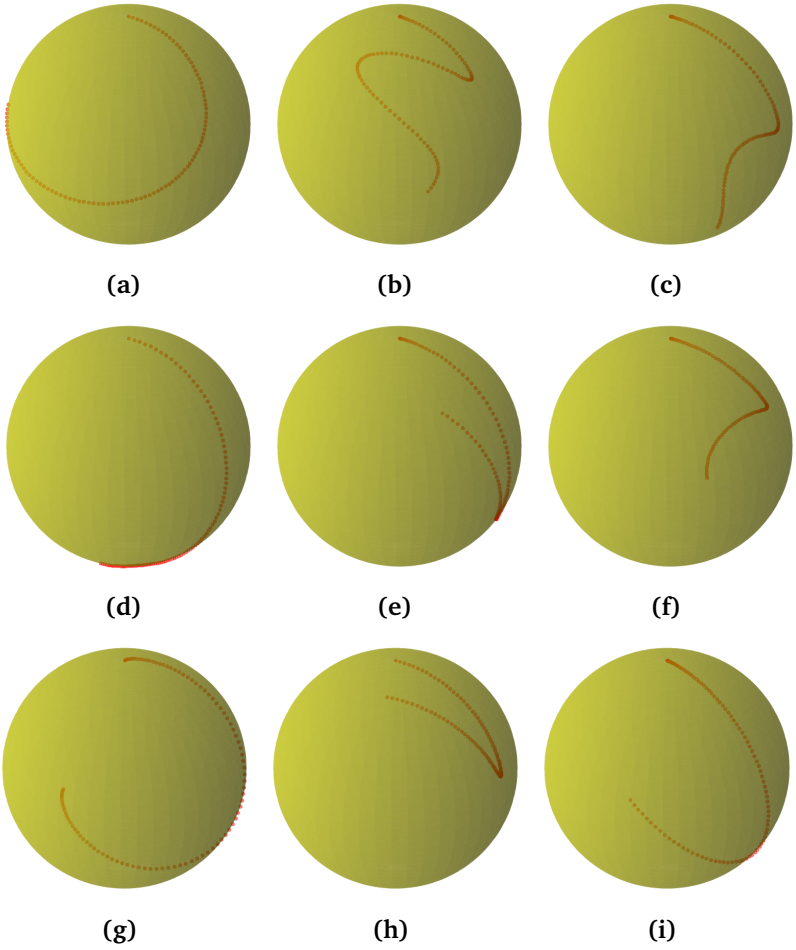


Figure 3.3: Rotations generated from synthetic data using (3.32). Each subfigure (a-i) shows the rotation of the initial vector $[0, 0, 1]$ governed by (3.31) with $\hat{u}(t) = \hat{\omega}_i(t)$, evaluated over $[0, 1]$ with 100 time points. Subfigures (a-i) correspond to $i = 1$ to $i = 9$, respectively.

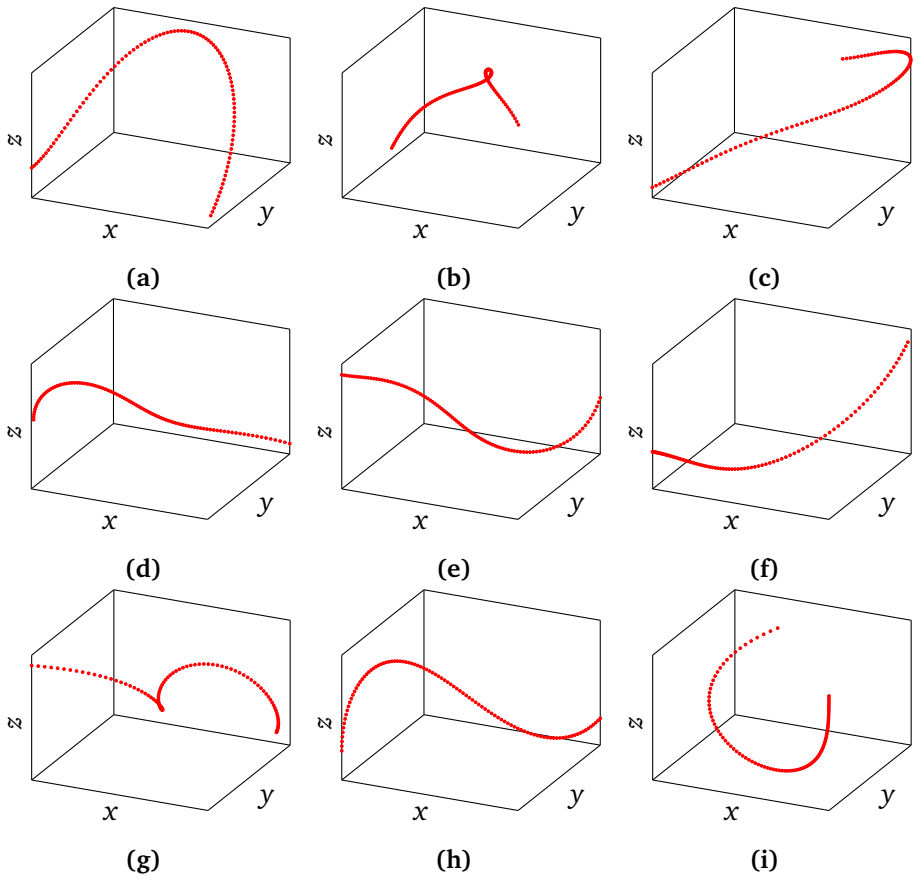


Figure 3.4: Translation part of the SE(3) elements generated from synthetic data by solving (3.31) with the functions defined in (3.32) and (3.33). Each subfigure (a-i) illustrates the translation governed by (3.31) with $\hat{u}(t) = \hat{\xi}_i(t)$, evaluated over $[0, 1]$ with 100 time points. Subfigures (a-i) correspond to $i = 1$ to $i = 9$, respectively.

Generate Synthetic Parameterizations

We generate three distinct synthetic parameterizations, each created by $\varphi(x) \in \text{Diff}^+([0,1])$:

$$\varphi(x) = x + F(x), \tag{3.34}$$

where $\varphi'(x) = 1 + F'(x) > 0$. The function $F(x)$ is constructed using an orthogonal sine basis function $\psi_n(x)$:

$$\psi_n(x) = \frac{\sin(n\pi x)}{n\pi}, \quad n \in \mathbb{N}. \tag{3.35}$$

These basis functions are orthogonal over $[0,1]$ and vanish at the endpoints, making them suitable for weighted sums. Consequently, $F(x)$ is represented as:

$$F(x) = \sum_{n=1}^{M} w_n \psi_n(x) = \sum_{n=1}^{M} w_n \frac{\sin(n\pi x)}{n\pi}, \tag{3.36}$$

where w_n are the weights associated with each basis function. The derivative $F'(x)$ is:

$$F'(x) = \sum_{n=1}^{M} w_n \cos(n\pi x). \tag{3.37}$$

Since each cosine term is bounded by 1, the derivative $F'(x)$ satisfies:

$$|F'(x)| \leq \|\mathbf{w}\|_1. \tag{3.38}$$

Therefore, the derivative of $\varphi(x)$ is:

$$\varphi'(x) = 1 + \sum_{n=1}^{M} w_n \cos(n\pi x). \tag{3.39}$$

To ensure $\varphi'(x) > 0$, we scale $F(x)$ by a factor $\frac{1-\epsilon}{\|\mathbf{w}\|_1}$, where ϵ is a small positive number. This guarantees that the sum of the cosine terms is less than 1, ensuring $\varphi'(x) > 0$:

$$1 + \sum_{n=1}^{M} w_n \frac{1-\epsilon}{\|\mathbf{w}\|_1} \cos(n\pi x) > 0. \tag{3.40}$$

The synthetic parameterizations generated using this method are illustrated in Figure 3.5. For this, we set $M = 4$, initialize the weights w_n with a normal distribution $\mathcal{N}(0,2)$, and use 100 equidistant steps with a tolerance of $\epsilon = 10^{-8}$.

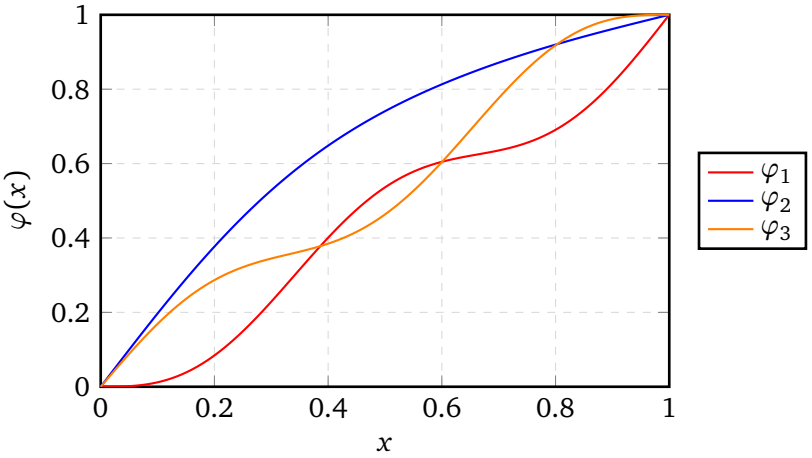


Figure 3.5: Visualization of three distinct synthetic parameterizations generated by setting $M = 4$, initializing the weights w_n with a normal distribution $\mathcal{N}(0, 2)$, and using 100 equidistant steps with a tolerance of $\epsilon = 10^{-8}$.

Generate Synthetic Curves

To evaluate our reparameterization framework, we generate synthetic curves by solving Equation (3.31) with the parameterizations $\varphi_1, \varphi_2, \varphi_3$ described in Section 3.3.3, as well as an equidistant parameterization. This differential equation is solved using the functions defined in Equations (3.32) and (3.33), which provide $u(t) = \omega_i(t)$ for SO(3) elements and $u(t) = \xi_i(t) = [\omega_i(t), v_i(t)]^T$ for SE(3) elements, where $i = 1, 2, 3$. This process creates single-element and three-element Cartesian curves, with each group comprising three shapes, each represented under four different parameterizations, yielding twelve curves per group.

For multi-component groups $\mathrm{SO}(3)^3$ and $\mathrm{SE}(3)^3$, we define composite parameters:

$$\begin{aligned} \omega_{i,i+1,i+2} &= [\omega_i, \omega_{i+1}, \omega_{i+2}], \\ \xi_{i,i+1,i+2} = [\xi_i, \xi_{i+1}, \xi_{i+2}] &= [[\omega_i, v_i], [\omega_{i+1}, v_{i+1}], [\omega_{i+2}, v_{i+2}]] \end{aligned} \tag{3.41}$$

for $i \in \{1, 4, 7\}$.

The idea is to create three interconnected curves that together form a single composite curve. These parameters are used to solve the differential equation under the same parameterizations, generating twelve curves per group. Each group of four curves shares identical geometric shapes, differentiated by parameterization.

The notation for the curves is as follows: single-element curves are denoted c_i^j, where i indicates the parameter used (ω_i or $\xi_i = [\omega_i, v_i]$) and j the specific parameterization (φ_j). Curves generated using the equidistant parameterization are denoted as c_i^{eq}. Cartesian curves are expressed as $c_{i,i+1,i+2}^j$ for individual parameterizations and $c_{i,i+1,i+2}^{\mathrm{eq}}$ for the equidistant parameterization.

3.3.4 Numerical Analysis

In this subsection we will use the data in 3.3.3 perform reparameterization, classification and measure performance in terms of accuracy and computation time as the search size increases.

Reparameterization of Curves

In this subsection, we evaluate the reparameterization framework using synthetic curves generated previously (Section 3.3.3). The goal is to reparameterize the equidistantly parameterized curves c_i^{eq} into the curves c_i^j for $i \in \{1,2,3\}$ and $c_{i,i+1,i+2}^{\text{eq}}$ into the curves $c_{i,i+1,i+2}^j$ for $i \in \{1,4,7\}$, for all parameterizations $j \in \{1,2,3\}$. This evaluation checks whether we can find the optimal reparameterization when the shapes are the same, demonstrating that we can effectively remove the effects of parameterization through reparameterization. It does not, however, determine whether we can differentiate between different shapes.

The objective is to find the optimal reparameterization $\hat{\varphi}_j$ that approximates φ_j. The relationship between the curves is given by:

$$c_i^j = c_i^{\text{eq}} \circ \varphi_j \tag{3.42}$$

Hence, each curve c_i^j is obtained by composing the equidistant curve c_i^{eq} with one of the parameterizations φ_1, φ_2, or φ_3. Thus, to find the optimal reparameterization $\hat{\varphi}_j$, we minimize the distance $d_{\mathcal{S}_*}$ between c_i^j and c_i^{eq}:

$$d_{\mathcal{S}_*}(c_i^j, c_i^{\text{eq}}) = \min_{\hat{\varphi}_j} d_{\mathcal{P}_*}(c_i^{\text{eq}} \circ \varphi_j, c_i^{\text{eq}} \circ \hat{\varphi}_j) \tag{3.43}$$

This minimization process helps to determine $\hat{\varphi}_j$ such that $\hat{\varphi}_j \approx \varphi_j$, indicating that the reparameterization framework can effectively approximate the original parameterizations.

The results after reparameterization for the curves in SO(3), SE(3), $\text{SO}(3)^3$, and $\text{SE}(3)^3$ are presented in Figures 3.6, 3.7, 3.8, and 3.9, respectively.

The reparameterization results show that the optimal reparameterizations $\hat{\varphi}_j^i$ for $i \in \{1,2,3\}$ and $\hat{\varphi}_j^{i,i+1,i+2}$ for $i \in \{1,4,7\}$ closely approximate the original parameterizations φ_j for all $j \in \{1,2,3\}$. This indicates that our framework can effectively remove the effects of parameterization. The results are consistent across all groups, attesting to its robustness. However, this evaluation only verifies the ability to find the parameterization for curves with the same shape.

Overall, the results affirm the reparameterization framework's effectiveness in approximating original parameterizations and suggest its potential for broader applications in shape analysis and related fields.

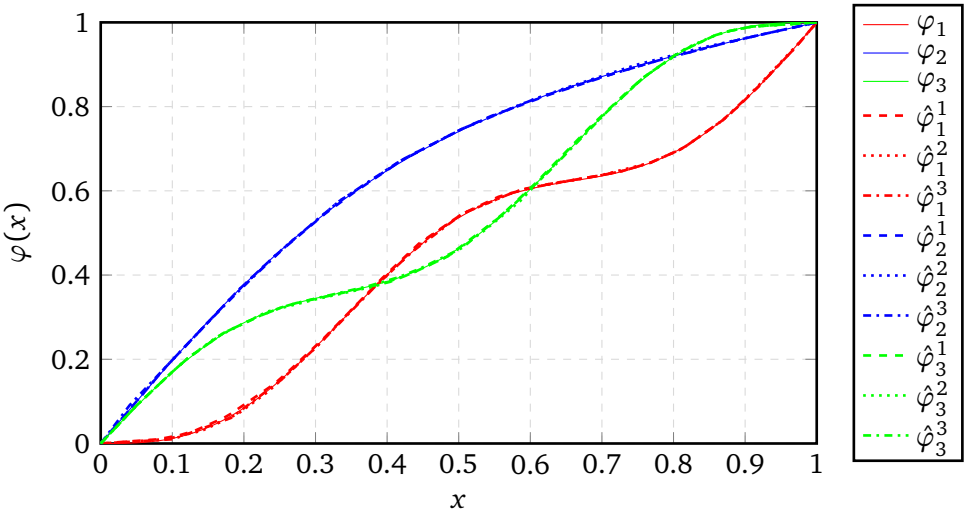


Figure 3.6: Comparison of reparameterized curves $\hat{\varphi}_j^i$ with the analytical parameterizations φ_j for curves in SO(3). The curves c_i^{eq} have been fitted to c_i^j for $j = 1, 2, 3$, and for all shapes $i = 1, 2, 3$. Optimal reparameterizations are indicated by $\hat{\varphi}_i^j$ being close to φ_j.

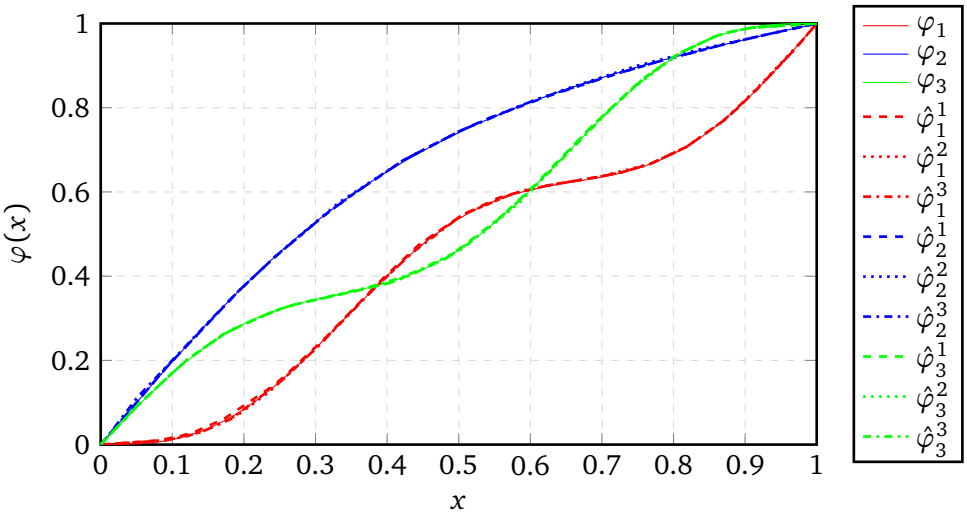


Figure 3.7: Comparison of reparameterized curves $\hat{\varphi}_j^i$ with the analytical parameterizations φ_j for curves in SE(3). The curves c_i^{eq} have been fitted to c_i^j for $j = 1, 2, 3$, and for all shapes $i = 1, 2, 3$. Optimal reparameterizations are indicated by $\hat{\varphi}_j^i$ being close to φ_j.

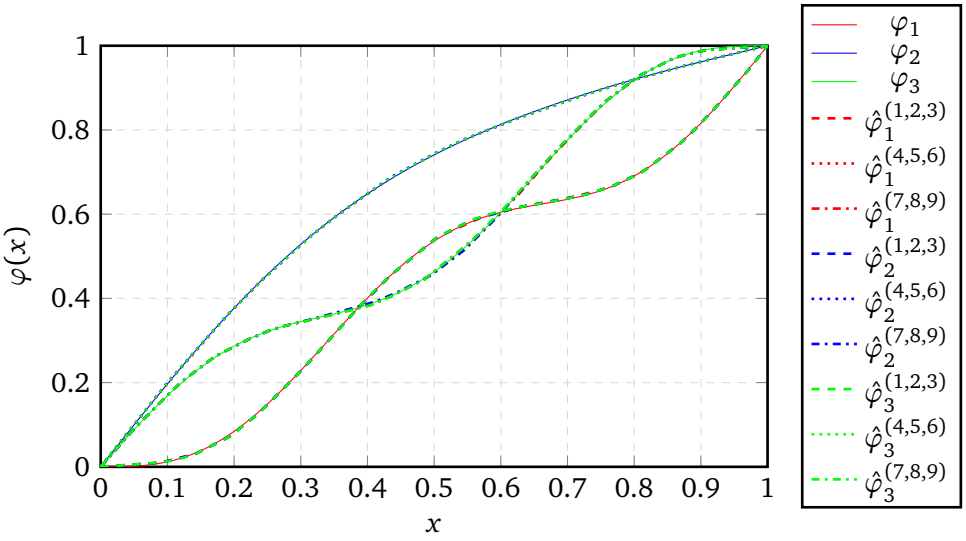


Figure 3.8: Comparison of reparameterized curves $\hat{\varphi}_j^{i,i+1,i+2}$ with the analytical parameterizations φ_j for curves in SE(3). The curves $c_{i,i+1,i+2}^{\text{eq}}$ have been fitted to $c_{i,i+1,i+2}^{j}$ for $i \in \{1,2,3\}$, and for all shapes $i \in \{1,4,7\}$. Optimal reparameterizations are indicated by $\hat{\varphi}_j^{i,i+1,i+2}$ being close to φ_j.

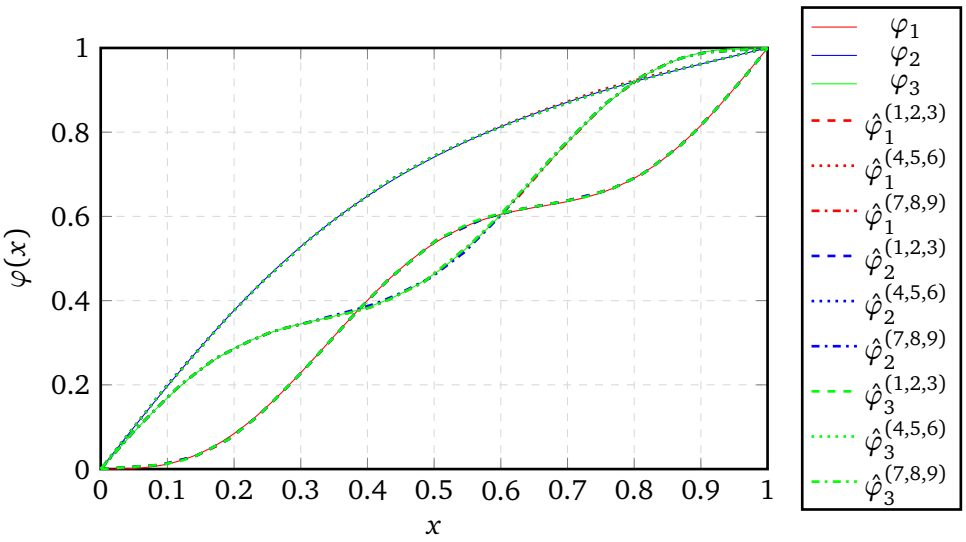


Figure 3.9: Comparison of reparameterized curves $\hat{\varphi}_j^{i,i+1,i+2}$ with the analytical parameterizations φ_j for curves in SE(3). The curves $c_{i,i+1,i+2}^{\text{eq}}$ have been fitted to $c_{i,i+1,i+2}^{j}$ for $j \in \{1,2,3\}$, and for all shapes $i \in \{1,4,7\}$. Optimal reparameterizations are indicated by $\hat{\varphi}_j^{i,i+1,i+2}$ being close to φ_j.

Classification of Curves

The primary objective here is to classify curves based on their geometric shapes using our reparameterization framework. This classification utilizes the curves generated in Section 3.3.3. Specifically, we use the curves c_i^j and c_i^{eq} for SO(3) and SE(3), as well as $c_{i,i+1,i+2}^j$ and $c_{i,i+1,i+2}^{\text{eq}}$ for SO(3)^3 and SE(3)^3. Our framework aims to find the shape space metric $d_{\mathcal{S}_*}$, as defined in Equation (2.18), between these curves to create a distance matrix. Instead of directly classifying the curves, we visualize the distances to observe clustering of similar shapes and separation of different shapes.

The color map of this matrix highlights the effectiveness of shape-based classification. We expect a block diagonal pattern in the matrix, with three 4x4 blocks along the diagonal indicating lower distances and thus, shape similarities. Conversely, the off-diagonal blocks should display higher, relatively uniform distances, suggesting distinct shapes. This pattern would confirm successful classification based on shape reparameterization, showing that geometrically similar curves cluster together, while different shapes are well-separated.

The classification outcomes for each dataset type SO(3), SE(3), SO(3)^3, and SE(3)^3 are depicted in Figures 3.10, 3.11, 3.12, and 3.13, respectively. These results highlight the efficacy of our methodology in distinguishing between curve shapes through reparameterization, demonstrating the robustness and utility of the proposed approach.

Notably, the classification is more precise for curves in SO(3)^3 and SE(3)^3. In these cases, the distances between curves of the same shape are significantly closer, while distances between curves of different shapes are more pronounced compared to those in SO(3) and SE(3). This suggests that our reparameterization framework is particularly effective in higher-dimensional settings, where it can better capture and distinguish subtle geometric differences.

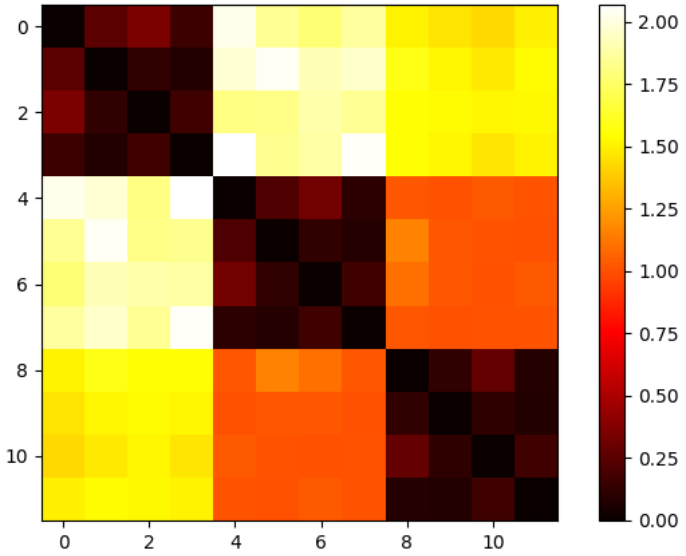


Figure 3.10: Heat map of the distance matrix for twelve synthetic SO(3) curves, denoted c_i^j, where $i \in \{1,2,3\}$ indicates the shape and $j \in \{1,2,3,\text{eq}\}$ represents the parameterization. The color intensity reflects the shape space distance, with lower distances indicating similarity in shape and higher distances signifying distinct shapes.

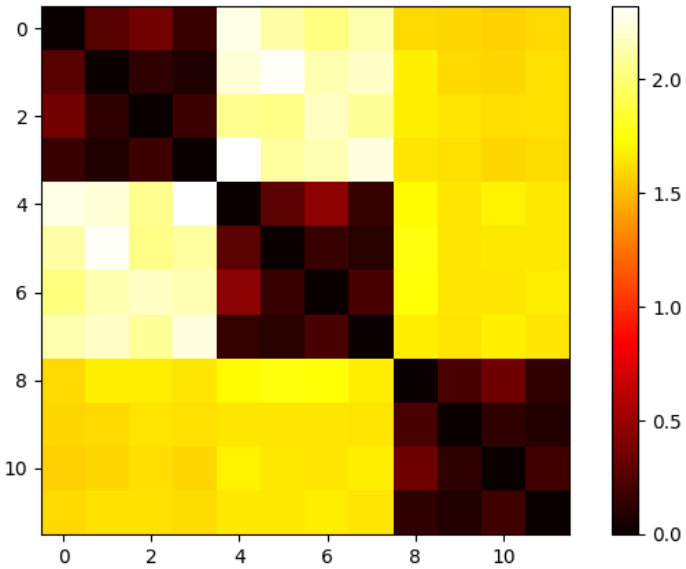


Figure 3.11: Heat map of the distance matrix for twelve synthetic SE(3) curves, denoted c_i^j, where $i \in \{1,2,3\}$ indicates the shape and $j \in \{1,2,3,\text{eq}\}$ represents the parameterization. The color intensity reflects the shape space distance, with lower distances indicating similarity in shape and higher distances signifying distinct shapes.

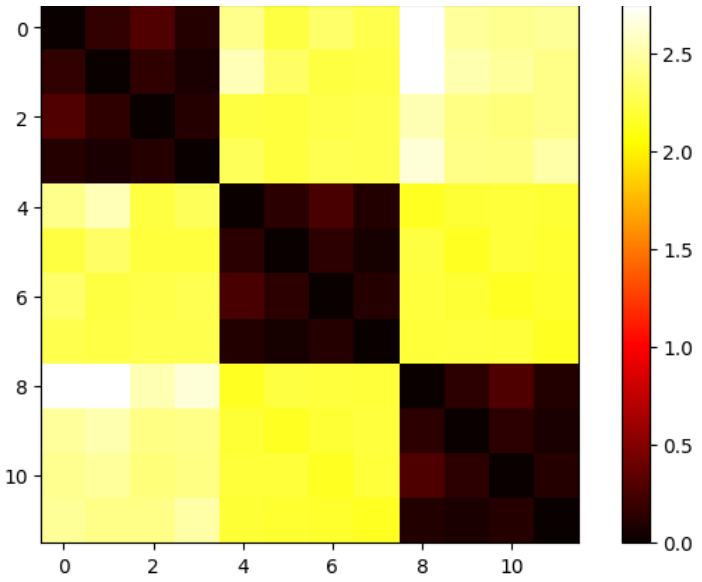


Figure 3.12: Heat map of the distance matrix for twelve synthetic $SO(3)^3$ curves, denoted $c^j_{(i,i+1,i+2)}$, where $i \in \{1,4,7\}$ indicates the shape and $j \in \{1,2,3,\text{eq}\}$ represents the parameterization. The color intensity reflects the shape space distance, with lower distances indicating similarity in shape and higher distances signifying distinct shapes.

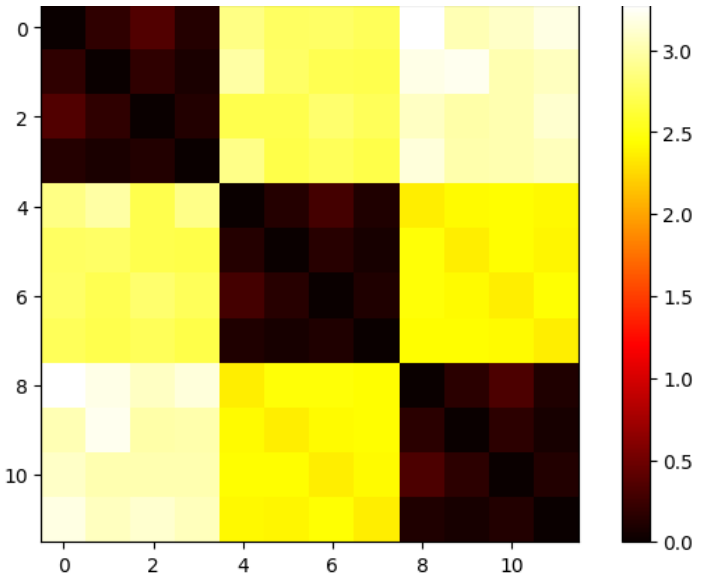


Figure 3.13: Heat map of the distance matrix for twelve synthetic $SE(3)^3$ curves, denoted $c^j_{(i,i+1,i+2)}$, where $i \in \{1,4,7\}$ indicates the shape and $j \in \{1,2,3,\text{eq}\}$ represents the parameterization. The color intensity reflects the shape space distance, with lower distances indicating similarity in shape and higher distances signifying distinct shapes.

Reduced Search Area

In practical applications, exhaustive domain searches can be computationally demanding, as outlined in Subsection 3.3.2. This becomes impractical for large datasets or higher-dimensional spaces. To address this, we evaluate the impact of varying search depths on performance while maintaining a constant sample count. Our goal is to balance computational efficiency and accuracy by optimizing the search depth.

Figures 3.14 and 3.15, shows that the error decreases with increasing search depth across curves in SO(3), SE(3), $SO(3)^3$, and $SE(3)^3$. Initially, the error reduction is rapid, but it levels off as depth increases, with minimal differences observed between depths 4 and 10. This suggests diminishing returns in error reduction beyond a certain depth, which appears to be around search depth 4 for this dataset.

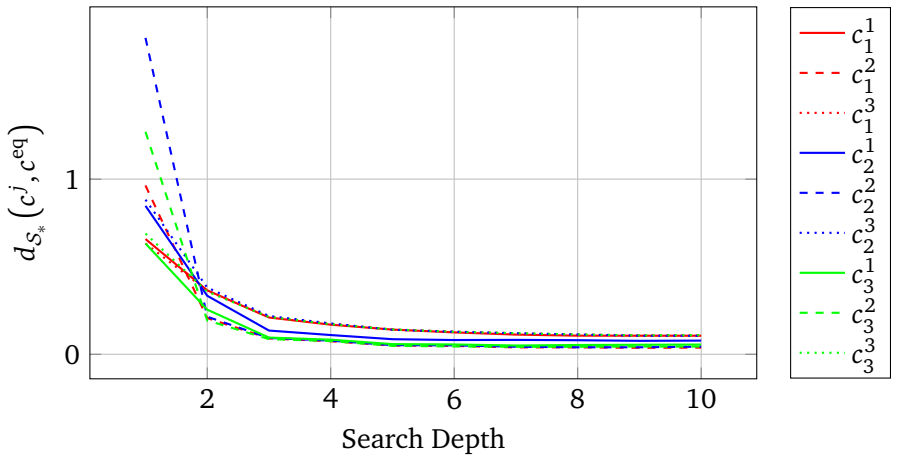
Figure 3.16 illustrates the computation time required for these calculations. As expected, computation time increases significantly with greater search depth, though the growth is not strictly quadratic due to various operational efficiencies and the moderate scale of n. These results highlight the need for an optimal search depth to balance accuracy and computational cost.

Based on these observations, we find that a search depth of 4 significantly reduces computation time while maintaining an acceptable error rate. This depth provides a practical balance between computational efficiency and accuracy, making it suitable for our reparameterization framework in both lower and higher-dimensional spaces. However, these findings are specific to the current dataset and may vary with different datasets.

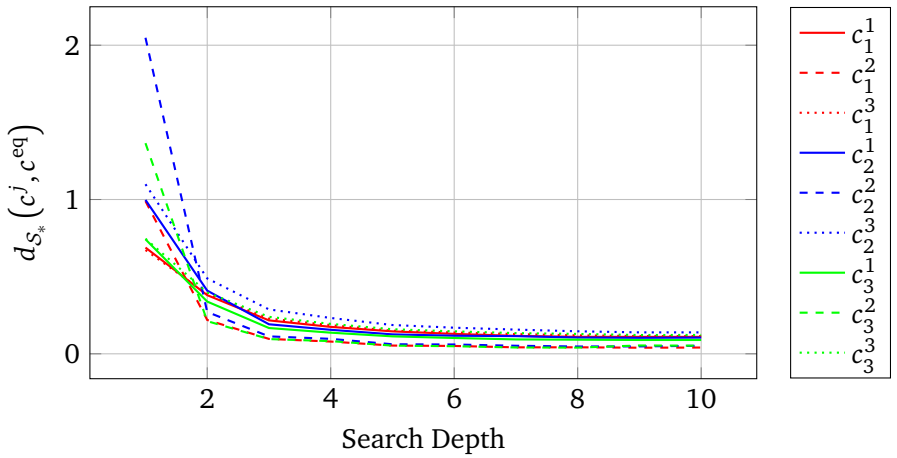
Perturbation Analysis

In this part, we examine the effects of perturbations on a curve by analyzing both the pseudometric $d_{\mathcal{P}_*}$ and the shape distance $d_{\mathcal{S}_*}$ between the equidistant parameterized curve c^{eq} and its perturbed counterpart c^{ϵ}. The perturbed curves c^{ϵ} are generated by solving the differential equation $g' = g\hat{u}(t)$, where $u(t) = \omega_i(t)$ or $u(t) = \xi_i(t)$, with t perturbed by a normal distribution. Previously In Section 3.3.4, we found that the order of the distance between the original and reparameterized curves was bounded from below by $1/2$. Here, we further analyze perturbations on curves in SO(3) and SE(3) using $d_{\mathcal{P}_*}$, and employ dynamic programming with depth 10 to examine the impact on $d_{\mathcal{S}_*}$. Figure 3.17 illustrates $d_{\mathcal{P}_*}$, and Figure 3.18 shows $d_{\mathcal{S}_*}$ for synthetic curves in SO(3) and SE(3) with varying perturbation magnitudes ϵ. Solid lines represent rotational components $\omega_1, \omega_2, \omega_3$, and dashed lines represent combined rotational and translational components ξ_1, ξ_2, ξ_3.

From Figure 3.17, the convergence rate of order 1 suggests our previous analysis may have been too pessimistic. In Figure 3.18, $d_{\mathcal{S}_*}$ is lower than $d_{\mathcal{P}_*}$ for large perturbations, but they converge as perturbation decreases, indicating $d_{\mathcal{S}_*}$ does not fully mitigate perturbations. To improve these results, a semidiscrete method as discussed in [48] may be employed.

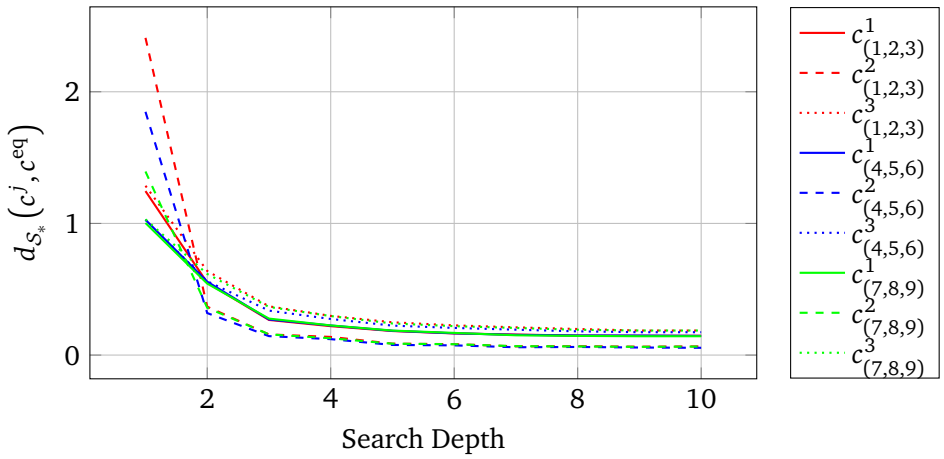


(a)

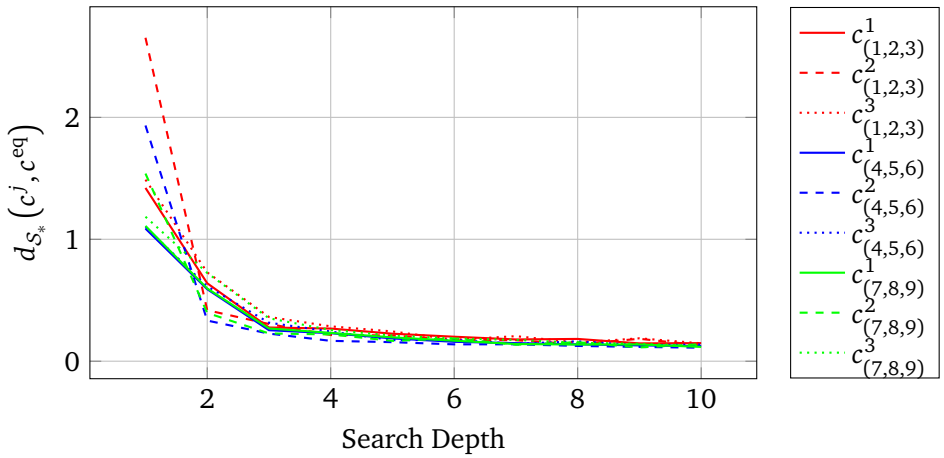


(b)

Figure 3.14: Plot showing the convergence of shape distance $d_{\mathcal{S}_*}$ (defined in Equation (2.18)) between curve c_i^{eq} and its reparameterization c_i^j for $j \in \{1,2,3\}$ and $i \in \{1,2,3\}$ within group G as search depth increases: (a) $G = \text{SO}(3)$, (b) $G = \text{SE}(3)$.



(a)



(b)

Figure 3.15: Plot showing the convergence of shape distance $d_{\mathcal{S}_*}$ (defined in Equation (2.18)) between curve $c^{\text{eq}}_{i,i+1,i+2}$ and its reparameterization $c^j_{i,i+1,i+2}$ for $j \in \{1,2,3\}$ and $i \in \{1,4,7\}$ within group G as search depth increases: (a) $G = \text{SO}(3)^3$, (b) $G = \text{SE}(3)^3$.

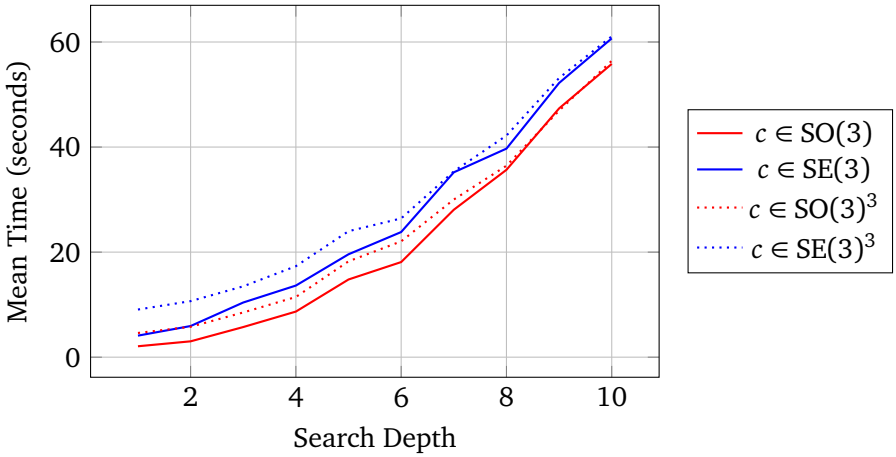


Figure 3.16: Mean computation time required for curve reparameterization across varying search depths (100 time steps). The mean values are calculated based on the time taken to compute errors for each curve, considering different reparameterizations and search depths. The associated errors are shown in the Figures 3.14 and 3.15.

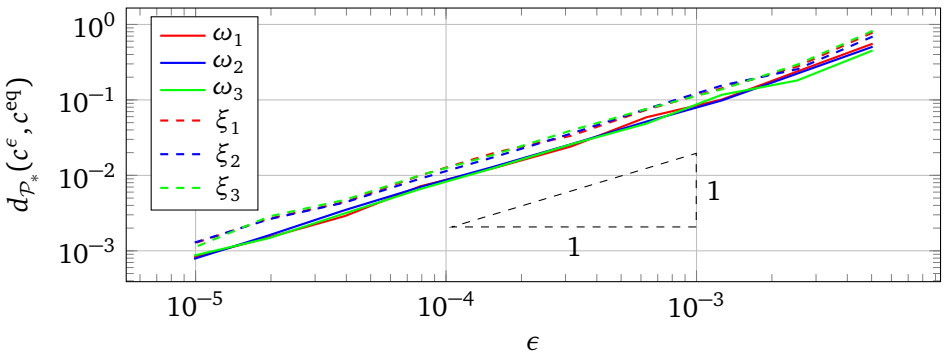


Figure 3.17: This figure illustrates the perturbation analysis of synthetic curves within the groups SO(3) and SE(3). Displayed is the pseudometric $d_{\mathcal{P}_*}$ between the baseline curve c^{eq} and its perturbed form c^ϵ, under perturbations of varying magnitudes ϵ. The dashed black triangle marks the convergence of order 1.

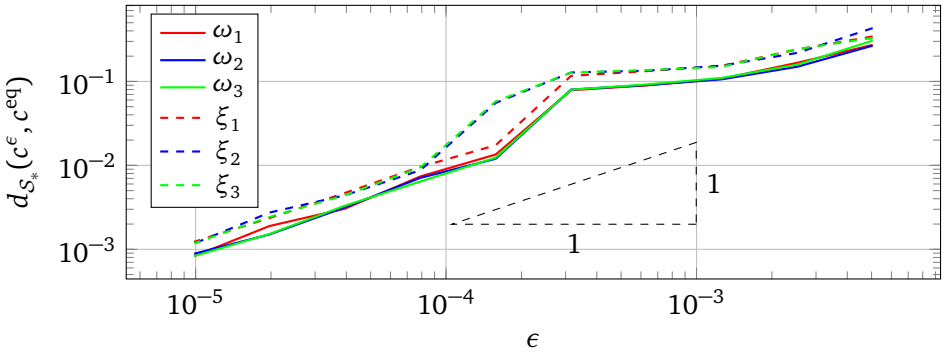


Figure 3.18: This figure presents a reparameterized perturbation analysis for synthetic curves in SO(3) and SE(3), focusing on the shape distance $d_{\mathcal{S}_*}$. It shows how $d_{\mathcal{S}_*}$ varies between the curve c^{eq} and its perturbed version c^ϵ under different perturbation magnitudes ϵ. The convergence rate of order 1 is indicated by a dashed black triangle. Reparameterization was executed using dynamic programming with a search depth of 10.

3.4 Optimal Curve Reparameterization through Geodesic Interpolation

This section presents an optimal reparameterization technique for curve alignment within Lie groups, particularly SO(3) and SE(3). Unlike the dynamic programming approach in Section 3.3, this method provides a continuous framework for curve comparison over the domain $[0,1]$ without a fixed set of nodes. As it does not use the SRVT framework (discussed in 3.1.3), this approach may result in very small geodesic distances for curves of different shapes. It is mainly applied to align curves with assumed identical shapes.

3.4.1 Overview of methodology

The goal is to reparameterize a discrete curve g in a Lie group G to minimize the shape distance to another curve h. This is achieved by identifying a diffeomorphism $\varphi \in \text{Diff}^+([0,1])$ that transforms g into $g_\varphi(t) = g(\varphi(t))$ for $t \in [0,1]$, aligning it with h at specific points. Geodesic interpolation facilitates detailed comparisons throughout $[0,1]$, not limited to discrete points.

The alignment's objective function is formulated as:

$$C(s_k^\varphi) = \sum_{k=0}^{N} \| \log(g(s_k^\varphi)^{-1} h(s_k)) \|_F^2, \tag{3.44}$$

where s_k are equidistant nodes along h and $s_k^\varphi = \varphi(s_k)$ are the transformed nodes of g.

For multi-component curves g_i and h_i, the function expands to:

$$C(s_k^{\varphi}) = \sum_{i=1}^{n} \sum_{k=0}^{N} \| \log(g_i(s_k^{\varphi})^{-1} h_i(s_k)) \|_F^2, \tag{3.45}$$

where g_i and h_i are the i-th components of the curves g and h.

Minimizing $C(s_k^{\varphi})$ can be done in several ways, but is here performed using Sequential Least Squares Quadratic Programming (SLSQP) via `scipy.optimize` [51], ensuring an increasing φ and mitigating local minima with multiple randomized initializations.

3.4.2 Numerical Analysis

This subsection evaluates the performance of our reparameterization framework on synthetic data. We first reparameterize synthetic curves of the same shape, and then explore its application for classification.

Reparameterization of Curves

For this analysis, we use synthetic data generated as described in 3.3.3. We reparameterize equidistant curves c_i^{eq} and $c_{i,i+1,i+2}^{\text{eq}}$ to match parameterized curves c_i^j and $c_{i,i+1,i+2}^j$, respectively, as in the dynamic programming approach described in 3.3.4. The quality of reparameterization is assessed by comparing the resulting reparameterizations $\hat{\varphi}_j^i$ with the original parameterizations φ_j.

The effectiveness of these reparameterizations is visually demonstrated in Figures 3.19, 3.20, 3.21, and 3.22. These figures show alignment improvements, where true parameterizations are represented by solid lines, and reparameterization results by dashed, dotted, and dash-dotted lines. Colors indicate shapes, and line styles represent mappings.

The results show that the reparameterization framework aligns the curves well with the true parameterizations. The curves are closely matched, demonstrating the method's effectiveness. While some discrepancies are observed, especially in Figure 3.19, this may be due to local minima or insufficient runtime due to the method's computational expense, and should be further investigated.

Classification of Curves

We use the same synthetic data to classify curves based on their shapes. The classification is performed by computing the shape space distance between the curves using the cost functions (3.44) and (3.45). We store the cost between each pair of curves in a distance matrix. The distance matrix is visualized through heatmaps, where color intensity indicates the shape space distance between the curves, as shown in Figures 3.23 and 3.24. We did not perform this classification on the curves in $\mathrm{SE}(3)^3$ and $\mathrm{SO}(3)^3$ due to the high computational expense and the method's unsuitability for comparing curves of different shapes.

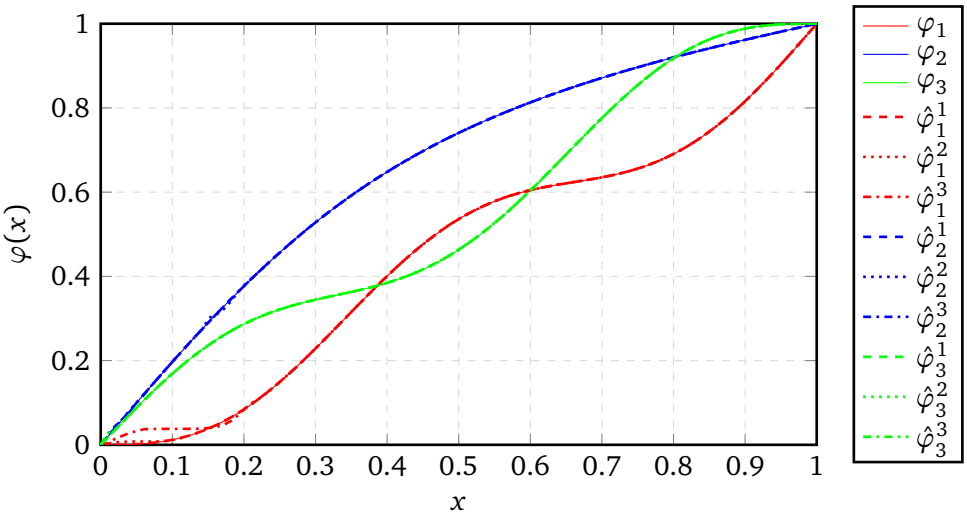


Figure 3.19: Comparison of reparameterization results for SO(3) data, showing $\hat{\varphi}_j^i$ obtained from our framework versus the analytical φ_j. Here, $j \in \{1,2,3\}$ denotes the mappings, and $i \in \{1,2,3\}$ indicates the shapes. Colors represent different mappings, while line styles distinguish the shapes, illustrating the accuracy of the fit.

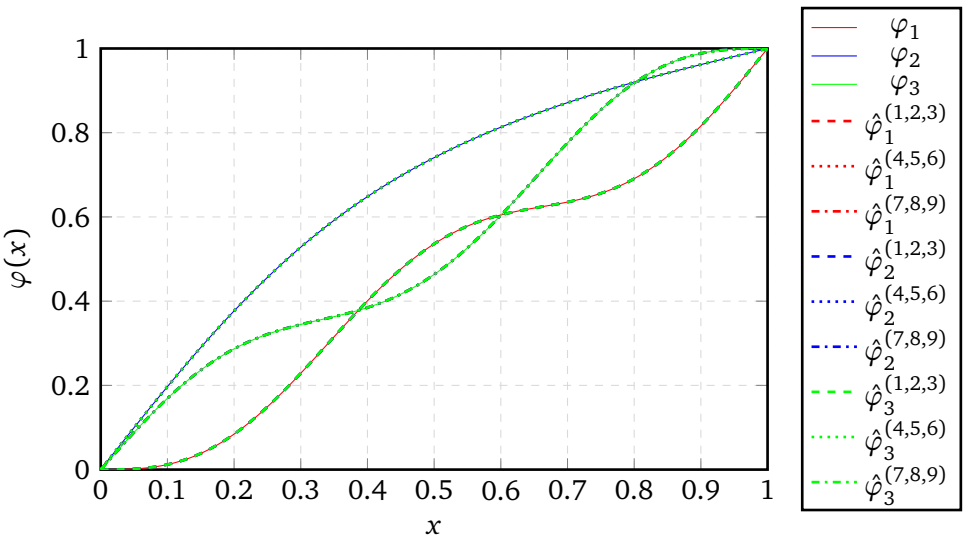


Figure 3.20: Comparison of reparameterization results for $\mathrm{SO}(3)^3$ data, showing $\hat{\varphi}_j^{i,i+1,i+2}$ obtained from our framework versus the analytical φ_j. Here, $j \in \{1,2,3\}$ denotes the mappings, and $i \in \{1,4,7\}$ indicates the shapes. Colors represent different mappings, while line styles distinguish the shapes, illustrating the accuracy of the fit.

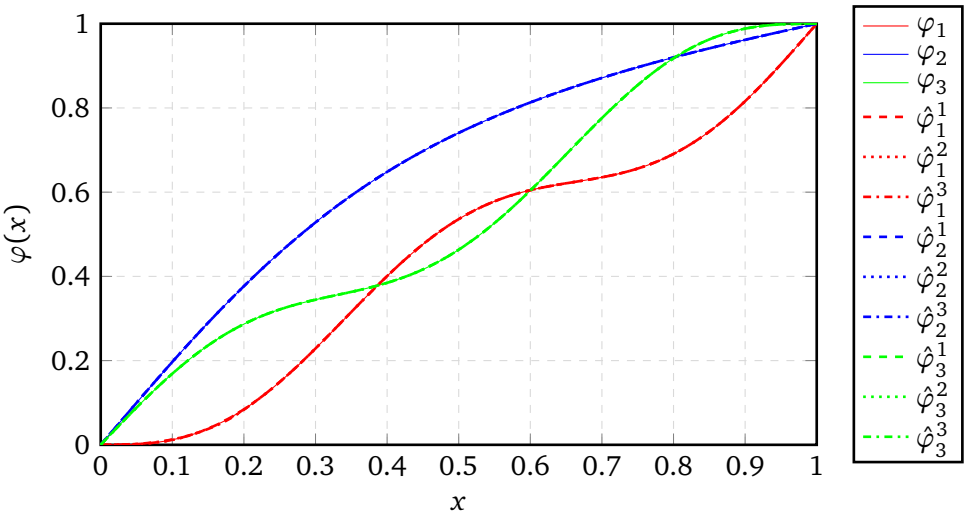


Figure 3.21: Comparison of reparameterization results for SE(3) data, showing $\hat{\varphi}_j^i$ obtained from our framework versus the analytical φ_j. Here, $j \in \{1,2,3\}$ denotes the mappings, and $i \in \{1,2,3\}$ indicates the shapes. Colors represent different mappings, while line styles distinguish the shapes, illustrating the accuracy of the fit.

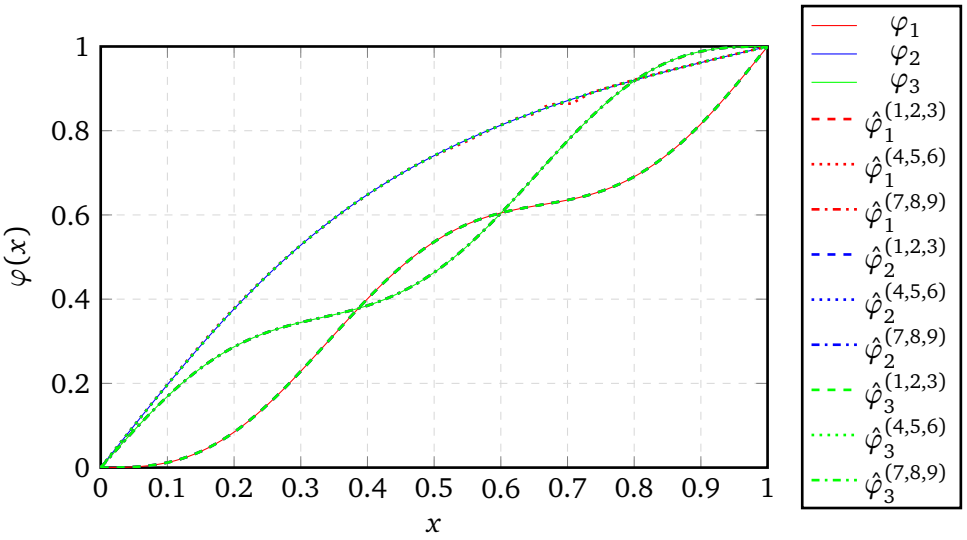


Figure 3.22: Comparison of reparameterization results for $\mathrm{SE}(3)^3$ data, showing $\hat{\varphi}_j^{i,i+1,i+2}$ obtained from our framework versus the analytical φ_j. Here, $j \in \{1,2,3\}$ denotes the mappings, and $i \in \{1,4,7\}$ indicates the shapes. Colors represent different mappings, while line styles distinguish the shapes, illustrating the accuracy of the fit.

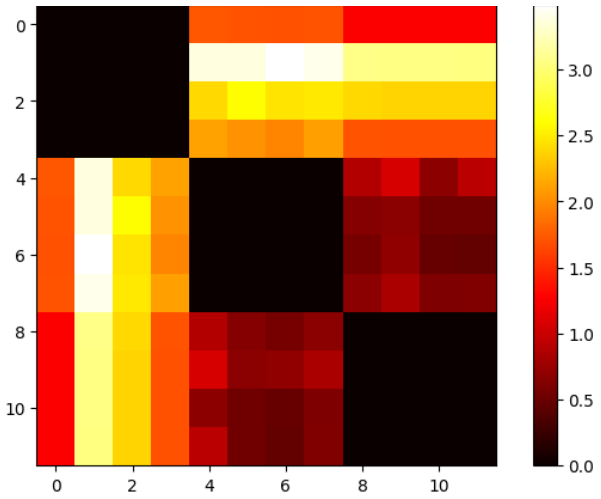


Figure 3.23: Classification of synthetic SO(3) curves across 100 timesteps, with geodesic interpolation to reparameterize, depicted as a distance matrix of twelve curves, denoted c_i^j, where $i = 1, 2, 3$ and $j = 1, 2, 3, 4$. The color intensity within the matrix indicates the shape space distance.

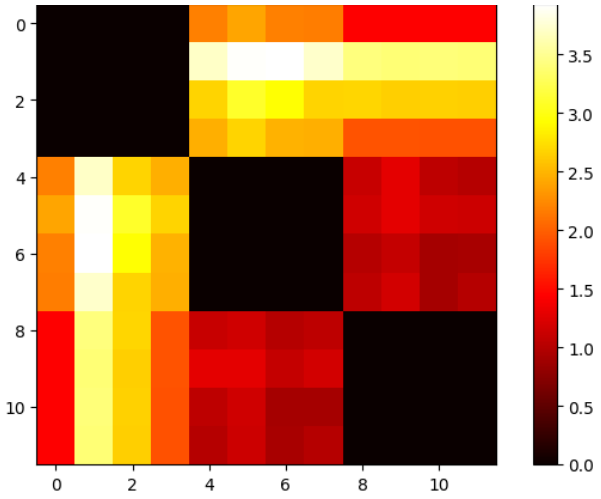


Figure 3.24: Classification of synthetic SE(3) curves across 100 timesteps, with geodesic interpolation to reparameterize, illustrated as a distance matrix of twelve curves, denoted c_i^j, where $i = 1, 2, 3$ and $j = 1, 2, 3, 4$. The color intensity within the matrix indicates the shape space distance.

As seen in the heatmaps, the classification is successful, with almost zero intra-class distances and high inter-class distances. This indicates that the curves are well separated in the shape space, and the classification is accurate. While these results are promising, we refrain from using this method for classifying motion capture data due to uncertainties about its robustness.

Chapter 4

Signature Method

This chapter delves into the intricacies of the signature method, a robust tool for analyzing paths and curves. We explore its fundamental concepts, including tensor algebra and the definition of the signature along with its properties. Additionally, we investigate the logarithmic signature for path representation. The chapter culminates with numerical examples illustrating the method's application in shape analysis.

4.1 Fundamentals of the Signature Method

This section delves into the foundational aspects of the signature method, including tensor algebra, the definition of the signature, and its properties.

4.1.1 Tensor Algebra and Its Dual Space

Within the context of a finite-dimensional vector space V with dimension d, the tensor algebra $T(V)$ encompasses all tensor powers of V:

$$T(V) := \bigoplus_{n=0}^{\infty} V^{\otimes n} = \mathbb{R} \oplus V \oplus (V \otimes V) \oplus (V \otimes V \otimes V) \oplus \cdots, \tag{4.1}$$

where $V^{\otimes n}$ denotes the n-th tensor product of V and $V^{\otimes 0} = \mathbb{R}$ [52].

The dual space of the tensor algebra, denoted $T((V)) := T(V)^*$ or $T(V^*)$, represents the domain of formal power series in d non-commuting variables, represented by the set $\{e_1, \ldots, e_d\}$ [53].

In a d-dimensional vector space, tensor series are depicted as infinite vectors indexed by words from the alphabet $\{1, \ldots, d\}$. Each word $w = i_1 \cdots i_n$, with i_j drawn from $\{1, \ldots, d\}$, corresponds to a fundamental element $e_w = e_{i_1} \otimes \cdots \otimes e_{i_n}$ [14]. This conceptual framework serves as the cornerstone for defining iterated integrals of a path in $\mathbb{R}^d$.

4.1.2 Definition of the Signature

For a smooth path $x : [s,t] \to \mathbb{R}^d$ over $[s,t] \subset [0,1]$, the n-fold iterated integral for a word $w = i_1 \cdots i_n$ is defined as:

$$\langle S(x)_{s,t}, e_w \rangle = \int_{s<u_n<\cdots<u_1<t} dx_{u_1}^{i_1} \cdots dx_{u_n}^{i_n}. \tag{4.2}$$

This provides geometric insights into the path x [54].

The signature $S(x)_{s,t}$ of x over $[s,t]$ is represented by the tensor series:

$$S(x)_{s,t} = 1 + \sum_{|w|\geq 1} \langle S(x)_{s,t}, e_w \rangle e_w \in T((\mathbb{R}^d)^*), \tag{4.3}$$

aggregating its iterated integrals [55].

To enhance computational efficiency, we truncate the signature at level n, capturing terms up to word length n. The truncated signature $S(x)_{s,t}^{(n)}$ is given by:

$$S(x)_{s,t}^{(n)} = 1 + \sum_{1\leq|w|\leq n} \langle S(x)_{s,t}, e_w \rangle e_w, \tag{4.4}$$

where $|w|$ represents the word length [56].

The number of terms in $S(x)_{s,t}^{(n)}$ is calculated as:

$$M = \frac{d(d^n - 1)}{d-1}, \tag{4.5}$$

where d is the path's dimension, crucial for balancing computational efficiency and information capacity [57].

4.1.3 Signatures on Piecewise Linear Paths

For linear paths in $\mathbb{R}^d$, such as $x(t) = a \cdot t + b$, the n-fold iterated integral for a word $w = i_1 \cdots i_n$ simplifies to:

$$\langle S(x)_{s,t}, e_w \rangle = \frac{(t-s)^n}{n!} \prod_{k=1}^{n} a_{i_k}. \tag{4.6}$$

Following prior work [18, 56], the signature for a linear path is derived as:

$$S(x)_{s,t} = 1 + \sum_{|w|\geq 1} \frac{(t-s)^n}{n!} \prod_{k=1}^{n} a_{ik} e_w = \exp_{\otimes}((t-s)a). \tag{4.7}$$

Applying Chen's Identity [12], for a path x and $0 \leq s \leq u \leq t \leq 1$, yields:

$$S(x)_{s,u} \otimes S(x)_{u,t} = S(x)_{s,t}. \tag{4.8}$$

For a piecewise linear path, combining equation (4.7) with (4.8) results in the signature:

$$S(x)_{s,t} = \bigotimes_{k=1}^{m} \exp_{\otimes}(\Delta t_k a_k) = \exp_{\otimes}(\Delta t_1 a_1) \otimes \cdots \otimes \exp_{\otimes}(\Delta t_m a_m), \tag{4.9}$$

where $\Delta t_k = t_k - t_{k-1}$ represents the lengths of the time intervals, and $a_1, \ldots, a_m$ are the slopes of the segments.

4.1.4 Signature for a Smooth Curve in a Lie Group

Following [18, 58], for a smooth curve $c : [0,1] \to G$ in a Lie group G, the n-fold iterated integral of the signature is defined recursively, with the base $\langle S(c)_{s,t}, 1\rangle := 1$ and the recursive step:

$$\langle S(c)_{s,t}, e_{i_1}, \ldots, e_{i_n}\rangle := \int_s^t \langle S(c)_{s,u}, e_{i_1}, \ldots, e_{i_{n-1}}\rangle \omega_{c(u)}^{i_n}(\dot{c}(u))du, \tag{4.10}$$

where $\omega_c^j(v)$ denotes the j-th component of $\omega_c(v) \in \mathfrak{g}$.

For a smooth curve $\kappa : [s_k, s_{k+1}] \to G$, where $G = SO(3)$ or $SE(3)$, representing the geodesic interpolation between configurations c_k and c_{k+1}, the term $\tilde{\eta}_k$ is defined as:

$$\tilde{\eta}_k = \frac{\log(c_{k+1}c_k^{-1})}{s_{k+1} - s_k}, \tag{4.11}$$

and the right logarithmic derivative $\omega_\kappa(\dot{\kappa})(t) = \hat{\eta}_k$ is formulated accordingly. Utilizing $\eta_k \in \mathfrak{g}$, derived from the inverse hat map, the n-fold integrals for the signature of κ are given by:

$$\langle S(\kappa)_{s_k,s_{k+1}}, e_{i_1,\ldots,i_n}\rangle = \frac{(s_{k+1} - s_k)^n}{n!} \prod_{j=1}^{n} \eta_k^{i_j}, \tag{4.12}$$

for each appropriate index set. Consequently, the signature over the interval $[s_k, s_{k+1})$ is:

$$S(\kappa)_{s_k,s_{k+1}} = \exp_{\otimes}\left((s_{k+1} - s_k)\tilde{\eta}_k\right). \tag{4.13}$$

This methodology extends to paths in Lie groups such as $SO(3)^d$ and $SE(3)^d$, by adapting the index set and employing Chen's rule for concatenation.

4.1.5 Properties of Signature Space for Paths

In the signature space of a path $x : [0,1] \to \mathbb{R}^d$, the identity is $S(x)_{s,s} = 1$, and its inverse is $S_{s,t}^{-1}(x) = S_{s,t}(\overleftarrow{x})$, where $\overleftarrow{x}(t) = x(1-t)$ [14]. This inverse relationship is captured by Chen's identity:

$$S(x) \otimes S(\overleftarrow{x}) = 1. \tag{4.14}$$

Chen's rule indicates that the signature acts as a homomorphism from the path space under concatenation to the dual tensor algebra [59]. For paths $x, y : [0,1] \to \mathbb{R}^d$, their concatenation $x * y$ satisfies:

$$S(x * y)_{0,1} = S(x)_{0,1} \otimes S(y)_{0,1} \tag{4.15}$$

Consequently, the identity for concatenation is:

$$S(x * \overleftarrow{x})_{0,1} = 1. \tag{4.16}$$

For the concatenation of two paths $c_1, c_2 : [0,1] \to G$, denoted $c_1 * c_2$, where G represents a Lie group, the homomorphism property is preserved:

$$S(c_1 * c_2)_{s,t} = S(c_1)_{s,t} \otimes S(c_2)_{s,t}. \tag{4.17}$$

Parameterization invariance in concatenation allows for flexibility in choosing a midpoint within the interval $(0,1)$.

4.1.6 Uniqueness of the Path Signature

The path signature provides a unique characterization of a path, subject to certain constraints: translation, parameterization, and irreducibility [59].

Translation invariance naturally arises from the signature's derivation. It stems from the manner in which iterated integrals are computed, where the differential $dx_t = \frac{dx}{dt}dt$ effectively eliminates any dependence on the initial position of the path.

Under parameterization that maintains the path's orientation, the signature remains invariant. For any orientation-preserving diffeomorphism φ defined over the interval $[s,t]$, the path's signature adheres to $S(x \circ \varphi)_{s,t} = S(x)_{s,t}$. Consequently, the signature transcends mere parameterization nuances.

Irreducibility denotes that a path cannot be expressed in a particular concatenated form involving path reversal. This notion is pivotal, as it underscores that the signature of the concatenated path $x * y * \overleftarrow{y} * z$ mirrors that of the simpler path $x * z$.

4.2 The Logarithmic Signature

In the realm of signature spaces, the emergence of a group structure naturally prompts inquiries into the existence of an associated Lie algebra and a corresponding logarithmic mapping. Chen first elucidated this aspect, confirming the presence of a Lie algebra structure within signature spaces and demonstrating that the logarithmic map yields a closed subspace within the tensor algebra [53]. Expanding upon Chen's concept, Lyons and Sidorova described this space as a

'formal Lie algebra', providing deeper insights into the structural dynamics within the group-like environment of signature spaces [55]. Building upon this foundation, we now extend our exploration from the conventional signature to the logarithmic signature. In this section, we will examine its properties and make a case for its utility in practical applications.

4.2.1 The Logarithmic Signature of a Path

The logarithmic signature constitutes a powerful transformation of the path signature, offering profound insights into the geometric and analytic properties of paths. This section introduces and explores the intricacies of this transformation within the framework of formal power series.

Beginning with the foundational principles outlined in [14], we establish the logarithmic signature as a formal power series. Represented by x, the series takes the form:

$$x = \sum_{k=0}^{\infty} \sum_{i_1,\ldots,i_k \in \{1,\ldots,d\}} \lambda_{i_1,\ldots,i_k} e_{i_1} \cdots e_{i_k}, \tag{4.18}$$

where $\lambda_0 > 0$ ensures its convergence. The logarithm of x is then defined as:

$$\log x = \log(\lambda_0) + \sum_{n=1}^{\infty} \frac{(-1)^n}{n} \left(\frac{x}{\lambda_0} - 1 \right)^{\otimes n}, \tag{4.19}$$

with $\otimes n$ denoting the n-th tensor power using the tensor product $\otimes$.

Exploring further, we illustrate the practical implications of this transformation through a concrete scenario. Consider a series x defined as:

$$x = 1 + \sum_{k=1}^{\infty} \frac{\lambda^k}{k!} e_1^{\otimes k}. \tag{4.20}$$

A straightforward computation reveals that $\log x = \lambda e_1$, showcasing how the logarithmic signature captures essential information from the original series while preserving finite coefficients.

The logarithmic signature of a path $X : [s,t] \to \mathbb{R}^d$ is formally represented as $\mathcal{LS}(X)_{s,t} := \log S(X)_{s,t}$, as per Definition 6 in [14]. This representation facilitates a deeper understanding of the path's geometric structure and analytical properties.

Moreover, Theorem 4 in the same work establishes that for any given path, the log signature can be expressed in terms of real coefficients $\lambda_{i_1,\ldots,i_k}$, as:

$$\mathcal{LS}(X)_{s,t} = \sum_{k=1}^{\infty} \sum_{i_1,\ldots,i_k \in \{1,\ldots,d\}} \lambda_{i_1,\ldots,i_k} [e_{i_1}, [e_{i_2}, \ldots, [e_{i_{k-1}}, e_{i_k}] \ldots]]. \tag{4.21}$$

It's essential to note that these coefficients may not be unique due to the inherent lack of linear independence among certain polynomial constructs, as exemplified by the Lie bracket.

Furthermore, the injective nature of the logarithmic signature, as established in [60], underscores its significance as a unique representation of a path. This uniqueness aligns with the properties of the path's signature, emphasizing its role as a fundamental analytical tool.

Additionally, [54] accentuates the logarithmic signature space's flat geometry, simplifying distance computations by allowing the use of Euclidean norms. This geometric property enhances its practical utility in various analytical and computational contexts.

4.2.2 Redundancy in Standard Signature Analysis

The issue of redundancy in standard signatures is notable. For n-fold integrals of the word w, $\langle S(x)_{s,t}, e_w \rangle$, with repetitions of letter i, the signature only reflects the increment of the path x

$$\begin{aligned} \langle S(x)_{s,t}, e_i \rangle &= x_t - x_s, \\ \langle S(x)_{s,t}, e_{ii} \rangle &= \frac{(x_t^i - x_s^i)^2}{2!}, \\ \langle S(x)_{s,t}, e_{iii} \rangle &= \frac{(x_t^i - x_s^i)^3}{3!}, \\ &\vdots \end{aligned} \tag{4.22}$$

which are just different powers of the increment.

In the work of Ree [61], the concept of the shuffle product is introduced to elucidate the inherent redundancy in the multiplication of signature terms. This concept is instrumental in demonstrating that the product of two such terms can be decomposed into a sum of more intricate signature terms. These terms are derived from the (p,q)-shuffles of words i and j, denoted as $\mathrm{Sh}(p,q)$. A (p,q)-shuffle consists of permutations σ that preserve the order of the letters in both i and j. Consequently, the product of the signatures of i and j can be expressed as an aggregate of all (p,q)-shuffles of k, represented mathematically as:

$$\langle S(x)_{s,t}, e_{i_1,\ldots,i_p} \rangle \cdot \langle S(x)_{s,t}, e_{j_1,\ldots,j_q} \rangle = \sum_{\sigma \in \mathrm{Sh}(p,q)} \langle S(x)_{s,t}, e_{k_{\sigma(1)},\ldots,k_{\sigma(p+q)}} \rangle. \tag{4.23}$$

An exemplary case is the term for the word ij, given by:

$$\langle S(x)_{s,t}, e_{ij} \rangle = \langle S(x)_{s,t}, e_{ji} \rangle - \langle S(x)_{s,t}, e_i \rangle \cdot \langle S(x)_{s,t}, e_j \rangle, \tag{4.24}$$

which does not offer additional geometric insight beyond the area encompassed by ji and the increments i and j.

4.2.3 Efficiency of Logarithmic Signatures

Logarithmic signatures, in contrast to standard signatures, eliminate redundancy and succinctly encapsulate the geometric essence of a path's signature. For a two-dimensional path, a signature $S(x)$ truncated at level $L = 2$ is represented as:

$$S(x) = (1, S^{(1)}(x), S^{(2)}(x), S^{(1,1)}(x), S^{(2,1)}(x), S^{(1,2)}(x), S^{(2,2)}(x)). \tag{4.25}$$

The logarithmic signature $\mathcal{LS}(x)$, truncated at the same level, is expressed as:

$$\mathcal{LS}(x) = \left(S^{(1)}(x), S^{(2)}(x), \tilde{S}^{(1,2)}(x)\right), \tag{4.26}$$

where

$$\tilde{S}^{(1,2)}(x) = \frac{1}{2}\left(S^{(1,2)}(x) - S^{(2,1)}(x)\right). \tag{4.27}$$

The logarithmic signature is the most compact description of the geometric properties of a path's signature, which can be deduced from the Rashevski-Chow Theorem [55].

4.3 Numerical Analysis

This section presents a numerical analysis of the signature-based framework for curve classification. We define the use of the logarithmic signature for classifying curves, introduce a distance metric to compare the signatures of two curves, and employ this metric for classification purposes.

4.3.1 Distance Metric for Logarithmic Signatures

The distance metric for comparing the logarithmic signatures of two curves c_1 and c_2 is defined as follows:

$$d_{\mathrm{sig}_*}(c_1, c_2) = \left\| \frac{\mathcal{LS}(c_1^*)}{\|\mathcal{LS}(c_1^*)\|_2} - \frac{\mathcal{LS}(c_2^*)}{\|\mathcal{LS}(c_2^*)\|_2} \right\|_2, \tag{4.28}$$

where $\mathcal{LS}(c_i^*)$ denotes the logarithmic signature of the curve c_i, adjusted to start at the identity by transforming all elements using the inverse of the first element. The signatures are normalized by their L^2 norm to ensure a standardized comparison. It should be noted that this choice is based on empirical observations and not theoretical reasons, which is a topic for future research.

We utilize the iisignature library [57], which implements logarithmic signature computations using the Lyndon basis. Although it is possible to use the Hall basis, we chose the Lyndon basis due to its superior results.

N	Level	u	$\sigma(u)$
1	1	1	$[1]$
2	1	2	$[2]$
3	1	3	$[3]$
4	2	12	$[1,2]$
5	2	13	$[1,3]$
6	2	23	$[2,3]$
7	3	112	$[1,[1,2]]$
8	3	113	$[1,[1,3]]$
9	3	122	$[[1,2],2]$
10	3	123	$[1,[2,3]]$
11	3	132	$[[1,3],2]$
12	3	133	$[[1,3],3]$
13	3	223	$[2,[2,3]]$
14	3	233	$[[2,3],3]$
		$\vdots$	

Table 4.1: The Lyndon words consisting of three letters, up to truncation level 3, and their images under the permutations σ.

4.3.2 Impact of Truncation on Curve Distances

This section explores the effects of truncation on curve distances and the transition from continuous to discrete settings when using logarithmic signatures. This analysis provides insights into the practical application of logarithmic signatures for data analysis.

Utilizing the iisignature package, we examine logarithmic signatures framed in the Lyndon basis, as detailed in [56]. For a triadic letter set, the Lyndon basis is tabulated in Table 4.1.

The dimensionality of truncated logarithmic signatures, dependent on truncation level n and dimension d, is computed using Witt's formula, essential in free Lie algebra dimensional analysis [62]. Witt's formula is given by:

$$W(d,n) = \sum_{l=1}^{n} \frac{1}{l} \sum_{x|l} \mu\left(\frac{l}{x}\right) d^x, \tag{4.29}$$

where μ represents the Möbius function [63]:

$$\mu(n) = \begin{cases} +1 & \text{if } n \text{ is square-free with an even number of prime factors,} \\ -1 & \text{if } n \text{ is square-free with an odd number of prime factors,} \\ 0 & \text{if } n \text{ has a squared prime factor.} \end{cases} \tag{4.30}$$

Table 4.2 presents the dimensions of truncated log signatures at various levels for a word length of three.

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	$\cdots$
$\mathbf{d = 3}$	3	6	14	32	80	196	508	$\cdots$

Table 4.2: Dimensionality of the logarithmic signature at different truncation levels

We observe that the distance between logarithmic signatures increases with higher truncation levels. For signatures at levels i and $i+1$, this can be expressed as:

$$\|\mathcal{LS}(x_{i+1}) - \mathcal{LS}(y_{i+1})\| \geq \|\mathcal{LS}(x_i) - \mathcal{LS}(y_i)\| + \|\text{Terms at } i+1\|, \tag{4.31}$$

where $\mathcal{LS}(x_i)$ and $\mathcal{LS}(y_i)$ are logarithmic signatures at level i, and $\mathcal{LS}(x_{i+1})$ and $\mathcal{LS}(y_{i+1})$ are at level $i+1$. This means that when we include more terms in the logarithmic signature (by increasing the truncation level), the distance between the signatures of two different paths either stays the same or becomes larger.

Future work could explore the existence of a scaling factor to enable the comparison of distances between different truncation levels.

4.3.3 Classification of Curves

This subsection aims to classify curves based on their geometric characteristics utilizing a signature-based framework. The approach utilizes synthetic data, as discussed in Subsection 3.3.3, and employs methodologies outlined in prior analyses for curve generation. These curves are produced according to the procedures specified in Subsection 3.3.4, for transformations in SO(3), SE(3), $SO(3)^3$, and $SE(3)^3$ spaces, respectively.

We employ the shape space metric d_{sig_*} (Equation (4.28)) to compute distances between each pair of curves, leading to the construction of a distance matrix. Currently, we have not addressed the scalar difference between the Frobenius norm and the 2-norm of the vector format for curves in SE(3) (Section 2.5), which may cause the method to focus excessively on translation. Visualizing this matrix with a color map facilitates the interpretation of these distances, where color intensity is inversely proportional to the geometric similarity between curves.

A key anticipated outcome is the formation of a block diagonal pattern within the distance matrix. This pattern is expected to manifest as three distinct 4x4 blocks along the diagonal, each block representing lower distances indicative of shape similarities. Conversely, off-diagonal blocks should display higher, relatively uniform distances, highlighting the presence of distinct geometric shapes. Such a configuration would underscore the efficacy of the signature-based classification method, confirming that curves sharing geometric properties cluster together in the shape space, while distinctly different shapes are effectively segregated.

In Figure 4.1, it is evident that SE(3) curves are more readily classified compared to SO(3) curves, as shown by the more distinct clustering observed within the same geometric types. Increasing truncation levels in SO(3) generally improves

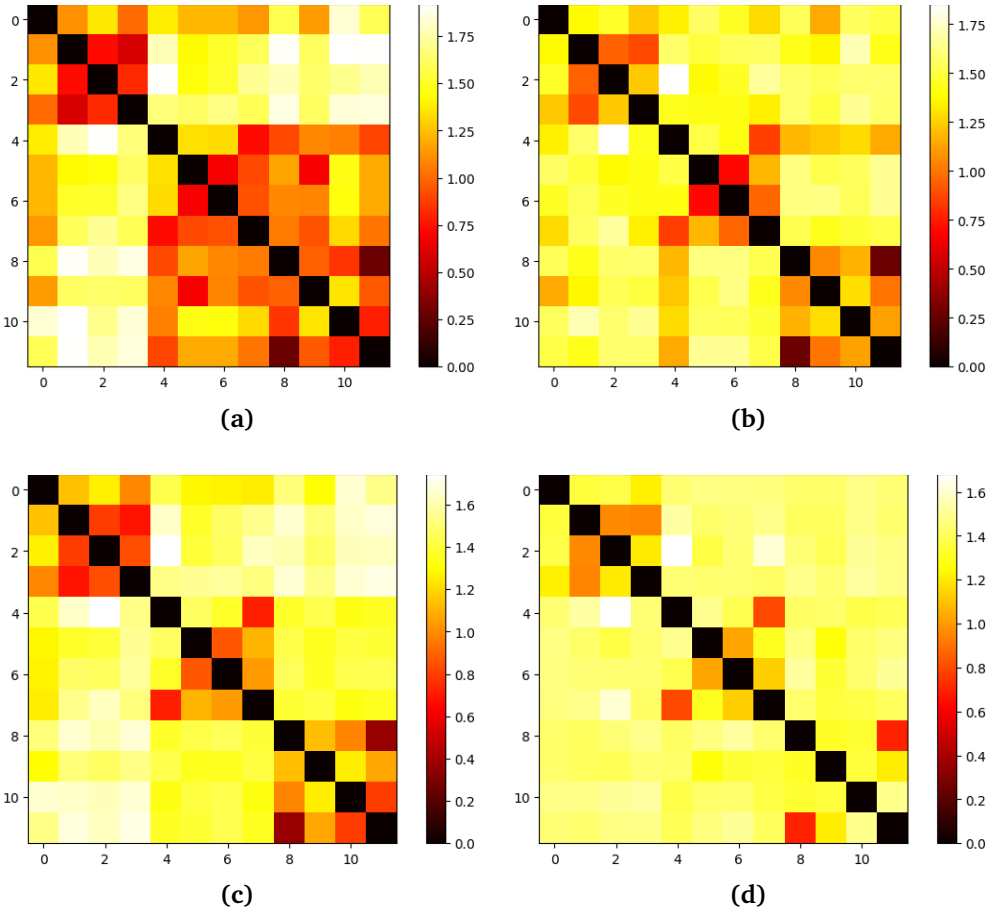


Figure 4.1: Distance matrices using the signature method for SO(3) and SE(3) at different truncation levels, visualized through the L^2 norm between normalized signatures. Panels: (a) SO(3), truncation level 3; (b) SO(3), truncation level 5; (c) SE(3), truncation level 3; (d) SE(3), truncation level 5. These matrices illustrate the variation in signature-based classification accuracy across truncation levels.

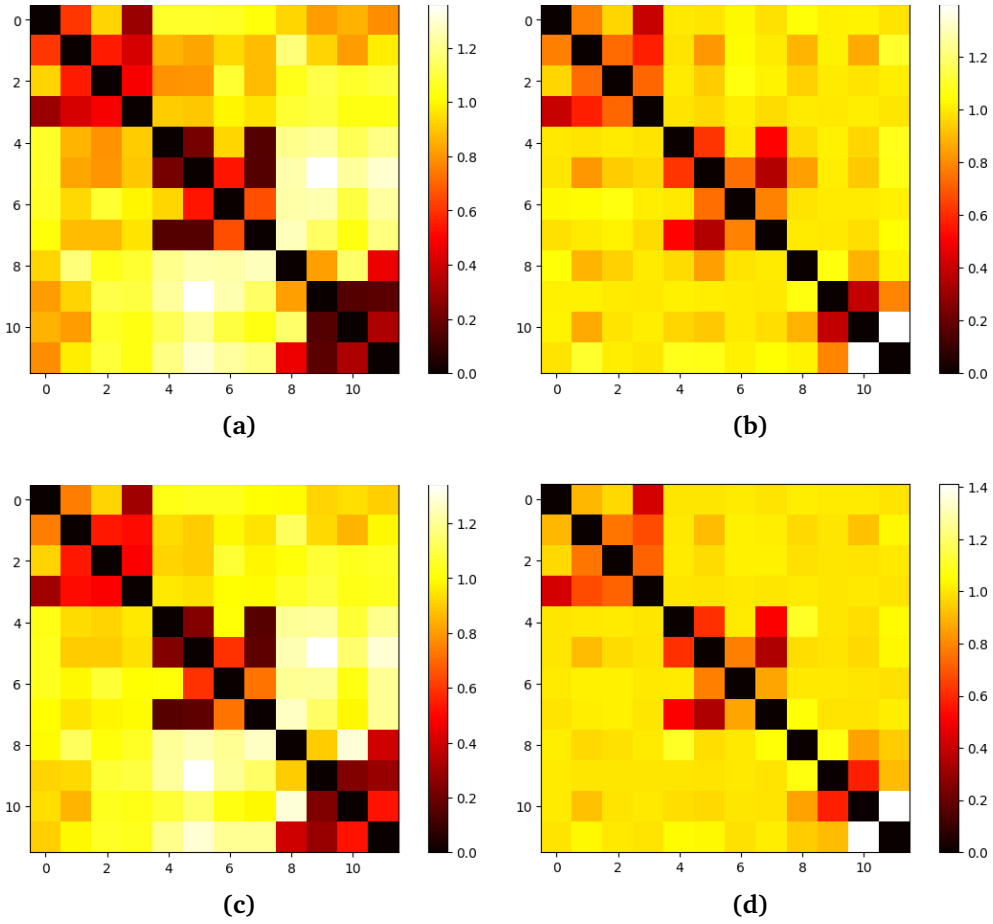


Figure 4.2: Distance matrices using the signature method for $SO(3)^3$ and $SE(3)^3$ at different truncation levels, visualized through the L^2 norm between normalized signatures. Panels: (a) $SO(3)^3$, truncation level 3; (b) $SO(3)^3$, truncation level 5; (c) $SE(3)^3$, truncation level 3; (d) $SE(3)^3$, truncation level 5. These matrices illustrate the variation in signature-based classification accuracy across truncation levels.

classification accuracy by incorporating more detailed information, resulting in a tighter grouping of curves with similar geometric characteristics. However, an interesting observation is made when the truncation level in SE(3) is increased to 5: some curves from the same shape space unexpectedly exhibit large distances between them.

Figure 4.2 highlights the classification results for the spaces $SO(3)^n$ and $SE(3)^n$, which are notably promising. Distances within the same shape space remain low, indicating effective classification. Interestingly, classification performs better in $SE(3)^n$ than in $SO(3)^n$, demonstrating a superior ability to resolve geometric differences in these extended spaces. Similar to SE(3), when the truncation level is increased to 5 in both $SO(3)^n$ and $SE(3)^n$, some curves from the same shape space begin to show significant distances between them.

These results highlight that the logarithmic signature is capable of distinguishing between curves of different shapes. By combining this with post-processing techniques, such as clustering, it may be possible to determine the shape of a curve based on its signature. However, the results also indicate that the choice of truncation level is crucial, as increasing the level may lead to unexpected results. Future work could explore the optimal truncation level for curve classification using the signature-based framework.

4.3.4 Perturbation Analysis

In this subsection, we investigate how the logarithmic signature-based distance metric (Equation (4.28)) behaves under perturbations. We utilize the curves c_i^{eq} and $c_{i,i+1,i+2}^{\text{eq}}$ created in Subsection 3.3.3. To create a perturbed version of these curves, we solve the same differential equation (3.31), but with timesteps perturbed by adding small, normally distributed noise. This results in the perturbed curves c_i^{ϵ} and $c_{i,i+1,i+2}^{\epsilon}$, with $i \in \{1,2,3\}$ and $i \in \{1,4,7\}$, respectively. We examine the effect of this perturbation on the signature-based distance metric by progressively reducing the standard deviation of the noise, performing this reduction 10 times. The results are shown in Figure 4.3.

We observe that as the perturbation size decreases, the signature-based distance metric also decreases. This indicates that the metric is robust to small perturbations, maintaining consistent performance even with minor noise in the parameterization of the curves.

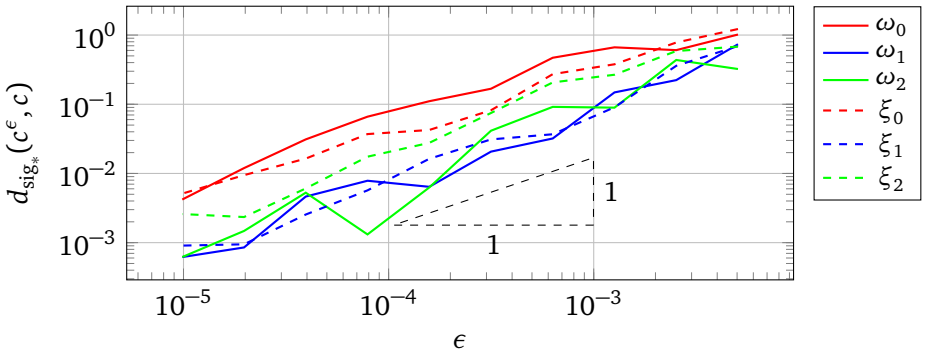


Figure 4.3: The signature based distance metric under perturbations in SO(3) and SE(3). The distance between the perturbed and non-perturbed curves is computed using the signature based distance metric. The curves are perturbed by adding a small normal distributed noise to the parameterization, where we halve the standard deviation, and the distance is computed 10 times.

Chapter 5

Analyzing Motion Capture Data

In this chapter, we evaluate our shape analysis frameworks using real motion capture data from the CMU Motion Capture Library. This real-world test aims to validate the robustness and accuracy of our methods and explore the potential for combining them with post-processing techniques.

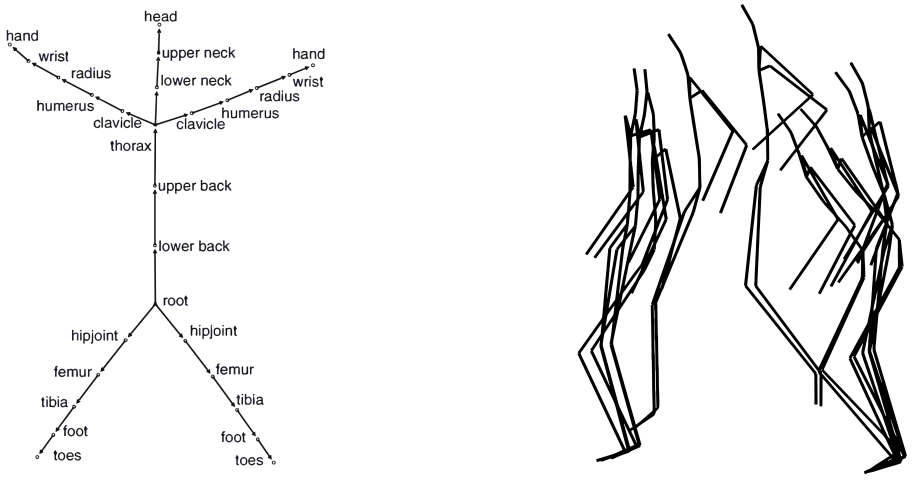
5.1 Methodology

We utilize files from the CMU Motion Capture Library [64], specifically XX.asf files and XX_YY.amc files, where XX represents the subject (describing the skeleton or body of the performer) and YY represents the motion (the action performed). In Figure 5.1a, we illustrate the skeleton structure, while Figure 5.1b shows a visual representation of a jump motion, combining the motion data with the skeleton structure.

Our study focuses on 44 motions, as listed in Table B.1: 9 'forward jumps', 9 'runs/jogs', 10 'walks', 9 'boxing' sequences, and 7 'stair climbing' sequences. These motions were chosen due to the availability of sufficient AMC files for each category. Each motion consists of 131 frames, matching the typical duration of a 'forward jump' motion, and all are recorded at a frame rate of 120 Hz. We ensured that the defining part of each motion was clearly visible, such as observing a punch in the boxing sequences.

The dataset includes ASF (Actor Skeleton File) and AMC (Actor Motion Capture Data) files. ASF files describe the skeleton structure and establish a base pose, while AMC files capture dynamic motion over time. We convert this data into joint-specific matrices, focusing on 23 joints to reduce noise from smaller joints such as thumbs, toes, and fingers. Each matrix contains rotational data for each timestep, resulting in a format of $(131, 23, 3, 3)$ representing (timestep, joint, rotation matrix).

Our hypothesis is that motions perceived as similar by humans (e.g., walking) are also mathematically similar, despite individual variations. To test this hypothesis, we will categorize the motions into five distinct groups and evaluate if our shape analysis framework accurately captures these similarities.



(a) **(b)**

Figure 5.1: (a) Illustration of a human skeleton model for computer animation, showing the joint structure with labeled points. Adapted from [25]. (b) Illustration of subject 16 motion 05, depicting one jump motion, 9 timeframes. Data for both illustrations from the CMU Graphics Lab Motion Capture Database [64].

5.2 Results

In this section, we utilize the frameworks from Chapter 3 and Chapter 4 to analyze motion capture data. The objective is to evaluate the performance of these methods on real-world data and compare their results.

5.2.1 Reparameterization with SRVT

To find the distance between all motions in the dataset, we use the framework from Chapter 3, which employs reparameterization of the SRVT (Square Root Velocity Transform) of the motions using Algorithm 1 for optimal reparameterization. We then calculate the L^2 distance between the SRVT of the motion and its reparameterized counterpart. This distance is stored in a distance matrix, where the diagonal is zero, as the distance between a motion and itself is zero.

The results are presented as a heatmap to show the unedited results, where the axes represent different motions, and the color indicates the distance between the two motions. These results are shown for various search depths (1, 4, and 10) to demonstrate that higher search depths yield improved outcomes. Ideally, the distance between similar motions should be smaller than the distance between dissimilar motions. Additionally, all motions from the same category should have similar distances to motions from other categories.

The motions are sorted before plotting the heatmap: motions 1-9 are forward jumps, 10-18 are runs/jogs, 19-28 are walks, 29-37 are boxing, and 38-44 are stair climbing. Thus, the heatmap should show blocks of the same color, where the

blocks along the diagonal should have close to zero distance.

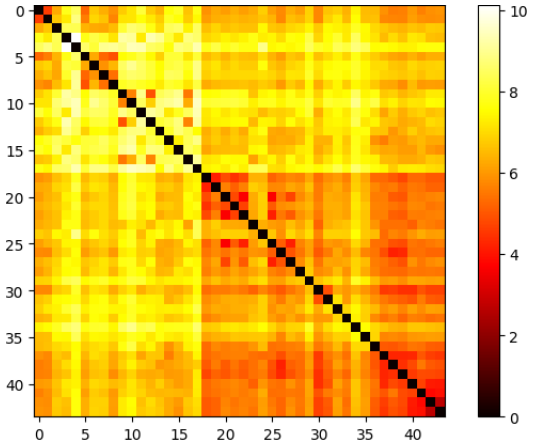
The results for different search depths using the SRVT method are shown in Figure 5.2. In 5.2a, little structure is visible in the heatmap. It appears that the first two motions, "forward jump" and "run/jog," are dissimilar to each other, while the last three motions show some similarity. In 5.2b, we observe four blocks along the diagonal, indicating that "forward jump," "run/jog," "walk," and "climbing stairs" are somewhat distinct. However, there are darker spots indicating some resemblance between walking and climbing stairs. In 5.2c, a similar structure to that of search depth 4 is evident. It is not surprising that "walk" and "climbing stairs" are somewhat similar, as they involve similar motions, with one being on a flat surface and the other on stairs. It appears that boxing is the motion that is hardest for our method to distinguish from the others, which may be intrinsic to the motion itself or may be a limitation of the method.

5.2.2 Logarithmic Signature

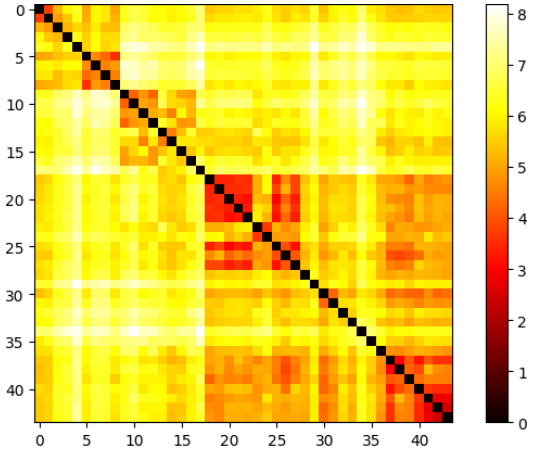
Another way to determine the shape space distance between two motions is to use the logarithmic signature method to find d_{sig_*} between all the curves, as defined in Equation (4.28). We compute the logarithmic signature of each motion, normalize the signatures, and calculate the L^2 norm of the difference between the normalized signatures. This distance is stored in a distance matrix, where the diagonal is zero since the distance between a motion and itself is zero.

The results for three different truncation levels (1, 2, and 3) are shown in Figure 5.3. In 5.3a there is significant noise, resembling Gaussian noise, indicating that it does not effectively distinguish between the motions. In 5.3b we see a block around "forward jump" and another block encompassing both "run/jog" and "walk" with a separate block forming at "climbing stairs". There is a significant distance between these and other motions. In 5.3c "forward jump" remains distinct, "run/jog" becomes distinct from "walk" and there are hints of "boxing" and "climbing stairs" being distinct.

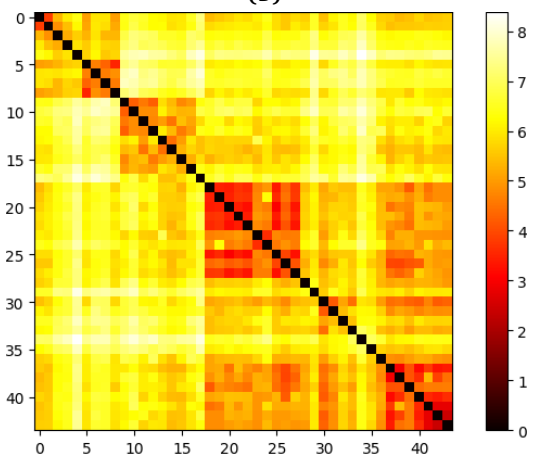
These results show that higher truncation levels yield more detailed distinctions, enabling the method to better differentiate between various motions due to the increased nuanced information. Additionally, the logarithmic signature method highlights different aspects of the data compared to the reparameterization method, which is expected given the fundamental differences between the two approaches.



(a)



(b)



(c)

Figure 5.2: Heatmaps of the distance matrix from motion capture data, where the distance matrix was created by reparameterization using dynamic programming and search depths of: (a) 1, (b) 4, and (c) 10.

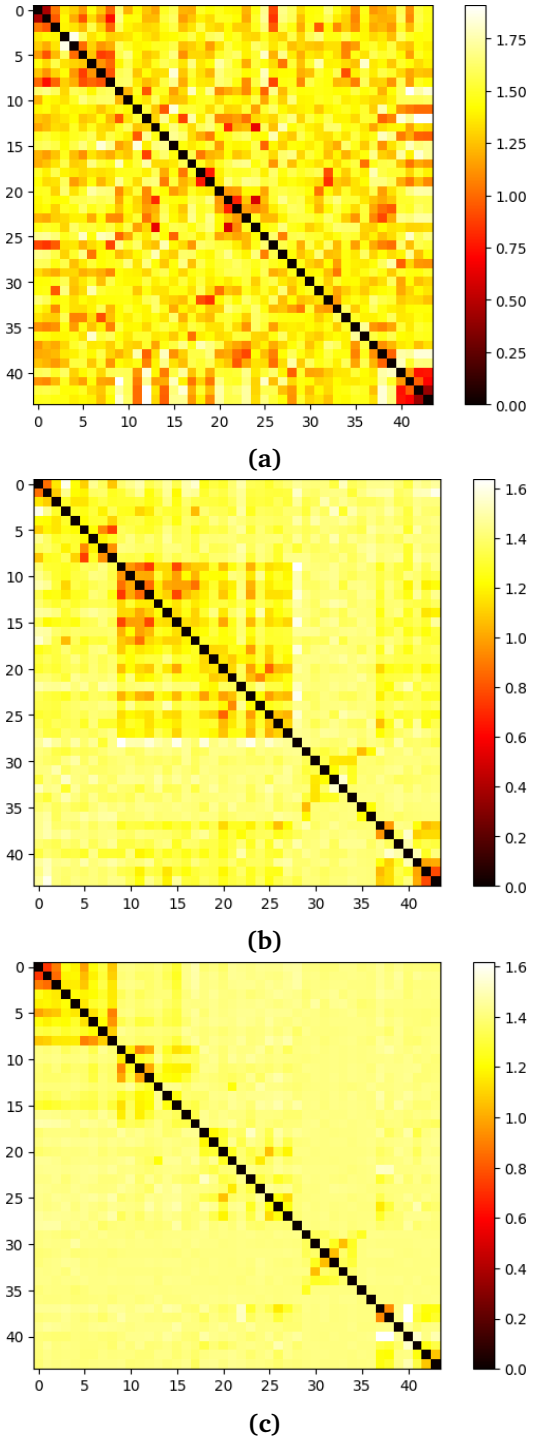


Figure 5.3: Heatmaps of the distance matrix from motion capture data, where the distance matrix was created by the Logarithmic Signature and the L^2 norm of these normalized signatures with truncation levels of: (a) 1, (b) 2, and (c) 3.

5.3 Post-Processing Techniques

To enhance our analysis results and quantify the performance of our methods, we will explore various post-processing techniques. These techniques aim to improve the results and provide a deeper understanding of the data. We will employ methods such as Principal Component Analysis (PCA), classical Multidimensional Scaling (cMDS), and the k-medoids clustering algorithm. Additionally, we will utilize the silhouette score as an evaluation metric to assess the quality of the clustering.

5.3.1 Principal Component Cosine Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique that extracts the most significant information from a dataset while minimizing noise. It achieves this by transforming the data into principal components using the eigenvalues and eigenvectors of the covariance matrix. The components with the highest variance are selected to efficiently represent the dataset [65, 66].

In practice, PCA converts an $n \times n$ distance matrix into an $n \times k$ matrix ($k < n$). After this transformation, pairwise distances in the reduced space are computed using the cosine distance. The cosine distance between two row vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ is defined by:

$$d_{\cos}(\mathbf{v_1}, \mathbf{v_2}) = 1 - \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{\|\mathbf{v_1}\|\|\mathbf{v_2}\|}, \tag{5.1}$$

This method calculates the cosine distances between all pairs (v_i, v_j) for $i, j \in \{1, 2, \ldots, n\}$, reconstructing a symmetric distance matrix. By using cosine distance, we emphasize directional similarity, which effectively handles high-dimensional data. This provides a computationally efficient approach to reconstruct the distance matrix, allowing us to discern and analyze the underlying structure in the data, isolating the most influential characteristics that define data relationships.

As shown in both subfigures, the cosine distance matrix after PCA captures the motions as distinct clusters. This indicates that the PCA method effectively reduces the dimensionality of the data, emphasizing the most significant features that define the motion capture data. This also reflects the categorization of the motions, demonstrating PCA's ability to highlight the underlying structure in the data.

5.3.2 K-Medoids Clustering

Given that we are working with a distance matrix, we aim to categorize data using a rigorous method rather than relying solely on visual inspection. K-Medoids clustering, as described in the literature [67, 68], enables this by partitioning datasets into clusters based on pairwise distances. Given a distance matrix $D = [d_{ij}]_{i,j=1}^{n}$, k-medoids are selected either randomly or heuristically, here chosen heuristically. Each point is assigned to the nearest medoid, forming clusters C_j.

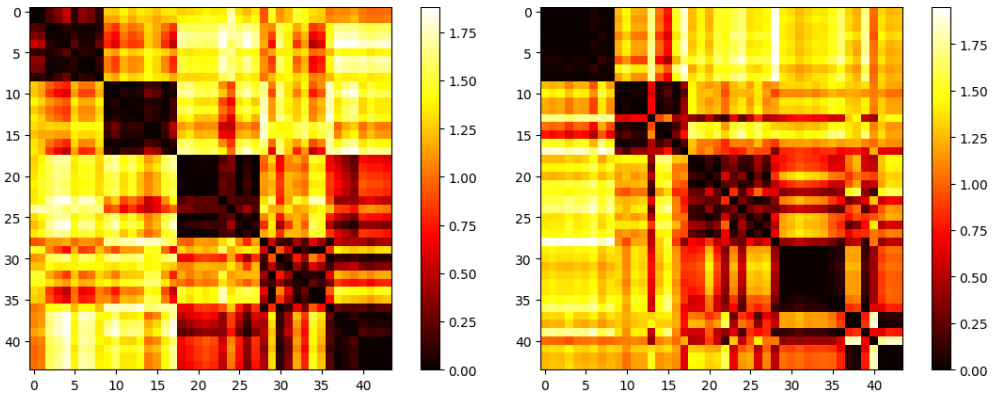


Figure 5.4: (a) Reparameterization through dynamic programming at a search depth of 10, (b) Logarithmic signature at a truncation level of 3. This figure displays heat maps of the cosine distance matrix following PCA, using data from reparameterization and logarithmic signature as seen in Figure 5.2 (search depth of 10) and Figure 5.3 (truncation level of 3).

Medoids are updated by selecting the point within each cluster that minimizes the total distance to all other points in the cluster, specifically

$$m_j = \arg\min_{x \in C_j} \sum_{x_i \in C_j} d(x, x_i). \tag{5.2}$$

This process iterates until cluster assignments stabilize or changes fall below a threshold. Unlike K-Means, which uses centroids, K-Medoids relies on medoids because means cannot be computed from a distance matrix. The algorithm operates solely on pairwise distances, effectively generalizing K-Means for use with distance matrices.

K-Medoids identifies clusters where points within the same cluster are more similar to each other than to points in different clusters. It retains the advantages of K-Means while being suitable for datasets where only pairwise distances are available, providing a robust and interpretable clustering method for rigorous mathematical analysis.

In Table 5.1, we present the results of K-Medoids clustering on the motion capture data. We applied this method to the distances created by reparameterization by dynamic programming search depth 10 and logarithmic signature with truncation level 3. This was done twice: once on the original distances and once on the reduced distances, where we used PCA combined with cosine similarity to reduce the dimensionality of the data (see Subsection 5.3.1). The values have been mapped to facilitate comparison, as the clusters are not necessarily ordered in the same way, as the number assigned to each cluster is arbitrary.

Table 5.1: K-Medoids clustering on motion capture data

Motion	Reparam	LogSig	Reparam (Red)	LogSig (Red)
Forward Jump	0	0	0	0
Forward Jump	0	0	0	0
Forward Jump	0	0	0	0
Forward Jump	0	0	0	0
Forward Jump	0	0	0	0
Forward Jump	0	0	0	0
Forward Jump	0	0	0	0
Forward Jump	0	0	0	0
Forward Jump	0	0	0	0
Run/Jog	1	1	1	1
Run/Jog	1	1	1	1
Run/Jog	1	1	1	1
Run/Jog	1	1	1	1
Run/Jog	4	2	1	2
Run/Jog	4	1	1	1
Run/Jog	1	1	1	1
Run/Jog	1	1	1	1
Run/Jog	4	1	1	1
Walk	2	2	2	2
Walk	2	2	2	2
Walk	2	2	2	2
Walk	2	2	2	2
Walk	2	2	2	2
Walk	4	4	2	2
Walk	2	2	2	2
Walk	2	2	2	2
Walk	2	4	2	2
Walk	2	2	2	2
Boxing	4	4	3	4
Boxing	3	3	3	3
Boxing	3	3	4	3
Boxing	3	3	3	3
Boxing	3	3	3	3
Boxing	3	3	3	3
Boxing	4	3	3	3
Boxing	4	2	3	3
Boxing	4	0	4	3
Climb Stairs	4	4	4	4
Climb Stairs	4	4	4	4
Climb Stairs	4	3	4	3

Motion	Reparam	LogSig	Reparam (Red)	LogSig (Red)
Climb Stairs	4	0	4	3
Climb Stairs	4	4	4	4
Climb Stairs	4	4	4	4
Climb Stairs	4	4	4	4

5.3.3 Classical Multidimensional Scaling (cMDS)

Classical multidimensional scaling (cMDS) is a technique used to visualize the distances between entities in a lower-dimensional space, typically two or three dimensions. Unlike Principal Component Analysis (PCA), which aims to maximize data variation, cMDS focuses on preserving the original distances between points as accurately as possible. This approach is detailed in Chapter 6 of Wang (2012) [69]. Your section on Classical Multidimensional Scaling (cMDS) is clear and well-structured. Below is a refined version to enhance rigor, flow, and conciseness.

Given a set of n entities represented by a distance matrix $D = [d_{ij}]_{i,j=1}^{n}$, cMDS seeks a configuration $\mathcal{Y} = \{y_1, \ldots, y_n\}$ in $\mathbb{R}^d$ such that the Euclidean distances $d_{\mathcal{Y}}(y_i, y_j)$ between points y_i and y_j in $\mathcal{Y}$ approximate the original distances d_{ij}:

$$d_{\mathcal{Y}}(y_i, y_j) \approx d_{ij}, \quad \forall i, j. \tag{5.3}$$

First, the distance matrix D is centered using the centering matrix $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$, where I is the identity matrix and $\mathbf{1}$ is a vector of ones. The centered Gram matrix G^C is calculated as:

$$G^C = -\frac{1}{2} H D^2 H, \tag{5.4}$$

where D^2 is the element-wise square of the distance matrix D. Next, eigendecomposition is performed on G^C to obtain its eigenvalues λ_i and corresponding eigenvectors u_i:

$$G^C = U \Sigma U^T, \tag{5.5}$$

where $U = [u_1, u_2, \ldots, u_r]$ and $\Sigma = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r)$. The eigenvalues and eigenvectors are sorted in descending order of the eigenvalues. To construct the configuration matrix Y, the top d eigenvalues and their corresponding eigenvectors are selected. Let $U_d = [u_1, u_2, \ldots, u_d]$ and $\Sigma_d = \text{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_d})$. The configuration matrix Y is then given by:

$$Y = U_d \Sigma_d. \tag{5.6}$$

For visualization, the first two columns of Y are typically plotted to provide a two-dimensional representation of the data.

By applying cMDS to the distance matrices generated by the reparameterization search at depth 10 and logarithmic signature truncation at level 3, the motion capture data can be visualized in two dimensions. This visualization is performed for both the original distances and the reduced distances obtained by PCA combined with cosine similarity. The cluster assignments obtained by K-Medoids clustering are included to better understand the data structure. The original reparameterization with clustering is shown in Figure 5.5, the reduced reparameterization with clustering in Figure 5.6, the original logarithmic signature with clustering in Figure 5.7, and the reduced logarithmic signature with clustering in Figure 5.8.

The figures demonstrate that the clustering utilizing the reparameterization method is quite distinct, especially when colored by the true cluster assignments. The reduced data show even clearer cluster formations in the 2D space, with the reduced clustering numbers more aligned with the true cluster assignments.

In the logarithmic signature method, the data are not as distinct as in the reparameterization method, for both the original and reduced data. However, the reduced data still show clearer cluster formations in 2D space.

In the reduced reparameterization data, "forward jump", "run/jog" and "walk" are distinct, while "boxing" and "climbing stairs" are somewhat mixed but still relatively distinct. For the logarithmic signature method, only "forward jump" is clearly separate, while "run/jog", "walk" and "climbing stairs" are somewhat distinct, with "boxing" mixed with the other clusters. This suggests that the "boxing" movement may resemble other movements, making it less distinct.

Overall, these results indicate that distinguishing real motion capture data using these methods is feasible. The application of post-processing techniques such as PCA with cosine distance, cMDS, and clustering enhances the clarity of the results compared to the heatmap figures 5.2, 5.3.

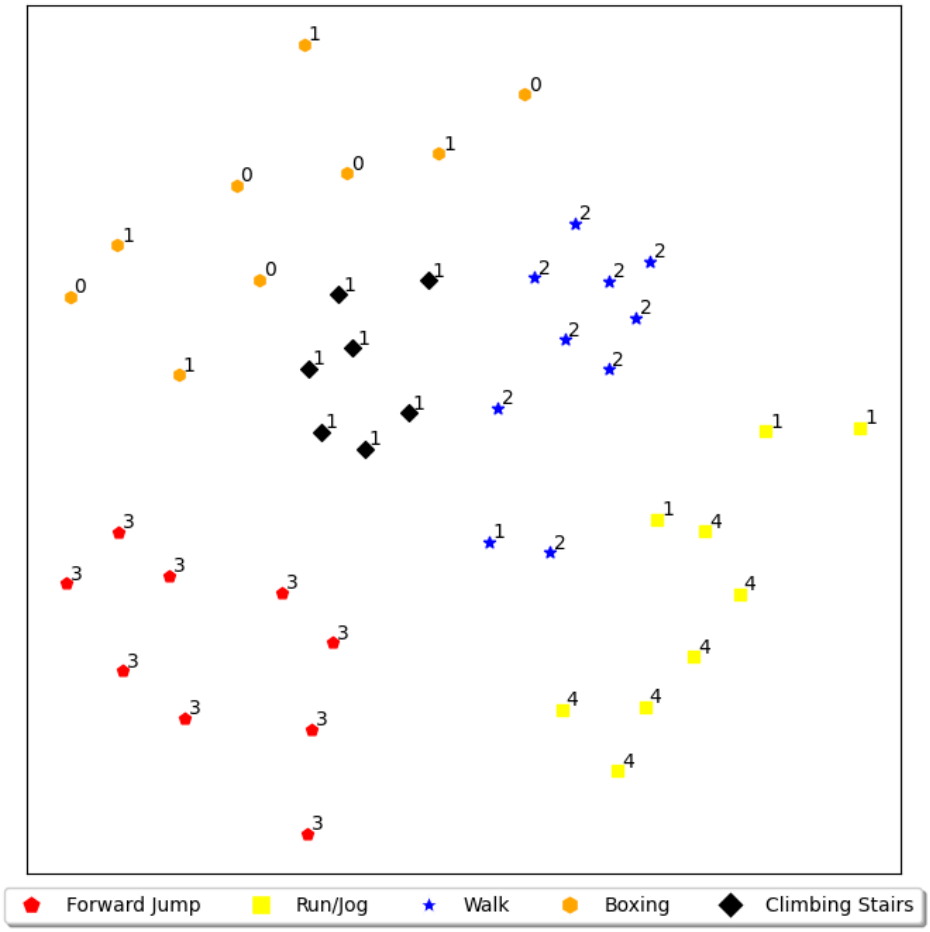


Figure 5.5: 2D visualization using cMDS of the reparameterization of motion capture data with dynamic programming using search depth 10. Colors and shapes represent the true cluster assignments, while the numbers indicate cluster assignments generated by K-Medoids clustering, with five clusters.

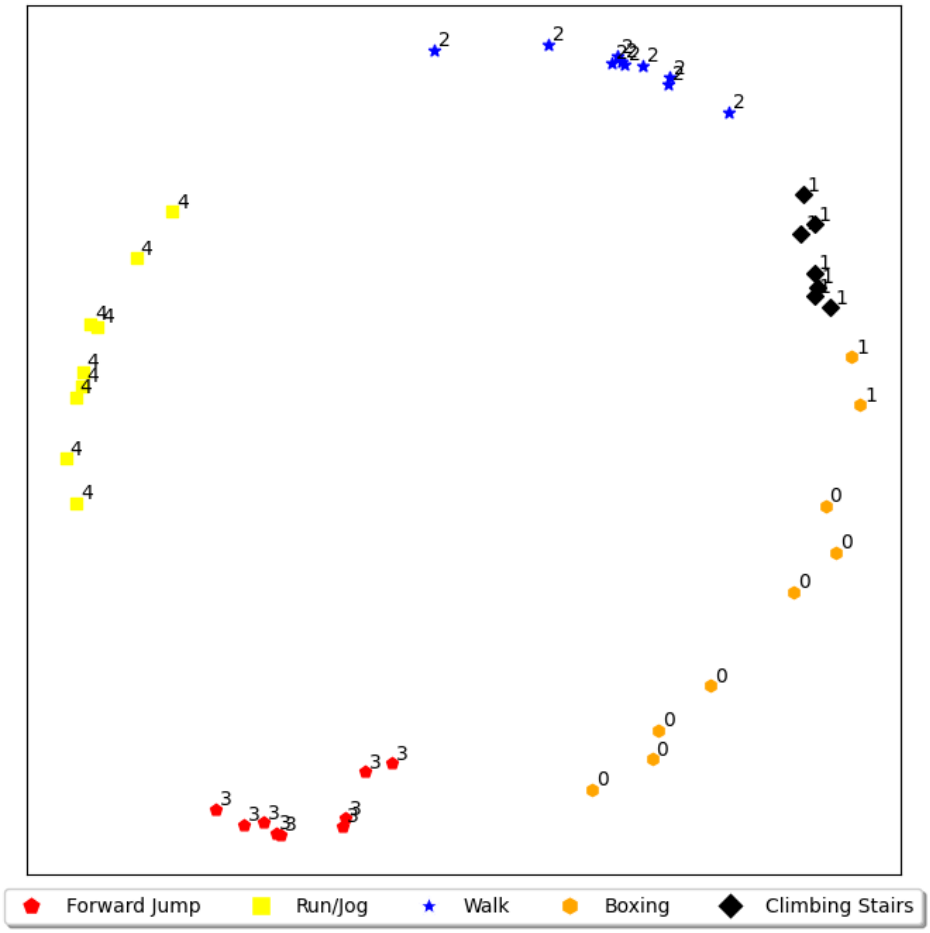


Figure 5.6: 2D visualization using cMDS of the reduced reparameterization of motion capture data, reduced to 4 dimensions using PCA and rebuilt with cosine distance, with dynamic programming using search depth 10. Colors and shapes represent the true cluster assignments, while the numbers indicate cluster assignments generated by K-Medoids clustering, with five clusters.

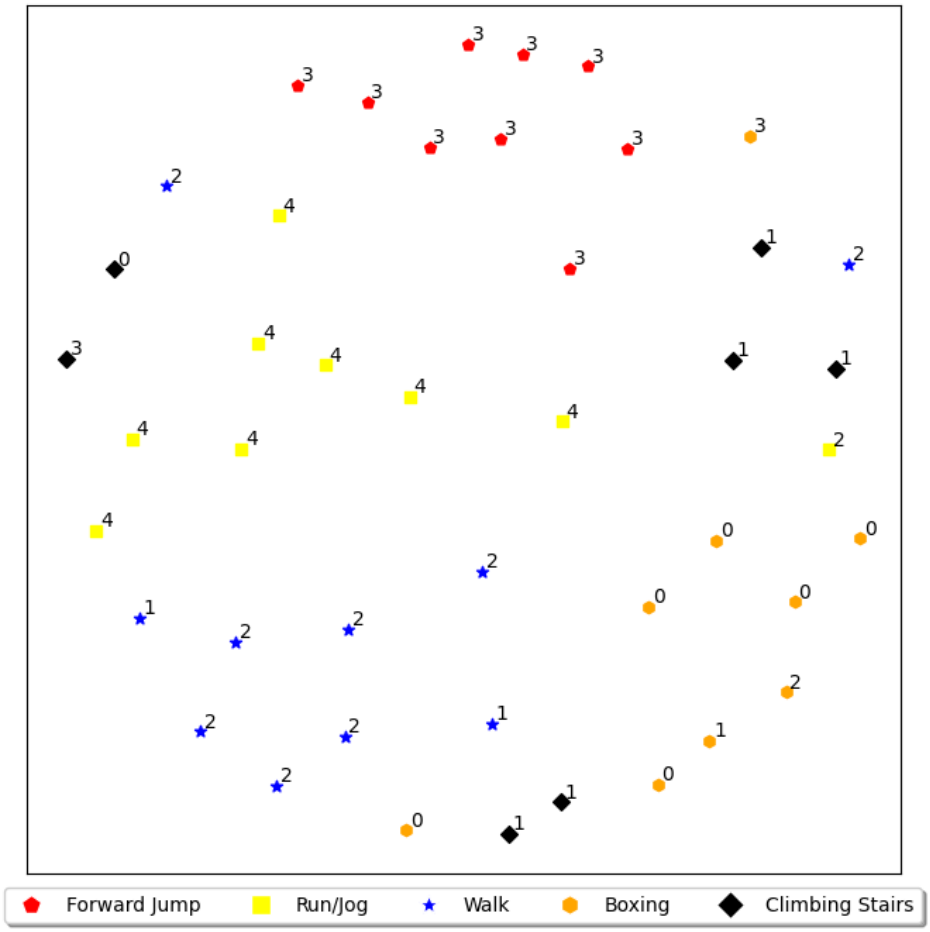


Figure 5.7: 2D visualization using cMDS of the logarithmic signature of motion capture data with truncation level 3. Colors and shapes represent the true cluster assignments, while the numbers indicate cluster assignments generated by K-Medoids clustering, with five clusters.

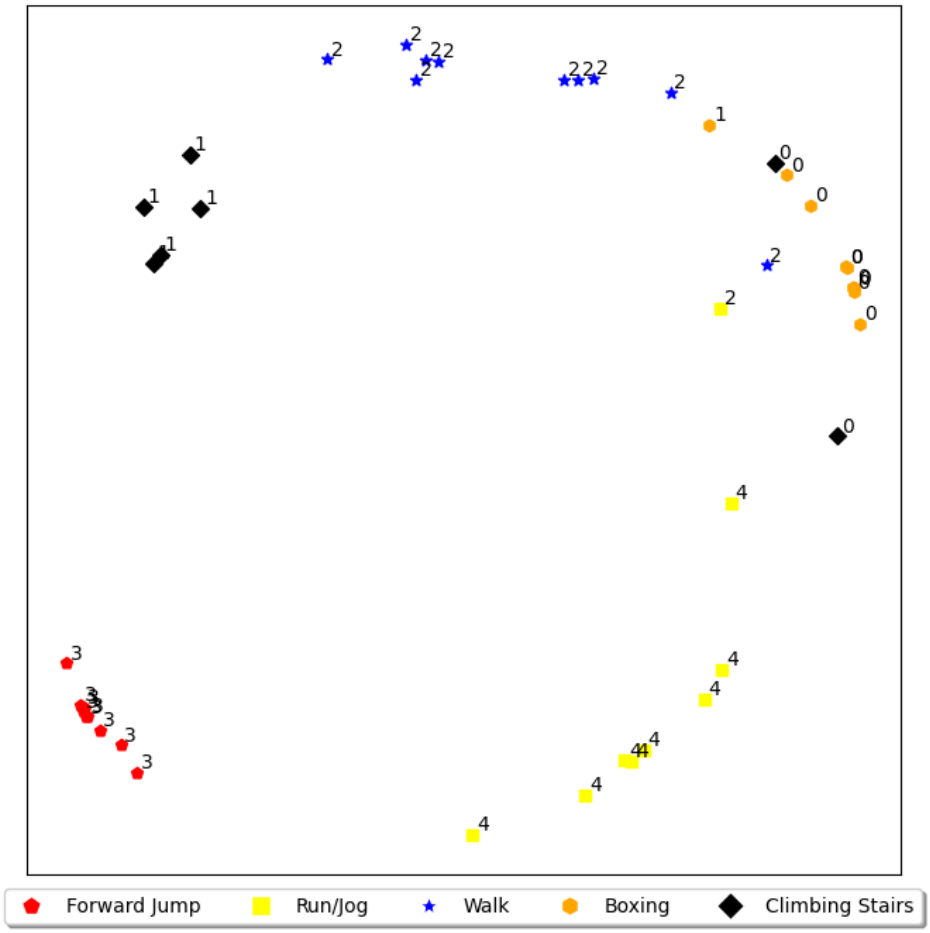


Figure 5.8: 2D visualization using cMDS of the reduced logarithmic signature of motion capture data, reduced to 4 dimensions using PCA and rebuilt with cosine distance, with truncation level 3. Colors and shapes represent the true cluster assignments, while the numbers indicate cluster assignments generated by K-Medoids clustering, with five clusters.

5.3.4 Silhouette Score

The Silhouette Score, introduced by Rousseeuw [70], evaluates clustering quality by quantifying how well objects match their own clusters compared to other clusters.

For each point i, let $a(i)$ be the average distance to all other points in the same cluster, and $b(i)$ be the average distance to all points in the nearest different cluster. The Silhouette Score for point i is:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \tag{5.7}$$

where $-1 \leq s(i) \leq 1$. A high $s(i)$ indicates that the point is well-matched to its own cluster and poorly matched to other clusters.

The overall Silhouette Score is the mean of the scores for all points, with higher scores indicating better-defined clusters. This metric robustly evaluates clustering performance, ensuring meaningful and interpretable groupings.

Reparam	LogSig	Reparam (Red)	LogSig (Red)
0.115690	0.072149	0.791245	0.751394

Table 5.2: Silhouette Scores for K-Medoids Clustering

In Table 5.2, we present the Silhouette Scores for K-Medoids clustering on the motion capture data (see Table 5.1). The scores indicate that the reparameterization method has a higher score (0.115) compared to the logarithmic signature method (0.072). Additionally, the scores significantly increase when the dimensionality of the data is reduced, reaching 0.791 and 0.751 when utilizing reparameterization and logarithmic signature, respectively. This suggests that clustering is more well-defined when the data is reduced.

Chapter 6

Conclusion

This thesis explored two techniques for reparameterization: the fully discretized method using the Square Root Velocity Transform (SRVT) with dynamic programming, and the geodesic interpolation method, where optimal nodes were determined using Sequential Least Squares Programming (SLSQP). The primary focus was on dynamic programming. Both methods showed promise in achieving optimal reparameterizations and effectively discretizing curves based on their geometric shapes. These techniques were applied to compute the shape space distance between curves, utilizing the normalized logarithmic signature of curves for the same purpose.

We extended previous research by adapting these methodologies to the special Euclidean group ($SE(3)^n$). Although we lacked real motion capture data for $SE(3)^n$, the methods were validated using synthetic data, indicating their potential utility.

Our framework was further expanded to include post-processing steps. Dimensionality reduction techniques, specifically Principal Component Analysis (PCA) and Classical Multi-Dimensional Scaling (cMDS), were employed, followed by clustering using K-Medoids. Silhouette scores were calculated to quantify the clustering quality, highlighting the importance of post-processing in enhancing the discriminative capability of our data.

Reparameterization proved particularly effective with motion capture data, discretizing five key moments identified in our study when post-processed. Although the logarithmic signature method was robust and faster, it was less effective.

A problem was identified when using the logarithmic signature on $SE(3)^n$ data due to discrepancies between the matrix norm and the vector norm in its vectorized form, which biased the results towards translation over rotation.

The post-processing phase was crucial in our analysis. By reducing the dimensionality of the data, we improved our ability to distinguish between different motions, facilitating effective clustering beyond mere visualization. This confirmed that data reduction enhances the classification of motion capture data. Silhouette scores further demonstrated the superiority of reparameterization utilizing SRVT over the logarithmic signature in terms of classification accuracy.

Future Work

Given the time and scope constraints of this thesis, several areas remain unexplored. Future research could focus on the following:

1. **Classification on $\mathrm{SE}(3)^n$ Data:** Exploring real motion capture data utilizing $\mathrm{SE}(3)^n$ and attempting classification on such data could be valuable, as our tools show promise for these applications.
2. **Mathematical Properties of Motions:** Investigate mathematical properties of motions, particularly $\hat{u}(t)$, under the assumption that motions can be described as:
$$\frac{d}{dt}g(t) = g(t)\hat{u}(t),$$
where we assume we can remove the parameterization, this aligns with our initial goals for reparameterization using geodesic interpolation.
3. **Pre- and Post-Processing Analysis:** A detailed examination of both pre-processing and post-processing techniques could yield significant insights and improvements, given their highlighted importance in our study.
4. **Semi-Discretized Methods on Motion Capture Data:** By utilizing semi-discretized methods to motion capture data we could achieve better optimal reparameterization and more accurate classification, as previous research has shown promising results for these methods.
5. **Theoretical Analysis:** A more thorough analysis of the theoretical foundations of our methods would enhance understanding and potentially improve their application.

Bibliography

[1] D. W. Thompson. 'On Growth and Form | Project Gutenberg.' (), [Online]. Available: https://www.gutenberg.org/files/55264/55264-h/55264-h.htm (visited on 22/05/2024).

[2] D. G. Kendall, 'Shape Manifolds, Procrustean Metrics, and Complex Projective Spaces,' *Bulletin of the London Mathematical Society*, vol. 16, no. 2, pp. 81–121, 1984, ISSN: 1469-2120. DOI: 10.1112/blms/16.2.81. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1112/blms/16.2.81 (visited on 22/05/2024).

[3] A. Srivastava, P. Turaga and S. Kurtek, 'On advances in differential-geometric approaches for 2D and 3D shape analyses and activity recognition,' *Image and Vision Computing*, vol. 30, no. 6, pp. 398–416, 1st Jun. 2012, ISSN: 0262-8856. DOI: 10.1016/j.imavis.2012.03.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885612000492 (visited on 22/05/2024).

[4] A. Srivastava, E. Klassen, S. H. Joshi and I. H. Jermyn, 'Shape Analysis of Elastic Curves in Euclidean Spaces,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1415–1428, Jul. 2011, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2010.184. [Online]. Available: https://ieeexplore.ieee.org/document/5601739 (visited on 23/11/2023).

[5] L. Younes, 'Computable Elastic Distances between Shapes,' *SIAM Journal on Applied Mathematics*, vol. 58, no. 2, pp. 565–586, 1998, ISSN: 0036-1399. DOI: 10.1137/S0036139995287685. JSTOR: 118345. [Online]. Available: https://www.jstor.org/stable/118345 (visited on 22/05/2024).

[6] M. Bauer, M. Bruveris and P. W. Michor, 'Overview of the Geometries of Shape Spaces and Diffeomorphism Groups,' *J Math Imaging Vis*, vol. 50, no. 1-2, pp. 60–97, Sep. 2014, ISSN: 0924-9907, 1573-7683. DOI: 10.1007/s10851-013-0490-z. arXiv: 1305.1150 [math]. [Online]. Available: http://arxiv.org/abs/1305.1150 (visited on 03/12/2023).

[7] A. Srivastava and E. P. Klassen, *Functional and Shape Data Analysis* (Springer Series in Statistics). New York, NY: Springer, 2016, ISBN: 978-1-4939-4018-9 978-1-4939-4020-2. DOI: 10.1007/978-1-4939-4020-2. [Online]. Available: http://link.springer.com/10.1007/978-1-4939-4020-2 (visited on 01/06/2024).

[8] W. Liu, A. Srivastava and J. Zhang, 'A Mathematical Framework for Protein Structure Comparison,' *PLoS computational biology*, vol. 7, e1001075, 3rd Feb. 2011. DOI: 10.1371/journal.pcbi.1001075.

[9] J. Eckhardt, R. Hiptmair, T. Hohage, H. Schumacher and M. Wardetzky, 'Elastic energy regularization for inverse obstacle scattering problems,' *Inverse Problems*, vol. 35, no. 10, p. 104 009, 1st Oct. 2019, ISSN: 0266-5611, 1361-6420. DOI: 10.1088/1361-6420/ab3034. arXiv: 1903.05074 [math]. [Online]. Available: http://arxiv.org/abs/1903.05074 (visited on 01/06/2024).

[10] H. Laga, S. Kurtek, A. Srivastava and S. J. Miklavcic, 'Landmark-free statistical analysis of the shape of plant leaves,' *Journal of Theoretical Biology*, vol. 363, pp. 41–52, 21st Dec. 2014, ISSN: 0022-5193. DOI: 10.1016/j.jtbi.2014.07.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022519314004500 (visited on 01/06/2024).

[11] J. Glaunès, A. Qiu, M. I. Miller and L. Younes, 'Large Deformation Diffeomorphic Metric Curve Mapping,' *Int J Comput Vis*, vol. 80, no. 3, pp. 317–336, 1st Dec. 2008, ISSN: 1573-1405. DOI: 10.1007/s11263-008-0141-9. [Online]. Available: https://doi.org/10.1007/s11263-008-0141-9 (visited on 01/06/2024).

[12] K.-T. Chen, 'Iterated Integrals and Exponential Homomorphisms†,' *Proceedings of the London Mathematical Society*, vol. s3-4, no. 1, pp. 502–512, 1954, ISSN: 1460-244X. DOI: 10.1112/plms/s3-4.1.502. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1112/plms/s3-4.1.502 (visited on 25/11/2023).

[13] T. J. Lyons, 'Differential equations driven by rough signals,' *Revista Matemática Iberoamericana*, vol. 14, no. 2, pp. 215–310, 31st Aug. 1998, ISSN: 0213-2230. DOI: 10.4171/rmi/240. [Online]. Available: https://ems.press/journals/rmi/articles/5137 (visited on 22/05/2024).

[14] I. Chevyrev and A. Kormilitzin. 'A Primer on the Signature Method in Machine Learning.' arXiv: 1603.03788 [cs, stat]. (11th Mar. 2016), [Online]. Available: http://arxiv.org/abs/1603.03788 (visited on 26/11/2023), preprint.

[15] I. Chevyrev, V. Nanda and H. Oberhauser, 'Persistence Paths and Signature Features in Topological Data Analysis,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 1, pp. 192–202, Jan. 2020, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2018.2885516. [Online]. Available: https://ieeexplore.ieee.org/document/8567922 (visited on 22/05/2024).

[16] M. Eslitzbichler, 'Modelling character motions on infinite-dimensional manifolds,' *Vis Comput*, vol. 31, no. 9, pp. 1179–1190, 1st Sep. 2015, ISSN: 1432-2315. DOI: 10.1007/s00371-014-1001-y. [Online]. Available: https://doi.org/10.1007/s00371-014-1001-y (visited on 02/12/2023).

[17] E. Celledoni, M. Eslitzbichler and A. Schmeding, 'Shape Analysis on Lie Groups with Applications in Computer Animation,' *JGM*, vol. 8, no. 3, pp. 273–304, Sep. 2016, ISSN: 1941-4889. DOI: 10.3934/jgm.2016008. arXiv: 1506.00783 [cs, math]. [Online]. Available: http://arxiv.org/abs/1506.00783 (visited on 23/11/2023).

[18] E. Celledoni, P. E. Lystad and N. Tapia, 'Signatures in Shape Analysis: An Efficient Approach to Motion Identification,' in vol. 11712, 2019, pp. 21–30. DOI: 10.1007/978-3-030-26980-7_3. arXiv: 1906.06406 [cs, math, stat]. [Online]. Available: http://arxiv.org/abs/1906.06406 (visited on 02/12/2023).

[19] L. Kovar and M. Gleicher, 'Automated extraction and parameterization of motions in large data sets,' *ACM Trans. Graph.*, vol. 23, no. 3, pp. 559–568, 1st Aug. 2004, ISSN: 0730-0301. DOI: 10.1145/1015706.1015760. [Online]. Available: https://dl.acm.org/doi/10.1145/1015706.1015760 (visited on 11/12/2023).

[20] T. Pejsa and I. Pandzic, 'State of the Art in Example-Based Motion Synthesis for Virtual Characters in Interactive Applications,' *Computer Graphics Forum*, vol. 29, no. 1, pp. 202–226, 2010, ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01591.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01591.x (visited on 09/12/2023).

[21] J. Lander, 'Working with Motion Capture File Formats,' *Game Developer*, vol. 5, no. 1, pp. 30–37, Jan. 1998.

[22] K. Adistambha, C. H. Ritz and I. S. Burnett, 'Motion classification using Dynamic Time Warping,' in *2008 IEEE 10th Workshop on Multimedia Signal Processing*, Oct. 2008, pp. 622–627. DOI: 10.1109/MMSP.2008.4665151. [Online]. Available: https://ieeexplore.ieee.org/document/4665151 (visited on 09/12/2023).

[23] W. Mio, A. Srivastava and S. Joshi, 'On Shape of Plane Elastic Curves,' *International Journal of Computer Vision*, vol. 73, pp. 307–324, 1st Jul. 2007. DOI: 10.1007/s11263-006-9968-0.

[24] E. Celledoni, S. Eidnes and A. Schmeding, 'Shape analysis on homogeneous spaces: A generalised SRVT framework,' in vol. 13, 2018, pp. 187–220. DOI: 10.1007/978-3-030-01593-0_7. arXiv: 1704.01471 [math]. [Online]. Available: http://arxiv.org/abs/1704.01471 (visited on 03/12/2023).

[25] M. Bauer, M. Eslitzbichler and M. Grasmair. 'Landmark-Guided Elastic Shape Analysis of Human Character Motions.' arXiv: 1502.07666 [cs]. (3rd Feb. 2015), [Online]. Available: http://arxiv.org/abs/1502.07666 (visited on 07/12/2023), preprint.

[26] H. Bærland, 'Optimal reparametrization of curves in shape analysis by introducing a deep neural network architecture,' M.S. thesis, NTNU, 2021. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2824270 (visited on 22/05/2024).

[27] 'THE 17 GOALS | Sustainable Development.' (), [Online]. Available: https://sdgs.un.org/goals (visited on 22/05/2024).

[28] J. N. Bakken, 'Lie Group Methods in Dynamic Shape Analysis,' Project report in TMA4500, Department of Mathematical Sciences, NTNU - Norwegian University of Science and Technology, Dec. 2023.

[29] J. M. Lee, *Introduction to Smooth Manifolds* (Graduate Texts in Mathematics). New York, NY: Springer, 2012, vol. 218, ISBN: 978-1-4419-9981-8 978-1-4419-9982-5. DOI: 10.1007/978-1-4419-9982-5. [Online]. Available: https://link.springer.com/10.1007/978-1-4419-9982-5 (visited on 22/11/2023).

[30] A. Kriegl and P. W. Michor. 'The Convenient Setting of Global Analysis.' (), [Online]. Available: https://bookstore.ams.org/surv-53 (visited on 22/11/2023).

[31] P. J. Olver, *Equivalence, Invariants and Symmetry*. Cambridge University Press, 1995, ISBN: 0-521-47811-1.

[32] P. Michor and D. Mumford, 'Vanishing geodesic distance on spaces of submanifolds and diffeomorphisms,' *Documenta Mathematica*, vol. 10, 17th Oct. 2004. DOI: 10.4171/dm/187.

[33] P. W. Michor and D. Mumford, 'An overview of the Riemannian metrics on spaces of curves using the Hamiltonian approach,' *Applied and Computational Harmonic Analysis*, Special Issue on Mathematical Imaging, vol. 23, no. 1, pp. 74–113, 1st Jul. 2007, ISSN: 1063-5203. DOI: 10.1016/j.acha.2006.07.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1063520307000243 (visited on 22/05/2024).

[34] M. Bauer, M. Bruveris, S. Marsland and P. W. Michor, 'Constructing reparametrization invariant metrics on spaces of plane curves,' *Differential Geometry and its Applications*, vol. 34, pp. 139–165, Jun. 2014, ISSN: 09262245. DOI: 10.1016/j.difgeo.2014.04.008. arXiv: 1207.5965 [math]. [Online]. Available: http://arxiv.org/abs/1207.5965 (visited on 21/12/2023).

[35] A. Schmeding, *An Introduction to Infinite-Dimensional Differential Geometry*. 31st Dec. 2022. DOI: 10.1017/9781009091251. arXiv: 2112.08114 [math]. [Online]. Available: http://arxiv.org/abs/2112.08114 (visited on 12/12/2023).

[36] A. B. Tumpach and P. Kán. 'Temporal Alignment of Human Motion Data: A Geometric Point of View.' arXiv: 2303.15259 [math]. (27th Mar. 2023), [Online]. Available: http://arxiv.org/abs/2303.15259 (visited on 21/12/2023), preprint.

[37] M. Bruveris, P. W. Michor and D. Mumford, 'GEODESIC COMPLETENESS FOR SOBOLEV METRICS ON THE SPACE OF IMMERSED PLANE CURVES,' *Forum of Mathematics, Sigma*, vol. 2, e19, Jul. 2014, ISSN: 2050-5094. DOI: 10.1017/fms.2014.19. [Online]. Available: https://www.cambridge.org/core/journals/forum-of-mathematics-sigma/article/geodesic-completeness-for-sobolev-metrics-on-the-space-of-immersed-plane-curves/90B8888AD48EA07AF5B02194C97D0547 (visited on 24/11/2023).

[38] T. Shingel, 'Interpolation in special orthogonal groups,' *IMA Journal of Numerical Analysis*, vol. 29, no. 3, pp. 731–745, 1st Jul. 2009, ISSN: 0272-4979. DOI: 10.1093/imanum/drn033. [Online]. Available: https://doi.org/10.1093/imanum/drn033 (visited on 23/11/2023).

[39] A. Marthinsen, 'Interpolation in Lie Groups and Homogeneous Spaces,' 2nd Apr. 1999.

[40] B. C. Hall, *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction* (Graduate Texts in Mathematics). Cham: Springer International Publishing, 2015, vol. 222, ISBN: 978-3-319-13466-6 978-3-319-13467-3. DOI: 10.1007/978-3-319-13467-3. [Online]. Available: https://link.springer.com/10.1007/978-3-319-13467-3 (visited on 23/11/2023).

[41] G. Gallego and A. Yezzi, 'A Compact Formula for the Derivative of a 3-D Rotation in Exponential Coordinates,' *J Math Imaging Vis*, vol. 51, no. 3, pp. 378–384, 1st Mar. 2015, ISSN: 1573-7683. DOI: 10.1007/s10851-014-0528-x. [Online]. Available: https://doi.org/10.1007/s10851-014-0528-x (visited on 23/11/2023).

[42] E. Celledoni, E. Çokaj, A. Leone, D. Murari and B. Owren, 'Lie group integrators for mechanical systems,' *International Journal of Computer Mathematics*, vol. 99, no. 1, pp. 58–88, 2nd Jan. 2022, ISSN: 0020-7160. DOI: 10.1080/00207160.2021.1966772. [Online]. Available: https://doi.org/10.1080/00207160.2021.1966772 (visited on 20/12/2023).

[43] E. Celledoni and B. Owren, 'Lie group methods for rigid body dynamics and time integration on manifolds,' *Computer Methods in Applied Mechanics and Engineering*, vol. 192, no. 3, pp. 421–438, 17th Jan. 2003, ISSN: 0045-7825. DOI: 10.1016/S0045-7825(02)00520-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782502005200 (visited on 20/12/2023).

[44] Y. Ma, S. Soatto, J. Košecká and S. S. Sastry, *An Invitation to 3-D Vision* (Interdisciplinary Applied Mathematics), S. S. Antman, J. E. Marsden, L. Sirovich and S. Wiggins, red. New York, NY: Springer, 2004, vol. 26, ISBN: 978-1-4419-1846-8 978-0-387-21779-6. DOI: 10.1007/978-0-387-21779-6. [Online]. Available: http://link.springer.com/10.1007/978-0-387-21779-6 (visited on 05/12/2023).

[45] Y. Wang and G. Chirikjian, 'Nonparametric Second-Order Theory of Error Propagation on Motion Groups,' *The International journal of robotics research*, vol. 27, pp. 1258–1273, 1st Nov. 2008. DOI: 10.1177/0278364908097583.

[46] J. L. Blanco-Claraco. 'A tutorial on $\mathbf{SE}(3)$ transformation parameterizations and on-manifold optimization.' arXiv: 2103.15980 [cs]. (7th Apr. 2022), [Online]. Available: http://arxiv.org/abs/2103.15980 (visited on 28/05/2024), preprint.

[47] M. Bruveris. 'Optimal reparametrizations in the square root velocity framework.' arXiv: 1507.02728 [math]. (30th Sep. 2016), [Online]. Available: http://arxiv.org/abs/1507.02728 (visited on 25/05/2024), preprint.

[48] E. N. Wøien, 'A Semi-Discretized Method for Optimal Reparametrization of Curves,' M.S. thesis, NTNU, 2019. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2624611 (visited on 24/05/2024).

[49] C. Runge, 'Ueber die numerische Auflösung von Differentialgleichungen,' *Math. Ann.*, vol. 46, no. 2, pp. 167–178, 1st Jun. 1895, ISSN: 1432-1807. DOI: 10.1007/BF01446807. [Online]. Available: https://doi.org/10.1007/BF01446807 (visited on 20/05/2024).

[50] J. C. Butcher, 'Coefficients for the study of Runge-Kutta integration processes,' *Journal of the Australian Mathematical Society*, vol. 3, no. 2, pp. 185–201, May 1963, ISSN: 0004-9735. DOI: 10.1017/S1446788700027932. [Online]. Available: https://www.cambridge.org/core/journals/journal-of-the-australian-mathematical-society/article/coefficients-for-the-study-of-rungekutta-integration-processes/457E8B0413C29B9D7BBCD7D2A45A23D5 (visited on 20/05/2024).

[51] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa and P. van Mulbregt, 'SciPy 1.0: Fundamental algorithms for scientific computing in Python,' *Nat Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2020, ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2. [Online]. Available: https://www.nature.com/articles/s41592-019-0686-2 (visited on 29/05/2024).

[52] K.-T. Chen, 'Algebras of Iterated Path Integrals and Fundamental Groups,' *Transactions of the American Mathematical Society*, vol. 156, pp. 359–379, 1971, ISSN: 0002-9947. DOI: 10.2307/1995617. JSTOR: 1995617. [Online]. Available: https://www.jstor.org/stable/1995617 (visited on 06/12/2023).

[53] K.-T. Chen, 'Integration of Paths, Geometric Invariants and a Generalized Baker- Hausdorff Formula,' *Annals of Mathematics*, vol. 65, no. 1, pp. 163–178, 1957, ISSN: 0003-486X. DOI: 10.2307/1969671. JSTOR: 1969671. [Online]. Available: https://www.jstor.org/stable/1969671 (visited on 21/11/2023).

[54] T. Lyons. 'Rough paths, Signatures and the modelling of functions on streams.' arXiv: 1405.4537 [math, q-fin, stat]. (18th May 2014), [Online]. Available: http://arxiv.org/abs/1405.4537 (visited on 21/11/2023), preprint.

[55] T. J. Lyons and N. Sidorova, 'On the radius of convergence of the logarithmic signature,' *Illinois Journal of Mathematics*, vol. 50, no. 1-4, pp. 763–790, Jan. 2006, ISSN: 0019-2082, 1945-6581. DOI: 10.1215/ijm/1258059491. [Online]. Available: https://projecteuclid.org/journals/illinois-journal-of-mathematics/volume-50/issue-1-4/On-the-radius-of-convergence-of-the-logarithmic-signature/10.1215/ijm/1258059491.full (visited on 21/11/2023).

[56] J. Reizenstein. 'Calculation of Iterated-Integral Signatures and Log Signatures.' arXiv: 1712.02757 [math]. (7th Dec. 2017), [Online]. Available: http://arxiv.org/abs/1712.02757 (visited on 29/11/2023), preprint.

[57] J. Reizenstein and B. Graham. 'The iisignature library: Efficient calculation of iterated-integral signatures and log signatures.' arXiv: 1802.08252 [cs, math]. (22nd Feb. 2018), [Online]. Available: http://arxiv.org/abs/1802.08252 (visited on 25/11/2023), preprint.

[58] D. Lee and R. Ghrist. 'Path Signatures on Lie Groups.' arXiv: 2007.06633 [cs, math, stat]. (15th Jul. 2020), [Online]. Available: http://arxiv.org/abs/2007.06633 (visited on 07/12/2023), preprint.

[59] K.-T. Chen, 'Integration of Paths–A Faithful Representation of Paths by Noncommutative Formal Power Series,' *Transactions of the American Mathematical Society*, vol. 89, no. 2, pp. 395–407, 1958, ISSN: 0002-9947. DOI: 10.2307/1993193. JSTOR: 1993193. [Online]. Available: https://www.jstor.org/stable/1993193 (visited on 03/12/2023).

[60] B. Hambly and T. Lyons, 'Uniqueness for the signature of a path of bounded variation and the reduced path group,' *Annals of Mathematics*, vol. 171, no. 1, pp. 109–167, 2010, ISSN: 0003-486X. DOI: 10.4007/annals.2010.171.109. JSTOR: 27799199. [Online]. Available: https://www.jstor.org/stable/27799199 (visited on 26/11/2023).

[61] R. Ree, 'Lie Elements and an Algebra Associated With Shuffles,' *Annals of Mathematics*, vol. 68, no. 2, pp. 210–220, 1958, ISSN: 0003-486X. DOI: 10.2307/1970243. JSTOR: 1970243. [Online]. Available: https://www.jstor.org/stable/1970243 (visited on 26/11/2023).

[62] N. Metropolis and G.-C. Rota, 'Witt vectors and the algebra of necklaces,' *Advances in Mathematics*, vol. 50, no. 2, pp. 95–125, 1st Nov. 1983, ISSN: 0001-8708. DOI: 10.1016/0001-8708(83)90035-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/000187088390035X (visited on 29/11/2023).

[63] M. Deléglise and J. Rivat, 'Computing the summation of the Möbius function,' *Experimental Mathematics*, vol. 5, no. 4, pp. 291–295, Jan. 1996, ISSN: 1058-6458, 1944-950X. DOI: 10.1080/10586458.1996.10504594. [Online]. Available: https://projecteuclid.org/journals/experimental-mathematics/volume-5/issue-4/Computing-the-summation-of-the-M%c3%b6bius-function/em/1047565447.full (visited on 29/11/2023).

[64] 'Carnegie Mellon University - CMU Graphics Lab - motion capture library.' (), [Online]. Available: http://mocap.cs.cmu.edu/ (visited on 30/11/2023).

[65] K. Pearson, 'LIII. On lines and planes of closest fit to systems of points in space,' 1st Nov. 1901. DOI: 10.1080/14786440109462720. [Online]. Available: https://zenodo.org/records/1430636 (visited on 26/05/2024).

[66] I. Jolliffe, *Principal Component Analysis* (Springer Series in Statistics). New York: Springer-Verlag, 2002, ISBN: 978-0-387-95442-4. DOI: 10.1007/b98835. [Online]. Available: http://link.springer.com/10.1007/b98835 (visited on 26/05/2024).

[67] L. Kaufmann and P. Rousseeuw, 'Clustering by Means of Medoids,' *Data Analysis based on the L1-Norm and Related Methods*, pp. 405–416, 1st Jan. 1987.

[68] E. Schubert and P. J. Rousseeuw, 'Fast and Eager k-Medoids Clustering: O(k) Runtime Improvement of the PAM, CLARA, and CLARANS Algorithms,' *Information Systems*, vol. 101, p. 101 804, Nov. 2021, ISSN: 03064379. DOI: 10.1016/j.is.2021.101804. arXiv: 2008.05171 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2008.05171 (visited on 27/05/2024).

[69] J. Wang, 'Classical Multidimensional Scaling,' in *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*, J. Wang, Ed., Berlin, Heidelberg: Springer, 2012, pp. 115–129, ISBN: 978-3-642-27497-8. DOI: 10.1007/978-3-642-27497-8_6. [Online]. Available: https://doi.org/10.1007/978-3-642-27497-8_6 (visited on 18/12/2023).

[70] P. Rousseeuw, 'Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Comput. Appl. Math. 20, 53-65,' *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1st Nov. 1987. DOI: 10.1016/0377-0427(87)90125-7.

Appendix A

Additional Material

A.1 Inequality for Concave Functions

Consider a function $f : \mathbb{R} \to \mathbb{R}$ that is concave, with the property that $f(0) \geq 0$. For any $a, b \in \mathbb{R}^+$, we aim to demonstrate the following inequality:

$$f(a) + f(b) \geq f(a+b).$$

The definition of concavity for any function f stipulates that:

$$f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda) f(x_2),$$

for any $x_1, x_2 \in \mathbb{R}$ and $\lambda \in [0,1]$. Applying this definition, we set $x_1 = 0$ and $x_2 = a + b$, and examine two specific cases of λ:

Let $\lambda = \frac{a}{a+b}$. Then,

$$f\left(\frac{a}{a+b} \cdot 0 + \frac{b}{a+b} \cdot (a+b)\right) \geq \frac{a}{a+b} f(0) + \frac{b}{a+b} f(a+b)$$
$$f(b) \geq \frac{b}{a+b} f(a+b),$$

where $\frac{a}{a+b} f(0) \geq 0$ due to the given condition that $f(0) \geq 0$.

Let $\lambda = \frac{b}{a+b}$. Then,

$$f\left(\frac{b}{a+b} \cdot 0 + \frac{a}{a+b} \cdot (a+b)\right) \geq \frac{b}{a+b} f(0) + \frac{a}{a+b} f(a+b)$$
$$f(a) \geq \frac{a}{a+b} f(a+b),$$

with $\frac{b}{a+b} f(0) \geq 0$ by the same premise.

Summing these inequalities yields:

$$f(a) + f(b) \geq \left(\frac{a}{a+b} + \frac{b}{a+b}\right) f(a+b) = f(a+b), \tag{A.1}$$

thus confirming the asserted inequality.

A.2 q-Hölder Continuity of Power Norm Functions

Consider the function $f : \mathbb{R}^n \to \mathbb{R}$ defined by $f(x) = \|x\|^q$ for a given $q \in (0,1)$. We aim to demonstrate that f is q-Hölder continuous, i.e., there exists a constant $C \in \mathbb{R}^+$ such that for any $x, y \in \mathbb{R}^n$, the following inequality holds:

$$|f(x) - f(y)| \leq C\|x - y\|^q. \tag{A.2}$$

Using the results from A.1, we know that if f is concave and $f(0) \geq 0$, then for any $a, b \in \mathbb{R}^+$:

$$f(a) + f(b) \geq f(a + b).$$

In this case, since $f(z) = z^q$ is concave for $z \geq 0$, we apply the definition of concavity directly. Assume without loss of generality that $\|x\| \geq \|y\|$. Then,

$$|\|x\|^q - \|y\|^q| = \|x\|^q - \|y\|^q.$$

By the concavity of $f(z) = z^q$, we have:

$$(\|y\| + \|x - y\|)^q \leq \|y\|^q + \|x - y\|^q.$$

Thus,

$$\|x\|^q = (\|y + (x - y)\|)^q \leq (\|y\| + \|x - y\|)^q \leq \|y\|^q + \|x - y\|^q.$$

Therefore,

$$\|x\|^q - \|y\|^q \leq \|x - y\|^q.$$

By symmetry, if $\|y\| \geq \|x\|$, the same argument applies. Hence, we conclude:

$$|f(x) - f(y)| = |\|x\|^q - \|y\|^q| \leq \|x - y\|^q,$$

confirming the q-Hölder continuity of f with constant $C = 1$.

A.3 α-Hölder Continuity

Consider the function $V : \mathbb{R}^n \to \mathbb{R}^n$ defined by $V(x) = x\|x\|^{\alpha-1}$ for $0 < \alpha \leq 1$. We aim to demonstrate that V is α-Hölder continuous, i.e., there exists a constant $C \in \mathbb{R}^+$ such that for any $x, y \in \mathbb{R}^n$:

$$\left\| x\|x\|^{\alpha-1} - y\|y\|^{\alpha-1} \right\| \leq C\|x - y\|^\alpha.$$

Using the results from A.2, we know that if $f(x) = \|x\|^q$ is q-Hölder continuous, then

$$|\|x\|^q - \|y\|^q| \leq \|x-y\|^q,$$

for any $x, y \in \mathbb{R}^n$ and $q \in (0,1)$. Applying this result, we proceed as follows:

$$\begin{aligned}
\left\|x\|x\|^{\alpha-1} - y\|y\|^{\alpha-1}\right\| &= \left\|x\|x\|^{\alpha-1} - \underbrace{\|x\|\frac{y}{\|y\|} + \|x\|\frac{y}{\|y\|}}_{=0} - y\|y\|^{\alpha-1}\right\| \\
&\leq \left\|x\|x\|^{\alpha-1} - \|x\|^{\alpha}\frac{y}{\|y\|}\right\| + \left\|\|x\|^{\alpha}\frac{y}{\|y\|} - y\|y\|^{\alpha-1}\right\| \\
&= \left\|\|x\|\frac{\|x\|^{\alpha}}{\|x\|} - \|x\|^{\alpha}\frac{y}{\|y\|}\right\| + \left\|\|x\|^{\alpha}\frac{y}{\|y\|} - \|y\|^{\alpha}\frac{y}{\|y\|}\right\| \\
&= \|x\|^{\alpha-1}\left\|x - \underbrace{y+y}_{=0} - \|x\|\frac{y}{\|y\|}\right\| + |\|x\|^{\alpha} - \|y\|^{\alpha}|\left\|\frac{y}{\|y\|}\right\| \\
&= \|x\|^{\alpha-1}\underbrace{\left\|x - y + y\frac{\|y\| - \|x\|}{\|y\|}\right\|}_{i} + |\|x\|^{\alpha} - \|y\|^{\alpha}|.
\end{aligned}$$

We can bound (i) as follows:

$$\begin{aligned}
\left\|x - y + y\frac{\|y\| - \|x\|}{\|y\|}\right\| &\leq \|x-y\| + \left\|y\frac{\|y\| - \|x\|}{\|y\|}\right\| \\
&= \|x-y\| + |\|y\| - \|x\|| \\
&\leq 2\|x-y\|.
\end{aligned}$$

Substituting this bound back into the inequality, we have:

$$\|x\|^{\alpha-1}\left\|x - y + y\frac{\|y\| - \|x\|}{\|y\|}\right\| + |\|x\|^{\alpha} - \|y\|^{\alpha}| \leq 2\|x\|^{\alpha-1}\|x-y\| + |\|x-y\|^{\alpha}|$$

Assuming without loss of generality that $\|x\| \geq \|y\| > 0$, we consider two cases:

If $\|x-y\| \leq \|x\|$, then:

$$\|x\|^{\alpha-1} \leq \|x-y\|^{\alpha-1},$$

such that

$$\begin{aligned}
2\|x\|^{\alpha-1}\|x-y\| + \|x-y\|^{\alpha} &\leq 2\|x-y\|^{\alpha-1}\|x-y\| + \|x-y\|^{\alpha} \\
&= 2\|x-y\|^{\alpha} + \|x-y\|^{\alpha} \\
&= 3\|x-y\|^{\alpha}.
\end{aligned}$$

If $\|x-y\| > \|x\|$, then:

$$\|x-y\| \leq \|x\| + \|y\| \leq 2\|x\|,$$

such that:

$$\begin{aligned} 2\|x\|^{\alpha-1}\|x-y\| + \|x-y\|^{\alpha} &\leq 2\|x\|^{\alpha-1}2\|x\| + \|x-y\|^{\alpha} \\ &\leq 4\|x-y\|^{\alpha} + \|x-y\|^{\alpha} \\ &= 5\|x-y\|^{\alpha}. \end{aligned}$$

Thus, we have shown that:

$$\left\| x\|x\|^{\alpha-1} - y\|y\|^{\alpha-1} \right\| \leq 5\|x-y\|^{\alpha},$$

which means that V is α-Hölder continuous with $C = 5$.

Appendix B

Movement data

B.1 Table of Movement Data

Table B.1: Table list of the motions used in the shape analysis. The movements are taken from the CMU Graphics Lab Motion Capture Database [64].

Movement Type	File	Start	End
Forward Jump	16_05.amc	90	220
Forward Jump	16_06.amc	200	330
Forward Jump	16_07.amc	200	330
Forward Jump	16_09.amc	240	370
Forward Jump	16_10.amc	260	390
Forward Jump	13_11.amc	190	320
Forward Jump	13_13.amc	160	290
Forward Jump	13_19.amc	205	335
Forward Jump	13_32.amc	125	255
Run/Jog	16_45.amc	0	130
Run/Jog	16_46.amc	0	130
Run/Jog	35_26.amc	0	130
Run/Jog	35_22.amc	0	130
Run/Jog	16_35.amc	0	130
Run/Jog	16_36.amc	0	130
Run/Jog	35_18.amc	0	130
Run/Jog	02_03.amc	0	130
Run/Jog	16_56.amc	0	130
Walk	16_16.amc	0	130
Walk	35_12.amc	0	130
Walk	16_58.amc	0	130
Walk	35_32.amc	0	130
Walk	35_11.amc	0	130
Walk	16_21.amc	0	130

Continued on next page

Table B.1 – *Continued from previous page*

Movement Type	File	Start	End
Walk	16_22.amc	0	130
Walk	16_15.amc	40	170
Walk	16_31.amc	40	170
Walk	16_47.amc	40	170
Boxing	13_17.amc	30	160
Boxing	13_18.amc	30	160
Boxing	14_01.amc	40	170
Boxing	14_02.amc	40	170
Boxing	14_03.amc	80	210
Boxing	15_13.amc	80	210
Boxing	17_10.amc	80	210
Boxing	15_04.amc	22200	22330
Boxing	15_05.amc	22400	22530
Climb Stairs	13_35.amc	200	330
Climb Stairs	13_36.amc	230	360
Climb Stairs	13_37.amc	220	350
Climb Stairs	13_38.amc	220	350
Climb Stairs	14_21.amc	220	350
Climb Stairs	14_22.amc	220	350
Climb Stairs	14_23.amc	220	350