2013

Statistical Analysis of Trajectories on Riemannian Manifolds

Jingyong Su
STATISTICAL ANALYSIS OF TRAJECTORIES ON RIEMANNIAN MANIFOLDS

By

JINGYONG SU

A Dissertation submitted to the
Department of Statistics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Summer Semester, 2013
Jingyong Su defended this dissertation on May 15, 2013.

The members of the supervisory committee were:

Anuj Srivastava
Professor Directing Thesis

Erik Klassen
University Representative

Fred Huffer
Committee Member

Jinfeng Zhang
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with the university requirements.
I lovingly dedicate this thesis to my wife, who supported me each step of the way.
ACKNOWLEDGMENTS

I would like to begin by thanking my dissertation advisor, Dr. Anuj Srivastava, for all his help and advice. He has been a wonderful mentor and teacher. His knowledge, excitement, patience and encouragement provided for a great research and learning environment. I am also thankful to Dr. Srivastava for providing me with many opportunities to travel to various conferences to present our research. Working with Dr. Srivastava has been a very rewarding experience and he has shown me how to be a successful teacher and researcher.

I also want to give sincere thanks to the members of my dissertation committee, Dr. Eric Klassen, Dr. Fred Huffer and Dr. Jinfeng Zhang. I would especially like to recognize Dr. Klassen as I have worked very closely with him on different aspects of research presented in my dissertation. I would like to thank Dr. Huffer and Dr. Zhang for taking the time to evaluate this dissertation and for providing helpful comments regarding this work.

I would like to acknowledge the many collaborators who have made significant contributions to this work, Dr. Sebastian Kurtek, Dr. Huiling Le, Dr. Ian Dryden and Dr. Zhaohua Ding. Thank you for all your input and help.

Finally, I would like to thank my family, friends and all of the members of the Department of Statistics. Your support and encouragement provided me with the drive to become a successful student and researcher.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td></td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td></td>
<td>viii</td>
</tr>
<tr>
<td>Abstract</td>
<td></td>
<td>xii</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Problem Statement</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Problem Description</td>
<td>5</td>
</tr>
<tr>
<td>1.4</td>
<td>Contributions</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Literature Review</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>Estimation of Full Trajectories</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Analysis of Temporal Trajectories</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Limitations</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>Estimation of Smooth Trajectories on Nonlinear Manifolds</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Mathematical Framework</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>Estimation of Trajectories on Nonlinear Manifolds</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Unit Sphere $S^2$</td>
<td>22</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Three-Dimensional Rotation Group $SO(3)$</td>
<td>23</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Symmetric Positive-Definite Matrices with Unit Determinant</td>
<td>28</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Kendall’s Shape Space</td>
<td>33</td>
</tr>
<tr>
<td>3.3</td>
<td>Estimation of Optimal Weight Ratio</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>Analysis of Trajectories on Nonlinear Manifolds</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Mathematical Framework</td>
<td>41</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Metric-Based Comparison of Trajectories</td>
<td>45</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Pairwise Temporal Registration of Trajectories</td>
<td>45</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Summarization and Registration of Multiple Trajectories</td>
<td>46</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Modeling and Evaluation of Trajectories</td>
<td>47</td>
</tr>
</tbody>
</table>
4.2 Analysis of Trajectories on Nonlinear Manifolds
   4.2.1 Unit Sphere $S^2$
   4.2.2 Special Euclidean Group $SE(2)$
   4.2.3 Symmetric Positive-Definite Matrices
   4.2.4 Shape Space of Planar Contours

5 Another Method for Analysis of Trajectories on $S^2$
   5.1 Mathematical framework
   5.2 Experimental Results
      5.2.1 Geodesic between trajectories before registration
      5.2.2 Geodesic between trajectories after registration

6 Detection, Classification and Estimation of Individual Shapes in 2D and 3D Point Clouds
   6.1 Introduction
      6.1.1 Past Work
      6.1.2 Our Approach
   6.2 Problem Formulation
      6.2.1 Model-Based Shape Detection
      6.2.2 Empirical Determination of Threshold
   6.3 Detection, Classification and Estimation of Shapes in Point Clouds
      6.3.1 Shape Detection
      6.3.2 Shape Classification
      6.3.3 Shape Estimation

7 Conclusion

A Another Method for Analysis of Trajectories on $S^2$
   A.1 Proof of Lemma 1
   A.2 Proof of Theorem 3

References

Biographical Sketch
LIST OF TABLES

3.1 Computational costs for trajectory updating on four different manifolds. . . . . . . 21

3.2 Energy terms and estimation errors for four methods in two different experiments
(λ₁ = 0.1, λ₂ = 1). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

4.1 Comparison of classification rates without and with alignment. . . . . . . . . . . . . 55
### LIST OF FIGURES

1.1 Examples of motivation for trajectory estimation: (a) bird migration; (b) human activity recognition. .......................................................... 3

1.2 Summary of trajectories on $S^2$: (a) a simulated example; (b) bird migration paths; (c) hurricane tracks; (d) cross-sectional mean of two trajectories without (top) and with (bottom) registration. ......................................................... 4

1.3 (a) smoothing spline in $\mathbb{R}^2$; (b) smoothing spline on manifolds. ................................................................. 6

2.1 Comparison of inverse exponential map and our representation: (a) a trajectory on $S^2$; (b) projected trajectory on the tangent space at a far away point $c$, using inverse exponential map; (c) our representation on the same tangent space. .................. 15

3.1 Smoothing splines of bird migration data ($\lambda_1 = \lambda_2 = 1$). ................................................................. 23

3.2 Minimization of $E_s(\lambda_1 = 0, \lambda_2 = 1)$: Each case shows the initial path (marked line) and the final path (solid line), along with the corresponding evolution of $E_s$. .... 25

3.3 Minimization of $E_d(\lambda_1 = 1, \lambda_2 = 0)$: Each case shows the initial path (marked line) and the final path (solid line), along with the evolution of $E_d$. ....... 25

3.4 Optimization of $E$ under different combinations of $\lambda_1$ and $\lambda_2$. In all these cases we use the same data points $(t_i, p_i)$ and the same initial path $\alpha$ for starting the algorithm. ................. 27

3.5 Comparison of three different methods. Solid black: true path, dotted yellow: spline on mean tangent space, dashdot green: piecewise geodesic, dashed blue: smoothing spline using our method. ................................................. 28

3.6 Estimation error versus: (a) noise variance; (b) $\lambda$. ................................................................. 28

3.7 Minimization of $E_s(\lambda_1 = 0, \lambda_2 = 1)$: In each case, the left panel shows the initial path (first row) and the final path (second row), the right panel shows the corresponding decrease in $E_s$ during evolution. ......................................................... 30
3.8 Minimization of $E_d(\lambda_1 = 1, \lambda_2 = 0)$: Each case shows the given data (first row), the initial path (second row) and the final path (third row), along with the evolution of $E_d$. 

3.9 Optimization of $E$ under different combinations of $\lambda_1$ and $\lambda_2$. In all these cases we use the same data points $(t_i, p_i)$ and the same initial path $\alpha$ for starting the algorithm. 

3.10 Smoothing splines of SPD data ($\lambda_1 = \lambda_2 = 1$). 

3.11 Minimization of $E_s(\lambda_1 = 0, \lambda_2 = 1)$: In each case, the left panel shows the initial path (top row) and the final path (bottom row), the right panel shows the corresponding decrease in $E_s$ during evolution. 

3.12 Minimization of $E_d(\lambda_1 = 1, \lambda_2 = 0)$. In each case, the left panel shows the given data (top row), the initial path (middle row) and the final path (bottom row), along with the evolution of $E_d$ on the right. 

3.13 Optimization of $E$ under different combinations of $\lambda_1$ and $\lambda_2$. In all these cases we use the same data points $(t_i, p_i)$ and the same initial path $\alpha$ for the gradient search. 

3.14 Optimization of $E$ under $\lambda_1 = \lambda_2 = 1$. In each case, the top row shows the initial path and the bottom row shows the final path, along with $E$ on the right. The shapes in the initial paths at time-indexed 1, 4, 7, and 10 form the data points. 

3.15 Comparison of four different methods on Kendall’s shape space: Case 1. 

3.16 Comparison of four different methods on Kendall’s shape space: Case 2. 

3.17 Estimation of $\hat{\lambda}$ using the cross-validation approach. Left: The optimal path under $\hat{\lambda}$. Right: Cost function for automatically estimating $\hat{\lambda}$. 

4.1 Registration of trajectories on $S^2$. 

4.2 Comparison of clustering between without and with registration. 

4.3 Average of trajectories on $S^2$. 

4.4 Swainson's Hawk migration: (a) $\{\alpha_i\}$ (top) and $\{\gamma_i^*\}$ (bottom); (b) $\mu$ and $\hat{\rho}$ without registration; (c) $\mu$ and $\hat{\rho}$ with registration. 

4.5 Comparison of $\hat{\rho}$ (1st row) and $p$-values (2nd row) without (red) and with (blue) registration. 

4.6 Summary of hurricane tracks without and with temporal registration.
4.7 (a): Real trajectories on SE(2) obtained from a traffic video. (b): Trajectories used for clustering. ................................................................. 54
4.8 Registration of trajectories on SE(2). ........................................ 54
4.9 Comparison of clustering without and with alignment. .............. 55
4.10 Registration of SPD trajectories. ............................................. 56
4.11 Data acquisition: (a) projection pathways; (b) and (c) 12 fibers and corresponding SPD trajectories. ............................................. 57
4.12 Summary of fibers: (a) and (b): Karcher mean computed in two different ways; (c): \( \hat{\rho} \). ................................................................. 57
4.13 Registration of two trajectories on shape space of planar contours. ................................................................. 59
4.14 Registration and summary of multiple trajectories. ..................... 60
4.15 Left: \( D = wD_1 + (1 - w)D_2 \) for \( w = 0.98 \). Middle: Measure of asymmetry in \( \tilde{D} \). Right: Classification rate versus the number of nearest neighbors. ................................. 60
5.1 Geodesic between unit length geodesics on \( S^2 \). .................... 65
5.2 Geodesic between arbitrary length geodesics on \( S^2 \). ................ 65
5.3 Geodesic between arbitrary trajectories on \( S^2 \). ....................... 66
5.4 Comparison of geodesics between arbitrary trajectories without and with registration. ................................. 66
6.1 Examples of 2D and 3D cluttered point clouds. .......................... 71
6.2 Likelihood maps \( \alpha_\sigma \) for some curves. The top row shows the case for the same curve but different \( \sigma \)’s and the bottom row shows different curves but with the same \( \sigma \). ......................................................... 74
6.3 (a): A curve \( \beta \), (b): simulated point cloud \( Y \), (c): the evolution of \( H_\sigma \) over \( (\gamma, \lambda) \) versus iteration index for a fixed \( \sigma \), and (d) \( H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma) \) versus \( \sigma \). ......................................................... 77
6.4 A point cloud, two hypothesized curves, \( \alpha_\sigma \) profiles, and \( \alpha_\sigma \) at \( y_i \)’s for the two curves. ......................................................... 77
6.5 Maximum likelihood estimation of shape transformations from given point clouds (black points) and starting curves (red). Left: rotation, middle: translation, right: scale. The top row shows the evolution of shapes while the bottom row plots the evolution of \( H \). ................................................................. 79
6.6 Point clouds (red) and estimated curves (blue), $R(Y) = -90.4$ (first) and $-96.0$ (second) strongly support $H_1$ ($\alpha = 0.05$ and $\mu = -3.8$). .......................... 80

6.7 Type II error probability versus $\sigma$ and $r$. ............................................. 80

6.8 Detection of a fixed face surface. First: face surface $\beta$, second: point cloud, third: point cloud (red from the surface and blue from the clutter) and $\beta$, fourth: $H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma)$ versus $\sigma$. .................................................. 81

6.9 Detection of a horse surface at unknown transformation. First: horse surface $\beta$, second: point cloud $Y$, third: $Y$ and the surface at the estimated position, rotation, and scale, fourth: $H$ versus iteration. ........................................... 83

6.10 2D shape classification. Left: 10 representing shapes. Right: comparison of 2D classification versus $k$ for different $\sigma$s. ................................. 84

6.11 Pre-processing of 2D images to obtain point clouds. ................................. 85

6.12 Shape classification in 2D point clouds from real images. In each example, the first column is the real image, the second column is the generated point cloud and the third column is the LLRs for 10 classes. ................................. 85

6.13 Database of 20 classes for 3D shape classification. A representative from each class forms the test set and the remaining 80 shapes form the training set. ............... 86

6.14 3D shape classification. Left panel: one example of 3D classification (1st column: chess template, 2nd column: point cloud, 3rd and 4th columns: two surfaces with the smallest LLRs from the training set). Right panel: comparison of 3D classification using different methods. .................................... 87

6.15 Examples of 2D shape estimation (black: point clouds, red: estimated shapes). ... 88

6.16 Shape estimation from 3D point clouds. In each example, the left shows the point cloud and the right shows the cloud with the estimated shape at the estimated pose and scale. .................................................. 89

6.17 Examples of partial estimation. Left: simulated point cloud, middle: estimated template in the cloud, right: estimated shape. .................................................. 89
ABSTRACT

This thesis consists of two distinct topics. First, we present a framework for estimation and analysis of trajectories on Riemannian manifolds. Second, we propose a framework of detecting, classifying, and estimating shapes in point cloud data.

This thesis mainly focuses on statistical analysis of trajectories that take values on nonlinear manifolds. There are many difficulties when analyzing temporal trajectories on nonlinear manifold. First, the observed data are always noisy and discrete at unsynchronized times. Second, trajectories are observed under arbitrary temporal evolutions. In this work, we first address the problem of estimating full smooth trajectories on nonlinear manifolds using only a set of time-indexed points, for use in interpolation, smoothing, and prediction of dynamic systems. Furthermore, we study statistical analysis of trajectories that are observed under arbitrary temporal evolutions. The problem of analyzing such temporal trajectories including registration, comparison, modeling and evaluation exist in a lot of applications. We introduce a quantity that provides both a cost function for temporal registration and a proper distance for comparison of trajectories. This distance, in turn, is used to define statistical summaries, such as the sample means and covariances, of given trajectories and Gaussian-type models to capture the variability. Both theoretical proofs and experimental results are provided to validate our work.

The problems of detecting, classifying, and estimating shapes in point cloud data are important due to their general applicability in image analysis, computer vision, and graphics. They are challenging because the data is typically noisy, cluttered, and unordered. We study these problems using a fully statistical model where the data is modeled using a Poisson process on the objects boundary (curves or surfaces), corrupted by additive noise and a clutter process. Using likelihood functions dictated by the model, we develop a generalized likelihood ratio test for detecting a shape in a point cloud. Additionally, we develop a procedure for estimating most likely shapes in observed point clouds under given shape hypotheses. We demonstrate this framework using examples of 2D and 3D shape detection and estimation in both real and simulated data, and a usage of this framework in shape retrieval from a 3D shape database.
CHAPTER 1

INTRODUCTION

This thesis considers the problems where features of interest evolve on nonlinear manifolds. Manifold-valued responses in curved spaces frequently arise in many disciplines including computer vision, medical imaging, computational biology, among many others. Many problems in these areas are naturally posed as problems of optimization or statistical inference on nonlinear manifolds. This is because there are some intrinsic constraints on the pertinent features that force the corresponding representations to these manifolds. Time series data is gaining importance in many applications and poses new challenges. Whereas historically the spatial aspect of data was analyzed, for instance to build atlases, we now need to consider both the spatial and the temporal aspects together. Trajectories are natural spatiotemporal data descriptors. The aim of this thesis is to develop a general framework for estimation and analysis of manifold-valued trajectories.

1.1 Problem Statement

There are two subproblems. First, the observed data are always noisy and discrete at unsynchronized times. We start by discussing the problem of estimating full trajectories on certain nonlinear manifolds using only a set of time-indexed points, for use in interpolation, smoothing, and prediction of dynamic systems. These trajectories are analogous to smooth splines on Euclidean spaces as they are optimal under a similar objective function, which is a weighted sum of a fitting-related (data term) and a regularity-related (smoothing term) cost functions. The first goal of our work can be stated as follows:

**Estimation of Full Trajectories:** Let $M$ be a finite-dimensional Riemannian manifold. Given a set of observation times $t_1, t_2, \ldots, t_n$ in $[0, 1]$ and the measurements $p_1, p_2, \ldots, p_n \in M$ at those times, our goal is to estimate a smooth trajectory $\alpha : [0, 1] \rightarrow M$ under a specific optimization criterion.

Second, we are interested in statistical analysis of trajectories that are observed under arbitrary temporal evolutions. The problem can be described in mathematical terms:
Analysis of Temporal Trajectories: We will study mappings of the type \( \alpha : [0, 1] \rightarrow M \), where \( M \) is a nonlinear manifold. We will call such \( \alpha \)'s trajectories and study them as elements of an appropriate subset of \( M^{[0,1]} \). We are especially interested in studying trajectories that are observed at arbitrary evolution rates. In other words, rather than observing a trajectory \( \alpha \) directly, say in the form of samples \( \alpha(t_1), \alpha(t_2), \ldots \), we instead observe the trajectory \( \alpha(\gamma(t_1)), \alpha(\gamma(t_2)), \ldots \). Here \( \gamma : [0, 1] \rightarrow [0, 1] \) is an unknown temporal evolution function (a function with certain constraints described later) that governs the rate of evolution.

The goal is to perform joint registration and comparison of trajectories. Based on that, we would like to compute summary statistics and build statistical modeling of temporal trajectories on Riemannian manifolds. These tasks can be mathematically stated as follows:

1. **Temporal Registration**: This is a process of establishing a one-to-one correspondence between points along multiple trajectories. That is, given any \( n \) trajectories, say \( \alpha_1, \alpha_2, \ldots, \alpha_n \), we are interested in finding functions \( \gamma_1, \gamma_2, \ldots, \gamma_n \) such that the points \( \alpha_i(\gamma_i(t)) \) are matched optimally for all \( t \in [0, 1] \).

2. **Metric Comparison**: We want to develop a metric that is invariant to different evolution rates of trajectories. Specifically, we want to define a distance \( d_p(\cdot, \cdot) \) such that for arbitrary evolution functions \( \gamma_1, \gamma_2 \) and arbitrary trajectories \( \alpha_1, \alpha_2 \), we have \( d_p(\alpha_1, \alpha_2) = d_p(\alpha_1 \circ \gamma_1, \alpha_2 \circ \gamma_2) \).

3. **Statistical Summaries**: The main use of this metric will be in defining and computing a (Karcher) mean trajectory \( \mu \) and a cross-sectional variance function \( \rho \), associated with any given set of trajectories. The main reason for performing registration is to reduce the cross-sectional variance that is artificially introduced in the data due to random observation times. This reduced variance is measured using \( \rho \).

4. **Statistical Modeling and Evaluation**: We will use the estimated mean and covariance of points along registered trajectories to define a “Gaussian-type” model on random trajectories. This model will then be used to evaluate \( p \)-values associated with new trajectories.

### 1.2 Motivation

First, we are interested in dynamical systems where an event of interest evolves over time on a nonlinear manifold and one observes this process only at limited times. The goal is to estimate/predict the remaining process using the observed values under some pre-determined criterion. The motivation of such a problem comes from many applications. Consider the phenomenon of bird
migration which is the regular seasonal journey undertaken by many species of birds. Fig. 1.1(a) shows geographic coordinates of Swainson’s Hawk migration observed at discrete times in 1995. Fitting a smooth spline becomes important for analyzing such noisy data. As another example, consider the evolution of the shape of a human silhouette in a video for activity recognition, in a situation where one has an unobstructed view of the person in only a few frames. Given these observed shapes, along with their observation times, one would like to estimate shapes at some intermediate times and perhaps even predict future shape evolution. Additionally, if the observed shapes are considered noisy (for example, due to the process of extracting silhouettes from image frames), one would like to account for the observation noise using the temporal information. A similar problem arises in tracking the rigid motion of an object using video data, e.g. in tracking the facial pose of a speaker using webcam sensing. Here one observes the face orientation at certain times and seeks to track/estimate it over the whole observation interval.

The need to summarize and model trajectories arises from many statistical procedures, especially those involving analysis of variance. An important issue in this analysis is that trajectories are often not observed at standard times and, in fact, at random times. If this temporal variability is not accounted for in the analysis, then the resulting statistical summaries will not be precise. The mean trajectory may not be a representative of individual trajectories and the cross-sectional variance will be artificially inflated. This, in turn, will greatly reduce the effectiveness of any subsequent modeling or analysis based on estimated mean and covariance. As a simple example consider the

Figure 1.1: Examples of motivation for trajectory estimation: (a) bird migration; (b) human activity recognition.
trajectory on $S^2$ shown in the top panel of Fig. 1.2(a). We simulate a set of random, discrete observation times and generate observations of this trajectory at these random times. These simulated trajectories are identical in terms of the points traversed but their evolutions, or parameterizations are quite different. If we compute cross-sectional mean and variance, the results are shown in the bottom panel. We draw the sample mean trajectory in black and the sample variance at discrete times using tangential ellipses. Not only is the mean fairly different from the original curve, the variance is purely due to randomness in observation times and is somewhat artificial. If we have observed the trajectory at fixed, synchronized times, then this problem will not exist.

To motivate this issue further, consider the phenomenon of bird migration which is the regular seasonal journey undertaken by many species of birds. There are variabilities in migration trajectories, even within the same species, including the variability in their rates of travels. In other words, either birds can travel along different paths or, even if they travel the same path, different birds (or subgroups) can fly at different speed patterns along the path. This results in variability in observation times of migration paths for different birds and artificially inflates the cross-sectional variance in the data. Another issue is that such trajectories are naturally studied as paths on a unit sphere which is a nonlinear manifold. We will study the migration data for Swainson’s Hawk, with some example paths shown in the top of Fig. 1.2(b). Swainson’s Hawk inhabits North America mainly in the spring and summer and winters in South America. It is probably the longest migrant of any North American raptor where each migration can last at least two months. [1] discovered that Swainson’s Hawk in migratory disposition exhibits reduced immune system. Therefore, it becomes important
to investigate and summarize such travels. The bottom panel in Fig. 1.2(b) shows cross-sectional sample mean and variance of the trajectories.

Another motivating application comes from hurricane tracking, where one is interested in studying the shapes of hurricane tracks in certain geographical regions. The statistical summaries and models of Hurricane tracks can prove very useful for monitoring and issuing warnings. Hurricanes potentially evolve at variable dynamical rates and any shape analysis of these tracks should be invariant of the evolution rates. As in the previous application, the hurricane tracks are also naturally treated as trajectories on a unit sphere. The top panel of Fig. 1.2(c) shows a set of hurricane tracks originating from the Atlantic region. The sample mean and the variance of these trajectories are adversely affected by this phase variability present in data, as shown in the bottom row of Fig. 1.2(c).

As the last motivating example, consider two trajectories, drawn in red and blue in the top of Fig. 1.2(d) These two trajectories have the same shape, i.e. two bumps each, and a curve representing their mean is also expected to have two bumps. A simple cross-sectional mean, shown by the black trajectory in the same picture, has three bumps! If we solve for the optimal temporal alignment, then such inconsistencies are avoided and the black trajectory in bottom panel shows the mean obtained using the method proposed in this paper, which accounts for the time-warping variability.

1.3 Problem Description

The most related work with our framework of estimating smooth trajectory is called smoothing spline \[2\] in Euclidean spaces. It is a method of smoothing (fitting a smooth curve to a set of noisy observations) using a spline function. Let \(\{(t_i, p_i); t_1 < t_2 < \cdots < t_n, i \in \mathbb{Z}\}\) be a sequence of observations, modeled by the relation \(p_i = \alpha(t_i)\). The smoothing spline estimate \(\hat{\alpha}\) of the function \(\alpha\) is defined to be the minimizer (over the class of twice differentiable functions) of \(\lambda_1 \sum_{i=1}^{n} (p_i - \alpha(t_i))^2 + \lambda_2 \int_{t_1}^{t_n} \dddot{\alpha}(t)^2 dt\). \(\lambda_1, \lambda_2 \geq 0\) are two parameters, controlling the trade-off between fidelity to the data and roughness of the function estimate. As \(\lambda_2/\lambda_1 \to 0\) (no smoothing), the smoothing spline converges to the interpolating spline. As \(\lambda_2/\lambda_1 \to \infty\) (infinite smoothing), the roughness penalty becomes paramount and the estimate converges to a linear least squares estimate. The roughness penalty based on the second derivative is the most common in modern statistics literature, although the method can easily be adapted to penalties based on other derivatives. An example of smoothing spline in \(\mathbb{R}^2\) is shown in Fig. 1.3(a).

For nonlinear manifolds, an important issue in the problem of fitting smooth trajectories is: What should be the criterion for this estimation? A simple solution is to connect the given points with shortest (geodesic) paths on the manifold, say \(M\). In other words, interpolate linearly between the given observations using piecewise geodesics. If \(M\) is a Riemannian manifold, it has a Riemannian metric to define and compute geodesic paths between arbitrary pairs of points. This estimation
is illustrated using a solid line in Fig. 1.3(b). As depicted, this estimate has discontinuities in the first derivative at the observation times, which does not often match the actual underlying process. If the given points are considered noisy, one does not require the estimated trajectory to strictly pass through them, i.e., the given points are treated as soft anchors rather than hard constraints. This argument leads to an optimization problem of following type. Let \( \{(t_1, p_1), (t_2, p_2), \ldots, (t_n, p_n)\} \) denote the given observations where \( t_i \in [0, 1] \) and \( p_i \in M \), and we are interested in a trajectory \( \hat{\alpha} \) such that

\[
\hat{\alpha} = \arg\min_{\alpha: [0, 1] \to M} \left( \frac{\lambda_1}{2} \sum_{i=1}^{n} d(\alpha(t_i), p_i)^2 + \frac{\lambda_2}{2} E_s(\alpha) \right).
\] (1.1)

The first term is denoted as the data term \( E_d = \sum_{i=1}^{n} d(\alpha(t_i), p_i)^2 \). \( E_s \) is a smoothness penalty on \( \alpha \) and \( d \) denotes the (Riemannian) geodesic distance on \( M \). \( \lambda = \lambda_2 / \lambda_1 \) denotes the ratio of two weights. The next issue is to select the smoothness penalty. There are two main ideas in this regard:

- Use a penalty that relates to the first derivative of \( \alpha \):

\[
E_{s,1}(\alpha) = \int_0^1 \langle \dot{\alpha}(t), \dot{\alpha}(t) \rangle \, dt.
\] (1.2)

Since \( \dot{\alpha}(t) \in T_{\alpha(t)}(M) \), the inner product inside the integral can be defined using the Riemannian metric. It can be shown that the solution of Eqn. 1.1 under this smoothness penalty are piecewise geodesics although they are not required to pass through the given points. This choice suffers from the same problem as the piecewise geodesic interpolations between \( p_i \)s, i.e. the discontinuity of the function \( \dot{\alpha}(t) \) at some points.
Another idea is to penalize the second derivative of $\alpha$ according to:

$$E_{s,2}(\alpha) = \int_0^1 \left\langle \frac{D^2\alpha}{dt^2}, \frac{D^2\alpha}{dt^2} \right\rangle dt.$$  \hspace{1cm} (1.3)

Here $\frac{D^2\alpha}{dt^2} = \frac{D}{dt}\dot{\alpha}(t)$ denotes the covariant derivative of the tangent vector field $\dot{\alpha}(t)$ and, once again, the inner product inside the integral is given by the Riemannian metric. Under this choice of $E_s$, the solution of Eqn. 1.1 will be smooth up to the second order and will provide more natural solutions than the previous choice. Such a solution is depicted using the broken line in Fig. 1.3(b).

Furthermore, we would like to take temporal variability into account, derive rate invariant metrics, generate statistical summaries and modeling. It is desirable that the proposed framework performs joint temporal registration and comparison.

For performing registration of trajectories, a majority of the existing methods/algorithms in Euclidean spaces formulate an objective function of the type:

$$\min_{\gamma} \left( \int_0^1 \|\alpha_1(t) - \alpha_2(\gamma(t))\|^2 dt + \lambda R(\gamma) \right),$$  \hspace{1cm} (1.4)

where $\| \cdot \|$ is the Euclidean norm, $R$ is a regularization term on the warping function $\gamma$, and $\lambda > 0$ is a constant. In case of a Riemannian manifold, one can modify the first term in Eqn. 1.4 to obtain:

$$\min_{\gamma} \left( \int_0^1 d(\alpha_1(t), \alpha_2(\gamma(t)))^2 dt + \lambda R(\gamma) \right),$$  \hspace{1cm} (1.5)

The main problem with this procedure is that: (a) it is not symmetric, i.e. the registration of $\alpha_1$ to $\alpha_2$ is not the same as that of $\alpha_2$ to $\alpha_1$, as pointed out by [3] and others, and (b) the minimum value is not a proper distance, so it cannot be used to compare trajectories.

For performing comparison and summarization of trajectories, we need a metric and, at first, we consider a more conventional solution. Since $M$ is a Riemannian manifold, using the geodesic distance between points on $M$, one can define a quantity between any two trajectories: $\alpha_1, \alpha_2 : [0,1] \to M$, as

$$d_x(\alpha_1, \alpha_2) = \int_0^1 d(\alpha_1(t), \alpha_2(t)) dt$$  \hspace{1cm} (1.6)

and use it for comparing trajectories. Although this quantity represents a natural extension of $d$ from $M$ to $M^{[0,1]}$, it suffers from the problem that $d_x(\alpha_1, \alpha_2) \neq d_x(\alpha_1 \circ \gamma_1, \alpha_2 \circ \gamma_2)$ in general. It is not preserved even when the same $\gamma$ is applied to both the trajectories, i.e. $d_x(\alpha_1, \alpha_2) \neq d_x(\alpha_1 \circ \gamma, \alpha_2 \circ \gamma)$ generally. This sums up the fundamental dilemma in trajectory analysis – Eqn. 1.6 provides a metric between trajectories but does not perform registration and Eqn. 1.5 performs
a registration but is not a metric! If we had an equality in the last case, for all $\gamma$'s, then one can develop a fully invariant distance and use it to properly register trajectories, as described later. So, the failure to have this equality is in fact a key issue that forces us to look for other solutions in situations where trajectories are observed at random temporal evolutions.

While the nonlinearities of different manifolds can be very diverse, certain types of manifolds are encountered more frequently. As an illustration, we discuss four of them and their applications as follows.

1. **Unit Spheres**: A simple constraint of unit length leads to a unit sphere. In case the vector representing a set of features in a vision system is scaled to have norm one, the set of such vectors is naturally a unit sphere. For example, in the study of directional data ([4]), where one is interested in analyzing the directions of moving objects, or imaging viewpoints, the space of all directions is $S^2$. In the landmark-based shape analysis of objects ([5]), where 2D objects are represented by configurations of salient points or landmarks, the set of all such configurations, after removing translation and scale is a real sphere $S^{2n-3}$ (for configurations with $n$ landmarks). Similarly, in an elastic shape analysis of curves on Euclidean spaces ([6]), the space of all unit length curves is a unit Hilbert sphere $S^\infty$. A number of applications involve functions that warp domains on interest to themselves in a diffeomorphic way. For example, in the problem of activity recognition, the sequence of events that form an activity may be performed at the same sequence but at different rates ([7]). A closely related problem is that of analyzing probability density functions because the derivatives of warping functions in one dimension are probability densities (or histograms, that are frequently used as features in texture analysis). It was illustrated in [8] that a natural way to study the warping functions in one dimension, the probability density functions, and the histograms is to use a square-root representation under the Fisher-Rao metric. The resulting set of such functions forms a Hilbert sphere $S^\infty$.

2. **Matrix Lie Groups**: The transformation groups represented by $GL(n)$, the set of $n \times n$ invertible matrices, and its subgroups are important due to their actions on Euclidean spaces. For example, for $x \in \mathbb{R}^n$, a transformation of the type $x \mapsto (Ax + b)$ where $b \in \mathbb{R}^n$, and (i) if $A \in GL(n)$ is called an *affine* transformation, (ii) if $A \in SL(n)$, the subgroup of $GL(n)$ consisting of matrices with determinant $+1$, is called a *volume-preserving* transformation, and (iii) if $A = aO$ where $a \in \mathbb{R}_+$ and $O \in SO(n)$, the set of $n \times n$ orthogonal matrices with determinant $+1$, the resulting transformation is called a *similarity* transformation. In the problem of tracking and recognizing objects in video data, their poses relative to the camera are important. The pose of a rigid object is conveniently denoted as an element of the special orthogonal group $SO(n)$ where $n = 2$ for planar objects and $n = 3$ for 3D objects ([9, 10]). The rotation matrices are preferred over the angles (e.g. pitch, yaw, and role for 3D pose) for
representing pose due to their uniqueness in representing pose.

3. **Quotient Spaces of Spheres**: Some problems involve the quotient spaces of spheres rather than the spheres themselves. For example, the landmark shape space of 2D configurations, after removing the rotation variability, becomes complex projective space \( \mathbb{CP}^{n-2} = \mathbb{S}^{2n-3}/\mathbb{S}^1 \) ([11, 12, 5]). In an elastic shape analysis of curves, the space of all unit length curves is a unit Hilbert sphere \( \mathbb{S}_\infty \), and after removing the rotation and re-parameterization variability one obtains a quotient space \( \mathbb{S}_\infty/(\text{SO}(2) \times \Gamma) \), where \( \Gamma \) is the re-parameterization group ([6]).

4. **Quotient Spaces of Matrix Lie Groups**: Perhaps the most prominent examples of this category are Grassmann and Stiefel manifolds that are useful in studying orthogonal linear transformations, including dimension reduction. Depending on the chosen criterion, one can generalize the problem of linear dimension reduction to include Fisher’s linear discriminant analysis ([13]) and optimal component analysis ([14]). Such problems are solved on Stiefel and/or Grassmann manifolds which, in turn, are quotient spaces of \( \text{SO}(n) \). In a number of applications, there has been a great interest in statistical analysis of \( n \times n \) symmetric, positive-definite (SPD) matrices \( \text{Sym}^+(n) \), which can be viewed as a quotient space of \( \text{GL}(n) \). For example, the basic unit in Diffusion Tensor - Magnetic Resonance Imaging (DT-MRI) is a \( \text{Sym}^+(3) \) matrix that represents the diffusion tensor at each point of a volume. The goal in this problem is to estimate, interpolate, and smooth a uniform field of diffusion tensors for further use in Tractography, and these tasks are performed using the Riemannian geometry of the underlying tensor space (see [15, 16, 17, 18]). Another application of this Riemannian framework is in analysis of Gaussian probability densities. Restricting to densities with mean zero, one can characterize them by their covariance matrices that are by definition \( \text{Sym}^+(n) \) ([19]).

### 1.4 Contributions

We begin by discussing the problem of fitting smoothing splines to time-indexed data points on a finite-dimensional Riemannian manifold \( M \) by means of a Palais-based gradient method. The first contribution is the particularization of this method to specific manifolds. We have described the relevant parts of their differential geometries. As a result we have derived the gradients of the cost function on these manifolds and have applied the gradient algorithm for finding smoothing splines for a variety of real and simulated data examples.

Next, we develop a framework for joint registration and comparison of trajectories, and obtain improvements in statistical summaries of time-warped trajectories on Riemannian manifolds. This framework is based on a novel mathematical representation called transported square-root vector field (TSRVF) and the \( L^2 \) norm between TSRVFs. This setup satisfies the invariance property
mentioned earlier, i.e. an identical time-warping of TSRVFs representing two trajectories preserves the $L^2$ norm of their difference and, therefore, this difference is used to define a warping-invariant distance between trajectories. The resulting distance is found useful in registration, comparison and summarization of trajectories on manifolds.

In the end, we present a fully statistical framework for detecting, classifying and estimating shapes in cluttered point cloud. This framework is based on a composite Poisson process: one for points generated from the shape and another for points belonging to the background clutter. This model allows computation of a log-likelihood ratio for each class against clutter and this ratio leads to a formal procedure for detection and classification of shapes. Furthermore, we estimate the shape from the cloud based on this framework and apply it to both 2D and 3D cases.
CHAPTER 2

LITERATURE REVIEW

In this chapter, we summarize previous work on two related but separate tasks: estimation and analysis of trajectories.

2.1 Estimation of Full Trajectories

We are interested in the problem of fitting smooth curves to given finite sets of points on Riemannian manifolds. Let \( p_1, p_2, \ldots, p_n \) be a finite set of points on a Riemannian manifold \( M \), and let \( 0 = t_1, t_2, \ldots, t_n = 1 \) be distinct and ordered instants of time. The problem of fitting a smooth curve \( \alpha \) on \( M \) to the given points at the given times involves two goals of a conflicting nature. The first goal is that the curve should fit the data as well as possible, as measured, e.g., by the real-valued function \( E_d \) defined by

\[
E_d = \sum_{i=1}^{n} d(\alpha(t_i), p_i)^2.
\]  

The second goal is that the curve should be sufficiently regular, as measured by a function \( \alpha \mapsto E_s(\alpha) \) such as Eqn. 1.2 and 1.3. We are thus facing an optimization problem with two objective functions - a fitting function \( E_d \) and a regularity function \( E_s \) - whose domain is a suitable set of curves on the Riemannian manifold \( M \).

One possible way of tackling an optimization problem with two objective functions is to turn it into a classical optimization problem where one of the objective functions becomes the objective function and the other one is turned into a constraint.

Let us first discuss the case where the fitting objective function \( E_d \) is minimized under a regularity constraint. When \( M = \mathbb{R}^n \), a classical regularity constraint is to restrict the curve \( \alpha \) to the family of polynomial functions of degree not exceeding \( m \) (\( m \leq n \)). This least-squares problem cannot be straightforwardly generalized to an arbitrary Riemannian manifold \( M \) because the notion of polynomial does not carry over to \( M \) in an obvious way. An exception is the case \( m = 1 \); the polynomial functions in \( \mathbb{R}^n \) are then straight lines, whose natural generalization on Riemannian
manifolds are geodesics. The problem of fitting a geodesic to data on Riemannian manifold $M$ was considered in [20] for the case where $M$ is the special orthogonal group $SO(n)$ or the unit sphere $S^n$.

The other case is when a regularity criterion $E_s$ is optimized under a constraint on $E_d$, in which case it is natural to impose the interpolation constraint $E_d(\alpha) = 0$. For example, when $M = \mathbb{R}^n$, minimizing the function $E_{s,1}(\alpha) = \int_0^1 \| \dot{\alpha}(t) \|^2 dt$ yields the piecewise-linear interpolant for the given data points and time instants (this follows from [21]), while minimizing $E_{s,2}(\alpha) = \int_0^1 \| \ddot{\alpha}(t) \|^2 dt$ yields solutions known as cubic splines. For the case where $M$ is a nonlinear manifold, several results on interpolation can be found in the literature. The generalization of cubic splines to more general Riemannian manifolds was pioneered by [22]. Cubic splines are then defined as curves that minimize the function in Eqn. 1.3. A necessary condition for optimality takes the form of a fourth-order differential equation. This variational approach was followed by other authors, including [23, 24]. Splines of class $C^k$ were generalized to Riemannian manifolds by [25]. Still in the context of interpolation on manifolds, but without a variational interpretation, we mention the literature on splines based on generalized Bézier curves, defined by a generalization to manifolds of the de Casteljau algorithm; see [26, 27, 28]. Recently, [29] presented a geometric two-step algorithm to generate splines of an arbitrary degree of smoothness on Euclidean spaces, then extended the algorithm to matrix Lie groups and applied it to generate smooth motions of 3D objects. [30] studied the affine structure of domain manifolds in depth and proved that the existence of manifold splines is equivalent to the existence of a manifold’s affine atlas. A set of practical algorithms was developed to generalize triangular B-spline surfaces from planar domains to manifold domains. Another approach to interpolation on manifolds consists of mapping the data points onto a vector space, mostly the tangent space at a particular point of manifold, then computing an interpolating curve on the tangent space, and finally mapping the resulting curve back to the manifold. The mapping can be defined, e.g., by a rolling procedure, see [31, 32, 33, 34, 35, 36]. For instance, [31] proposed a method that can be described as ”unwrapping” the data onto the plane, where standard curve fitting techniques can then be applied. The papers [32, 36, 33] studied the problem of fitting a smooth curve on the planar shape space.

Another way of tackling an optimization problem with two objective functions is to optimize a weighted sum of the objective functions. Spherical smoothing splines on the two-dimensional unit sphere were originally studied by [31]. This approach was followed on general manifolds by [37] using the first-order smoothing term in Eqn. 1.2 and by [20] for the second-order smoothing term in Eqn. 1.3. Specifically, in [37], the objective function is defined to be Eqn. 1.1 with $E_{s,1}$, over the class of all piecewise smooth curves $\alpha : [0, 1] \rightarrow M$. Solutions to this variational problem are piecewise geodesics that best fit the given data. As shown in [37], when $\lambda$ goes to $\infty$, the optimal curve converges to a single point which is shown in [38] to be the Riemannian mean of the data points. When $\lambda$ goes to zero, the optimal curve goes to a broken geodesic on $M$ interpolating the
data points. In [20], the objective function is defined to be Eqn. 1.1 with $E_{s,2}$ over a certain set of admissible $C^2$ curves. The authors give a necessary condition of optimality that takes the form of a fourth-order differential equation involving the covariant derivative and the curvature tensor along with certain regularity conditions at the time instants. The optimal curves are approximating cubic splines: they are approximating because in general $\alpha(t_i)$ differs from $p_i$, and they are cubic splines because they are obtained by smoothly piecing together segments of cubic polynomials on $M$, where cubic polynomial on $M$ is understood in the sense of [22]. It is also shown in [20] that, as the smoothing parameter $\lambda$ goes to $\infty$, the optimal curves converge to a geodesic curve on $M$ fitting the given data points at the given instants of time. When $\lambda$ goes to zero, the approximating cubic spline converges to an interpolating cubic spline.

The next category of papers presents interpolating curves between given points, with a goal of generating smooth rigid-body motion. [39] used Bézier curves to interpolate between points representing 3D motion. [40] defined minimum acceleration and minimum jerk curves between given pairs of points on $SE(3)$ to perform smooth interpolation of motion. [41] utilized an intrinsic scheme for computing averages on spheres and used that framework to compute interpolating splines between given points. Since these methods require curves to pass through the given points, they do not account for noise in the data.

The last set of papers takes a filtering or a smoothing approach to path fitting at discrete times, using explicit models for the unknown process and the observation process. While such problems on Euclidean spaces are solved using Kalman filters, extended Kalman filters, and particle filters, the corresponding problems on manifolds are solved mostly using particle filtering. For example, [42] considered state estimation for partially observed processes evolving in a Riemannian manifold and obtained some new results on un-normalized nonlinear filters. [43] studied the problem of estimating trajectories on Grassmann manifolds using a particle filtering approach. [44] considered a similar problem on a Stiefel manifold. [45] formulated a particle filtering algorithm in the geometric active contour framework that can be used for tracking moving and deforming objects. [46, 47, 48] developed solutions to the problem of tracking landmark-based shapes in video data. These solutions assume that one has an observation, albeit nonlinear and noisy, at each discrete time of interest in the interval and in this sense differ from the current problem where only a few measurements are given.

Estimation of trajectories can be interpreted as the problem of regression analysis. There are limited studies on the regression analyses of manifold-valued data. Even for the ‘simplest’ directional data, there is a sparse literature on regression modeling of a single directional response and a set of covariates ([4, 49]). In addition, these regression models of directional data are primarily based on a specific distribution, such as the von Mises-Fisher distribution ([50, 4, 51]). However, it can be challenging to assume parametric distributions for general manifold-valued data, and thus it is difficult to generalize these regression models of directional data to general manifold-valued data.
data. Regression analysis on the group of diffeomorphisms has been proposed as growth models by [52], and nonparametric regression by [53]. Recently, [54, 55] have proposed a semiparametric intrinsic model with multiple covariates for manifold-valued response data in Sym+\((n)\) and \(\mathbb{S}^2\). [56] and [57] have each independently developed geodesic regression which generalizes the notion of linear regression to Riemannian manifolds.

### 2.2 Analysis of Temporal Trajectories

Our goal is to take time-warping into account, derive a warping-invariant metric, and generate statistical summaries (sample mean, covariance, etc.) for trajectories on a set \(M\). The fact that \(M\) is a Riemannian manifold presents a formidable challenge in developing a comprehensive framework. But this is not the only challenge. To clarify this part, let us consider the question: How has this registration and analysis problem been handled for trajectories in Euclidean spaces?

- **Previous work on Euclidean spaces**
  
  In case \(M = \mathbb{R}\), i.e. one is interested in registration and modeling of real-valued functions under random time-warpings, the solution has been studied by many authors, including [58, 59, 60, 61].

  In case \(M = \mathbb{R}^2\), where the problem involves registration and shape analysis of planar curves, the solution is discussed in [62, 63, 64, 65]. [6] proposed a solution that applies to curves in arbitrary \(\mathbb{R}^n\). One can also draw an inspiration from problems in image registration where 2D and 3D images are registered to each other using spatial warping instead of a temporal warping (see e.g. LDDMM technique [66]).

- **Previous work on nonlinear manifolds**

  Although there has been progress in removal of this temporal variability, often termed the *phase variability*, in Euclidean spaces ([58, 67, 68]), there has not been any treatment dealing with trajectories on Riemannian manifolds. There are many other applications involving analysis of trajectories on Riemannian manifolds [69, 70, 71]. Take the problem of human activity recognition that has attracted a tremendous interest in recent years because of its potential in applications such as surveillance, security, and human body animation. There are several survey articles [72, 73, 74] that provide a detailed review of research in this area. Each observed activity is represented by a sequence of silhouettes in video frames, each silhouette being an element of the shape space of planar contours. The shape sequences have also been called shape curves or curves on shape spaces [75, 76]. Since activities can be performed at different execution rates, their corresponding shape curves will exhibit different evolution rates. Generally, activity recognition has either been studied using probabilistic
graphical models such as hidden Markov models [77, 78, 79] or dynamic Bayesian networks [80, 81, 82, 83]. [7] studied this problem of time-warping variability and showed that the activity classification performance can improve if the shape trajectories are temporally aligned in some fashion. They provide a systematic model-based approach to learn the nature of such temporal variations (time-warpings) while simultaneously allowing for the spatial variations in the descriptors. Let $\alpha_1$ and $\alpha_2$ denote two shape trajectories. The distance used there for registering shape trajectories is $\int_0^1 \|\alpha_1 - \alpha_2(\gamma(t))\|^2 dt$. However, this criterion is not a proper metric on the space of trajectories. In fact, it is not even symmetric. To the best of our knowledge, there is no method in the literature that uses a proper warping-invariant metric for temporal registration of trajectories on nonlinear manifolds.

Another potential approach is to map the trajectory onto a vector space, say the tangent space at a point, using the inverse exponential map, and then compare the mapped trajectories using the Euclidean solutions in the vector space. While this idea is feasible, the results may not be consistent since the inverse exponential map is a local and highly nonlinear operator. For example, on a sphere, two points near the south pole will map to two distant points in the tangent space at the north pole, and their distance will be highly distorted. In contrast, the solution proposed here transports vector fields associated with trajectories, rather than trajectories themselves, into a standard tangent space and this provides a more stable alternative. To illustrate this stability, Fig. 2.1(a) shows a simulated trajectory on $S^2$. If we project it to the tangent space at a point $c$ which is on the opposite side of sphere using inverse exponential map, the resulting curve is heavily distorted as shown in Fig. 2.1(b). In contrast, the mapping to the same tangent space, obtained using our framework, is still nice and smooth, as shown on the right.
2.3 Limitations

For trajectory estimation, most papers study optimality criteria of the type given in Eqn. 1.1 and use calculus of variation approach to proceed. They typically derive some differential equations that the solution should satisfy and to find the solutions of these fourth and higher-order differential equations is non-trivial, except in some simple cases. In other words, such approaches may lead to a characterization of the solution that is not constructive or algorithmic.

While for analysis of trajectories, current state-of-the-art approaches treat the problems of registration and comparison separately. Different objective functions are used for registration and comparison respectively. This leads to a dilemma. The distance used for temporal registration is not proper, even not symmetric. While the distance for comparison does not perform registration although it is a metric. We would like a joint framework for registration and comparison together. This will allow us to define meaningful summary and modeling of trajectories.
CHAPTER 3

ESTIMATION OF SMOOTH TRAJECTORIES ON NONLINEAR MANIFOLDS

This chapter discusses the problem of estimating full trajectories on certain nonlinear manifolds using only a set of time-indexed points. These trajectories are analogous to smooth splines on Euclidean spaces as they are optimal under a similar objective function, which is a weighted sum of a fitting-related (data term) and a regularity-related (smoothing term) cost functions. The search for smoothing splines on manifolds is based on a Palais-based steepest-decent algorithm developed in [84]. Using four representative manifolds: the unit sphere for bird migration, the rotation group for pose tracking, the manifold of $3 \times 3$ symmetric positive-definite matrices with unit determinant for DTI image analysis, and Kendall’s shape space for video-based activity recognition, we demonstrate the effectiveness of the proposed algorithm for optimal smoothing trajectories. In this chapter, we derive certain geometrical elements, namely the exponential map and its inverse, parallel transport of tangents, and the curvature tensor, on these manifolds, that are needed in the gradient-based search for smoothing trajectories. These ideas are illustrated using experimental results involving both simulated and real data, and comparing the results to some current algorithms such as piecewise geodesic and splines on tangent spaces, including the method by [35].

3.1 Mathematical Framework

We adapt an approach presented in [84] that develops a gradient-based method for solving the optimization problem in Eqn. 1.1, with the smoothness penalty given by $E_{s,2}$ in Eqn. 1.3. This paper considers a general, finite-dimensional Riemannian manifold $M$ and derives the gradient of the cost function on $M$ using a second-order Palais metric. Let $\alpha$ be a twice-differentiable path on $M$ and let $v, w$ be two smooth vector fields along $\alpha$. That is, for any $t \in [0, 1]$, the vector
The second-order Palais metric ([85]) is given by:

\[ \langle \langle v, w \rangle \rangle_{2,\alpha} = \langle v(0), w(0) \rangle_{\alpha(0)} + \left\langle \frac{Dw}{dt}(0), \frac{Dw}{dt}(0) \right\rangle_{\alpha(0)} + \int_0^1 \left\langle \frac{D^2v}{dt^2}, \frac{D^2w}{dt^2} \right\rangle_{\alpha(t)} \, dt. \quad (3.1) \]

The objective function is a functional on an appropriate space of twice-differentiable paths on \( M \) and one needs a metric on this space to express the gradient as a vector field on the current path. Although there are several choices of metrics, including the standard \( \mathbb{L}^2 \) metric, the use of the second-order Palais metric greatly simplifies the expression for gradient of the second term in Eqn. 1.1, albeit at the cost of slight increase in complexity in gradient of the first term. [84] provides an expression for the gradient vector on a general manifold in terms of its differential geometry, and demonstrates these ideas using examples from two simple manifolds: \( \mathbb{R}^2 \) and \( S^2 \).

Now we summarize the main result presented in [84] for fitting smooth curves to time-indexed points on Riemannian manifolds. Let \( M \) be a Riemannian manifold and \( \alpha : [0, 1] \to M \) be an appropriately differentiable path on \( M \). Our goal is to find a path that minimizes the energy function:

\[ E(\alpha) = \frac{\lambda_1}{2} \sum_{i=1}^{n} d^2(\alpha(t_i), p_i) + \frac{\lambda_2}{2} \int_0^1 \left\langle \frac{D^2\alpha}{dt^2}, \frac{D^2\alpha}{dt^2} \right\rangle \, dt. \quad (3.2) \]

The first term is referred as the data term or \( E_d \) and the second term is referred as the smoothing term or \( E_s \) without the weights respectively. As mentioned in [84], the asymptotic limits of the solution are as following. As \( \lambda_1 \) goes to zero, for a fixed \( \lambda_2 \geq 0 \), one obtains a geodesic curve as the optimal curve under \( E_s \). Similarly, as \( \lambda_2 \) goes to zero, for a fixed \( \lambda_1 > 0 \), the optimal curve is analogous to a piecewise cubic polynomial that interpolates between the given points. At the limit, however, for \( \lambda_2 = 0 \) and \( \lambda_1 > 0 \) any interpolating curve will be optimal.

We will use the steepest-descent algorithm for minimizing \( E \), where the steepest-descent direction is defined with respect to the second-order Palais metric. This Palais metric is used in deriving the gradient of \( E \) and, once the gradient is derived, it does not play any further role in the computational solution. According to [84], the gradients of these two terms are given as follows:

1. **Data Term**: The gradient of the function \( E_d : \alpha \to \frac{1}{2} \sum_{i=1}^{n} d^2(p_i, \alpha(t_i)) \) at \( \alpha \in \Gamma \), w.r.t. the second-order Palais metric (Eqn. 3.1), is given by the vector field \( G = \sum_{i=1}^{n} g_i(t) \) along \( \alpha \) where

\[ g_i(t) = \begin{cases} 
(1 + t_1 + \frac{1}{6} t_1 t_2 - \frac{1}{6} t_3^3) \tilde{v}_i(t) & 0 \leq t \leq t_i, \\
(1 + tt_i + \frac{1}{2} tt_i^2 - \frac{1}{6} t_i^3) \tilde{v}_i(t) & t_i \leq t \leq 1,
\end{cases} \quad (3.3) \]

where \( \tilde{v}_i \) is the parallel translation of \( v_i \) along \( \alpha \) and \( v_i = -\exp_{\alpha(t_i)}^{-1}(p_i) \).

In other words, we first obtain a tangent vector \( v_i \) at the point \( \alpha(t_i) \) on the manifold using the inverse exponential map and parallel translate it over the whole path \( \alpha \) to obtain a (tangent) vector field along \( \alpha \). Then, we re-scale this vector field by multiplying a real-valued function
(given above) to obtain $g_i$ and, finally, we add all $g_i$s to obtain the gradient of $E_d$.

2. **Smoothing Term**: The gradient vector field of the function $E_s$ along $\alpha$, w.r.t. the second-order Palais metric, is a vector field given by $H_2(t) + H_3(t)$, where

- $H_2$ is a vector field given by:
  
  $$H_2(t) = \hat{H}_2(t) - (1/6) t^3 \tilde{S}(t) - (1/2) t^2 (\tilde{Q}(t) - \tilde{S}(t)) - t (\tilde{Q}(t) - \tilde{S}(t)) + \tilde{S}(t), \quad (3.4)$$

- $\hat{H}_2$ is also a vector field given by:
  
  $$\frac{D^4 \hat{H}_2}{dt^4}(t) = R\left(\frac{D^2 \dot{\alpha}}{dt^2}(t), \dot{\alpha}(t)\right)(\dot{\alpha}(t))$$

  with initial conditions $\hat{H}_2(0) = \frac{DH_2}{dt}(0) = \frac{D^2 H_2}{dt^2}(0) = \frac{D^3 H_2}{dt^3}(0) = 0.$

- $\tilde{Q}$ and $\tilde{S}$ are the parallel transports along $\alpha$ of $Q = \frac{D^2 H_2}{dt^2}(1)$ and of $S = \frac{D^3 H_2}{dt^3}(1)$.

- $H_3$ is given by $\frac{D^2 H_3}{dt^2} = \frac{D^2 \dot{\alpha}}{dt^2}$ with initial conditions $H_3(0) = \frac{DH_3}{dt}(0) = 0$.

- The mapping $R(\cdot,\cdot)(\cdot)$ denotes the Riemannian curvature of the manifold at a point. A physical interpretation of this curvature is as follows. When a vector at a point on the manifold is parallel transported around that point, it may not reach its original version. The Riemannian curvature tensor measures the difference between the starting vector and the ending vector. More precisely, it is given in terms of the Levi-Civita connection $\nabla$ by the formula: $R(u,v)w = \nabla_u \nabla_v w - \nabla_v \nabla_u w - \nabla_{[u,v]} w$, where $[u,v]$ is the Lie bracket of vector fields. For details, we refer readers to [86].

Combining the two terms, the gradient of $E$ under the second-order Palais metric is given by a vector field along $\alpha$:

$$\nabla E = \lambda_1 G + \lambda_2 (H_2 + H_3).$$

This gradient expression is used in deriving a steepest-descent algorithm for finding the optimal path $\hat{\alpha}$. This is an iterative algorithm that is summarized below. There are two parameters that need to be chosen by the user: $T$, the size of a discrete partition of $[0,1]$ on which computations are made and $\epsilon$, the step size for updating curves in the gradient direction. The choice of $T$ is determined by considerations of the computational cost and $\epsilon$ is through trial and error.

**Algorithm 1 Iterative Algorithm for Gradient-Based Minimization of $E$.** At each iteration $k = 1, 2, \ldots$, we have the current path $\alpha_k : [0,1] \rightarrow M$ sampled at $T$ times $\{t\delta | t = 0, 1, 2, \ldots, T\}$, $\delta = 1/T$.

1. **Data Term Gradient**: For $i = 1, 2, \ldots, n,$
(a) Find the shooting vector \( v_i = -\exp^{-1}_{\alpha_k(t_i)}(p_i) \).

(b) Parallel transport \( v_i \) to each point along \( \alpha_k \) to form a vector field \( \tilde{v}_i \). We start at \( \alpha_k(t_i) \) and proceed in both directions (increasing and decreasing \( t \)) in steps of size \( \delta \) and transport the tangent vector from the previous point to the current point.

(c) Compute \( g_i(\delta) \) using Eqn. 3.3 for all \( t \in \{0, 1, \ldots, T - 1\} \).

Evaluate \( G(t\delta) = \sum_{i=1}^{n} g_i(t\delta) \) for all \( t \in \{0, 1, \ldots, T - 1\} \).

2. **Smoothing Term Gradient**: For each \( t \in \{0, 1, \ldots, T - 1\} \),

(a) Compute \( \dot{\alpha}_k(t\delta) = \exp^{-1}_{\alpha_k(t\delta)} \alpha_k((t + 1)\delta) \).

(b) Parallel transport \( \dot{\alpha}_k(t\delta) \) to \( \alpha_k((t + 1)\delta) \); call it \( \dot{\alpha}_k^{||}(t\delta) \).

Compute \( \frac{D^2\dot{\alpha}_k(t)}{dt^2}((t + 1)\delta) = (\dot{\alpha}_k((t + 1)\delta) - \dot{\alpha}_k^{||}(t\delta))/\delta \).

(c) Compute the curvature tensor \( R((\frac{D^2\dot{\alpha}_k(t)}{dt^2}(t\delta), \dot{\alpha}_k(t\delta)))(\dot{\alpha}_k(t\delta)) \) at \( \alpha_k(t\delta) \).

(d) Solve for \( \tilde{H}_2 \) by covariantly integrating \( R \) four times (by repeatedly applying Algorithm 2), each time with initial condition zero.

(e) Compute \( \hat{Q}(t\delta) \) and \( \hat{S}(t\delta) \) as backward parallel translations of \( \frac{D^2\tilde{H}_2}{dt^2}(1) \) and \( \frac{D^2\tilde{H}_2}{dt^2}(1) \), respectively.

For any vector \( v(1) \) at \( \alpha_k(1) \), its backward parallel translation along \( \alpha_k \) is computed in backward steps: start from \( t = T - 1 \), and for any \( t \delta \) parallel transport \( v((t + 1)\delta) \) at \( \alpha_k((t + 1)\delta) \) to \( \alpha_k(t\delta) \) and call that vector \( v(t\delta) \).

(f) Use Eqn. 3.4 to compute \( H_2 \).

(g) Similarly, solve for \( H_3 \) by covariantly integrating \( \frac{D^2\hat{Q}(t)}{dt^2} \) twice (using Algorithm 2), each time with initial condition zero.

3. **Total Gradient**: The gradient of \( E \) is now given by the vector field \( F(t\delta) = \lambda_1 G(t\delta) + \lambda_2(H_2(t\delta) + H_3(t\delta)) \) for all \( t \in \{0, 1, \ldots, T - 1\} \).

4. **Update of Curve**: For a pre-defined step size \( \epsilon \), we update the current path according to: for a fixed \( \tau \),

\[
\alpha_{k+1}^\tau(t\delta) = \exp_{\alpha_k(t\delta)}(-\tau \epsilon F(t\delta)), \text{ for } t = 0, 1, 2, \ldots, T, 
\]

and compute \( E(\alpha_{k+1}^\tau) \). At this stage we try different values of \( \tau \) as follows. We start at \( \tau = 1 \) and continue increasing \( \tau \) with the increment of one until \( E(\alpha_{k+1}^{\tau+1}) > E(\alpha_{k+1}^\tau) \). Then, we select the best \( \tau \) value based on the minimum \( E \) achieved, and the set \( \alpha_{k+1} \) to be the corresponding \( \alpha_{k+1}^\tau \).

5. Check for convergence. If not converged, then set \( k = k + 1 \) and return to Step 1.
Algorithm 2 Compute covariant integral \( u \) of a vector field \( v \) along \( \alpha \). Initialize \( w(0) = 0 \). For each \( t = 0, 1, \ldots, T - 1 \):

1. Parallel transport \( w(t\delta) \) from \( \alpha(t\delta) \) to \( \alpha((t+1)\delta) \); call it \( w^\parallel((t+1)\delta) \).
2. Also, parallel transport \( v(t\delta) \) from \( \alpha(t\delta) \) to \( \alpha((t+1)\delta) \); call it \( v^\parallel((t+1)\delta) \).
3. Define \( u((t + 1)\delta) = v^\parallel((t + 1)\delta) + \delta w^\parallel((t + 1)\delta) \).

To implement this method (including Algorithms 1 and 2), we need the following items from the differential geometry of \( M \):

- **Exponential Map**: For any point \( p \in M \) and a tangent vector \( v \in T_p(M) \), we should be able to find the point \( \exp_p(v) \in M \). If we appropriately restrict the domain of this map, then it becomes invertible.

- **Inverse Exponential Map**: For any two points \( p_1, p_2 \) on \( M \), we should be able to compute the inverse exponential \( \exp_{p_1}^{-1}(p_2) \). This part is needed in the gradient of \( E_d \). Of course, this inverse is well defined only when the domain of the forward map is restricted appropriately.

- **Parallel Transport**: For any two points \( p_1 \) and \( p_2 \), and a tangent vector \( v_1 \in T_{p_1}(M) \), we should be able to parallel transport \( v_1 \) to \( p_2 \) along a geodesic path connecting \( p_1 \) and \( p_2 \). This item is used in numerical implementations of covariant integration and differentiation using finite sums and differences, respectively.

- **Riemannian Curvature**: For any point \( p \in M \) and tangent vectors \( X, Y, \) and \( Z \in T_p(M) \), we should be able to compute the Riemannian curvature tensor \( R(X, Y)(Z) \). This calculation is needed in evaluating the gradient of \( E_s \).

**Computational Cost**: As mentioned above, we have implemented this algorithm for four manifolds: unit sphere \( \mathbb{S}^2 \), rotation group \( \text{SO}(3) \), the space of SPD matrices with unit determinant \( \mathcal{P}(3) \) and Kendall’s shape space \( \mathbb{C}P^{n-2} \). Although we present the detailed results in the later sections, we mention the computational costs of this method here. To illustrate the computational efficiency, we present computational costs per iteration for updating a path on each of the four manifolds (using Matlab on a 2.4GHz Intel processor) in Table 3.1.

<table>
<thead>
<tr>
<th>Manifold</th>
<th>Time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{S}^2 )</td>
<td>0.082</td>
</tr>
<tr>
<td>( \text{SO}(3) )</td>
<td>0.245</td>
</tr>
<tr>
<td>( \mathcal{P}(3) )</td>
<td>0.233</td>
</tr>
<tr>
<td>( \mathbb{C}P^{n-2}, n = 100 )</td>
<td>0.485</td>
</tr>
</tbody>
</table>
In the next few sections, we will take specific examples of manifolds, describe the relevant parts of their differential geometries, and implement the Algorithms 1 and 2 for finding optimal paths on those manifolds.

### 3.2 Estimation of Trajectories on Nonlinear Manifolds

#### 3.2.1 Unit Sphere $\mathbb{S}^2$

The case of unit sphere $\mathbb{S}^2$ has already been studied in [84]. Here, we list the required tools for reference. Later on, we apply the framework to a specific application: bird migration. The resulted smoothing trajectories will be further analyzed in later chapters.

- **Basic Geometric Tools:** Here, we list the required tools for competing the gradient.

  - **Exponential Map:** The exponential map, $\exp: T_p(\mathbb{S}^2) \mapsto \mathbb{S}^2$ has a simple expression:
    
    $$
    \exp_p(v) = \cos(\|v\|)p + \sin(\|v\|) \frac{v}{\|v\|}.
    $$

  - **Inverse Exponential Map:** The inverse exponential map $\exp^{-1}_{p_1}(p_2)$ is:
    
    $$
    \exp^{-1}_{p_1}(p_2) = \frac{\theta}{\sin(\theta)}(p_2 - p_1 \cos(\theta)), \text{ where } \theta = \cos^{-1} \langle p_1, p_2 \rangle.
    $$

  - **Parallel Transport:** A vector $v \in T_{p_1}(\mathbb{S}^2)$ can be transported along a great circle to a point $p_2 \neq -p_1 \in \mathbb{S}^2$ using the formula:
    
    $$
    v \mapsto \left( v - \frac{2 \langle v, p_2 \rangle}{\|p_1 + p_2\|^2} (p_1 + p_2) \right).
    $$

  - **Riemannian Curvature:** Since the unit sphere has constant sectional curvature one, the curvature tensor is given by:
    
    $$
    R(X, Y)Z = \langle Y, Z \rangle X - \langle X, Z \rangle Y.
    $$

- **Experimental Results:** We apply the framework to the bird migration data. The dataset has 35 migration trajectories of Swainson’s Hawk, measured from 1995 to 1997. The observed data includes geographic coordinates measured at discrete times. Two examples of estimated trajectories are shown as red in Fig. 3.1.
3.2.2 Three-Dimensional Rotation Group $\text{SO}(3)$

Here we are interested in fitting smooth paths to given time-indexed points on the rotation group $\text{SO}(3)$ under the standard Euclidean metric.

- **Basic Geometric Tools**: This space is a Lie group under matrix multiplication and the identity element is given by the $3 \times 3$ identity matrix $I_3$. The tangent space $T_{I_3}(\text{SO}(3))$ is the set of all $3 \times 3$ skew-symmetric matrices, and $T_O(\text{SO}(3)) = \{OA|A \in T_{I_3}(\text{SO}(3))\}$. The standard Riemannian metric is given by $\langle X_1, X_2 \rangle = \text{trace}(X_1^T X_2)$. For any point $O \in \text{SO}(3)$ and a tangent vector $X = OA$ (where $A^T = -A$), the geodesic path from $O$ in the direction $X$ is given by: $t \mapsto Oe^{tA}$ where $e$ denotes the matrix exponential. The required items for computing the gradient of $E$ for paths on $\text{SO}(3)$ are:

  - **Exponential Map**: Given a point $O \in \text{SO}(3)$ and a tangent vector $X \in T_O(\text{SO}(3))$, the exponential map is given by:
    \[ \exp_O(X) = Oe^{O^TX}. \]
– **Inverse Exponential Map:** For any two points $O_1, O_2 \in \text{SO}(3)$, we can compute the inverse exponential using the formula:

$$\exp^{-1}_{O_1}(O_2) = O_1 \log(O_1^T O_2),$$

where log denotes the matrix logarithm.

– **Parallel Transport:** For any two elements $O_1$ and $O_2$ in $\text{SO}(3)$, and a tangent vector $W \in T_{O_1}(\text{SO}(3))$, the tangent vector at $O_2$ which is the parallel transport of $W$ along the shortest geodesic from $O_1$ to $O_2$ is:

$$W' = O_1(O_1^T O_2)^{1/2}(O_1^T W)(O_1^T O_2)^{1/2}.$$  

Since $W$ is tangent to $\text{SO}(3)$ at $O_1$, it follows that $O_1^T W$ is tangent at the identity, i.e., $O_1^T W$ is skew-symmetric. The natural way to calculate $(O_1^T O_2)^{1/2}$ is to let $A$ be a skew-symmetric matrix which is the matrix log of $O_1^T O_2$; then $(O_1^T O_2)^{1/2} = e^{A/2}$. If we let $A$ be the matrix log of $O_1^T O_2$ as above, then the geodesic from $O_1$ to $O_2$ is just $O_1 e^{tA}$ where $t$ goes from 0 to 1. So, the parallel transport becomes:

$$W' = O_1 e^{A/2}(O_1^T W)e^{A/2}.$$  

– **Riemannian Curvature:** For any point $O \in \text{SO}(3)$ and tangent vectors $X, Y, and Z \in T_O(\text{SO}(3))$, the Riemannian curvature tensor $R(X, Y)(Z)$ at the point $O$ is given by:

$$R(X, Y)(Z) = \frac{1}{4}O[[A, B], C], \text{ where } A = O^T X, \ B = O^T Y, \ C = O^T Z.$$  

Here $[A, B]$ denotes $AB - BA \in \mathbb{R}^{3 \times 3}$.  

**Experimental Results:** We implement Algorithm 1 for minimizing the total energy $E$ associated with a given path on $\text{SO}(3)$. To display a path $\alpha : [0, 1] \rightarrow \text{SO}(3)$, we use the following idea. Take the unit vector $v = (1, 1, 1)/\sqrt{3} \in \mathbb{R}^3$, generate the process $X(t) = v\alpha(t)$ in $\mathbb{R}^3$ and display points along $X(t)$ at any iteration. Note that this is only one way to visualize the results and several alternative ideas can also be used here. The value of $T = 40$ is used in the experiments in this section.

1. **Smoothing Term Only:** As the first experiment, we set $\lambda_1 = 0$ and minimize only $E_s$ using its gradient vector field given by $H_2 + H_3$. In other words, we ignore the points $\{(t_i, p_i)\}$ and the data term $E_d$ and simply try to minimize $E_s$. Two examples are shown in Fig. 3.2, where each case shows the initial path and the final path, along
Figure 3.2: Minimization of $E_s(\lambda_1 = 0, \lambda_2 = 1)$: Each case shows the initial path (marked line) and the final path (solid line), along with the corresponding evolution of $E_s$.

with the evolution of $E_s$. As seen in both examples, the gradient method proves very effective in minimizing $E_s$ to zero rapidly. As mentioned earlier, a minimizer of $E_s$ is simply a geodesic path and this is precisely what we get here. When visualized via its action on $(1, 1, 1)/\sqrt{3}$, this optimal path traces an arc on the sphere.

Figure 3.3: Minimization of $E_d(\lambda_1 = 1, \lambda_2 = 0)$: Each case shows the initial path (marked line) and the final path (solid line), along with the evolution of $E_d$.

2. **Data Term Only**: In this case we set $\lambda_1 = 1$ and $\lambda_2 = 0$, and use the gradient $G$ to minimize only the data term $E_d$. We generate a set of indexed points $(t_i, p_i)$ on SO(3), select a smooth path as the initial $\alpha$ and use Algorithm 1 to perform gradient-based updates of $\alpha$. Some of the results are shown in Fig. 3.3. Each pair of panels shows the initial path and the final path, along with the evolution of $E_d$. We note that, in each case, the decrease in $E_d$ is very rapid at the start but slows down considerably in
the later stages. This is due to the use of second-order Palais metric in the evaluation of the gradient; while \( E_d \) is a zeroth-order quantity, as it simply involves computing distances between points and no derivatives, the Palais metric is a second-order metric involving second derivatives of \( \alpha(t) \). From a practical perspective, there is a possibility of stopping at a point that is not a global minimizer of \( E_d \). This is visible in the examples shown in Fig. 3.3 where the final curve does not strictly pass through the given points. In a practical situation we can possibly address this problem by using an initial path that is close to the given points, e.g. a piecewise geodesic. Another notable phenomenon occurs when \( t_i \)s are non-uniform and a big part of the interval \([0, 1]\) is not represented by any data point. In this case, the evolution of the curve in that part can be unintuitive since this part is evolving under the influence of far away data points.

3. **Joint Optimization:** In this case we study the full gradient term \( \lambda_1 G + \lambda_2 (H_2 + H_3) \) for different values of \( \lambda_1 \) and \( \lambda_2 \). Some results are presented in Fig. 3.4. In this experiment we generate a set of \( n = 4 \) data points and keep them fixed. The algorithm is initialized in each case with the same path that passes through the data points at the corresponding times and, hence, \( E_d \) is zero for the starting path \( \alpha \). Different combinations of \( \lambda_1 \) and \( \lambda_2 \) result in different evolutions of \( \alpha \) and the final energy \( E \). In the first few cases, where \( \lambda_2 \) is significant, the smoothing term dominates the evolution. Here, the energy quickly decreases to a very small value with the optimal path being close to a geodesic path on \( SO(3) \). As \( \lambda_1 \) starts getting larger, and/or \( \lambda_2 \) starts getting smaller, the data term starts having an effect and consequently the solution stays close to the given points. The last two solutions appear to be reasonable solutions in a practical situation for smooth interpolating between noisy points.

**Comparison of different methods:** In order to compare our solutions to the curve fitting problem, we consider two common ideas:

1. **Piecewise geodesic:** In this case one generates piecewise geodesic paths by connecting the data points via geodesics at the given time indices.

2. **Spline on mean tangent space:** In this case, we first compute the extrinsic mean of the given data points on \( SO(3) \), project each data point \( p_i \in SO(3) \) onto tangent space at the extrinsic mean, and then fit a cubic smoothing spline on that Euclidean space. Finally, we project the fitted curve back to \( SO(3) \) using the exponential map.

To evaluate performance of a method, we first generate a smooth path on \( SO(3) \) to serve as the ground truth, denoted by \( \alpha_0 \). Then, we choose a certain number of random points on the path as the data points, add some noise to them and use them as data to fit curves using the three methods. Since any element of \( SO(3) \) can be written as \( e^W \), where \( W \) is a skew-symmetric
Figure 3.4: Optimization of $E$ under different combinations of $\lambda_1$ and $\lambda_2$. In all these cases we use the same data points $(t_i, p_i)$ and the same initial path $\alpha$ for starting the algorithm.

matrix, we can add noise to it using $e^{(W+G)}$, where

$$G = [0, -g(3), -g(2); g(3), 0, -g(1); g(2), g(1), 0]$$

and $g$ is multivariate normal with mean zero. The level of additive noise can be quantified by the variance of $g$. Finally, we evaluate the estimation performance using

$$\varepsilon = \int_0^1 d(\alpha(t), \alpha_0(t))^2 dt,$$

where $\alpha$ is the fitted path and $d$ denotes the geodesic distance. Fig. 3.5 shows some pictorial examples of the estimated curves, under two different noise levels. In each case, solid line denotes the ground truth $\alpha_0$ and red points denote the given data. The dashdot green line denotes the piecewise geodesic fitted curve, the dotted yellow line denotes the spline on tangent space and the dash blue line denotes the solution obtained using our method under $\lambda_1 = 100$ and $\lambda_2 = 1$. Note that the smoothing parameter for the spline on mean tangent space is chosen by minimizing the estimation error $\varepsilon$ in these examples. The left two figures show that three solutions are close when the additive noise in data points is low while the right two figures show that when the noise is large, our smoothing spline is closer to the ground truth than the other two solutions.
In order to study the effect of noise on estimation performance, we have averages $\varepsilon$ over $k = 100$ realizations of the random noise at each level of noise variance, and have plotted it against the variance in Fig. 3.6 (a). As shown in the plot, the error gets worse as noise level increases for all the three methods. However, of the three, our method provides the lowest error and results in a stable estimator, especially when the noise gets high. Fig. 3.6 (b) shows the changes in estimation error $\varepsilon$ versus the weight ratio $\lambda = \lambda_2/\lambda_1$ used in our method. In case the noise level is low and the data points are reflective of the ground truth, a small or negligible value of $\lambda$ is optimal in terms of the estimation error $\varepsilon$. No smoothing is required and most interpolating curves work well in this situation. However, when the noise level gets high, the role of the smoothing term becomes important and the optimal result is obtained for a significant value of $\lambda$. A further increase in $\lambda$ results in over smoothing and the error starts growing again.

### 3.2.3 Symmetric Positive-Definite Matrices with Unit Determinant

In this section we present results on the space of $3 \times 3$ SPD matrices with determinant 1.
• **Basic Geometric Tools:** Let $\tilde{\mathcal{P}}(n)$ be the space of $n \times n$ SPD matrices and $\mathcal{P}(n) = \{ P \mid P \in \tilde{\mathcal{P}}(n) \text{ and } \det(P) = 1 \}$. It is known that for any $\tilde{P} \in \tilde{\mathcal{P}}(n)$, $P = \tilde{P}/\det(\tilde{P})^{1/n} \in \mathcal{P}(n)$. The space $\mathcal{P}(n)$ is a well known Riemannian symmetric manifold, i.e., the quotient of the special linear group $SL(n) = \{ G \in GL(n) \mid \det(G) = 1 \}$ by its closed subgroup $SO(n)$ acting on the right ([87]) and with an $SL(n)$-invariant metric. (Note that there are numerous Riemannian metrics used in the literature for analyzing SPD matrices ([19]), but we choose the current metric in view of its invariance properties.) Since $n = 3$ is important in diffusion tensor imaging, we will focus on this case. We are interested in fitting smooth paths to given time-indexed points on $\mathcal{P}(3)$ and it can be easily extended to $\tilde{\mathcal{P}}(3)$ by separately interpolating the determinants. The tangent space of $\mathcal{P}(3)$ at $I$ is $T_I(\mathcal{P}(3)) = \{ A \mid A^T = A \text{ and } \text{tr}(A) = 0 \}$, and the tangent space at $P$ is $T_P(\mathcal{P}(3)) = \{ PA \mid A \in T_I(\mathcal{P}(3)) \}$. For any point $P \in \mathcal{P}(3)$ and a tangent vector $V$, the geodesic path from $P$ in the direction $V$ is given by:

$$t :\mapsto \sqrt{Pe^{2V}P}.$$

- **Exponential Map:** Given a point $P \in \mathcal{P}(3)$ and a tangent vector $V \in T_P(\mathcal{P}(3))$, the exponential map is given by:

$$\exp_P(V) = \sqrt{Pe^{2V}P}.$$

- **Inverse Exponential Map:** For any two points $P_1, P_2 \in \mathcal{P}(3)$, we can compute the inverse exponential using the formula:

$$\exp_{P_1}^{-1}(P_2) = P_1 \log(\sqrt{P_1^{-1}P_2^2P_1^{-1}}).$$

- **Parallel Transport:** For any two points $P_1, P_2 \in \mathcal{P}(3)$, and a tangent vector $V \in T_{P_1}(\mathcal{P}(3))$, the tangent vector at $P_2$ which is the parallel transport of $V$ along the shortest geodesic from $P_1$ to $P_2$ is:

$$P_2 T_{12}^T B T_{12}, \text{ where } B = P_1^{-1}V, \ T_{12} = P_{12}^{-1}P_{12}^{-1} \text{ and } P_{12} = \sqrt{P_1^{-1}P_2^2P_1^{-1}}.$$

- **Riemannian Curvature:** For any point $P \in \mathcal{P}(3)$ and tangent vectors $X, Y, \text{ and } Z \in T_P(\mathcal{P}(3))$, the Riemannian curvature tensor $R(X, Y)(Z)$ is given by:

$$R(X, Y)(Z) = -P[[A, B], C], \text{ where } A = P^{-1}X, \ B = P^{-1}Y, \ C = P^{-1}Z.$$

- **Experimental Results:** An important issue in studying experimental results on $\mathcal{P}(3)$ is how to visualize the solution. We will visualize an element $P$ of this space using an ellipsoid
formed as a level curve of function \( X \mapsto X^T P X \) and \( T = 10 \) is used in this section.

1. **Smoothing Term Only**: As the first experiment, we set \( \lambda_1 = 0 \) and minimize only \( E_s \) using its gradient \( H_2 + H_3 \). In other words, we ignore the points \((t_i, p_i)\) and the data term \( E_d \) and simply try to obtain a geodesic that minimizes \( E_s \). Some results are shown in Fig. 3.7 where the left panels show the initial path (first row) and the final path (second row), along with the evolution of \( E_s \) on the right panels. As seen in both examples, the gradient method proves very effective in minimizing \( E_s \) to zero rapidly. The minimizer of \( E_s \) is simply a geodesic path on \( \mathcal{P}(3) \).

2. **Data Term Only**: In this case we set \( \lambda_1 = 1 \) and \( \lambda_2 = 0 \), and use the gradient \( G \) to minimize only the data term \( E_d \). In this experiment we generate a set of \( n = 4 \) data points, select a smooth path as the initial \( \alpha \) and use Algorithm 1 to perform gradient-based updates of \( \alpha \). Some of the results are shown in Fig. 3.8. In both cases, the top row shows the given time-indexed points, the middle row shows the initial \( \alpha \) and the bottom row shows the final \( \alpha \). The right panel shows the evolution of \( E_d \). Note that, in
each case, the decrease in $E_d$ is very rapid at the start but slows down considerably in the later stages.

3. **Joint Optimization**: In this case we study the full gradient term $\lambda_1 G + \lambda_2 (H_2 + H_3)$ for different values of $\lambda_1$ and $\lambda_2$, and some of the results are presented in Fig. 3.9. We initialize the algorithm each time with a piecewise geodesic path that is passing through the given points at the corresponding times and, hence $E_d$ is zero for the starting path $\alpha$. Different combinations of $\lambda_1$ and $\lambda_2$ result in different evolutions of $\alpha$ and the energy $E$. In the earlier cases where $\lambda_2$ is relatively significant, the smoothing term dominates
Figure 3.9: Optimization of $E$ under different combinations of $\lambda_1$ and $\lambda_2$. In all these cases we use the same data points $(t_i, p_i)$ and the same initial path $\alpha$ for starting the algorithm.

The evolution. Here, the energy quickly decreases to a very small value on the resulting path, i.e. a geodesic path on $\mathcal{P}(3)$. As $\lambda_1$ starts getting larger, and/or $\lambda_2$ starts getting negligible, the data term starts having an effect and consequently the solution stays close to the given points.

Finally, we apply our method to real data. The data includes 12 fibers and the corresponding trajectories of SPD matrices. These 12 fibers are representative projection pathways passing through internal capsule, extracted from 12 subjects. Fig. 3.10 shows two examples of smoothing splines of such SPD data. We will compute statistical summaries of these splines in the next chapter.
3.2.4 Kendall’s Shape Space

As the last example, we consider the shape space formed by \( n \) landmarks in \( \mathbb{R}^2 \). We first describe the relevant elements from geometry of this shape space and then present some experimental results.

- **Basic Geometric Tools:** Let \( x \in \mathbb{R}^{n \times 2} \) represent \( n \) ordered points selected from the boundary of an object. It is often convenient to identify points in \( \mathbb{R}^2 \) with elements of \( \mathbb{C} \), i.e., \( x^i \equiv z^i = (x^{i,1} + jx^{i,2}) \), where \( j = \sqrt{-1} \). Thus, in this complex representation, a configuration of \( n \) points \( x \) is now \( z \in \mathbb{C}^n \). We remove the translations by restricting to those elements of \( \mathbb{C}^n \) whose average is zero and the scale variability by rescaling the complex vector to have norm one. This results in a set: \( D = \{ z \in \mathbb{C}^n | \frac{1}{n} \sum_{i=1}^{n} z^i = 0, \| z \| = 1 \} \). Here, \( D \) is a unit sphere and one can utilize the geometry of a sphere to analyze points on it. The tangent space to \( D \) at a point \( z \in D \) is given by \( T_z(D) = \{ v \in \mathbb{C}^n | \Re(\langle v, z \rangle) = 0, \frac{1}{n} \sum_{i=1}^{n} v^i = 0 \} \).

Let \([z]\) be the set of all rotations of a configuration \( z \) according to \([z] = \{ e^{j\phi}z | \phi \in \mathbb{S}^1 \}\). One defines an equivalence relation on \( D \) by setting all elements of this set as equivalent, i.e., \( z_1 \sim z_2 \) if there exists an angle \( \phi \) such that \( z_1 = e^{j\phi}z_2 \). The set of all such equivalence classes is the quotient space \( D/\mathbb{S}^1 \). This space is called the complex projective space and is denoted by \( \mathbb{CP}^{n-2} \). The tangent space to the orbit of point \( z \) is given by \( T_z([z]) = \{ jxz | x \in \mathbb{R} \} \).

Since \( T_z(D) = T_z([z]) \oplus T_z(\mathbb{CP}^{n-2}) \), the tangent space is \( T_z(\mathbb{CP}^{n-2}) = \{ v \in \mathbb{C}^n | \langle v, z \rangle = 0, \frac{1}{n} \sum_{i=1}^{n} v^i = 0 \} \).

A geodesic between two elements \([z_1], [z_2] \in \mathbb{CP}^{n-2} \) is given by computing a geodesic between \( z_1 \) and \( z_2^* = e^{j\phi^*}z_2 \) in \( D \), where \( \phi^* = \theta \) and \( \langle z_1, z_2 \rangle = re^{j\theta} \). Now consider the items required for computing the gradient of \( E \) for paths on \( \mathbb{CP}^{n-2} \).
– **Exponential Map**: The exponential map, \( \exp : T_z(\mathbb{C}P^{n-2}) \mapsto \mathbb{C}P^{n-2} \), has a simple expression:
\[
\exp_z(v) = \cos(\|v\|)z + \sin(\|v\|) \frac{v}{\|v\|}.
\]

– **Inverse Exponential Map**: The exponential map is a bijection if we restrict \( \|v\| \) so that \( \|v\| \in [0, \frac{\pi}{2}) \). The inverse exponential map \( \exp^{-1}_z(z_2) \) is:
\[
\exp^{-1}_z(z_2) = \frac{d}{\sin(d)}(z_2^* - rz_1), \text{ where } d = \cos^{-1}(r).
\]

– **Parallel Transport**: If we parallel transport a vector \( v \in T_{z_1}(\mathbb{C}P^{n-2}) \) along the shortest geodesic (i.e., great circle) from \( z_1 \) to \( z_2 \), the result is:
\[
v - \frac{(v, z_2^*) (z_1 + z_2^*)}{1 + r} \in T_{z_2^*}(\mathbb{C}P^{n-2}).
\]

– **Riemannian Curvature**: For \( X, Y, Z \in T_z(\mathbb{C}P^{n-2}) \), the curvature is given by
\[
R(X, Y)Z = -\langle Y, Z \rangle X + \langle X, Z \rangle Y + \langle Y, jZ \rangle jX - \langle X, jY \rangle jZ - 2\langle X, jY \rangle jZ.
\]

• **Experimental Results**: Now we show some experimental results for fitting curves to given data points on \( \mathbb{C}P^{n-2} \). We will show results from two sets of experiments. In one case we simulate shape data using simple shapes while in another we use real shape sequences from the INRIA 4D repository. \( T = 10 \) is used for all examples in this section.

– **Simulated Data**

1. **Smoothing Term Only**: As the first experiment, we set \( \lambda_1 = 0 \) and minimize only \( E_s \) using its gradient \( H_2 + H_3 \). In other words, we ignore the points \( (t_i, p_i) \) and the data term \( E_d \) and simply try to minimize \( E_s \). Some results are shown in Fig. 3.11. In each case, the top row shows the initial path and the bottom row shows the final path, along with the evolution of \( E_s \) on the right. As seen in these examples, the gradient method proves very effective in minimizing \( E_s \) to zero rapidly and the final curve is simply a geodesic path on \( \mathbb{C}P^{n-2} \).

2. **Data Term Only**: In this case we set \( \lambda_1 = 1 \) and \( \lambda_2 = 0 \), and use the gradient \( G \) to minimize only the data term \( E_d \). In this experiment we generate a set of \( n = 4 \) data points, select as the initial \( \alpha \) the geodesic between the two end points and use Algorithm 1 to perform gradient-based updates of \( \alpha \). Some of the results are shown in Fig. 3.12. In each case, the top row shows the given time-indexed points, the middle row shows the initial \( \alpha \) and the bottom row shows the final \( \alpha \). The right
Figure 3.11: Minimization of $E_s(\lambda_1 = 0, \lambda_2 = 1)$: In each case, the left panel shows the initial path (top row) and the final path (bottom row), the right panel shows the corresponding decrease in $E_s$ during evolution.

panel shows the evolution of $E_d$. Note that, in each case, the decrease in $E_d$ is very rapid at the start but slows down considerably in the later stages.

3. **Joint Optimization**: In this case we study the full gradient term $\lambda_1 G + \lambda_2 (H_2 + H_3)$ for different values of $\lambda_1$ and $\lambda_2$. Some results are presented in Fig. 3.13. We initialize the algorithm each time with a piecewise geodesic path that is passing through the given points at the corresponding times and, hence $E_d$ is zero for the starting path $\alpha$. Different combinations of $\lambda_1$ and $\lambda_2$ result in different evolutions of $\alpha$ and the energy $E$. The results of this experiment are similar to that in the two manifolds we talked earlier. In the earlier cases where $\lambda_2$ is not negligible, the smoothing term dominates the evolution. Here, the energy quickly decreases to a very small value on the resulting path, i.e. a geodesic path on $\mathbb{C}P^{n-2}$. As $\lambda_1$ starts getting larger, and/or $\lambda_2$ starts getting negligible, the data term starts having an effect and consequently the solution stays close to the given points. Some more examples when $\lambda_1 = \lambda_2 = 1$ are given in Fig. 3.14.

**Comparison of different methods using real data:**

In order to evaluate our approach, we compare our results with three existing methods for data interpolation on Kendall’s shape space. They are:
Figure 3.12: Minimization of $E_d(\lambda_1 = 1, \lambda_2 = 0)$. In each case, the left panel shows the given data (top row), the initial path (middle row) and the final path (bottom row), along with the evolution of $E_d$ on the right.

1. **Spline on mean tangent space**: Similar to the SO(3) case, we take an extrinsic mean of the given points on $\mathbb{C}P^{n-2}$ and project all the given points onto the tangent space at that mean shape using the inverse exponential map. On that Euclidean space, we fit a smoothing spline using standard Euclidean algorithm and project the fitted spline back to $\mathbb{C}P^{n-2}$ using the exponential map.

2. **Piecewise geodesic**: Here we simply construct geodesics between successive points, and concatenate them to form a fitted curve.

3. **Shape spline of [35]**: This method involves projecting the given data points onto the tangent space at the first data point, but the projection itself uses a piecewise geodesic curve and the projection is performed by passing through successive tangent spaces so as to minimize the distortion in the projection. Then, one uses a smoothing spline on the tangent space to fit a curve and finally project it back to the shape space using the exponential map.

To quantify the performance, we have taken two real sequences from INRIA 4D repository, shown in the top rows of Fig. 3.15 and 3.16. In each case, we choose 4 data points from the real sequence, add (point wise) noise to the 4th and 7th shapes and fit splines using the noisy data points. The 1st row shows the real sequence, the 2nd row shows the data points, the
Figure 3.13: Optimization of $E$ under different combinations of $\lambda_1$ and $\lambda_2$. In all these cases we use the same data points $(t_i, p_i)$ and the same initial path $\alpha$ for the gradient search.

3rd row shows the spline on mean tangent space, the 4th row shows the piecewise geodesic, the 5th row shows the shape spline of [35] and the bottom row shows the optimal sequence obtained using our method under $\lambda_1 = 0.1$, $\lambda_2 = 1$. In both cases, the spline of [35] and our spline look more natural and more regular than the piecewise geodesic and the spline on mean tangent space. For each spline in both cases, Table 3.2 lists the energy terms $E_d$, $E_s$, and $E$ and the estimation error $\varepsilon$, which measures the difference of the spline and the real sequence, for each of the four methods. The smoothing parameters for the methods of spline on mean tangent space and [35] are chosen by minimizing the estimation error. It shows that the spline of [35] and our spline do have the smoothing effect when noise exists and our spline has the smallest estimation error in both cases.
Figure 3.14: Optimization of $E$ under $\lambda_1 = \lambda_2 = 1$. In each case, the top row shows the initial path and the bottom row shows the final path, along with $E$ on the right. The shapes in the initial paths at time-indexed 1, 4, 7, and 10 form the data points.

Table 3.2: Energy terms and estimation errors for four methods in two different experiments ($\lambda_1 = 0.1, \lambda_2 = 1$).

<table>
<thead>
<tr>
<th>Method</th>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_d$</td>
<td>$E_s$</td>
</tr>
<tr>
<td>Spline on tangent space</td>
<td>0.026</td>
<td>0.0290</td>
</tr>
<tr>
<td>Piecewise geodesic</td>
<td>0.043</td>
<td>0.0845</td>
</tr>
<tr>
<td>Spline of [35]</td>
<td>0.051</td>
<td>0.0004</td>
</tr>
<tr>
<td>Our spline</td>
<td>0.050</td>
<td>0.0003</td>
</tr>
</tbody>
</table>

### 3.3 Estimation of Optimal Weight Ratio

An important open issue in this framework is the choice of the weight ratio $\lambda = \lambda_2 / \lambda_1$ in estimating the smoothing spline. Clearly, the different values of $\lambda$ will lead to different results and this choice becomes important. While, in some cases, it may be possible to choose $\lambda$ according
to the application, this is a difficult issue in most situations. Here we suggest a cross-validation approach, that uses the training data, to automatically determine an optimal value of $\lambda$. The optimal value of $\lambda$ is given by:

$$\hat{\lambda} = \arg\min_{\lambda} \left( \sum_{i=1}^{n} d(\alpha_{-i}(t_i, \lambda), p_i)^2 \right),$$

where $\alpha_{-i}(\cdot, \lambda)$ is the optimal curve obtained when $p_i$ is omitted from the data, under the weight ratio $\lambda$. In other words, we omit $p_i$, estimate the optimal curve for the current $\lambda$ and compute the distance of the estimated point $\alpha(t_i)$ from the correct value $p_i$. Summing these distances squared over all the points gives the cost function for finding the optimal $\lambda$.

We demonstrate this method using the example shown in Fig. 3.17. Given the data points shown in the top row of left panel, we find the optimal path in the bottom row according to $\hat{\lambda}$, which is the
Real sequence 2

Data points

Spline on mean tangent space

Piecewise geodesic

Shape spline of [35]

Our spline under $\lambda_1 = 0.1$, $\lambda_2 = 1$

Figure 3.16: Comparison of four different methods on Kendall’s shape space: Case 2.

Figure 3.17: Estimation of $\hat{\lambda}$ using the cross-validation approach. Left: The optimal path under $\hat{\lambda}$. Right: Cost function for automatically estimating $\hat{\lambda}$.

minimizer of the cost function shown on the right.
CHAPTER 4

ANALYSIS OF TRAJECTORIES ON NONLINEAR
MANIFOLDS

In this chapter, we investigate statistical analysis of trajectories on Riemannian manifolds that are observed under arbitrary temporal evolutions. The past methods rely on cross-sectional analysis, with the given temporal registration, and consequently lose the mean structure and artificially inflate the observed variance. We introduce a quantity that provides both a cost function for temporal registration and a proper distance for comparison of trajectories. This distance, in turn, is used to define statistical summaries, such as the sample means and covariances, of synchronized trajectories, and “Gaussian-type” models to capture their variability at discrete times. This distance is invariant to identical time-warpings (or temporal re-parameterizations) of trajectories. This is based on a novel mathematical representation of trajectories, termed transported square-root vector field (TSRVF), and the $L^2$ norm on the space of TSRVFs. We will illustrate this framework using four representative manifolds – $S^2$, $SE(2)$, Symmetric Positive-Definite(SPD) Matrices and shape space of planar contours – involving both simulated and real data. In particular, we will demonstrate: (1) improvements in mean structures and significant reductions in cross-sectional variances using real datasets, (2) statistical modeling for capturing variability in aligned trajectories, and (3) evaluating arbitrary trajectories under these models. Experimental results are used to demonstrate this framework in bird migration, hurricane tracking, and activity recognition in videos.

4.1 Mathematical Framework

Let $\alpha$ denote a smooth trajectory on a Riemannian manifold of interest $M$, where $M$ is endowed with a Riemannian metric $\langle \cdot, \cdot \rangle$. Let $\mathcal{M}$ denote the set of all such trajectories: $\mathcal{M} = \{ \alpha : [0, 1] \to M | \alpha \text{ is smooth} \}$. Also, define $\Gamma$ to be the set of all orientation preserving diffeomorphisms of $[0, 1]$: $\Gamma = \{ \gamma : [0, 1] \to [0, 1] | \gamma(0) = 0, \gamma(1) = 1, \gamma \text{ is a diffeomorphism} \}$. It is important to note that $\Gamma$ forms a group under the composition operation. If $\alpha$ is a trajectory on $M$, then $\alpha \circ \gamma$ is a trajectory that follows the same sequence of points as $\alpha$ but at the evolution rate governed by $\gamma$. 

41
More technically, the group $\Gamma$ acts on $\mathcal{M}$, $\mathcal{M} \times \Gamma \to \mathcal{M}$, according to $(\alpha, \gamma) = \alpha \circ \gamma$.

Given two smooth trajectories $\alpha_1, \alpha_2 \in \mathcal{M}$, we want to register points along the trajectories and compute a time-warping invariant distance between them. As mentioned earlier, the quantity given in Eqn. 1.5 would be a natural choice for this purpose but it fails for several reasons, including the fact that it is not even symmetric. Fundamentally speaking, this and other quantities used in previous literature are not appropriate for solving the registration problem because they are not measuring registration in the first place. To highlight this issue, take the registration of points between the pair $(\alpha_1, \alpha_2)$ and the pair $(\alpha_1 \circ \gamma, \alpha_2 \circ \gamma)$, for any $\gamma \in \Gamma$. It can be seen that the pairs $(\alpha_1, \alpha_2)$ and $(\alpha_1 \circ \gamma, \alpha_2 \circ \gamma)$ have exactly the same registration of points. In fact, any identical time-warping of two trajectories does not change the registration of points between them. But the quantities given in Eqns. 1.6 and 1.5 provide different values for these pairs, despite the same registration. Hence, they are not good measures of registration. We emphasize that the invariance under identical time-warping is a key property that is needed in the desired framework.

We introduce a new representation of trajectories that will be used to compare and register them. We will assume that for any two points $p, q \in \mathcal{M}$, we have an expression for parallel transporting any vector $v \in T_p(\mathcal{M})$ along the geodesic from $p$ to $q$, denoted by $(v)_{p \to q}$. As long as $p$ and $q$ do not fall in the cut loci of each other, the geodesic between them is unique and the parallel transport is well defined. Also, let $c$ be a point in $\mathcal{M}$ that we will designate as a reference point. We will assume that none of the observed trajectories pass through the cut locus of $c$ to avoid the problem mentioned above.

**Definition 1** For any smooth trajectory $\alpha \in \mathcal{M}$, define its transported square-root vector field (TSRVF) to be a parallel transport of a scaled velocity vector field of $\alpha$ to a reference point $c \in \mathcal{M}$ according to: $h_\alpha(t) = \frac{\dot{\alpha}(t)_{\alpha(t) \to c}}{\sqrt{|\dot{\alpha}(t)|}} \in T_c(\mathcal{M})$, where $| \cdot |$ denotes the norm related to the Riemannian metric on $\mathcal{M}$.

Since $\alpha$ is smooth, so is the vector field $h_\alpha$. Let $\mathcal{H} \subset T_c(\mathcal{M})^{[0,1]}$ be the set of smooth curves in $T_c(\mathcal{M})$ obtained as TSRVFs of trajectories in $\mathcal{M}$, $\mathcal{H} = \{ h_\alpha | \alpha \in \mathcal{M} \}$. If $\mathcal{M} = \mathbb{R}^n$ with the Euclidean metric then $h$ is exactly the square-root velocity function defined in [6].

The choice of reference point $c$ used in Definition 1 is important in this framework and can potentially affect the results. The choice of $c$ would typically depend on the application, the data and the manifold $\mathcal{M}$ under study. In case all the trajectories pass through a point or pass close to a point, then that point is a natural candidate for $c$. This would be true, for example, in case of hurricane tracks, if we are focused on all hurricanes starting from the same region in the Atlantic Ocean. Another remark is that instead of parallel transporting of scaled velocity vectors along geodesics, one can translate them along trajectories themselves, as was done by [31], but that requires $c$ to be a common point of all trajectories. While the choice of $c$ can, in principle, affect distances, our experiments suggest that the results of registration, distance-based clustering and classification are
Before we apply this framework for different tasks laid out earlier, we study the uniqueness of this representation of a trajectory \( \alpha \) by its TSRVF \( h_\alpha \). Can we find another trajectory \( \beta \) such that the TSRVF of \( \beta \) is \( h_\alpha \)? The answer is yes. In fact, there are an infinite number of them. For any time \( t \), let \( V_t \) be a time-varying vector field on \( M \) obtained by parallel transporting \( h_\alpha(t) \) over the whole \( M \) (except the cut locus of \( \alpha(t) \)), i.e. for any \( p \in M \), \( V_p(t) = (h_\alpha(t))_{c \to p} \). Then, define an integral curve \( \beta \) such that \( \dot{\beta}(t) = V_{\beta(t)}(t) \) with an arbitrary starting point \( \beta(0) = p_0 \in M \). One can construct a different trajectory for each point in \( M \) as a potential starting point. However, the TSRVF of all these trajectories is the same! So, the representation of \( \alpha \) by \( h_\alpha \) is not one-to-one. To handle this issue, we will define an equivalence relation: \( \alpha \sim_1 \beta \) if \( h_\alpha = h_\beta \), and form equivalence classes of the type: \( [\alpha]_1 = \{ \beta \in M | h_\alpha = h_\beta \} \). (The subscript 1 is used to distinguish this equivalence relation from a different one introduced later in the paper.) With this definition, it is easy to see that the mapping between \( M/\sim_1 \) and \( H \), given by \( [\alpha]_1 \mapsto h_\alpha \), is a bijection. Therefore, it is reasonable to compare any two equivalence classes of trajectories by comparing their corresponding TSRVFs.

Since a TSRVF is a path in \( T_c(M) \), the Riemannian structure of \( M \) can be used to compare TSRVFs as follows.

**Definition 2** Let \( \alpha_1 \) and \( \alpha_2 \) be two smooth trajectories on \( M \) and let \( h_{\alpha_1} \) and \( h_{\alpha_2} \) be the corresponding TSRVFs. Define the distance between them as:

\[
d_{h}(h_{\alpha_1}, h_{\alpha_2}) \equiv \left( \int_0^1 |h_{\alpha_1}(t) - h_{\alpha_2}(t)|^2 dt \right)^{\frac{1}{2}}.
\]

Since \( d_h \) is the standard \( L^2 \) norm, it satisfies symmetry, positive definiteness, and triangle inequality. Also, note that due to the bijective relation relation between \( M/\sim_1 \) and \( H \), one can use \( d_h \) to define a distance on \( M/\sim_1 \). The main importance of this setup – TSRVF representation and \( L^2 \) norm – comes from the following fact. If a trajectory \( \alpha \) is warped by \( \gamma \), to result in \( \alpha \circ \gamma \), the TSRVF of \( \alpha \circ \gamma \) is given by:

\[
h_{\alpha \circ \gamma}(t) = \frac{(\dot{\alpha}(\gamma(t)))_{\alpha(\gamma(t)) \to c}}{\sqrt{\left| \dot{\alpha}(\gamma(t))\right|}} \frac{\gamma(t)}{\sqrt{\left| \dot{\alpha}(\gamma(t))\right|}} = h_\alpha(\gamma(t)) \sqrt{\gamma(t)}, \text{ which is also denoted as } (h_\alpha, \gamma)(t).
\]

As stated earlier, we need a distance for registration that is invariant to identical time-warpings of trajectories. Next, we show that \( d_h \) satisfies this property.

**Theorem 1** For any \( \alpha_1, \alpha_2 \in M \) and \( \gamma \in \Gamma \), the distance \( d_h \) satisfies \( d_h(h_{\alpha_1 \circ \gamma}, h_{\alpha_2 \circ \gamma}) = d_h(h_{\alpha_1}, h_{\alpha_2}) \).

In geometric terms, this implies that the action of \( \Gamma \) on \( H \) under the \( L^2 \) metric is by isometries.
Proof: Starting from the left side, we get

\[
d_h(h_{\alpha_1 \gamma}, h_{\alpha_2 \gamma}) = \left( \int_0^1 |h_{\alpha_1}(\gamma(t)) \sqrt{\gamma'(t)} - h_{\alpha_2}(\gamma(t)) \sqrt{\gamma'(t)}|^2 dt \right)^{\frac{1}{2}}
\]

\[
= \left( \int_0^1 |h_{\alpha_1}(s) - h_{\alpha_2}(s)|^2 ds \right)^{\frac{1}{2}} = d_h(h_{\alpha_1}, h_{\alpha_2}), \text{ where } s = \gamma(t). \square
\]

Next we define a quantity that can be used as a distance between trajectories while being invariant to their temporal variability. To set up this definition, we first introduce another equivalence relation between trajectories. For any two trajectories \( \alpha_1 \) and \( \alpha_2 \), we define them to be equivalent, \( \alpha_1 \sim_2 \alpha_2 \), if there exists a sequence \( \{ \gamma_k \} \in \Gamma \) such that \( \lim_{k \to \infty} h_{(\alpha_1 \gamma_k)} = h_{\alpha_2} \); this convergence is measured under the \( \mathbb{L}^2 \) metric. Two trajectories are equivalent if the TSRVF of one can be time-warped into the TSRVF of the other using a sequence of warping. It is interesting to note that this equivalence relation includes the first one defined earlier. In other words, if \( \alpha_1 \sim_1 \alpha_2 \), this means that \( h_{\alpha_1} = h_{\alpha_2} \) and that, in turn, implies that \( \alpha_1 \sim_2 \alpha_2 \).

It can be easily checked that \( \sim_2 \) forms an equivalent relation on \( \mathcal{H} \) (and correspondingly \( \mathcal{M}/ \sim_1 \)). For a TSRVF \( h \in \mathcal{H} \), its equivalence class is \( [h]_2 = \text{closure}\{ (h, \gamma) | h \in \mathcal{H}, \gamma \in \Gamma \} \) and the set of these orbits, denoted by quotient space \( \mathcal{H}/ \sim_2 \), can be bijectively identified with the set \( \mathcal{M}/ \sim_2 \) using the mapping \( [h] \sim_2 \mapsto [\alpha]_2 \). The reason of closure is that orbits under \( \Gamma \) are not closed originally and their closure is needed to make the following definition of \( d_s \) precise. For a discussion of this issue for \( \mathcal{M} = \mathbb{R}^n \), please refer to [88]. From now onwards we will drop the subscript 2 in stating the orbits since this is the only equivalence relation used.

Now we are ready to define the quantity that will serve as both the cost function for registration and the distance for comparison. This quantity is essentially \( d_h \) measured between not the individual trajectories but their equivalence classes.

Definition 3 Define a distance \( d_s \) on \( \mathcal{H}/ \sim \) (or \( \mathcal{M}/ \sim \)) by computing the shortest \( d_h \) distance between equivalence classes in \( \mathcal{H} \):

\[
d_s([\alpha_1], [\alpha_2]) \equiv \inf_{\gamma \in \Gamma} d_h(h_{\alpha_1}, h_{\alpha_2}, \gamma) = \inf_{\gamma \in \Gamma} \left( \int_0^1 |h_{\alpha_1}(t) - h_{\alpha_2}(\gamma(t)) \sqrt{\gamma'(t)}|^2 dt \right)^{\frac{1}{2}}.
\]

Theorem 2 The distance \( d_s \) is a proper distance on \( \mathcal{H}/ \sim \).

Proof: Since the action of \( \Gamma \) is by isometries (Theorem 1) it is easy to show that \( d_s \) is symmetric. For positive definiteness, we need to show that \( d_s([\alpha_1], [\alpha_2]) = 0 \Rightarrow [\alpha_1] = [\alpha_2] \). Suppose that \( d_s([\alpha_1], [\alpha_2]) = 0 \). By definition, it follows immediately that for all \( \epsilon > 0 \), there exists a \( \gamma \in \Gamma \) such that \( d_h(h_{\alpha_1}, h_{\alpha_2}, \gamma) < \epsilon \). From this, it follows that \( h_{\alpha_1} \) is in the closure of the orbit \( h_{\alpha_2} \). Since we are assuming that orbits are closed, it follows that \( h_{\alpha_1} \in [h_{\alpha_2}] \), so \( [\alpha_1] = [\alpha_2] \).
To establish the triangle inequality, we need to prove \( d_s([h_{\alpha_1}], [h_{\alpha_3}]) \leq d_s([h_{\alpha_1}], [h_{\alpha_2}]) + d_s([h_{\alpha_2}], [h_{\alpha_3}]) \) for any \( h_{\alpha_1}, h_{\alpha_2}, h_{\alpha_3} \in \mathcal{H} \). Seeking contradiction, suppose that \( d_s([h_{\alpha_1}], [h_{\alpha_3}]) > d_s([h_{\alpha_1}], [h_{\alpha_2}]) + d_s([h_{\alpha_2}], [h_{\alpha_3}]) \). Let \( \epsilon = \frac{1}{3} (d_s([h_{\alpha_1}], [h_{\alpha_2}]) - d_s([h_{\alpha_1}], [h_{\alpha_3}]) - d_s([h_{\alpha_2}], [h_{\alpha_3}])) \); by our supposition, \( \epsilon > 0 \). From the definition of \( \epsilon \), it follows that \( d_s([h_{\alpha_1}], [h_{\alpha_3}]) = d_s([h_{\alpha_1}], [h_{\alpha_2}]) + d_s([h_{\alpha_2}], [h_{\alpha_3}]) + 3\epsilon \). By the definition of \( d_s \), we can choose \( \gamma_1, \gamma_2 \in \Gamma \), such that \( d_h((h_{\alpha_1}, \gamma_1), h_{\alpha_2}) \leq d_s([h_{\alpha_1}], [h_{\alpha_2}]) + \epsilon \) and \( d_h(h_{\alpha_2}, (h_{\alpha_3}, \gamma_2)) \leq d_s([h_{\alpha_2}], [h_{\alpha_3}]) + \epsilon \). Now by the triangle inequality for \( d_h \), we know that \( d_h((h_{\alpha_1}, \gamma_1), (h_{\alpha_3}, \gamma_2)) \leq d_h((h_{\alpha_1}, \gamma_1), h_{\alpha_2}) + d_h(h_{\alpha_2}, (h_{\alpha_3}, \gamma_2)) \leq d_s([h_{\alpha_1}], [h_{\alpha_2}]) + d_s([h_{\alpha_2}], [h_{\alpha_3}]) + 2\epsilon \). It follows that \( d_s([h_{\alpha_1}], [h_{\alpha_3}]) \leq d_s([h_{\alpha_1}], [h_{\alpha_2}]) + d_s([h_{\alpha_2}], [h_{\alpha_3}]) + 2\epsilon \). But this contradicts that fact that \( d_s([h_{\alpha_1}], [h_{\alpha_3}]) = d_s([h_{\alpha_1}], [h_{\alpha_2}]) + d_s([h_{\alpha_2}], [h_{\alpha_3}]) + 3\epsilon \). Hence our supposition that \( d_s([h_{\alpha_1}], [h_{\alpha_3}]) > d_s([h_{\alpha_1}], [h_{\alpha_2}]) + d_s([h_{\alpha_2}], [h_{\alpha_3}]) \) must be false. The triangle inequality follows. Q.E.D.

The minimization over \( \Gamma \) in Eqn. 4.1 can be solved in practice using the dynamic programming (DP) algorithm [89]. Here one samples the interval \([0, 1]\) using \( T \) discrete points and then restricts to only piecewise linear \( \gamma \) that passes through that \( T \times T \) grid. The search for the optimal trajectory on this grid is accomplished in \( O(T^2) \) steps. While it is possible that the optimal mapping \( \gamma^* \) lies on the boundary of \( \Gamma \), the DP algorithm provides an approximation using a piecewise linear map on a finite grid.

### 4.1.1 Metric-Based Comparison of Trajectories

Our goal of warping-invariant comparisons of trajectories is achieved using \( d_s \). For any \( \gamma_1, \gamma_2 \in \Gamma \), we have:

\[
d_s([\alpha_1 \circ \gamma_1], [\alpha_2 \circ \gamma_2]) = \inf_{\gamma \in \Gamma} d_h(h_{\alpha_1 \circ \gamma_1}, h_{\alpha_2 \circ \gamma_2})
\]

Theorem 1 \( \inf_{\gamma \in \Gamma} d_h(h_{\alpha_1}, h_{\alpha_2 \circ \gamma_2 \circ \gamma_{-1}}) = \inf_{\gamma \in \Gamma} d_h(h_{\alpha_1}, h_{\alpha_2 \circ \gamma}) = d_s([\alpha_1], [\alpha_2]). \)

The second to last equality comes from the group structure of \( \Gamma \). Examples of this metric are presented later.

### 4.1.2 Pairwise Temporal Registration of Trajectories

The next goal is to perform registration of points along trajectories. Let the optimal warping be:

\[
\gamma^* = \arg\min_{\gamma \in \Gamma} \left( \int_0^1 |h_{\alpha_1}(t) - h_{\alpha_2}(\gamma(t))\sqrt{\gamma'(t)}|^2 dt \right)^{\frac{1}{2}}. \tag{4.2}
\]

This solves for the optimal registration between \( \alpha_1 \) and \( \alpha_2 \). It says that the point \( \alpha_1(t) \) on the first trajectory is optimally matched to the point \( \alpha_2(\gamma^*(t)) \) on the second trajectory.
If we compare Eqn. 4.2 with Eqn. 1.5, we can immediately see the advantages of the proposed framework. Both equations present a registration problem between $\alpha_1$ and $\alpha_2$, but only the minimum value resulted from Eqn. 4.2 is a proper distance (and hence symmetric). Also, in Eqn. 1.5, we have two separate terms for matching and regularization, with an arbitrary weight $\lambda$, but in Eqn. 4.2 the two terms have been merged into a single natural form. The change in TSR VF $h$ due to the time-warping of $\alpha$ by $\gamma$ is given by $(h, \gamma) = (h \circ \gamma)\sqrt{\dot{\gamma}}$, and the distance $d_s$ is based on these warped TSRVFs. It turns out that the term $\sqrt{\dot{\gamma}}$ provides an intrinsic regularization on $\gamma$ in the matching process. This term provides an elastic penalty against excessive warping since $\dot{\gamma}$ becomes large at those places. Lastly, the optimal registration in Eqn. 4.2 remains the same if we change the order of the input functions. In that case, we simply get the inverse, $(\gamma^*)^{-1}$, as the optimal warping function. That is, the registration process is inverse consistent!

### 4.1.3 Summarization and Registration of Multiple Trajectories

An additional advantage of this framework is that one can compute an average of several trajectories and use it as a template for future classification. Furthermore, this template can, in turn, be used for registering multiple trajectories. We will use the notion of the Karcher mean to define and compute average trajectories. Given a set of sample trajectories $\alpha_1, \ldots, \alpha_n$ on $M$, their Karcher mean is defined by: $\mu = \arg\min_\alpha \sum_{i=1}^n d_s([\alpha], [\alpha_i])^2$. Note that $\mu$ is actually an equivalence class of trajectories and one can select any element of this mean class to help in alignment of multiple trajectories. The standard algorithm to compute the Karcher mean [90], adapted to this problem is given as follows:

**Algorithm 3 Karcher Mean of Multiple Trajectories**

1. Initialization step: Select $\mu$ to be one of the original trajectories.

2. Align each $\alpha_i$, $i = 1, \ldots, n$ to $\mu$, using Eqn. 4.2. That is, solve for $\gamma_i^*$ using the DP algorithm and set $\tilde{\alpha}_i = \alpha_i \circ \gamma_i^*$.

3. Compute TSRVFs of the warped trajectories, $h_{\tilde{\alpha}_i}$, $i = 1, 2, \ldots, n$, and form a curve in $T_c(M)$ according to: $\bar{h}(t) = \frac{1}{n} \sum_{i=1}^n h_{\tilde{\alpha}_i}(t)$.

4. Define $\tilde{\mu}$ to be the integral curve associated with a time-varying vector field on $M$ generated using $\bar{h}$, i.e. $\frac{d\tilde{\mu}(t)}{dt} = \bar{h}(t)_{c \rightarrow \tilde{\mu}(t)}$, with the initial condition $\tilde{\mu}(0)$ computed separately as the Karcher Mean of $\{\alpha_i(0)\}$s.

5. Compute $E = \sum_{i=1}^n d_s([\tilde{\mu}], [\alpha_i])^2 = \sum_{i=1}^n d_h(h_{\tilde{\mu}}, h_{\tilde{\alpha}_i})^2$ and check it for convergence. If not converged, set $\mu = \tilde{\mu}$ and return to step 2.
At any iteration, the original mean is given by $\mu$, the optimal warping from $\alpha_i$ to $\mu$ by $\gamma_i^*\alpha$, and the updated mean by $\bar{\mu}$. Then,

$$\sum_{i=1}^{n} d_s([\mu], [\alpha_i])^2 = \sum_{i=1}^{n} d_h(h_\mu, h_{\bar{\alpha}_i})^2 = \sum_{i=1}^{n} \int_0^1 |h_\mu(t) - h_{\bar{\alpha}_i}(t)|^2 dt \geq \sum_{i=1}^{n} \int_0^1 |\bar{h}(t) - h_{\bar{\alpha}_i}(t)|^2 dt = \sum_{i=1}^{n} d_h(h_\bar{\mu}, h_{\bar{\alpha}_i})^2 \geq \sum_{i=1}^{n} d_s([\bar{\mu}], [\alpha_i])^2.$$ 

Thus, the cost function decreases iteratively and as zero is a natural lower bound, $\sum_{i=1}^{n} d_s([\mu], [\alpha_i])^2$ will always converge. This algorithm provides two outputs: an average trajectory denoted by the final $\mu$ and the set of aligned trajectories $\bar{\alpha}_i$’s. This actually solves the problem of aligning multiple trajectories too.

For computing and analyzing the second and higher moments of a sample trajectory, the tangent space $T_{\mu(t)}(M)$, for $t \in [0, 1]$, is used. This is convenient because it is a vector space and one can apply more traditional methods here. First, for each aligned trajectory $\bar{\alpha}_i(t)$ at time $t$, the vector $v_i(t) \in T_{\mu(t)}(M)$ is computed such that a geodesic that goes from $\mu(t)$ to $\bar{\alpha}_i(t)$ in unit time has the initial velocity $v_i(t)$. This is also called the shooting vector from $\mu(t)$ to $\bar{\alpha}_i(t)$. Let $\hat{K}(t)$ be the sample covariance matrix of all the shooting vectors from $\mu(t)$ to $\bar{\alpha}_i(t)$. The sample Karcher covariance at time $t$ is given by $\hat{K}(t) = \frac{1}{n} \sum_{i=1}^{n} v_i(t)v_i(t)^T$, with the trace $\hat{\rho}(t) = \text{trace}(\hat{K}(t))$. This $\hat{\rho}(t)$ represents a quantification of the cross-sectional variance, as a function of $t$, and can be used to study the level of alignment of trajectories. Also, for capturing the essential variability in the data, one can perform Principal Component Analysis (PCA) of the shooting vectors. The basic idea is to compute the Singular Value Decomposition (SVD) $\hat{K}(t) = U(t)\Sigma(t)U^T(t)$, where $U(t)$ is an orthogonal matrix and $\Sigma(t)$ is the diagonal matrix of singular values. Assuming that the entries along the diagonal in $\Sigma(t)$ are organized in a non-increasing order, $U_1(t), U_2(t)$ etc. represent the dominant directions of variability in the data.

### 4.1.4 Modeling and Evaluation of Trajectories

An important use of means and covariances of trajectories is in devising probability models for capturing the observed statistical variability, and for using these models in evaluating $p$-values of future observations. By $p$-values we mean the proportion of random trajectories that will have lower probability density under a given model when compared to the test trajectory. Several models are possible in this situation but since our main focus is on temporal registration of trajectories, we will choose a simple model to demonstrate our ideas. After the registration, we treat a trajectory $\alpha$ as a discrete-time process, composed of $m$ points as $\{\alpha(t_1), \alpha(t_2), \ldots, \alpha(t_m)\}$, for a fixed partition $\{0 = t_1, t_2, \ldots, t_m = 1\}$ of $[0, 1]$. Given the mean and the covariance at each $t_j$, we model the points $\alpha(t_j) \in M, j = 1, 2, \ldots, m$ independently, and obtain the joint density by taking the product.
The difficulty in this step comes from the fact $M$ is a nonlinear manifold but we can use the tangent space $T_{\mu(t_j)}(M)$, instead, to impose a probability model since this is a vector space. We will impose a multivariate normal density on the tangent vector $v(t_j) = \exp^{-1}_{\mu(t_j)}(\alpha(t_j))$, with mean zero and variance given by $\hat{K}(t_j)$ (as defined above). It is analogous to the model of additive white gaussian noise when $M = \mathbb{R}$. Then, for any trajectory $\alpha$, one can compute a joint probability of the full trajectory as $P(\alpha) = \prod_{m=1}^{m} f(\alpha(t_j)) \equiv \prod_{j=1}^{m} N(v(t_j); 0, \hat{K}(t_j))$. This model is potentially useful for many situations: (1) It can be used to simulate new trajectories via random sampling. Given $\{\mu(t_j), \hat{K}(t_j) \mid t \in [0, 1] \}$, we can simulate the tangent vectors and compute the corresponding trajectory points $\alpha(t_j)$, for the desired $t_j$. (2) Given any new trajectory $\alpha_{new}$, we evaluate its $p$-value under the model imposed, using parametric bootstrap as follows. We simulate a large number, say $N = 10000$, trajectories from the model, and denote them as $X_i, i = 1, 2, \ldots, N$. Then, we estimate the $p$-value of $\alpha_{new}$ as $p(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(P(X_i) < P(\alpha))$.

Since we are interested in studying the effects of temporal registration, we demonstrate these ideas with the following experiment. We take a set of trajectories as the given data and estimate model parameters in both cases: with and without temporal alignment. Then, we treat each of the observed trajectories as a “test” trajectory and compute its $p$-value under the two cases. Note that in case of temporal alignment, the test trajectory is first temporally aligned with the mean trajectory $\mu$ while there is no alignment in the other case.

In the following sections, we will take four specific examples of $M$, define the parallel transport on these manifolds and present experimental results to validate our framework.

### 4.2 Analysis of Trajectories on Nonlinear Manifolds

#### 4.2.1 Unit Sphere $S^2$

To illustrate this framework, in a simple setting, we start with $M = S^2$, with the standard Euclidean Riemannian metric. For any two points $p, q \in S^2$ and a tangent vector $v \in T_p(S^2)$, the parallel transport $(v)_{p \rightarrow q}$ along the shortest geodesic (i.e. great circle) from $p$ to $q$ is given by $v - \frac{2(v \cdot q)}{|p+q|^2} (p + q)$.

- **Registration and clustering of Trajectories:** As mentioned earlier, for any two trajectories on $S^2$, we can use their TSRVFs and DP algorithm in Eqn. 4.2 to find the optimal registration between them. In Fig. 4.1 we show one example of registering such trajectories. The parameterization of trajectories is displayed using colors. In the top row, the left column shows the given trajectories $\alpha_1$ and $\alpha_2$, the middle column shows $\alpha_1$ and $\alpha_2 \circ \gamma^{c}$ and the right column shows $\gamma^{c}$ using $c = [0, 0, 1]$. The correspondences between two trajectories are depicted by black lines connecting points along them. Due to optimization of $\gamma$ in Eqn. 4.2, the $d_h$ value between them reduces from 1.67 to 0.36 and the correspondences become more natural after
the alignment. We also try different choices of $c$’s ($c = [0, 0, -1], [-1, 0, 0], [0, 1, 0]$). The registration results are very close despite different $c$’s as shown in the bottom row.

In addition, we simulate two trajectories, one with two bumps and another with three bumps. We apply randomly generated $\gamma$’s to each and build a large synthetic dataset of trajectories. There are 20 trajectories in total with 2 classes (one with two bumps indexed from 1 to 10 and another with three bumps indexed from 11 to 20). We compute two distance matrices and display MDS plots, with and without registration in Fig. 4.2. It shows clearly that our method has a better clustering and a clear pattern after registration.

- **Averaging of Trajectories**: Next we present two examples of averaging multiple trajectories on $\mathbb{S}^2$ in Fig. 4.3. In each example, the average of these trajectories obtained using Algorithm 3 is denoted by black. Both examples show that the average after temporal alignment retains the same structure as the given trajectories, i.e., two bumps in the top panel and common part shared in the bottom panel. The corresponding energies $E$, defined in Step 5 of Algorithm 3, converge in both examples.

In the following, we will apply it to two specific applications: bird migration data and hurricane tracks and show how the cross-sectional variance of mean trajectories is reduced by registration. We use the mean of starting points of trajectories as the reference point $c$ in Definition 1 for both applications.
Figure 4.2: Comparison of clustering between without and with registration.

Application in Bird Migration Data: This dataset has 35 migration trajectories of Swainson’s Hawk, measured from 1995 to 1997, each having geographic coordinates measured at some random times. Several sample paths are shown at the top row in Fig. 4.4(a). In the bottom panel of Fig. 4.4(a), we show the optimal warping functions \( \{\gamma_i^*\} \) used in aligning them and this clearly highlights a significant temporal variation present in the data. In Fig. 4.4(b) and (c), we show the Karcher mean \( \mu \) and the cross-sectional variance \( \hat{\rho} \) without and with registration, respectively. In the top row, \( \mu \) is displayed using colors, where red areas correspond to higher variability in the given data. In the bottom row, the principal modes of variation are displayed using ellipses on tangent spaces. We use the first and second principal tangential directions as the major and minor axes of ellipses, and the corresponding singular values as their lengths. We observe that: (1) the mean after registration better preserves the shapes of trajectories, and (2) the variance ellipses before registration have major axis along the trajectory while the ellipses after registration exhibit the actual variability in the data. Most of the variability after registration is limited to the top end where the original trajectories indeed had differences. The top row of Fig. 4.5(a) shows a decrease in the function \( \hat{\rho} \) due to the registration.

Next we construct a “Gaussian-type” model for these trajectories using estimated summaries for two cases (with and without temporal registration), as described previously, and compute \( p \)-values of individual trajectories using Monte Carlo simulation. The results are shown in the bottom of Fig. 4.5(a), where we note a general increase in the \( p \)-values for the original trajectories after the
alignment. This is attributed to a reduced variance in the model due to temporal alignment and the resulting movement of individual samples closer to the mean values.

**Application in Hurricane Tracks:** We choose two subsets of Atlantic Tracks File 1851-2011, available on the National Hurricane Center website \(^1\). The first subset has 10 tracks and another has 7 tracks, with observations at six-hour separation. We show the data, their Karcher mean and variance without and with registration in Fig. 4.6 for each subset. The decrease in the value of \(\hat{\rho}\) is shown in the top of Fig. 4.5(b) and (c). Although the decrease here is not as large as the previous example, we observe about 20% reduction in \(\hat{\rho}\) due to registration. The \(p\)-values of tracks, without and with registration, are plotted in the bottom of (b) and (c).

### 4.2.2 Special Euclidean Group \(SE(2)\)

Here we study the problem of classifying vehicle trajectories into broad motion patterns using data obtained from traffic videos. While the general motion of a vehicle at a traffic intersection is predictable – left turn, right turn, U turn, or straight line, the travel speeds of vehicles may be different in different instances due to traffic variations. Since we are interested in tracking position

---

\(^1\)http://www.nhc.noaa.gov/pastall.shtml
and orientation of a vehicle, we will consider individual tracks as parameterized trajectories on $SE(2)$, which is a semidirect product of $SO(2)$ and $\mathbb{R}^2$, i.e. $SE(2) = SO(2) \ltimes \mathbb{R}^2$. For the rotation component $O \in SO(2)$ and tangent vectors $X_1, X_2 \in T_O(SO(2))$, the standard Riemannian metric is given by $\langle X_1, X_2 \rangle = \text{trace}(X_1^T X_2)$, while we use the Euclidean metric for $\mathbb{R}^2$. We choose the
rotation component of $c$ as the identity matrix and the translation component as $[0, 0]$. We found that the results of registration, clustering and classification are quite stable with respect to different choices of $c$. For a tangent vector $W \in T_O(SO(2))$, the parallel transport of $W$ from $O$ to $I_{2 \times 2}$ is $O^T W$. The formulae for the $\mathbb{R}^2$ component are standard.

- **Registration**: The data for this experiment comes from traffic videos available at the Image Sequence Server website\(^2\). In Fig. 4.7(a), we show an example trajectory for each of the three classes: right turn (first panel), left turn (second panel) and straight line (third panel). In this small experiment, the data includes 14 trajectories with 5 trajectories corresponding to right turn indexed from 1 to 5, 5 trajectories of straight line indexed from 6 to 10, and 4 trajectories of left turn indexed from 11 to 14. In Fig. 4.8, we show two examples of temporally aligning these trajectories. In Example 1, we first choose a trajectory as $\alpha_1$, apply to it a simulated $\gamma$ and consider this time-warped trajectory as $\alpha_2$. The right plot of $\gamma^{-1}$ (dashed) and $\gamma^*$ shows that we are able to recover the simulated time-warping using the framework. In Example 2, we align two trajectories coming from different classes. In this case the distance $d_h$ between the trajectories is large, since they are from different classes, but it decreases from 14.2 to 10.8 after registration. Furthermore, the registration result is quite intuitive since it matches as much of the common features (straight line part) as possible.

\(^2\)http://i21www.ira.uka.de/image_sequences/
Figure 4.7: (a): Real trajectories on SE(2) obtained from a traffic video. (b): Trajectories used for clustering.

Example 1

Example 2

Figure 4.8: Registration of trajectories on SE(2).

- **Clustering and Classification**: Now we study the effects of temporal alignment on clustering and classification results. In the first example, we introduce simple speed variations in the vehicle motions; these variations represent either fast-slow or slow-fast movements of a vehicle and apply them randomly to the 14 given trajectories, shown in Fig. 4.7(b). In Fig. 4.9, we display the resulting pairwise distance matrices, multidimensional scaling (MDS) plots and dendrograms computed with and without alignment. It can be clearly seen that the alignment helps in revealing the underlying patterns of the data and greatly improves the performance.
In the second experiment, we introduce more drastic, random speed-variations, corresponding to multiple stop-and-go patterns of a vehicle. We again apply them to the given trajectories and compute the distance matrices with and without temporal alignment. In Table 4.1, we report the classification performances based on 1-, 3-, 5-nearest neighbor (NN) classifiers. We note that the method described in this paper produces superior classification of driving patterns. In particular, we can achieve a 100% classification rate using 1-NN classifier.

4.2.3 Symmetric Positive-Definite Matrices

In a number of applications, there has been a great interest in statistical analysis of SPD matrices using the Riemannian geometry of the underlying tensor space (see [19, 91, 17, 18]). Let $\bar{\mathcal{P}}(n)$ be the space of $n \times n$ SPD matrices and $\mathcal{P}(n) = \{ P \mid P \in \bar{\mathcal{P}}(n) \text{ and } \det(P) = 1 \}$. The identity
matrix $I_{3 \times 3}$ is chosen as $c$ in Definition 1. For any $\tilde{P} \in \tilde{P}(n)$, since $\det(\tilde{P}) > 0$, by writing $\tilde{P} = (P, \frac{1}{n} \log(\det(\tilde{P})))$, we see that $\tilde{P}(n)$ is the product space of $\mathcal{P}(n)$ and $\mathbb{R}$. If $\tilde{V}$ is a tangent vector to $\tilde{P}(n)$ at $\tilde{P}$, where $\tilde{P}$ is identified with $\tilde{P} = (P, x)$ and $x = \frac{1}{n} \log(\det(\tilde{P}))$, we can express $\tilde{V}$ as $\tilde{V} = (V, v)$ with $V$ being a tangent vector of $\mathcal{P}(n)$ at $P$. The parallel transport of $\tilde{V}$ is the parallel transport of each of its two components in the corresponding spaces. The parallel transport of $V$ from $P$ to $I_{3 \times 3}$ is $P^{-1}V$ and the transport of $v$ is itself.

- **Registration of Simulated Trajectories**: We visualize each SPD matrix as an ellipsoid. Two examples in Fig. 4.10 show that the algorithm can register such trajectories very well.

- **Statistical Summary of Real Trajectories**: Next, we apply our method on real data. The data includes 12 fibers, denoted as $X_i, i = 1 \ldots 12$, shown in the middle of Fig. 4.11 and the corresponding trajectories of SPD matrices, denoted as $\alpha_i, i = 1 \ldots 12$, shown on the right. These 12 fibers are representative projection pathways passing through internal capsule, extracted from 12 subjects. We compute the mean of fibers in two different ways. First, consider 12 fibers as open curves, we computed a mean fiber using the method in [92], shown in Fig. 4.12(a). [92] provides tools for statistical modeling of shapes of 3D curves in conjunction with various features such as rotation and scale. Second, we computed a mean tensor-based path using Algorithm 3, denoted as $\mu_\alpha$, then for each $\alpha_i$, find the registration $\gamma_i^*$ in Eqn 4.2. Apply $\gamma_i^*$ to the fiber $X_i$, and find the mean fiber path, shown in Fig. 4.12(b). The two mean fibers look fairly similar. But there’s still some local differences. Through zooming in
Figure 4.11: Data acquisition: (a) projection pathways; (b) and (c) 12 fibers and corresponding SPD trajectories.

Figure 4.12: Summary of fibers: (a) and (b): Karcher mean computed in two different ways; (c): \( \hat{\rho} \).

the bottom of mean fiber in (b), there is an indent which could suggest more structure using tensor-based approach. The cross-sectional variance \( \hat{\rho} \) using tensor-based registration on the right in Fig. 4.12(c) is smaller than using elastic shape analysis in general except that the variance near the lower end, which connects the spinal cord, is larger. This is because diffusion imaging of the spinal cord is highly susceptible to heart beating, which creates quite some difficulties for accurate fiber tracking.

4.2.4 Shape Space of Planar Contours

Motivated by the problem of analyzing human activities using video data, we are interested in alignment, comparison, and averaging of trajectories on the shape space of planar, closed curves.
There are several mathematical representations available for this analysis, and we will use the representation used in [6]. The benefits of using this representation over other methods are also presented in the paper. Here is a brief description of that method. Let \( \beta : S^1 \mapsto \mathbb{R}^2 \) denote a planar closed curve. Its corresponding \( q \)-function is defined as \( q(s) = \frac{\dot{\beta}(s)}{\sqrt{|\dot{\beta}(s)|}}, s \in S^1 \). A major advantage of using \( q \)-functions to represent shapes of curves is that the translation variability is automatically removed (\( q \) only depends on \( \dot{\beta} \)). To remove the scaling variability, we re-scale all curves to be of unit length. This restriction translates to the following condition for \( q \)-functions:

\[
\int_{S^1} |q(s)| ds = 1.
\]

Therefore, the \( q \)-functions associated with unit length curves are elements of a unit hypersphere in the Hilbert space \( L^2(S^1, \mathbb{R}^2) \). In order to study shapes of closed curves, we impose an additional condition, which ensures that the curve starts and ends at the same point. This condition is given by:

\[
\int_{S^1} q(s) q(s) ds = 0.
\]

Using these two conditions and the \( q \)-function representation, we can define the pre-shape space of unit length, closed curves as

\[
C = \{ q \in L^2(S^1, \mathbb{R}^2) | \int_{S^1} |q(s)|^2 ds = 1, \int_{S^1} q(s) q(s) ds = 0 \}.
\]

The shape space of these curves is obtained by removing the re-parameterization group \( \Psi \), the set of diffeomorphisms from \( S^1 \) to itself, and rotation, i.e. \( S = C / (\Psi \times SO(2)) \). A unit circle is used as the standard shape and \( c \) in Definition 1 is given by its \( q \)-representation. For algorithms on computing parallel transports of tangent vectors along geodesic trajectories, we refer the reader to [6].

To illustrate our framework, we apply it to real sequences in the UMD common activities dataset. We use a subset of 8 classes from this dataset with 10 instances in each class. Each instance consists of 80 consecutive planar closed curves. As a first step, we down-sample each of these trajectories to 17 shapes.

- **Registration**: First, we choose a sequence from the dataset as \( \alpha_1 \), apply it with a simulated \( \gamma \) and consider this time-warped trajectory as \( \alpha_2 \). Using our registration framework we are able to recover the simulated \( \gamma \), as shown in Example 1 in Fig. 4.13. Example 2 shows another result of aligning two real trajectories coming from the same activity class. The distance \( d_h \) between the two trajectories decreases from 4.27 to 3.26. In each example, we show \( \alpha_1, \alpha_2 \), the temporally aligned trajectory \( \alpha_2 \circ \gamma^* \), the estimated \( \gamma^* \) (solid) in the right. For Example 1, we also plot the simulated \( \gamma^{-1} \) (dashed) on top of \( \gamma^* \). Furthermore, in both examples, the distance \( d_h(\alpha_1, \alpha_2) \) is experimentally verified to be the same as \( d_h(\alpha_2, \alpha_1) \), which shows the symmetric property of the metric.

- **Statistical Summaries**: We demonstrate an example of averaging and registration of multiple trajectories using Algorithm 3 in Fig. 4.14. The aligned sample trajectories within the same class are much closer to each other than before temporal alignment. The energy when computing the Karcher mean converges quickly, as shown at the left bottom corner in Fig. 4.14. The right bottom plot shows that the cross-sectional variance \( \hat{\rho} \) has a significant reduction after temporal registration.

58
Classification: Since our metric is defined in terms of a vector field that depends only on $\dot{\alpha}(t)$, and is invariant to the “translation” of $\alpha$ on $S$, we need to include information about the translation for activity classification. We do this by incorporating the Riemannian distance between the first point of each trajectory, i.e., $d(\alpha_1(0), \alpha_2(0))$. We first compute two distance matrices, $D_1$ using Eqn. 4.1 and $D_2$ using $d(\alpha_1(0), \alpha_2(0))$. Then we compute a weighted distance matrix $D = wD_1 + (1-w)D_2$ by optimizing over a set of weights $w \in [0,1]$ based on the leave-one-out nearest neighbor classification rate. In Fig. 4.15, the distance matrix of $D$ for $w = 0.98$ is plotted on the left. It shows the obvious blocks corresponding to each activity class. The resulting classification performance is 95% compared to only 87.5% when time-warping is ignored. We also compute the distance matrix $\tilde{D}$ using the method in [7]. The middle plot in Fig. 4.15 shows the matrix $100 \times |\tilde{D} - \tilde{D}^T|/\max|\tilde{D}|$ to highlight the fact that the distance used in [7] is not symmetric since the image contains values much larger than 0. This results in rather arbitrary solutions to the correspondence problem: A matched to B gives a different correspondence than B matched to A. Also, the leave-one-out nearest neighbor classification rate based on $\tilde{D}$ and the same setup is 97%. The right panel in Fig. 4.15 shows the increase of the classification rate as we increase the number of nearest neighbors considered. This shows that we can get a comparable result as the method in [7].
while using less information (17 out of 80 shapes in each sequence) and still have a formal metric. While [7] happens to provide a good classification result, its lack of being a proper distance does not allow us to compute mean trajectory or dominant modes of variations of trajectories. Many times in computer vision a certain ad hoc similarity measure works well for database retrieval or pairwise comparisons but it cannot be used for developing something more ambitious. For example, a Riemannian framework gives us the requisite tools, such as the ability to compute proper distances, summary statistics, and even generative models.
CHAPTER 5

ANOTHER METHOD FOR ANALYSIS OF
TRAJECTORIES ON $S^2$

In our previous work, we have studied the problem of statistically analyzing trajectories that take values on nonlinear Riemannian manifolds and are observed under arbitrary temporal evolutions. We have introduced a quantity that provides both a cost function for temporal registration and a proper distance for comparison of trajectories. An essential property of this distance is that it is invariant to identical time-warpings (or temporal re-parameterizations) of trajectories. This results from a novel mathematical representation of trajectories, termed transported square-root vector field (TSRVF), and the $L^2$ norm on the space of TSRVF. The space of TSRVF is a tangent space at a reference point on the underlying manifold. For different applications with the different underlying spaces, the choice of the reference point becomes an important issue. We would like to have a framework that has the same nice properties while avoiding the issue mentioned above. In this chapter, we focus on the two dimensional unit sphere $S^2$. We define a new representation of trajectories on $S^2$, and a proper distance invariant to time-warpings. In the end, we show some preliminary results of geodesics between trajectories.

5.1 Mathematical framework

Let $\alpha$ denote a smooth trajectory on $S^2$ and $\mathcal{M}$ denote the set of all such trajectories: $\mathcal{M} = \{\alpha : [0, 1] \to S^2|\alpha \text{ is smooth}\}$. Also, define $\Gamma$ to be the set of all orientation preserving diffeomorphisms of $[0, 1]$: $\Gamma = \{\gamma : [0, 1] \to [0, 1]|\gamma(0) = 0, \gamma(1) = 1, \gamma \text{ is a diffeomorphism}\}$. It is important to note that $\Gamma$ forms a group under the composition operation. If $\alpha$ is a trajectory on $S^2$, then $\alpha \circ \gamma$ is a trajectory that follows the same sequence of points as $\alpha$ but at the evolution rate governed by $\gamma$. More technically, the group $\Gamma$ acts on $\mathcal{M}$, $\mathcal{M} \times \Gamma \to \mathcal{M}$, according to $(\alpha, \gamma) = \alpha \circ \gamma$.

We introduce a new representation of trajectories that will be used to compare and register them. Given a trajectory $\alpha$, we will assume that for any two points $\alpha(t_1), \alpha(t_2) \in S^2, t_1 \neq t_2$, we have an expression for parallel transporting any vector $v \in T_{\alpha(t_1)}(S^2)$ along $\alpha$ from $\alpha(t_1)$ to $\alpha(t_2)$, denoted
by \((v)_{\alpha(t_1)} \rightarrow (v)_{\alpha(t_2)}\).

**Definition 4** Let \(\alpha_p : [0, 1] \rightarrow S^2\) denote a smooth trajectory starting with \(\alpha_p(0) = p\). Given a trajectory \(\alpha_p\), one can define its transported square-root vector field (TSRVF) to be a scaled parallel transport of the vector field on \(\alpha\) to the starting point \(p\) according to: 
\[\dot{q}(t) = \frac{\dot{\alpha}(t)}{\sqrt{|\dot{\alpha}(t)|}} \quad \text{in} \quad T_p(S^2),\]
where \(| \cdot |\) denotes the norm that is defined through the Riemannian metric on \(S^2\).

Now, we have the set of trajectories on the tangent space at \(p\), denoted as \(C_p = L^2([0, 1], T_p(S^2))\). The space of interest becomes a vector bundle \(C = \bigoplus_{p \in S^n} L^2([0, 1], T_p(S^2))\). The TSRVF representation is not bijective. However, one can reconstruct the trajectory from its TSRVF and a given starting point as follows:

**Algorithm 4 Covariant integral of \(q\) along \(\alpha\).** Given a TSRVF \(q\) sampled at \(T\) times \(\{t\delta | t = 0, 1, 2, \ldots, T\}, \delta = 1/T\), and the starting point \(p\):

1. Set \(\alpha(0) = p\).
2. Compute \(\alpha(\delta) = \exp_{\alpha(0)}(\delta q(0) \| q(0) \|)\).
3. For \(t = 1, 2, \ldots, T - 1\),
   
   (a) Parallel transport \(q(t\delta)\) to \(\alpha(t\delta)\) along the current trajectory \(\{\alpha(0), \alpha(\delta), \ldots, \alpha(t\delta)\}\); call it \(q^{\parallel}(t\delta)\).

   (b) Compute \(\alpha(t + 1)\delta = \exp_{\alpha(t\delta)}(\delta q^{\parallel}(t\delta) \| q^{\parallel}(t\delta) \|)\), where \(\exp\) denotes the exponential map.

This covariant integral results in a trajectory \(\alpha\) whose TSRVF is \(q\).

Given two arbitrary trajectories \(\alpha_1, \alpha_2\), we are interested in finding the geodesic between them and the geodesic distance.

**Lemma 1** The starting points of geodesic trajectories between \(\alpha_1\) and \(\alpha_2\) form a circular arch on \(S^2\) connecting \(p_1\) and \(p_2\), the starting points of \(\alpha_1\) and \(\alpha_2\) respectively.

The proof is given in A.1. Let \(q_1, q_2 \in \mathbb{C}\) denote the corresponding TSRVF of \(\alpha_1, \alpha_2\) and \(\beta(\tau), \tau \in [0, 1]\) denote a circular arch of length \(l_{\beta} = \int_0^1 \| \dot{\beta}(\tau) \| d\tau\). The parallel transport of \(q_1\) along \(\beta\) to the tangent space \(T_{p_2(0)}S^2\) is denoted as \(q_{1, \beta}^{\parallel}\). The optimal arch \(\hat{\beta}\) is given by:

\[\hat{\beta} = \arg\min_{\beta} l_{\beta}^2 + \int_0^1 \| q_{1, \beta}(t) - q_2(t) \|^2 dt .\quad (5.1)\]

To solve for the minimization problem in Eqn. 5.1, we have to find all circular arcs from \(p_1\) and \(p_2\) on \(S^2\), where \(p_1\) and \(p_2\) denote the starting points of \(\alpha_1\) and \(\alpha_2\) respectively. Here is one idea.
Algorithm 5 Circular arcs from \( p_1 \) to \( p_2 \). Generate a unit vector \( v_1 \) on the tangent space \( T_{p_1}(S^2) \) and a unit vector \( v_2 \) on the tangent space \( T_{p_2}(S^2) \). For each \( \theta \in [0, 2\pi] \),

1. Compute the matrix for a rotation by an angle of \( \theta \) about an axis in the direction of \( p_2 \), given by \( R = I \cos \theta + \sin \theta \left[p_2\right]_{\times} + (1 - \cos \theta) \left[p_2\right]_{\times} \left[p_2\right]_{\times} \), where \( \left[p_2\right]_{\times} \) is the cross product matrix of \( p_2 \), \( \otimes \) is the tensor product and \( I \) is the identity matrix.

2. Compute two frames \( f_1 = [p_1, v_1, w_1] \) and \( f_2 = [p_2, Rv_2, w_2] \), where \( w_1 \) is the cross product of \( p_1 \) and \( v_1 \) and \( w_2 \) is the cross product of \( p_2 \) and \( Rv_2 \).

3. Generate \( \beta_\theta(\tau) = e^{\tau A} p_1 \), \( \tau \in [0, 1] \), where \( A = \log m (f_2 f_1^T) \).

In this way, one can generate all circular arcs as \( \beta_\theta \)'s. These arcs can be treated as a one-parameter family w.r.t. \( \theta \in [0, 2\pi] \). The optimization problem in Eqn. 5.1 now becomes:

\[
\hat{\beta} = \arg\min_{\theta \in [0, 2\pi]} t^2_{\beta_{\theta}} + \int_0^1 \|q_1(\beta_{\theta}(t)) - q_2(2(t))\|^2 dt.
\]

(5.2)

where \( \hat{\beta} \) is solved using an exhaustive grid search over \( \theta \). Once \( \hat{\beta} \) is obtained, one can build geodesic trajectories between \( \alpha_1 \) and \( \alpha_2 \) as follows:

Algorithm 6 Geodesic trajectories between \( \alpha_1 \) and \( \alpha_2 \).

1. Compute parallel transport of \( q_1 \) along \( \hat{\beta} \) to the tangent space \( T_{p_2(0)} S^2 \); call it \( \hat{q}_1 \).

2. In the tangent space \( T_{p_2(0)} S^2 \), build the geodesic between \( \hat{q}_1 \) and \( q_2 \), given by \( \eta(\tau) = (1 - \tau)\hat{q}_1 + \tau q_2 \), \( \tau \in [0, 1] \).

3. For each \( \tau \in [0, 1] \), compute backward parallel transport of \( \eta(\tau) \) along \( \hat{\beta} \) to the point \( \hat{\beta}(\tau) \), denoted by \( \eta^\dagger(\tau) \).

4. Compute covariant integral of \( \eta^\dagger(\tau) \) with starting point \( \hat{\beta}(\tau) \) using Algorithm 4; call as \( \alpha_{\hat{\beta}(\tau)} \).

The geodesic trajectories between \( \alpha_1 \) and \( \alpha_2 \) are \( \alpha_{\hat{\beta}(\tau)} \), \( \tau \in [0, 1] \). The geodesic distance between \( q_1 \) and \( q_2 \) on \( \mathbb{C} \) is given as:

\[
d(q_1, q_2) = \sqrt{t^2_{\beta} + \int_0^1 \|\hat{q}_1(t) - q_2(t)\|^2 dt}.
\]

(5.3)

The main importance of this setup – TSRVF representation and the metric \( d \) – comes from the following. If a trajectory \( \alpha \) is warped by \( \gamma \), to result in \( \alpha \circ \gamma \), what is the TSRVF of the time-warped trajectory? The new TSRVF is given by:

\[
q_{\alpha \circ \gamma}(t) = \frac{(\hat{\alpha}(\gamma(t))\hat{\gamma}(t))_{\alpha(\gamma(t)) \rightarrow p}}{\sqrt{|\hat{\alpha}(\gamma(t))\hat{\gamma}(t)|}} = \frac{(\hat{\alpha}(\gamma(t)))_{\alpha(\gamma(t)) \rightarrow p}\sqrt{\gamma(t)}}{\sqrt{|\hat{\alpha}(\gamma(t))|}} = q_\alpha(\gamma(t))\sqrt{\gamma(t)} \equiv (q_\alpha, \gamma)(t).
\]
The advantage of this framework is that the action of $\Gamma$ on $C$ under the metric $d$ in Eqn. 5.3 is by isometries.

**Theorem 3** For any $\alpha_1, \alpha_2 \in \mathcal{M}$ and $\gamma \in \Gamma$, the distance $d$ satisfies $d(q_{\alpha_1 \gamma}, q_{\alpha_2 \gamma}) = d(q_1, q_2)$.

The proof is given in A.2.

Our first goal is to preform registration of points along trajectories. The optimization problem is

$$\min_{\theta, \gamma} \left( l^2_{\bar{\beta}_{\theta}} + \int_0^1 \| \frac{q_1}{\| q_1 \|, \beta_{\theta}(t) - (q_2, \gamma(t)) \|^2 dt \right)$$

(5.4)

For a fixed $\theta$, one have the corresponding arch $\beta_{\theta}$ and the length is fixed. So the problem becomes equivalent to optimization on $\Gamma$, which can be solved using Dynamic Programming (DP). For $\theta$, the situation is different and we use an exhaustive grid search over $\theta$. This combined algorithm is summarized below:

**Algorithm 7 Minimization over $(\theta, \gamma)$**.

1. For each $\theta \in [0, 2\pi]$, solve $\hat{\gamma}$ by DP

   $$\hat{\gamma} = \arg\min_{\gamma \in \Gamma} \int_0^1 \| q_1 \|, \beta_{\theta}(t) - (q_2, \gamma(t)) \|^2 dt$$

2. Set the current values to be $(\theta, \hat{\gamma}_{\theta})$.

3. Define $(\hat{\theta}, \hat{\gamma}_{\hat{\theta}})$, where $\hat{\theta} = \arg\min_{\theta \in [0, 2\pi]} \left( l^2_{\bar{\beta}_{\theta}} + \int_0^1 \| \frac{q_1}{\| q_1 \|, \beta_{\theta}(t) - (q_2, \hat{\gamma}_{\theta}(t)) \|^2 dt \right)$

### 5.2 Experimental Results

In this section, we will show some preliminary results of geodesics between trajectories without and with registration. First, we show geodesics without registration in three cases.

#### 5.2.1 Geodesic between trajectories before registration

- **Case 1: Geodesics with unit length**

  The vector bundle described earlier is the vector bundle of all trajectories. In this case, we want to restrict to unit length geodesics. The corresponding subspace of this vector bundle is called a fiber bundle in which one replaces each Hilbert space $C_p = L^2([0, 1], T_p(S^2))$ by the unit sphere in this Hilbert space. The constructed geodesics in this case are actually geodesics in the total space of the corresponding sphere bundle.

  The optimal arch $\hat{\beta}$ is one having the same angle with $\alpha_1$ and $\alpha_2$. Specifically, a geodesic on $S^2$ can be viewed as a frame $f = (p, v, w)$, where $p$ is the unit vector of the starting point, $v$
is the unit shooting vector at \( p \) and \( w \) is the normal vector, which is the cross product of \( p \) and \( v \). Now the two geodesics can be uniquely expressed as two frames: \( f_1 = (p_1, v_1, w_1) \) and \( f_2 = (p_2, v_2, w_2) \). The optimal arch is given by:

\[
\hat{\beta}(\tau) = e^{\tau A} p_1, \quad \tau \in [0, 1], \text{ where } A = \logm (f_2 f_1^T).
\]

And the geodesic trajectories between \( \alpha_1 \) and \( \alpha_2 \) are also unit length geodesics. At each \( \tau \in [0, 1] \), the geodesic is uniquely determined by the starting point \( \hat{\beta}(\tau) \) and the shooting vector is \( e^{\tau A} v_1 \). Fig. 5.1 shows three examples of geodesic trajectories between unit length geodesics on \( \mathbb{S}^2 \). In each case, both trajectories are geodesics on \( \mathbb{S}^2 \). The optimal arch is denoted by the solid yellow line. For comparison, the dashed yellow line denotes the geodesic between the starting points of trajectories. The intermediate lines show the geodesic trajectories starting from points along the arch.

- **Case 2: Geodesics with arbitrary length** Now, assume \( \alpha_1 \) and \( \alpha_2 \) are geodesics with arbitrary length. The optimal arch is obtained using Eqn. 5.2 and geodesic trajectories are solved by Algorithm 6. Some results are shown in Fig. 5.2.
Case 3: General trajectories In this case, $\alpha_1$ and $\alpha_2$ are arbitrary trajectories. Fig. 5.3 shows three examples of geodesic trajectories obtained by Algorithm 6.

5.2.2 Geodesic between trajectories after registration

Now, for two arbitrary trajectories, we compare the results between without registration and with registration. Both examples in Fig. 5.4 show that the geodesic trajectories after registration are more...
reasonable in preserving the structures and the resulting distances are much smaller. The distance reduces from 9.56 to 6.78 in Example 1, and from 4.41 to 3.05 in Example 2. In particular, the geodesic after registration in Example 1 preserves the "bump" pattern in the data. The fact that the optimal circular arch is very different after registration makes the geodesic much more reasonable.
CHAPTER 6

DETECTION, CLASSIFICATION AND ESTIMATION OF INDIVIDUAL SHAPES IN 2D AND 3D POINT CLOUDS

The problems of detecting, classifying and estimating shapes in point cloud data are important due to their general applicability in image analysis, computer vision, and graphics. They are challenging because the data is typically noisy, cluttered, and unordered. We study these problems using a fully statistical model where the data is modeled using a Poisson process on the object’s boundary (curves or surfaces), corrupted by additive noise and a clutter process. Using likelihood functions dictated by the model, we develop a generalized likelihood ratio test for detecting a shape in a point cloud. This ratio test is based on optimizing over some unknown parameters, including the pose and scale associated with hypothesized objects, and an empirical evaluation of the log-likelihood ratio distribution. Additionally, we develop a procedure for estimating most likely shapes in observed point clouds under given shape hypotheses. We demonstrate this framework using examples of 2D and 3D shape detection and estimation in both real and simulated data, and a usage of this framework in shape retrieval from a 3D shape database.

6.1 Introduction

An important feature for characterizing objects in images is their shapes and, as a consequence, shape analysis has become an integral part of object classification. One way to use shape analysis is to estimate the boundaries of the objects (in images) and to analyze the shapes of those boundaries. While analyses of curves and surfaces are important parts of a shape theory, the practical situations mostly involve heavily under-sampled, noisy, and cluttered discrete data, often because the process of estimating boundaries uses low-level techniques that extract a set of primitives (points, edges, arcs, etc.) in the image domain. Therefore, the problem of detecting and estimating shapes in point clouds is an important problem. In some cases one can obtain labeled points where each
point carries a binary label assigning it to either the object or the background. Sometimes these labels can also be associated with specific parts of objects. However, in general, the points are unlabeled and do not carry any information about which part of the scene they belong to. Since point clouds are so general, as they contain no interpretation of the data, they are broadly applicable in different scientific domains. Furthermore, the acquisition and processing of digital 3D point clouds has received increasing attention over the last few years. While visualization of very detailed and complex point clouds has become possible, our ability to draw inferences at a semantic level are still very limited. Even tasks as basic as selecting all windows in a scan of a house currently require a disproportional amount of user interaction. This is due to the fact that the acquired raw data does not provide any structure let alone semantic information. Therefore, the extraction of structured shapes from 2D and 3D point clouds is an important topic for a wide field of applications.

What makes the problem of shape detection in point clouds difficult? Here are some of the issues: (1) **Unknown Pose and Scale**: A shape can be present in the data at an arbitrary pose and scale, as shown in Fig. 6.1, and one does not know these variables a-priori. The key is to be able to search over the corresponding parameter spaces in an exhaustive fashion and reach global solutions. (2) **Noise and Clutter**: There is invariably some observation noise associated with shape measurements. Also, the data is commonly corrupted by the presence of points that belong to either the background or other objects, termed as clutter in this chapter, as shown in Fig. 6.1. In this setting it is natural to develop statistical models, and seek efficient global solutions for estimating unknown variables. Note that this problem is different from the problem of comparing unlabeled point patterns which has been addressed by a number of landmark papers, including [93, 94, 95, 96, 97, 98]. In our problem context, one of the two objects being compared is a well-defined shape while the other is a point cloud. In contrast, in the earlier papers both the objects can be point clouds. Also, while most previous works focus on registration or comparison of two point clouds, our problem is actually to detect and estimate a shape.

### 6.1.1 Past Work

There has been a large body of work detecting objects in 2D and 3D point clouds. The general theory of statistical analysis of spatial data, especially that involving point patterns, can be found in [99, 100]. The work on object detection in point clouds can be broadly divided into the following categories.

The first category of papers detects shapes from 2D point clouds. [101] presents a statistical approach for identification of objects in digital images where a point distribution model is fitted using Procrustes analysis to a set of training images and used as a prior distribution for the shape of a deformable template. A recent paper by [102] develops a Bayesian approach for shape classification in 2D point clouds. The authors estimate the posterior probability of a given shape by integrating over unknown variables such as pose, scale, and point labels using a Monte Carlo method. The
second category of papers works on surface reconstruction from 3D point clouds including [103, 104, 105, 106, 107]. Most methods call for some kind of connectivity information and are not well equipped to deal with a large amount of outliers. A region growing approach has also been used to detect planes in 3D point clouds by [108]. This approach often delivers a superior segmentation but still suffers from the problems of noisy data. The point clouds have also received attention in the computer graphics community. For example, [109] have proposed a general variational framework for approximation of surfaces by planes, which is extended to more elaborate approximations by [110].

The iterative closest point (ICP) algorithm by [111] is a method that uses the nearest-neighbor relationship to assign a binary correspondence at each step. This estimate of the correspondence is then used to refine the transformation, and vice versa. [112] finds the locations of target objects using single spin image matching and then retrieves the orientation and quality of the match using the ICP algorithm. [113] proposes a point matching algorithm for non-rigid registration. They develop an algorithm with the thin-plate splines as the parameterization of the non-rigid spatial mapping and estimate correspondence between points. [114] presents a technique that uses a vector space representation of shape (3D Morphable Model) to infer missing vertex coordinates. Another approach that models 3D objects using mixtures of point patterns and draws inferences using MCMC algorithm in a Bayesian setup is described by [115].

### 6.1.2 Our Approach

We present a fully statistical framework for detecting pre-determined shapes in point clouds. An important goal is to provide a likelihood, and thus a confidence, of finding a shape in a given data. We develop a model-based approach where the data is modeled using a Poisson process on the object’s boundary, corrupted by an additive noise and a clutter process. The clutter process itself is modeled using an independent Poisson process on the image domain. Using analytical likelihood functions dictated by the model, we develop a generalized likelihood ratio test for detecting a shape, as described in Section 6.2.1. The classification threshold is based on an empirical distribution of the log-likelihood ratio estimated using Monte Carlo method under the null distribution; this is presented in Section 6.2.2. The ratio test requires optimizing the pose and scale associated with hypothesized shapes which, in turn, is performed using a combination of grid search and gradient descent. We use analytical expressions for the gradients of the log-likelihood function. This is illustrated for the 2D and 3D problems in Section 6.3.1. The classification of shapes in 2D and 3D point clouds, based on comparing their log-likelihood ratios, is described in Section 6.3.2, while the problem of estimating shapes in the given clouds using selected points is studied in Section 6.3.3.
6.2 Problem Formulation

We consider the following problem: We are given a point cloud $Y = \{y_i \in \mathbb{R}^n, i = 1, 2, \ldots, m\}$ in a domain $U \subset \mathbb{R}^n$ and we want to develop a statistical framework for deciding if there is a pre-determined shape contained in this set. Only the shape is known but its location, orientation, and scale in the scene is unknown. One can extend this idea to detection of full shape classes, i.e., a set of shapes belonging to the same population, using statistical shape models ([116]) but that idea is not explored in this section.

To illustrate the problem, some examples of 2D and 3D point clouds are shown in Fig. 6.1. In the top row we are interested in finding the likelihood that the shape of a “runner” is present in these clouds. A quick inspection ascertains the presence of a runner in the first two panels, even though the second one is more cluttered than the first one, but the situation is not so clear for the last case. The bottom row shows a similar problem for 3D point clouds. We would like to develop a framework to perform this detection automatically.

We will treat this as a problem of binary hypothesis testing – the null hypothesis is that $Y$ is simply clutter, i.e., the shape of interest is NOT present in $Y$, and the alternate hypothesis is that $Y$ is generated from that shape, i.e., a shape is present in $Y$.

\[ H_0 : \text{Shape is absent, Likelihood } P(Y|C) \]
\[ H_1 : \text{Shape is present, Likelihood } P(Y|S) \]

Here, $C$ denotes the clutter and $S$ denotes the shape of interest. The challenge, of course, is to develop appropriate probability models that will enable us to evaluate the two likelihoods.
6.2.1 Model-Based Shape Detection

We take a model-based approach where we evaluate the likelihood of a point cloud containing a shape, and compare that with the likelihood of the cloud being pure clutter. The points present in a given point cloud can be one of two types: (i) points belonging to a shape and (ii) points associated with the background clutter. We will propose an observation model for each of them separately.

Shape is a characteristic that is invariant to similarity transformations, but when a shape occurs in a scene, it has a specific scale, position, and orientation. From the perspective of shape detection, these variables are considered nuisance variables that have to be either estimated or integrated out.

In order to better explain the model description and a detection solution, we will start with a simpler problem where we seek a specific object, i.e. known shape, position, orientation, and scale. Let $\beta : D \rightarrow \mathbb{R}^n$ be a parameterized object, where $D$ is a domain for the parameterization. For finding shapes of curves in 2D images we will have $D = [0, 1]$ and $n = 2$, while for finding surfaces in 3D images we will have $D = [0, 1]^2$, $n = 3$. We will assume that the curves are parameterized by a constant speed parameter in the 2D case, i.e. $\beta(s) \in \mathbb{R}^2$ such that $|\dot{\beta}(s)| = \text{constant}$. Since no such canonical parameterization exists for surfaces, we will work with arbitrarily parameterized surfaces, i.e. $\beta(s) \in \mathbb{R}^3$ where $s \in [0, 1]^2$ parameterizes the surface $\beta$. In both the cases, we are going to restrict to those $\beta$’s that are absolutely continuous on $D$.

To develop the data model, we make the following assumptions:

1. **Points belonging to $\beta$:** We assume that these points are realizations of a Poisson process on the parameterized object $\beta$. Let $\gamma : D \rightarrow \mathbb{R}_{\geq 0}$ be the intensity function of the Poisson process along $\beta$. The number of points generated from any part of the object is a Poisson random variable with mean being the integral of $\gamma$ on that part. In particular, $k$, the total number of points belonging to the object, is a Poisson random variable with mean $\Gamma = \int_D \gamma(s)ds \in \mathbb{R}_{\geq 0}$. Let the points sampled from $\beta$ be denoted by $X = [x_1, x_2, \ldots, x_k], x_j \in \mathbb{R}^n$. The actual observations $y_j$ are assumed to be noisy versions of $x_j$. For given $x_1, x_2, \ldots, x_k$, the $y_j$’s are assumed to be independent of each other with the identical density $f(y_j|x_j)$. Under this model, the two hypotheses can be re-written as:

$$
\begin{align*}
H_0 & : \gamma = 0, \text{ Likelihood } P(Y|C) \\
H_1 & : \gamma > 0, \text{ Likelihood } P(Y|S)
\end{align*}
$$

2. **Points associated with clutter:** This subset of observations, independent of the first subset, comes from the clutter and we model them as realizations of a Poisson process with the intensity $\lambda : U(\subset \mathbb{R}^n) \rightarrow \mathbb{R}_{\geq 0}$, where $U$ is the region containing observed points, e.g. $U = [a, b]^2$ for 2D and $U = [a, b]^3$ for 3D point clouds. Let $\Lambda = \int_U \lambda(y)dy \in \mathbb{R}_{\geq 0}$.

The full observation $Y$ can now be modeled as a Poisson process with the intensity function:
\( \xi(y) = \lambda(y) + \int_D f(y|\beta(s))\gamma(s) \, ds \). The probability density function of \( Y \), given \( \beta, \gamma, \lambda \), and for a fixed \( m \), is given by: \( P_m(Y|\beta, \gamma, \lambda) = (\prod_{i=1}^m \xi(y_i))e^{-\Lambda - \Gamma} \), where \( m \) is the total number of points in the data. The null hypothesis is that all the points belong to the Poisson clutter. In that case, the likelihood function is given by: \( Q_m(Y|\lambda) = e^{-\Lambda} \prod_{i=1}^m \lambda(y_i) \). The likelihoods for both the cases, \( H_0 \) and \( H_1 \), involve certain parameters that are generally not known beforehand. Thus, taking a simple likelihood ratio is not possible and we resort to the generalized likelihood ratio test (GLRT). This is based on maximum likelihood estimates (MLEs) of parameters, under the respective hypotheses, and uses the MLEs for evaluating the likelihood ratio. The generalized likelihood ratio is given by:

\[
\frac{Q_m(Y|C)}{P_m(Y|S)} = \frac{\max_{\lambda, \gamma} Q_m(Y|\lambda)}{\max_{\lambda, \gamma} P_m(Y|\beta, \gamma, \lambda)} = \frac{\max_{\lambda}(e^{-\Lambda} \prod_{i=1}^m \lambda(y_i))}{\max_{\lambda, \gamma}(e^{-\Gamma - \Lambda}(\prod_{i=1}^m \xi(y_i)))}.
\]

(6.1)

So far the unknown parameters are full functions and that involves tremendous computational complexity. We will simplify the evaluation of GLR in Eqn. 6.1 by making the following additional assumptions:

1. The noise added to the points sampled from \( \beta \) is i.i.d. Gaussian with mean zero and variance \( \sigma^2 I_{n \times n} \). Therefore, \( f(y|x) \) takes the form \( \frac{1}{(2\pi)^{n/2} \sigma^n} e^{-\frac{1}{2\sigma^2}||y-x||^2} \) for \( y, x \in \mathbb{R}^n \).

2. Both the Poisson intensities are constant, i.e., \( \lambda(y) = \lambda \) and \( \gamma(s) = \gamma \) and we get \( \Lambda = \lambda \int_U dy \) and \( \Gamma = \gamma \int_D ds \). To simplify the discussion, we scale both the integrals to be one such that \( \Lambda = \lambda \) and \( \Gamma = \gamma \).

With these assumptions, the likelihood ratio simplifies to:

\[
\frac{Q_m(Y|C)}{P_m(Y|S)} = \frac{\max_{\lambda}(e^{-\lambda} \prod_{i=1}^m \lambda)}{\max_{\lambda, \gamma, \sigma}(e^{-\gamma - \lambda}(\prod_{i=1}^m (\lambda + \gamma \alpha_\sigma(y_i))))}.
\]

The numerator on maximization becomes \( e^{-\lambda m m^m} \). The quantity \( \alpha_\sigma : \mathbb{R}^n \to \mathbb{R}_+ \) in the denominator is a scalar map given by \( \alpha_\sigma(y_i) = \frac{1}{(2\pi)^{n/2} \sigma^n} \int_D e^{-\frac{1}{2\sigma^2}||y_i - \beta(s)||^2} ds \). Notice that \( \alpha_\sigma(y_i) \) is high if a point \( y_i \) is close to the object \( \beta \), with the closeness being measured relative to the scale \( \sigma \). Some illustrations of \( \alpha_\sigma \) in \( \mathbb{R}^2 \) are shown in Fig. 6.2. The top row shows the case for different \( \sigma \)'s (from left to right: \( \sigma = 0.01, 0.02, 0.03 \)) but a fixed curve. As \( \sigma \) increases, the region of high likelihood spreads further away from the curve. The bottom row shows \( \alpha_\sigma \) maps for different curves but a fixed \( \sigma = 0.02 \).

Define a function \( H \) to be the logarithm of \( P_m(Y|\beta, \gamma, \lambda) \). Let \( \theta = [\gamma, \lambda, \sigma] \in \mathbb{R}^3 \) denote three unknown parameters associated with the shape. Then, the function \( H : \mathbb{R}^3 \to \mathbb{R}_+ \) is given by \( H(\theta) = -\gamma - \lambda + \sum_{i=1}^m \log(\lambda + \gamma \alpha_\sigma(y_i)) \), and let \( \hat{\theta} = \arg\max_{\theta} H(\theta) \) be maximizer (actually, \( \hat{\theta} \) is the MLE of \( \theta \) under the hypothesis \( H_1 \)). We solve for the MLE of \( \theta \) using a gradient approach. Of the three components of \( \theta \), we search exhaustively for the parameter \( \sigma \) and use a gradient-based
approach to search over the remaining two $\gamma$ and $\lambda$. For each value of $\sigma$ in a certain range, say $[\sigma_l, \sigma_u]$, we maximize $H$ over the pair $(\gamma, \lambda)$.

For a fixed $\sigma$, the function $H: \mathbb{R}^2 \rightarrow \mathbb{R}$, given by $H_\sigma(\lambda, \gamma) = -\gamma - \lambda + \sum_{j=1}^{m} \log(\lambda + \gamma \alpha_\sigma(y_j))$ has the following properties:

1. Its derivatives with respect to $\lambda$ and $\gamma$ are given by:
   \[
   \frac{\partial H_\sigma}{\partial \lambda} = -1 + \sum_{i=1}^{m} \frac{1}{\lambda + \gamma \alpha_\sigma(y_i)}, \quad \frac{\partial H_\sigma}{\partial \gamma} = -1 + \sum_{i=1}^{m} \frac{\alpha_\sigma(y_i)}{\lambda + \gamma \alpha_\sigma(y_i)}. \tag{6.2}
   \]

2. Its Hessian matrix is given by:
   \[
   \begin{pmatrix}
   \sum_{i=1}^{m} \frac{-\alpha_\sigma(y_i)^2}{(\lambda + \gamma \alpha_\sigma(y_i))^2} & \sum_{i=1}^{m} \frac{-\alpha_\sigma(y_i)}{(\lambda + \gamma \alpha_\sigma(y_i))^2} \\
   \sum_{i=1}^{m} \frac{-\alpha_\sigma(y_i)}{(\lambda + \gamma \alpha_\sigma(y_i))^2} & \sum_{i=1}^{m} \frac{-1}{(\lambda + \gamma \alpha_\sigma(y_i))^2}
   \end{pmatrix}.
   \]

It is easy to show that the two eigenvalues of the Hessian matrix are non-positive so that $H_\sigma$ is a concave function in $\lambda$ and $\gamma$.

Therefore, one can use the gradient search over $\lambda$ and $\gamma$, and reach a global optimizer. For $\sigma$ the situation is different and we use an exhaustive grid search over allowable values of $\sigma$ to reach a
global maximizer. This combined gradient and grid search algorithm is summarized below:

Algorithm 8 MLE of $\theta$

- For each $\sigma \in [\sigma_l, \sigma_u]$ perform the following:

1. Set $t = 0$ and initialize the pair $[\gamma_t, \lambda_t]$ with random values in the range $[0, m]$.
2. Update the estimates using:
   
   $$
   \begin{bmatrix}
   \gamma_{t+1} \\
   \lambda_{t+1}
   \end{bmatrix} =
   \begin{bmatrix}
   \gamma_t \\
   \lambda_t
   \end{bmatrix} + \delta
   \begin{bmatrix}
   \frac{\partial H}{\partial \gamma}(\gamma_t, \lambda_t) \\
   \frac{\partial H}{\partial \lambda}(\gamma_t, \lambda_t)
   \end{bmatrix}, \text{ for a small } \delta > 0.
   $$
3. If the norm of the gradient vector is small, then stop the loop. Else, set $t = t + 1$ and return to step 2.

- Set the current values to be $([\hat{\gamma}(\sigma), \hat{\lambda}(\sigma)]$).

- Define the MLE $\hat{\theta}$ to be $([\hat{\gamma}(\hat{\sigma}), \hat{\lambda}(\hat{\sigma}), \hat{\sigma}]$ where $\hat{\sigma} = \operatorname{argmax}_{\sigma \in [\sigma_l, \sigma_u]} H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma)$.

With the estimated parameters, the log-likelihood ratio (LLR) becomes

$$
R(Y) = \log \frac{Q_m(Y|C)}{P_m(Y|S)} = -m + m \log(m) - H(\hat{\theta}). \quad (6.3)
$$

The generalized likelihood ratio test is given by: $R(Y) \geq \mu$. Next, we will focus on how to choose the threshold $\mu$.

6.2.2 Empirical Determination of Threshold

In the binary test, the LLR $R(Y)$ is to be compared with a threshold $\mu$ to decide if a shape is present in the data or not. Ideally, this threshold is dictated by the probability distributions of $R(Y)$ under the null hypothesis. In practical situations, where it is difficult to ascertain these distributions, one uses either the asymptotic theory or an empirical approach to reach an optimal value of $\mu$.

Taking an empirical approach, we estimate the probability density function of $R(Y)$ under the null hypothesis. We generate 1000 realizations of $Y$, each using $\gamma = 0$ (null hypothesis = clutter) and a fixed $m$ equal to the observed number of points in our data, and compute a histogram of $R(Y)$ values. Using this estimated density function, we can decide the threshold $\mu$ for a specific Type I error rate, denoted by $\alpha$. One can repeat this for different values of $m$ to catalog distributions of $R(Y)$ for different $m$'s.

The main advantage of a numerical evaluation of the threshold is that we need not assume any specific form for the underlying density, nor do we need to invoke any asymptotics. The disadvantage, however, is that we need to do this for every shape we are interested in, since we do not have an
analytical expression. We point out that this computation is offline and can be performed for each of the shapes beforehand. It takes approximately 16 minutes to generate an empirical distribution of $R(Y)$, for a fixed shape, in the 2D case (with $m = 100$ points) and two hours in the 3D case (with $m = 500$ points).

### 6.3 Detection, Classification and Estimation of Shapes in Point Clouds

#### 6.3.1 Shape Detection

We will describe the problem of shape detection in 2D and 3D point clouds separately in the next two sections.

**Shape Detection in 2D Point Clouds.** In this case the domain of parameterization is $D = [0, 1]$ and the observation space is $\mathbb{R}^2$, so that $\beta : [0, 1] \to \mathbb{R}^2$ is a closed, continuous contour parameterized at a constant speed. For our experiments, we have taken closed curves from the MPEG4 shape database. The points belonging to the curve are assumed to be realizations of a 1D Poisson process on the parameterized curve $\beta$. The intensity function of the Poisson process is $\gamma \in \mathbb{R}_{\geq 0}$. And points associated with clutter are modeled as realizations of a 2D Poisson process with the intensity $\lambda \in \mathbb{R}_{\geq 0}$. At first we will assume that the position, orientation, and scale of the curve $\beta$ are fully known but later we will generalize to curves with unknown transformations.

- **Detection of a Known Curve**: In this section we focus on detection of a shape formed by a curve $\beta$ in a square domain $U$. Now the scalar map $\alpha_\sigma : \mathbb{R}^2 \to \mathbb{R}_+$ is given by
  \[
  \alpha_\sigma(y) = \frac{1}{2\pi\sigma^2} \int_0^1 e^{-\frac{1}{2\sigma^2} \|y - \beta(s)\|^2} \, ds.
  \]
  With this setup, one can use Algorithm 8 to find MLE of $\theta$ and use the resulting LLR $R(Y)$ to detect the presence of $\beta$ in $Y$.

We demonstrate MLE of $\theta$ using Algorithm 8 on some simulated datasets where we simulate the point cloud data $Y$ as follows. First, we sample the curve $\beta$ randomly (uniformly) with $k$ points, with $k$ being a Poisson random variable with a certain mean. Then, we add an independent Gaussian perturbation (mean zero, fixed variance) to their positions. Additionally, we generate a clutter sample from a two-dimensional homogeneous Poisson process with certain mean intensity in the region $U$. Fig. 6.3 presents an example of maximizing the function $H$ using the gradient method. The first two panels show the curve and the simulated point cloud using the parameters $\gamma = 30, \lambda = 25, \sigma = 0.030$. The next panel shows the evolution of $H_\sigma$ over $(\gamma, \lambda)$ versus iteration index for a fixed $\sigma = 0.060$, while the rightmost panel shows the plot of $H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma)$ versus $\sigma$. The MLE of $\theta$ is $\hat{\gamma} = 32.45, \hat{\lambda} = 25.74, \hat{\sigma} = 0.032$. The maximum value of $H$ lies close to the true $\sigma = 0.03$.

In addition, for each point $y_i$ of an observation, we calculate the corresponding value $\alpha_\sigma(y_i)$. 

76
Figure 6.3: (a): A curve $\beta$, (b): simulated point cloud $Y$, (c): the evolution of $H_\sigma$ over $(\gamma, \lambda)$ versus iteration index for a fixed $\sigma$, and (d) $H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma)$ versus $\sigma$.

Figure 6.4: A point cloud, two hypothesized curves, $\alpha_{\hat{\sigma}}$ profiles, and $\alpha_{\hat{\sigma}}$ at $y_i$’s for the two curves.

For the same point cloud, shown in the left column of Fig. 6.4, we apply Algorithm 8 for two curves – a runner and a wineglass. and display the results in the two rows respectively. The third panel of each row shows the estimated $\alpha_{\hat{\sigma}}$ map for that curve and the rightmost panel shows the value of $\alpha_{\hat{\sigma}}(y_i)$ for each of the data points using its thickness. Recall that $\alpha_{\hat{\sigma}}(y_i)$ is large if the point $y_i$ is close to the curve $\beta$. Since the observation $Y$ here was generated from the runner curve, $\alpha_{\hat{\sigma}}(y_i)$’s have larger values for that shape and lower values for the wineglass shape.

• **Detection of a Curve with Unknown Pose and Scale:** So far we have assumed a fixed curve $\beta$ but a contour can be present in an image at an arbitrary position, orientation, and scale. Therefore, $\beta$ can have variable shape, position, rotation, and scale. Let $O \in SO(2)$ denote the orientation, $T \in \mathbb{R}^2$ denote its translation, and $\rho \in \mathbb{R}_+$ denote its scale. First let $\beta_0$ be a standardized curve, i.e., it has a fixed shape, its centroid is at the origin, its length is one,
and its major axes are aligned with the canonical axes. Then, define \( \beta(t) = \rho O \beta_0(t) + T \) to be a transformed version of that curve. Here, \( O = \begin{bmatrix} \cos(\tau) & -\sin(\tau) \\ \sin(\tau) & \cos(\tau) \end{bmatrix} \), \( \tau \in [0, 2\pi] \) and 
\( T = [T_1, T_2] \in \mathbb{R}^2 \). Even though we allow unknown transformations to the curve, denoted by \( \omega = (\rho, O, T) \), the function \( H \) keeps the same form, except the scalar map \( \alpha_\sigma \) now also depends on these other variables, i.e.,
\[
\alpha_\sigma(y|\omega) = \frac{1}{2\pi\sigma^2} \int_0^1 e^{-\frac{1}{2\sigma^2}\|y_i-T-\rho O \beta_0(s)\|} ds.
\]
These additional variables are also estimated, along with \( \theta \), during the maximum-likelihood estimation procedure. The gradients of \( H \) with respect to \( \lambda \) and \( \gamma \) remain same as in Eqn. 6.2.

The gradient for the three elements of \( \omega \) takes a general form:
\[
\frac{1}{2\pi\sigma^4} \sum_{i=1}^{m} \frac{\gamma}{\lambda + \gamma \alpha_\sigma(y_i|\omega)} \int_0^1 e^{-\frac{1}{2\sigma^2}\|y_i-\beta(s)\|^2} \langle y_i - \beta(s), g(s) \rangle ds,
\]
where (i) \( g(s) = O \beta_0(s) \) for \( \frac{\partial H}{\partial \rho} \), (ii) \( g(s) = \rho \dot{O}(\tau) \beta_0(s) \) for \( \frac{\partial H}{\partial \tau} \), and (iii) \( g(s) = e_j \) for \( \frac{\partial H}{\partial \omega} \). Here \( \dot{O}(\tau) = \begin{bmatrix} -\sin(\tau) & -\cos(\tau) \\ \cos(\tau) & -\sin(\tau) \end{bmatrix} \), \( e_1 = (1, 0)' \), and \( e_2 = (0, 1)' \). We will use \( \frac{\partial H}{\partial \omega} \) to denote the gradient with respect to the position, rotation, and scale variables: \( \left[ \frac{\partial H}{\partial \rho}, \frac{\partial H}{\partial \tau}, \frac{\partial H}{\partial \lambda}, \frac{\partial H}{\partial \omega} \right] \). The algorithm for jointly estimating \( \theta \) and \( \omega \) is summarized below:

**Algorithm 9 Joint MLE of \( \theta \) and \( \omega \) for 2D:** To start, we first translate the center of \( Y \) to the origin and align the major axes with canonical axes using Procrustes rotation.

1. For each \( \sigma \in [\sigma_l, \sigma_u] \) perform the following:
   1. Set \( t = 0 \) and initialize the pair \([\gamma_t, \lambda_t]\) with random values in the range \([0, m]\) and initialize \( \omega_t = [1, I_{2\times2}, [0, 0]' \).
   2. Update the estimates using:
      \[
      \begin{bmatrix} \gamma_{t+1} \\ \lambda_{t+1} \\ \omega_{t+1} \end{bmatrix} = \begin{bmatrix} \gamma_t \\ \lambda_t \\ \omega_t \end{bmatrix} + \delta \begin{bmatrix} \frac{\partial H}{\partial \gamma} \gamma_t \lambda_t \omega_t \\ \frac{\partial H}{\partial \lambda} \gamma_t \lambda_t \omega_t \\ \frac{\partial H}{\partial \omega} \gamma_t \lambda_t \omega_t \end{bmatrix},
      \]
      where \( \delta \) is a positive, diagonal matrix. Each diagonal component of \( \delta \) corresponds to a small step size for each variable in \( \theta \) and \( \omega \).
   3. If the norm of the gradient vector is small, then stop. Else, set \( t = t + 1 \) and return to step 2.
   - Set the current values to be \((\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \hat{\omega}(\sigma))\).
   - Define the MLE \((\hat{\theta}, \hat{\omega})\) to be \((\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \hat{\omega}(\sigma), \hat{\sigma})\), where
      \[
      \hat{\sigma} = \arg \max_{\sigma \in [\sigma_l, \sigma_u]} H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \hat{\omega}(\sigma), \sigma).
      \]

Fig. 6.5 is an illustration of this gradient process. For the same point cloud, we estimate the
Figure 6.5: Maximum likelihood estimation of shape transformations from given point clouds (black points) and starting curves (red). Left: rotation, middle: translation, right: scale. The top row shows the evolution of shapes while the bottom row plots the evolution of $H$.

rotation $O$ (left), translation $T$ (middle), and scale $\rho$ (right) in the top panel with the evolution of $H$ versus iteration in the bottom panel. Each plot in the bottom displays a maximization over just one of them with the other two held fixed.

Next we provide a set of detection results involving simulated point clouds. For a fixed $m$, say $m = 50$, we find $k$ (number of points from the curve) and $n$ (number of clutter points) such that $m = n + k$. Then, we transform a curve with a random rotation, translation and scale, and select $k$ random points on it uniformly, and finally add Gaussian noise to the selected points. Next we generate $n$ clutter points uniformly on the domain $U$ and mix the two sets to obtain $Y$. For the resulting $Y$, we compute LLR under $S$ and perform the binary detection by comparing it with $\mu$. Fig. 6.6 shows two examples of this setup. In both cases, the data is simulated from the runner shape, and the resulting large negative values of $R(Y)$, $R(Y) = -90.4$ (left) and $-96.0$ (right), support rejecting $H_0$ where $\mu = -3.8$ for $\alpha = 0.05$. We also plot the original curve $\beta_0$ under estimated position, orientation, and scale on top of the cloud data.

Beyond the experiments on individual point clouds, we are interested in evaluating the average performance of our method on a large dataset. To study the detection performance systematically, we will study the variability in the probability of Type II error by changing the model parameters. The probability of Type I error is kept fixed at $\alpha = 0.05$ in these experi-
Figure 6.6: Point clouds (red) and estimated curves (blue), $R(Y) = -90.4$ (first) and $-96.0$ (second) strongly support $H_1$ ($\alpha = 0.05$ and $\mu = -3.8$).

Figure 6.7: Type II error probability versus $\sigma$ and $r$.

The probability of Type II error is estimated using the equation: $P(\text{Type II Error}) \approx \frac{1}{N} \left( \text{No. of times} \ (R(Y) > \mu) \right)$, where $N = 1000$. As the number of sampled points on the curve decreases, or as the noise or clutter increases, the detection performance suffers. We have evaluated the shape detection performance as a function of $r = \frac{n}{k}$ (# of points on the curve)/# of clutter points), with $m = n + k$, and noise std dev. $\sigma$, and have plotted the estimated Type II error probability versus these variables for the runner shape in Fig. 6.7. The left of those results shows the probability of Type II error versus the noise level $\sigma$, for different values of the ratio $r$. In the case where $r = 0$, there is obviously no dependence on $\sigma$ and the probability of Type II error is simply $1 - \alpha = 0.95$ as expected. In other cases, where $r$ is strictly positive, there is a steady increase in this error probability as the noise level increases. The right plot shows the estimated error probability versus the ratio $r$ while $\sigma$ is kept fixed at 0.1. For a fixed noise level, the error probability decreases with an increase in $r$; this is intuitively clear as more points on the curve allow for a better detection performance.
Shape Detection in 3D Point Clouds. In this section, we focus on detection of a shape formed by a parameterized surface $\beta$ in a cubical domain $U \subset \mathbb{R}^3$. We set $U$ to be a cube that contains all of $Y$. Now, the given point cloud is $Y = \{ y_i \in \mathbb{R}^3, i = 1, 2, \ldots, m \}$. Let $\beta : [0, 1]^2 \rightarrow \mathbb{R}^3$ be a parameterized surface. Using the same Poisson models, points belonging to the surface are assumed to be realizations of a 2D Poisson process on the parameterized surface $\beta$ with a fixed intensity $\gamma \in \mathbb{R}_{\geq 0}$. Points associated with clutter are modeled as realizations of a 3D Poisson process with a fixed intensity $\lambda \in \mathbb{R}_{\geq 0}$. Similar to the 2D shape detection, we will introduce the detection of a fixed surface in 3D point clouds first and then extend it to detection of surface under unknown transformations.

- **Detection of a Known Surface**: Starting with the problem of detecting the shape formed by a fixed surface $\beta$, we notice that the expressions for gradients of $H$ with respect to the parameters $\lambda$ and $\gamma$ are the same as in Eqn. 6.2 except $\alpha_\sigma$ becomes $\alpha_\sigma : \mathbb{R}^3 \rightarrow \mathbb{R}_+$ given by $\alpha_\sigma(y_i) = \frac{1}{\sqrt{2\pi\sigma}} \int_{[0,1]^2} e^{-\frac{1}{2\sigma^2}\|y_i - \beta(s)\|^2} ds$. Thus Algorithm 8 directly applies and we can estimate the unknown parameters and consequently evaluate $R(Y)$ using that algorithm. We demonstrate this gradient method on a simple example. For a facial surface shown in the left panel of Fig. 6.8, we simulate a point cloud using the parameters $\gamma = 250, \lambda = 150, \sigma = 0.01$, shown in the second panel of Fig. 6.8. The face surface $\beta$ and the point cloud $Y$ are plotted together in the third panel where the red points are from the surface and the blue ones are from the clutter. The right panel shows the evolution of $H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma)$ versus $\sigma$ under Algorithm 8. As shown there, the function $H$ achieves the maximum at $\hat{\sigma} = 0.01$. The MLE of $\theta$ is found to be $\hat{\gamma} = 262.62, \hat{\lambda} = 152.53, \hat{\sigma} = 0.01$ and the LLR is $R(Y) = -716.66$. 

We also compare the error rates of detection in two situations: curve with known rotation and curve with unknown rotation. As shown in the plot, the error increases for the latter case, quantifying the penalty paid for in estimating this additional unknown.
Detection of a Surface with Unknown Pose and Scale: So far we have assumed a fixed surface $\beta$, but a surface can be present in $Y$ at an arbitrary position, orientation, and scale. Therefore, $\beta$ can have variable position, rotation, and scale. Let $O \in SO(3)$ denote the orientation, $T \in \mathbb{R}^3$ denote its translation, and $\rho \in \mathbb{R}_+$ denote its scale. Let $\beta_0$ to be a standardized surface (center of mass is at the origin, and major axes are aligned with canonical axes) with a fixed shape and define $\beta = \rho O \beta_0 + T$ be a transformed version of that surface. When we allow unknown transformations, denoted by $\omega = (\rho, O, T) \in \mathbb{R}_+ \times SO(3) \times \mathbb{R}^3$, the cost function $H$ keeps the same form, except the function $\alpha_\sigma$ now depends on these transformation variables, i.e., $\alpha_\sigma(y_i|\omega) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^3 \int_{[0,1]^2} e^{-\frac{1}{2\sigma^2}\|y_i - \rho O \beta_0(s) - T\|^2} ds$. These additional variables are also estimated, along with $\theta$, using maximum-likelihood estimation.

The gradientsof $H$ with respect to $\theta$ remain same as earlier, and its gradient with respect to the translation is given by:

$$\frac{\partial H}{\partial T} = (2\pi)^{-\frac{3}{2}\sigma^{-5}} \sum_{i=1}^{m} \frac{\gamma}{\lambda + \gamma \alpha_\sigma(y_i|\omega)} \int_{[0,1]^2} e^{-\frac{1}{2\sigma^2}\|y_i - \beta(s)\|^2} (y_i - \beta(s), e_j) ds,$$

where $j = 1, 2, 3$, and $e_1 = (1, 0, 0)'$, $e_2 = (0, 1, 0)'$, $e_3 = (0, 0, 1)'$. To define its gradient with respect to $O \in SO(3)$, we first define the directional derivative $\eta_j$ of $H$ in the direction of $E_j, j = 1, 2, 3$, where

$$E_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, E_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, E_3 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix},$$

This directional derivative is given by: $\eta_j, j = 1, 2, 3$, where

$$\eta_j = (2\pi)^{-\frac{3}{2}\sigma^{-5}} \sum_{i=1}^{m} \frac{\gamma}{\lambda + \gamma \alpha_\sigma(y_i|\omega)} \int_{[0,1]^2} e^{-\frac{1}{2\sigma^2}\|y_i - \beta(s)\|^2} (y_i - \beta(s), \rho O E_j \beta_0(s)) ds.$$

Then, we get the full gradient of $H$ with respect to $O$ in the form of a matrix $M = \eta_1 E_1 + \eta_2 E_2 + \eta_3 E_3$.

As opposed to the 2D case, the function $H$ in the 3D case may have several local maximums in the transformation $\omega$ space, especially in the scale space. So, we try to search globally using a grid search in the scale space. The MLE procedure is summarized below:

**Algorithm 10 MLE of $\theta$ and $\omega$ for 3D:** Translate the center of $Y$ to the origin and align the major axes with canonical axes using Procrustes rotation.

- For each pair of $\sigma \in [\sigma_l, \sigma_u]$ and $\rho \in [\rho_l, \rho_u]$, perform the following:
1. Set $t = 0$ and initialize the pair $[\gamma_t, \lambda_t]$ with random values in the range $[0, m]$ and initialize $\omega_t = [\rho, I_{3x3}, [0, 0, 0]^t]$.

2. Compute the gradient terms $\frac{\partial H_\sigma}{\partial \gamma}$, $\frac{\partial H_\sigma}{\partial \lambda}$, $\frac{\partial H_\sigma}{\partial T_j}$, and $M$.

3. Update the estimates using:

   $$
   \begin{bmatrix}
   \gamma_{t+1} \\
   \lambda_{t+1} \\
   T_{t+1}
   \end{bmatrix} =
   \begin{bmatrix}
   \gamma_t \\
   \lambda_t \\
   T_t
   \end{bmatrix} + \delta
   \begin{bmatrix}
   \frac{\partial H_\sigma}{\partial \gamma}(\gamma_t, \lambda_t, \omega_t) \\
   \frac{\partial H_\sigma}{\partial \lambda}(\gamma_t, \lambda_t, \omega_t) \\
   \frac{\partial H_\sigma}{\partial T}(\gamma_t, \lambda_t, \omega_t)
   \end{bmatrix},
   $$

   where $\delta$ is a diagonal matrix of step sizes.

4. The update for the rotation matrix is performed by $O_{t+1} \rightarrow O_t e^{\delta_1 M_t}$, where $\delta_1$ is the step size and $M_t$ is the gradient matrix for orientation.

5. If the norm of the gradient vector is small, then stop. Else, set $t = t + 1$ and return to step 2.

   - Record the limiting $(\gamma, \lambda, T, O)$ for each pair of $\sigma$ and $\rho$ and select the solution that results in the largest $H$ value.

An example of this algorithm is shown in Fig. 6.9. Starting with a parameterized surface with the shape of a horse, shown in the left panel of Fig. 6.9, we transform it with random rotation, translation and scale, and simulate a point cloud using the parameters $\gamma = 80, \lambda = 20, \sigma = 0.010, \rho = 0.717$, shown in the second panel of Fig. 6.9. Using Algorithm 10 on this point cloud, the estimated values are $\hat{\gamma} = 83.29, \hat{\lambda} = 28.49, \hat{\sigma} = 0.016, \hat{\rho} = 0.712$ and the maximum log-likelihood ratio is $R(Y) = -471.12$. The third panel shows the original surface drawn at the estimated position, rotation, and scale along with the original $Y$ and the fourth panel shows the evolution of $H$ under Algorithm 10 for the optimal $\sigma$ and $\rho$.

### 6.3.2 Shape Classification

The procedure of detecting shapes can be easily extended to classification of shapes by computing the LLR for a representative for each candidate shape class and then selecting the class with the

83
smallest LLR. Let $R(Y|S_i)$ denote the LLR for the $i^{th}$ shape $S_i$. The estimated shape class is given by: 
$$\hat{i} = \arg\min_{i} R(Y|S_i).$$
We will demonstrate this idea for both 2D and 3D shape classification.

**Shape Classification in 2D Point Clouds.** We will illustrate the problem of shape classification in 2D point clouds using both simulated and real point clouds. The use of simulated data allows one to measure exhaustive classification performance versus noise, clutter, and other variables, while real data is useful to study individual cases.

- **Simulated Data:** In this experiment we take curves representing 10 shapes shown in the left of Fig. 6.10, generate point clouds according to the data model and perform classification, i.e., we compute LLR for each of the 10 classes and select the one with the smallest LLR. To estimate probability of correct classification, we have used 1000 runs (simulations of $Y$) for each value of $k$ (the number of points from the shape) and $\sigma$ at randomly generated transformations. The number of points from clutter is chosen to be the half of points from the curve. For example, $\gamma = 20, \sigma = 0.03$ implies that $Y$ is simulated by sampling 20 points on the curve and 10 clutter points on average. For these simulations, the underlying shape is picked from the set of ten randomly with equal probability. The results are shown in the right of Fig. 6.10 where the probability of correct classification is plotted versus $k$, for three different Gaussian noise levels. This plot suggests that, in case of low noise, the sampling of shapes by $k = 10$ results in approximately 90% classification rate.

- **Real Images:** Now we present some classification results using point clouds extracted from real images as follows. Starting with an image, we use a median filter on the image to remove noise and calculate image gradients with an auto-adaptive threshold. We then merge rectangular regions that roughly contain high gradients and find the largest such rectangular region. Then, we obtain a binary image for the rectangular region. Next, we apply an image thinning algorithm to the rectangular region and find the contour of the object. Finally, we
sample random points on the contour to obtain $Y$. Some steps of this process are shown in Fig. 6.11 using an example image.

Using this technique, we extract point clouds $Y$ from a number of test images: the images and the corresponding point clouds are shown in the left two panels of Fig. 6.12 in each example. For these images, we try to classify them according to the ten shapes $\beta$ shown in the left of Fig. 6.10. That is, in this experiment $Y$ comes from images and $\beta$ comes from the shapes shown in Fig. 6.10. The results, in form of the LLRs for each of the 10 classes, are shown in the right panels for each case in Fig. 6.12. In most cases we obtain a very low LLR for the correct shape (and sometimes a similar shape). It must be noted that the test and training data came from different sources, the shapes present in the test images are slightly different from the training shapes, and that causes some of the misclassifications in these experiments.
Shape Classification in 3D Point Clouds. To investigate shape estimation and classification in 3D, we construct a database of 20 classes from Princeton shape benchmark in [117], where each class contains 5 examples, shown in Fig. 6.13. The test set consists of 20 examples, generated by choosing one example from each class, and the training set contains the remaining 80 shapes. To generate cloud data, we take a surface from the test set, transform it with random transformation, and simulate \( Y \) by sampling \( k \) points from the transformed surface with a Gaussian noise and half number of clutter points. For each \( Y \), we compute the LLRs for each of the 80 training shapes using Algorithm 10 and select the class containing the shape with the smallest LLR. One such example is shown in the left panel of Fig. 6.14 where the two shapes with smallest LLRs are found to be from the same class as the point cloud. The performance curve versus number of sampling points \( k \) is plotted in the right of Fig. 6.14. In order to compare our performance with some current methods, we have studied the performance of classification using the Gromov-Hausdorff distance ([118, 119]) and the iterative closest point (ICP) algorithm ([111]) based on 1-nearest neighbor classifier. This plot suggests that the sampling of shapes by \( k = 500 \) points can result in approximately 90% classification rate using our method and the Gromov-Hausdorff distance. It also shows that our method and Gromov-Hausdorff distance are better than the ICP algorithm. It is intuitive that the classification performance varies from class to class. That is, we expect better performance for identifying classes that have small variations within the class and are distinct from other classes, compared to classes that have large variations and are similar to other classes. For example, for a fixed \( k = 250 \), the classification rate of nearest neighbor classifier is 78% for the class of “bottle” rather than 69% for the class of “couch”. Under the same setup, the rates are 70.4% for chair, 72.6% for airplane, 79.5% for shoe, and 78.4% for duck.
6.3.3 Shape Estimation

Beyond the problems of shape detection and classification, there is an interesting problem of estimating the shape itself from the data. By shape estimation we mean that we select a relevant subset of points in the given cloud and join them in an appropriate order to form our best estimate of the underlying shape. This brings up two important questions: How should we select the relevant points and how should we determine the ordering? For the first question, the issue can be handled using marked point process models, where each point is characterized by a binary mark: 0 for clutter and 1 for object boundary. Then, one can estimate these marks or labels according to a chosen model. While it is possible to impose a prior distribution on the parameters and use MCMC-based Bayesian estimation of labels, we have simply used the estimated parameters to select a subset of $\hat{Y}$ as follows. We set $n = \hat{\gamma}$ (after rounding) as the number of points to be selected, and then pick $n$ data points with the largest values of $\alpha_{\hat{\sigma}}(y_i)$. For the second question, the one relating to the ordering of points, we use the ordering borrowed from the fitted hypothesized shape. That is, for each selected point, we find the nearest point on the object $\beta$ (under estimated position, orientation and scale parameters) and inherit the ordering from those corresponding points. We illustrate this process using examples from 2D and 3D domains.

**Shape Estimation in 2D Point Clouds.** For a given point cloud and a hypothesis shape $\beta_0$, we first detect if the shape is present in the data. If yes, using the estimates $\hat{\gamma}$, $\hat{\sigma}$, and $\hat{\omega}$, we select $\hat{\gamma}$ number of points in $Y$ with the largest values of $\alpha_{\hat{\sigma}}$. Then, we connect them in the same order as their nearest neighbors on $\beta_0$. Examples of this process are shown in Fig. 6.15.

**Shape Estimation in 3D Point Clouds.** Given a 3D point cloud $Y$ we first test the presence of given shape hypothesis $\beta_0$. If the shape is detected then, as earlier, we select $\hat{\gamma}$ points with the largest $\alpha_{\hat{\sigma}}$ values. Next we use the triangle information of the matched shape surface, i.e., $\beta_0$ at the estimated transformation, to reconstruct the shape from the point cloud. A basic vertex contraction
Figure 6.15: Examples of 2D shape estimation (black: point clouds, red: estimated shapes).

algorithm taken from [120] is used here. Some examples are shown in Fig. 6.16. The point cloud in each example is simulated from the surface at randomly generated position, orientation, and scale with some added clutter. Each pair of panels shows a point cloud and estimated shape from the cloud. Fig. 6.17 shows that the method can also estimate part of shape correctly. Here each row shows a point cloud simulated from part of the shape template, estimated template under the estimated transformation in the cloud, and the shape estimated from the cloud.

**Computational Cost:** We summarize the computational cost of our procedure. This cost is computed using MATLAB on a single-processor desktop PC. For computing LLR for a 2D shape involving $m = 100$ data points, the program takes approximately one second, and for computing LLR for a 3D shape involving $m = 500$ data points, it takes about eight seconds.
Figure 6.16: Shape estimation from 3D point clouds. In each example, the left shows the point cloud and the right shows the cloud with the estimated shape at the estimated pose and scale.

Figure 6.17: Examples of partial estimation. Left: simulated point cloud, middle: estimated template in the cloud, right: estimated shape.
CHAPTER 7
CONCLUSION

Trajectory analysis on Riemannian manifolds is important in many different application areas including computer vision, medical imaging, meteorology and bioinformatics. In this work, we have discussed the problem of fitting smoothing splines to time-indexed data points on a finite-dimensional Riemannian manifold $M$ by means of a Palais-based gradient method. We have studied four specific manifolds and described the relevant parts of their differential geometries. As a result we have derived the gradients of the cost function on these manifolds and have applied the gradient algorithm for finding smoothing splines for a variety of real and simulated data examples.

Next, we have introduced a quantity that provides both a cost function for temporal registration and a proper distance for comparison of trajectories. This distance, in turn, is used to define statistical summaries, such as the sample means and covariances, of given trajectories and “Gaussian-type” models to capture their variability. An essential property of this distance is that it is invariant to identical time-warpings (or temporal re-parameterizations) of trajectories. This is based on a novel mathematical representation of trajectories, termed transported square-root vector field (TSRVF), and the $L^2$ norm on the space of TSRVFs. We have illustrate this framework using four representative manifolds with related applications. Both theoretical proofs and experimental results are provided to validate this framework.

In addition, we are exploring a new method for analysis of trajectories on spheres. We have defined a new representation of trajectory and a new metric. This will avoid the problem of selecting a reference point and becomes more intuitive. We will focus on that direction for future work.

Finally, we have presented a fully statistical framework for detecting, classifying and estimating shapes in cluttered point cloud and have demonstrated it using interesting examples of both real and simulated data. This framework is based on a composite Poisson process: one for points generated from the shape and another for points belonging to the background clutter. This model allows computation of a log-likelihood ratio for each class against clutter and this ratio leads to a formal procedure for detection and classification of shapes. Furthermore, we can also estimate the shape from the cloud based on this framework and have applied it to both 2D and 3D cases.
APPENDIX A

ANOTHER METHOD FOR ANALYSIS OF TRAJECTORIES ON $S^2$

A.1 Proof of Lemma 1

To prove Lemma 1, let us start from a simple claim in the following:

- **Claim:** Given two points $p_1$ and $p_2$ in $S^2$, and given an isometry $L : T_{p_1}S^2 \rightarrow T_{p_2}S^2$, the shortest path from $p_1$ to $p_2$ that induces $L$ as its parallel translation map is an arc of a circle (but not necessarily a great circle) from $p_1$ to $p_2$.

- **Proof of this Claim:** Let $\beta$ be the shortest geodesic joining $p_1$ to $p_2$ and let $L : T_{p_1}S^2 \rightarrow T_{p_2}S^2$ be the parallel translation map induced by $\beta$. Let $\zeta$ be any other arc from $p_1$ to $p_2$ that is disjoint from $\beta$. The Gauss Bonnet Theorem states that the angle of rotation of the parallel translation map $T_{p_1}S^2 \rightarrow T_{p_2}S^2$ induced by the concatenation $\zeta^{-1} \ast \beta$ is equal to the integral of the Gaussian curvature over the region enclosed by the loop $\beta \cup \zeta$. Since the Gaussian curvature of $S^2$ equals $+1$ at every point, this implies that this angle of rotation is equal to the area enclosed by the loop. However, it is well known that of all curves that enclose a given area, a circle is the shortest! From this, it is easy to prove that if part of your loop is already given (by the geodesic, as in this case), then the shortest way to fill in the rest of your arc to enclose a given area is by a circular arc. This proves the claim and can be extended to prove Lemma 1.
A.2 Proof of Theorem 3

First, the optimal circular arc \( \hat{\beta} \) will not change with respect to arbitrary \( \gamma \)'s since the starting points are fixed, so for \( l_{\hat{\beta}} \). Starting with the left side, we get:

\[
d^2(q_{\alpha_1 \circ \gamma}, q_{\alpha_2 \circ \gamma}) = l_{\hat{\beta}}^2 + \int_0^1 \|q_{\alpha_1}, \gamma)(t) - (q_{\alpha_2}, \gamma)(t)\|^2 dt
\]

\[
= l_{\hat{\beta}}^2 + \int_0^1 \|\dot{q}_{\alpha_1}(\gamma(t))\sqrt{\gamma'(t)} - q_{\alpha_2}(\gamma(t))\sqrt{\gamma'(t)}\|^2 dt
\]

\[
= l_{\hat{\beta}}^2 + \int_0^1 \|\dot{q}_{\alpha_1}(\gamma(t)) - q_{\alpha_2}(\gamma(t))\|^2 \gamma'(t) dt
\]

\[
= l_{\hat{\beta}}^2 + \int_0^1 \|\dot{q}_{\alpha_1}(s) - q_{\alpha_2}(s)\|^2 \gamma'(t) ds
\]

\[
= d^2(q_1, q_2) \text{, where we have used } s = \gamma(t). \square
\]
REFERENCES


BIOGRAPHICAL SKETCH

Jingyong Su was born on November 12, 1983 in Anhui, China. He enrolled at Harbin Institute of Technology in September, 2002 and graduated with a Bachelor of Engineering in Electrical Engineering in July, 2006. He began his studies in Harbin Institute of Technology Shenzhen Graduate School in September, 2006 and received a Master of Engineering in Information Engineering under the supervision of Prof. Zheming Lu in July, 2008. Then, he moved to Tallahassee, FL in the fall of 2008. At the Florida State University, he studied toward a Doctor of Philosophy degree under the supervision of Prof. Anuj Srivastava. He will graduate with the Doctor of Philosophy degree in Statistics in August, 2013.