2012

Functional Data Analysis and Partial Shape Matching in the Square Root Velocity Framework

Daniel T. Robinson

THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

FUNCTIONAL DATA ANALYSIS AND PARTIAL SHAPE MATCHING IN THE SQUARE ROOT VELOCITY FRAMEWORK

By

DANIEL T. ROBINSON

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

Degree Awarded:
Fall Semester, 2012

Daniel T. Robinson defended this dissertation on August 22, 2012.

The members of the supervisory committee were:

Eric Klassen
Professor Directing Dissertation

Laura Reina
University Representative

Steven Bellenot
Committee Member

Washington Mio
Committee Member

Anuj Srivastava
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.

ACKNOWLEDGMENTS

Parts of this work were supported by NSF grant DMS-0915003. I also extend the deepest gratitude to Dr. Eric Klassen and Dr. Anuj Srivastava for their guidance, expertise, and support throughout this project. Thanks also to Dr. Jinfeng Zhang and Jose Laborde for their advice on protein structural analysis, and to Hongquan Sun for his input on partial shape matching. I also acknowledge the support of my other colleagues and professors in the Mathematics and Statistics departments, from whom I learned a great deal during my time at Florida State University. Finally, I would like to thank my family for their constant support and encouragement, which have been especially important to me over the past few years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABSTRACT

We investigate two problems in elastic curve shape analysis, working within the context of the square root velocity (SRV) framework. The first of these is to develop specialized algorithms for the analysis of one-dimensional curves, which are just real-valued functions. In this particularly simple case, the elastic matching problem can be stated as a finite combinatorial problem in which the optimal solution can be found exactly. We also develop a method for groupwise alignment, and use it to compute Karcher means of collections of functions. Second, we consider the problem of finding optimal partial matches between curves in $\mathbb{R}^n$ within the SRV framework, and present algorithms and heuristics to solve this problem. Finally, we give a brief overview of `libsrvf`, an open-source software library providing implementations of the algorithms developed in the course of this work.

CHAPTER 1

INTRODUCTION

In this chapter, we give a brief overview of shape analysis and its applications, focusing on shape analysis of curves in Euclidean space. Section 1.1 outlines some of the goals of shape analysis in general, and provides a bit of historical background. In Section 1.2, we begin discussing shape analysis of curves, and we describe several of the methods and frameworks that have been proposed over the years. Finally, in Section 1.3, we give a detailed description of the square root velocity framework, which we will be using in later chapters.

1.1 Overview of Shape Analysis

Objects have many physical attributes that can be measured and compared, such as volume, weight, height, color, etc. *Shape analysis* aims to develop mathematical descriptions of shape, so that shapes of objects can also be measured and quantitatively compared. Specifically, goals of shape analysis include [33]

- **Quantifying similarity/difference between shapes**
  The space of shape descriptors is equipped with some sort of a distance function. This distance function is taken as a measure of the difference between shapes, enabling us to make statements such as "shapes A and B are more similar than shapes C and D."

- **Shape classification**
  Suppose we have a database of previously-observed shapes from each of several different classes. Given a test shape, we would like to determine which of these classes the test shape most likely belongs to.

- **Finding the most efficient way to deform one shape into another**

- **Defining a mean shape to represent a shape class**

- **Modeling shape variability in a class**

1.1.1 Shape Representations

In order to do any of these things, we need a mathematical representation for the shapes that we will be analyzing. Many representations have been proposed, including point clouds, landmark-based representations, parametric representations, representations based on the medial axis, level sets, and several frameworks which represent shapes using sequences of local feature vectors.. Whatever representation we choose, we usually need to achieve invariance to one or more shape-preserving transformations. In fact, Kendall and Le [17], [23] define shape as all of the geometric information that remains after the effects of certain shape-preserving transformations (translation, rotation, and scale) have been filtered out of an object. If two objects differ only by some combination of these transformations, then we want to consider their shapes to be identical, and if we have a distance measure on the set of all shapes, then the distance between such shapes in that metric should be zero. This is one of the major challenges in shape analysis, and the choice of representation has a large impact on how difficult it is to achieve this invariance.

1.2 Shape Analysis of Curves

In many different applications of shape analysis, the objects of interest can be modeled as a continuous curves. For example, in medical image analysis, boundaries of anatomical parts are modeled as closed planar curves. The backbone of a protein chain or an RNA molecule can be approximated by a smooth curve in $\mathbb{R}^3$. Handwriting samples are naturally represented using open planar curves. In this section, we briefly describe a few of the many frameworks that have been proposed for shape analysis of curves.

1.2.1 Landmarks and Active Shape Models

One of the earliest frameworks was the landmark-based shape analysis of Kendall and Le [17], [23]. In this method, each shape is represented by a *landmark set*, which is just a sequence of k points in the ambient space $\mathbb{R}^m$. The set of all possible landmark sets is then $\mathbb{R}^{m \times k}$. Starting with this space, the authors methodically and very rigorously proceed to remove variability due to translation, rotation, and (optionally) scaling to arrive at their space of shape descriptors, which is the space of orbits under the action of the rotation group. They also equip this orbit space with a Riemannian metric, making it possible to compute distances and geodesics between shapes.

1.2.2 Sliding Landmarks

In applications such as medical image analysis, landmarks are typically placed at select points along the boundary of the object of interest, and then these landmarks are used to compare the shapes of the objects. However, such a comparison depends on the placement of the landmarks: even identical object boundaries will have a nonzero dissimilarity measure if the placement of the landmarks differs from one to the other. Bookstein [5] addressed this issue by allowing the landmarks to slide along object boundary curves to achieve a better correspondence.

1.2.3 Medial Axis and Shock Graph

A planar shape can be represented by its *medial axis*, which is the locus of centers of circles lying inside the shape which are multiply-tangent to the boundary; this idea was introduced by Blum [3]. Sebastian *et al.* [29] extended the medial axis representation with dynamic flows, which arise from letting waves propagate inward from the object's boundary and observing the *shocks* (singularities) where the waves collide with each other. The medial axis, together with these flows, is the object's *shock graph*; in this framework, one shape is deformed into another by editing its shock graph.

1.2.4 Parametric Representations

Shapes can also be represented by continuous functions, rather than collections of discrete points. For boundaries of planar shapes and other objects that can be represented by curves, there are many ways to do this; the most obvious is to parametrize the curve with a function $\beta : I \to \mathbb{R}^n$, and then let β (or one of its derivatives) serve as the representative. If β itself is used, then translations, rotations, scalings, and reparametrizations will in general all result in a different function, so this representation offers no invariance. The first derivative is translation-invariant, but still varies under rotations, scalings, and reparametrizations. The curvature function is invariant to both translations and rotations.

Klassen *et al.* [18] introduced a representation for planar curves parametrized by arc length in which each curve is represented by its *angle function* $\theta(t)$, defined as the elevation angle of the tangent vector of the curve at t (values are chosen so that θ is continuous). This representation is invariant to translation and reparametrization, and can be made rotation-invariant by vertically shifting each angle function so that it has an average value of π (say). However, since all curves are required to be parametrized by arc length, it is not possible to reparametrize curves to improve the registration between them.

Other parametric shape analysis frameworks do allow the curves to be reparametrized; these are referred to as *elastic* shape analysis frameworks, since they model deformations as combinations of bending and stretching. For example, Michor, Mumford and others ([38, 25]) have studied several different metrics on the space of smooth planar curves modulo reparametrization. Mio *et al.* [27], [26] used a parametric curve representation along with an elastic metric obtained as a weighted combination of stretching and bending energies. The square root velocity framework used in this paper also falls into the category of elastic shape analysis frameworks.

1.2.5 Feature Vector Representations

Several representations have been proposed which introduce a local feature descriptor, along with a metric on the space of all possible descriptor values. Curves are then represented as finite sequences of feature points, and the dissimilarity between two curves is defined in terms of the distances between corresponding feature points with respect to the metric. One of the best-known examples is the *shape context* descriptor for planar shapes, introduced by Belongie *et al.* [2] in 2002. Given points $p_1, p_2, \ldots, p_n \in \mathbb{R}^2$ representing a shape, they compute a log-polar histogram for each of the points p_i, which encodes the distribution of the positions of the other points relative to p_i. The descriptor is translation-invariant, and can be made invariant to rotations and scaling as well. Other examples of this type of representation include the turning angle / distance across the shape feature used by Chen [9], and the chord distribution representation of Donoser *et al.* [10].

1.3 The Square Root Velocity Framework

In the following chapters, we will use the *square root velocity (SRV) framework*, which was introduced in [27] and [15], for elastic shape analysis of curves. In this framework, each curve is represented by its square root velocity function (SRVF), which we define below. This representation depends only on the first derivative of the curve, and so is inherently translation-invariant. It can optionally be made scaling invariant by first scaling all curves to unit length as a preprocessing step. Invariance to rotation and reparametrization is achieved by modding out by the actions of $\mathrm{SO}(\mathrm{n}, \mathbb{R})$ and the group Γ of orientation-preserving diffeomorphisms of the domain. Both of these group actions are isometric with respect to the $\mathbb{L}^2$ metric, which gives rise to a distance function on the space of orbits.

We now introduce the square root velocity representation. Given a curve $\beta : [0,1] \to \mathbb{R}^n$ satisfying certain requirements (to be discussed in a moment), the associated SRVF $q : [0,1] \to \mathbb{R}^n$ is defined as

$$q(t) = \begin{cases} \frac{\dot{\beta}(t)}{\sqrt{\|\dot{\beta}(t)\|}} & \text{if } \dot{\beta}(t) \text{ exists and is nonzero} \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

As noted above, q depends only on the first derivative of β, so that any two curves which differ only by a translation will have the same SRVF representative. Also note that for all t, we have $\|q(t)\| = \sqrt{\left\|\dot{\beta}(t)\right\|}$, which is the reason for the name *square root velocity function.* A consequence of this is that the squared $\mathbb{L}^2$ norm of q is just the arc length of β:

$$\int_0^1 \|q(t)\|^2 \, dt = \int_0^1 \left\|\dot{\beta}(t)\right\| dt = L[\beta]$$

In applications that require scaling invariance, we rescale all curves to unit arc length as a preprocessing step, before doing any analysis. In this case, all SRVFs q will lie on the unit sphere in $\mathbb{L}^2$. Also note that if we are given q, we can recover β (up to a translation) via the formula

$$\beta(t) = \int_0^t q(s) \, \|q(s)\| \, ds$$

1.3.1 The Class of Admissible Curves

We now identify the class of functions $\beta : [0,1] \to \mathbb{R}^n$ in which we are interested. For this, we will need a result from real analysis.

Definition 1. *A function $f : [a,b] \to \mathbb{R}$ is* ***absolutely continuous on*** *$[a,b]$ if for all $\varepsilon > 0$, there exists $\delta > 0$ such that, for any collection $(a_1, b_1), \ldots, (a_n, b_n)$ of disjoint open subintervals of $[a,b]$ with total length at most δ, we have*

$$\sum_{i=1}^{n} |f(b_i) - f(a_i)| \leq \varepsilon$$

Theorem 1. *Let $f : [a, b] \to \mathbb{R}$. Then f is absolutely continuous on $[a, b]$ iff there exists an integrable function $g : [a, b] \to \mathbb{R}$ such that*

$$f(x) - f(a) = \int_a^x g(t)dt \text{ , for } a \leq x \leq b$$

Furthermore, if f is absolutely continuous on $[a, b]$, then on all but a set of measure zero, f is differentiable and $f' = g$. Therefore, f is absolutely continuous on $[a, b]$ iff

$$f(x) - f(a) = \int_a^x f'(t)dt \text{ , for } a \leq x \leq b$$

Proof. See Ash and Doléans-Dade [1], section 2.3. □

To extend this result to the case of vector-valued functions, we say that a function $f : [a, b] \to \mathbb{R}^n$ is absolutely continuous if and only if all of its component functions are absolutely continuous. Then the vector-valued analogue of Theorem 1 is an immediate corollary.

Now, let $\mathcal{AC}_0$ denote the set of all absolutely continuous functions $\beta : [0, 1] \to \mathbb{R}^n$ such that $\beta(0) = 0$. Let $\mathcal{A} \subset \mathcal{AC}_0$ denote the set of all $\beta \in \mathcal{AC}_0$ such that $\int_0^1 \left\|\dot{\beta}(t)\right\| dt = 1$ (note that β may fail to be differentiable on a set of measure zero, but this will not affect the value of this integral). This will be the space of curves which we will consider; note that we have already removed translation and scaling variability by imposing appropriate conditions on these curves. Let $\mathcal{C}$ denote the unit sphere in $\mathbb{L}^2$, and let $\mathcal{Q} : \mathcal{A} \to \mathcal{C}$ be the map that sends each curve to its SRV function.

Proposition 1. *$\mathcal{Q}$ has inverse function $\mathcal{Q}^{-1} : \mathcal{C} \to \mathcal{A}$ given by*

$$\mathcal{Q}^{-1}(q)(t) = \int_0^t q(s) \left\|q(s)\right\| ds.$$

Hence, we have a one-to-one correspondence between the set of admissable curves and $\mathcal{C}$.

Proof. For $\beta \in \mathcal{A}$, let $q = \mathcal{Q}(\beta)$. Then by equation 1.1 and Theorem 1, $q(t) \left\|q(t)\right\| = \dot{\beta}(t)$ almost everywhere, so we have

$$\int_0^t q(s) \left\|q(s)\right\| ds = \beta(t) - \beta(0) = \beta(t) \text{ for } t \in [0, 1]$$

Hence, $\mathcal{Q}^{-1} \circ \mathcal{Q} = id_{\mathcal{A}}$.

Now, if $q \in \mathcal{C}$, then let $\beta = \mathcal{Q}^{-1}(q)$, and note that $\beta(0) = 0$. Also, by the above, β is absolutely continuous and differentiable almost everywhere, and $\dot{\beta}(t) = q(t) \left\|q(t)\right\|$ almost everywhere. Therefore,

$$\int_0^1 \left\|\dot{\beta}(t)\right\| dt = \int_0^1 \left\|q(t)\right\|^2 dt = \left\|q\right\|^2 = 1$$

so that $\mathcal{Q}^{-1}$ does in fact map into $\mathcal{A}$. Finally, we have

$$\mathcal{Q}(\beta)(t) = \begin{cases} \frac{q(t)\|q(t)\|}{\sqrt{\|q(t)(\|q(t)\|)\|}} = q(t) & \text{if } q(t) \neq 0 \\ 0 & \text{if } q(t) = 0 \end{cases}$$

for almost all $t \in [0,1]$, so that $\mathcal{Q}(\beta) = q$ almost everywhere (i.e., $\mathcal{Q}(\beta) = q$ in $\mathbb{L}^2$). Thus, $\mathcal{Q} \circ \mathcal{Q}^{-1} = id_{\mathcal{C}}$. □

Definition 2. *The set $\mathcal{C} = \mathcal{Q}(\mathcal{A})$ of all SRV functions of admissible curves is called the **preshape space**.*

1.3.2 The Geometry of the Preshape Space $\mathcal{C}$

Now that we have identified the preshape space, we say a little bit about its geometry. $\mathcal{C}$ is a C^∞ submanifold of $\mathbb{L}^2([0,1], \mathbb{R}^n)$. The tangent space at a point $q \in \mathcal{C}$ is the set of all vectors which are orthogonal to q:

$$T_q\mathcal{C} = \{w \in \mathbb{L}^2 \mid \langle w, q \rangle = 0\}$$

Each tangent space is equipped with the usual $\mathbb{L}^2$ inner product, and the geodesics on $\mathcal{C}$ are the great circle paths (where a great circle path is the intersection of $\mathcal{C}$ with a two-dimensional subspace of $\mathbb{L}^2$). We denote by $d(q_1, q_2)$ the geodesic distance on $\mathcal{C}$ between q_1 and q_2, which is given by

$$d(q_1, q_2) = \arccos(\langle q_1, q_2 \rangle) \tag{1.2}$$

$\mathcal{C}$ is a space of shape descriptors which are invariant with respect to translations and scalings – that is, if two curves differ only by a scaling or a translation, then both of these curves will have the same representative in $\mathcal{C}$. However, $\mathcal{C}$ is called a *pre*shape space because curves that differ only by a rotation or a reparametrization will still have different representatives in $\mathcal{C}$. The set of *shape* descriptors will be a quotient of $\mathcal{C}$, obtained by modding out rotations and reparametrizations. Let Γ denote the group of orientation-preserving C^∞ diffeomorphisms of $[0,1]$ (representing reparametrizations), and let $\mathrm{SO(n, \mathbb{R})}$ denote the group of orthogonal $n \times n$ real matrices with determinant 1 (representing rotations). Our goal is to mod out by appropriate actions of these groups on $\mathcal{C}$, and then define distances and geodesics in the resulting quotient space.

1.3.3 The Actions of $\mathrm{SO(n, \mathbb{R})}$ and Γ

First, the rotation group $\mathrm{SO(n, \mathbb{R})}$ acts from the left on $\mathcal{C}$ by pointwise multiplication. This action is isometric: for any $q_1, q_2 \in \mathcal{C}$ and any $O \in \mathrm{SO(n, \mathbb{R})}$, we have

$$\langle Oq_1, Oq_2 \rangle = \int_0^1 \langle Oq_1(t), Oq_2(t) \rangle \, dt = \int_0^1 \langle q_1(t), q_2(t) \rangle \, dt = \langle q_1, q_2 \rangle \tag{1.3}$$

(the second equality comes from the fact that O is an isometry of $\mathbb{R}^n$).

The reparametrization group Γ acts on $\mathcal{C}$ from the right as follows: for $\gamma \in \Gamma$ and $q \in \mathcal{C}$,

$$q * \gamma = \sqrt{\dot{\gamma}}(q \circ \gamma) \tag{1.4}$$

The action is defined in this way so that acting on an SRVF is equivalent to reparametrizing the corresponding curve. That is, if $q = \mathcal{Q}(\beta)$ is the SRV function for a curve $\beta \in \mathcal{A}$, and

$\gamma \in \Gamma$, then the SRVF for $\beta \circ \gamma$ is given by

$$\begin{aligned}\mathcal{Q}(\beta \circ \gamma)(t) &= \frac{\dot{\beta}(\gamma(t))\dot{\gamma}(t)}{\sqrt{\|\dot{\beta}(\gamma(t))\dot{\gamma}(t)\|}} = \sqrt{\dot{\gamma}(t)}\frac{\dot{\beta}(\gamma(t))}{\sqrt{\|\dot{\beta}(\gamma(t))\|}} \\ &= \sqrt{\dot{\gamma}(t)}q(\gamma(t)) = (q * \gamma)(t)\end{aligned}$$

at points where $\dot{\beta}(t)$ exists and is nonzero, and is zero elsewhere.

Now, to see that Γ acts by isometries, let $q_1, q_2 \in \mathcal{C}$, and let $\gamma \in \Gamma$. Then

$$\begin{aligned}\langle q_1 * \gamma, q_2 * \gamma \rangle &= \int_0^1 \left\langle \sqrt{\dot{\gamma}(t)}q_1(\gamma(t)), \sqrt{\dot{\gamma}(t)}q_2(\gamma(t)) \right\rangle dt \\ &= \int_0^1 \langle q_1(\gamma(t)), q_2(\gamma(t)) \rangle \dot{\gamma}(t) dt \\ &= \int_0^1 \langle q_1(\gamma), q_2(\gamma) \rangle d\gamma \\ &= \langle q_1, q_2 \rangle\end{aligned} \tag{1.5}$$

Another useful fact is that these two actions commute: reparametrizing and then rotating gives the same result as rotating and then reparametrizing. To see this, let $q \in \mathcal{C}$, $O \in \mathrm{SO(n,\mathbb{R})}$, and $\gamma \in \Gamma$. Then

$$O(q * \gamma) = O(\sqrt{\dot{\gamma}}(q \circ \gamma)) = \sqrt{\dot{\gamma}}((Oq) \circ \gamma) = (Oq) * \gamma$$

1.3.4 The Shape Space $\mathcal{S}$

Our space of shape descriptors will be a sort of quotient of $\mathcal{C}$ under the actions of $\mathrm{SO(n,\mathbb{R})}$ and Γ. For $q \in \mathcal{C}$, we define $[q]$ as follows:

$$[q] = \text{closure}(\{O\sqrt{\dot{\gamma}}(q \circ \gamma) \mid O \in \mathrm{SO(n,\mathbb{R})}, \gamma \in \Gamma\}) \tag{1.6}$$

We then define our *shape space* as the set of all such closed-up orbits:

$$\mathcal{S} = \{[q] \mid q \in \mathcal{C}\} \tag{1.7}$$

An important fact is that the closed-up orbits are equivalence classes, and therefore the set of all closed-up orbits forms a partition of $\mathcal{C}$. Define the relation $\sim$ on $\mathcal{C} \times \mathcal{C}$ as

$$v \sim w \text{ if and only if } v \in [w] \tag{1.8}$$

Proposition 2. *The relation $\sim$ is an equivalence relation.*

Proof. Clearly $v \in [v]$ for all $v \in \mathcal{C}$, so $\sim$ is reflexive. To show that $\sim$ is symmetric, suppose that $v \in [w]$, and let $\{A_i w * \gamma_i\}$ be a sequence of points in the orbit of w which converges to v in $\mathbb{L}^2$. Then we have

$$\lim_{i\to\infty} \|A_i w * \gamma_i - v\| = 0$$
$$\lim_{i\to\infty} \left\|w - A_i^{-1} v * \gamma_i^{-1}\right\| = 0$$

where the second equality is an easy consequence of the fact that both of the group actions are isometric. Therefore, $A_i^{-1} v * \gamma_i^{-1} \longrightarrow w$, and $w \in [v]$. Finally, to show that $\sim$ is transitive, suppose that $u \in [v]$ and $v \in [w]$. Consider sequences $\{A_i v * \gamma_i\}$ and $\{B_i w * \eta_i\}$ of points in the orbits of v and w, respectively, with $A_i v * \gamma_i \longrightarrow u$ and $B_i w * \eta_i \longrightarrow v$. Then

$$\begin{aligned} \|A_i B_i w * \eta_i * \gamma_i - u\| &\le \|A_i B_i w * \eta_i * \gamma_i - A_i v * \gamma_i\| + \|A_i v * \gamma_i - u\| \\ &= \|B_i w * \eta_i - v\| + \|A_i v * \gamma_i - u\| \\ &\longrightarrow 0 \end{aligned}$$

so $\{A_i B_i w * \eta_i * \gamma_i\}$ is a sequence of points in the orbit of w which converges to u, and $u \in [w]$. □

This shows that the space $\mathcal{S}$ gives us a partition of the preshape space $\mathcal{C}$ into closed sets such that any two SRVFs which differ only by a rotation or a reparametrization will belong to the same equivalence class. Further, it can be shown that q and w are SRVFs in $\mathcal{C}$ with $w \in [q]$, then the curves corresponding to q and w pass through exactly the same points in $\mathbb{R}^n$, so placing q and w in the same shape class makes sense (*c.f.* Srivastava *et al.* [34]). We will take $\mathcal{S}$ as our set of shape descriptors; our next task is to define a metric on this space.

1.3.5 A Metric on the Shape Space

To define a distance between two closed-up orbits $[q_1], [q_2] \in \mathcal{S}$, we simply take the smallest distance between any pair of representatives:

$$\begin{aligned} \rho([q_1],[q_2]) &= \inf_{w \in [q_1], v \in [q_2]} d(w, v) \\ &= \inf_{O_1, O_2, \gamma_1, \gamma_2} d(O_1 q_1 * \gamma_1, O_2 q_2 * \gamma_2) \\ &= \inf_{O,\gamma} d(q_1, O q_2 * \gamma) \end{aligned} \tag{1.9}$$

The last equality follows from the fact that both group actions are isometric, so we can apply the inverses of O_1 and γ_1 to both functions without changing the inner product:

$$\langle O_1 q_1 * \gamma_1, O_2 q_2 * \gamma_2 \rangle = \left\langle q_1, O_1^{-1}(O_2 q_2 * \gamma_2)\gamma_1^{-1} \right\rangle = \left\langle q_1, (O_1^{-1} O_2) q_2 * (\gamma_2 \circ \gamma_1^{-1}) \right\rangle$$

Therefore, it is enough to optimize over one orbit. Next, we show that this is a proper distance function.

Proposition 3. *$\rho(\cdot,\cdot)$ is a proper metric on $\mathcal{S}$.*

Proof. Clearly, $\rho([q],[q]) = 0$ for any $[q] \in \mathcal{S}$. On the other hand, if $\rho([q_1],[q_2]) = 0$, then there must be a sequence of points in $[q_2]$ converging to q_1, so $[q_1] = [q_2]$. As for symmetry, we have

$$\begin{aligned} \rho([q_2],[q_1]) &= \inf_{O,\gamma} d(q_2, O q_1 * \gamma) \\ &= \inf_{O,\gamma} d(O^{-1} q_2 * \gamma^{-1}, q_1) \\ &= \inf_{O,\gamma} d(q_1, O^{-1} q_2 * \gamma^{-1}) \\ &= \rho([q_1],[q_2]) \end{aligned}$$

Finally, ρ satisfies the triangle inequality: for $[q_1], [q_2], [q_3] \in \mathcal{S}$, we have

$$\begin{aligned}
\rho([q_1],[q_2]) &= \inf_{A,\gamma} d(q_1, Aq_2 * \gamma) \\
&\leq \inf_{A,B,\gamma,\eta} \left(d(q_1, Bq_3 * \eta) + d(Bq_3 * \eta, Aq_2 * \gamma)\right) \\
&= \inf_{A,B,\gamma,\eta} \left(d(q_1, Bq_3 * \eta) + d(q_3, B^{-1}Aq_2 * \gamma * \eta^{-1})\right) \\
&= \inf_{A,B,C,\gamma,\eta,\delta} \left(d(q_1, Bq_3 * \eta) + d(q_3, Cq_2 * \delta)\right) \\
&= \inf_{B,\eta} d(q_1, Bq_3 * \eta) + \inf_{C,\delta} d(q_3, Cq_2 * \delta) \\
&= \rho([q_1],[q_3]) + \rho([q_3],[q_2])
\end{aligned}$$

□

1.3.6 Computing Distances in the Shape Space

In practice, we approximate the distance in equation 1.9 by alternately optimizing over rotations and reparametrizations as described in Algorithm 1.1 until the distance stabilizes, or until a specified number of iterations has been completed. While optimizing over rotations, we keep the parametrizations of both curves fixed, and while optimizing over reparametrizations, we keep the rotational alignments of both curves fixed. Optimizing over rotations can be done efficiently using the Procrustes alignment techniques described in [12]. We describe the procedure for optimizing over reparametrizations in detail in Chapter 3.

Algorithm 1.1 Approximate the shape distance $\rho([q_1],[q_2])$

```
[Initial rotation] Compute $\hat{R} = \operatorname{argmin}_{R \in \mathrm{SO(n,\mathbb{R})}} d(q_1, Rq_2)$ and set $q_2 = \hat{R}q_2$
Set $d_0 = d(q_1, q_2)$
for $i = 1$ to MAX_ITERS do
    Compute $\hat{\gamma} = \operatorname{argmin}_{\gamma \in \Gamma} d(q_1, q_2 * \gamma)$ and set $q_2 = q_2 * \hat{\gamma}$
    Compute $\hat{R} = \operatorname{argmin}_{R \in \mathrm{SO(n,\mathbb{R})}} d(q_1, Rq_2)$ and set $q_2 = \hat{R}q_2$
    Set $d_1 = d(q_1, q_2)$
    if $d_1 > d_0 -$ MIN_PROGRESS then
        Break
    else
        Set $d_0 = d_1$ and continue
    end if
end for
Return $d_1$
```

CHAPTER 2

FUNCTIONAL DATA ANALYSIS IN THE SQUARE ROOT VELOCITY FRAMEWORK

In this chapter, we focus on the analysis of 1-D curves, which are just functions $I \to \mathbb{R}$, using the square root velocity framework. While the framework is applicable to curves in any Euclidean space, the special case of 1-D curves is particularly interesting. While any absolutely continuous curve is rectifiable, the square root velocity function (SRVF) corresponding to a 1-D arc-length parametrized curve is particularly simple, because it takes on only two values: 1 and -1. In this chapter, we develop an elastic matching algorithm which capitalizes on this simple structure.

Before we give a detailed explanation of the new method, we should say a few words about its advantages over the existing framework for curves in $\mathbb{R}^n$ as described in Chapter 1. Both the new method and the existing method require us to solve the *elastic matching problem* (the problem of reparametrizing one or both curves so that their features match as well as possible) as part of the shape distance calculation. However, in the existing SRV framework, one must search over the entire set of increasing diffeomorphisms of $[0, 1]$, which is a very large space. In practice, this problem is discretized and solved using dynamic programming techniques, and the solutions are piecewise linear approximations to the actual best reparametrization. By contrast, the method that we will present in this chapter reduces the elastic matching problem to a combinatorial optimization problem with a finite search space, so the solutions produced by the new method are exact.

2.1 Applying the Square Root Velocity Framework to Real-Valued Functions

The square root velocity framework was introduced in Chapter 1 for the general case of curves in $\mathbb{R}^n$. However, in the case where $n = 1$, the framework becomes simpler in several ways, so we give a quick review here, taking advantage of simplifications where possible.

2.1.1 Definitions

As in Chapter 1, we let $\mathcal{AC}_0$ denote the set of absolutely continuous functions $f : I \to \mathbb{R}$ such that $f(0) = 0$, where $I = [0, 1]$. Then the square root velocity representation gives a bijection between $\mathcal{AC}_0$ and $\mathbb{L}^2(I,\mathbb{R})$:

$$\mathcal{Q}(f)(t) = \begin{cases} \frac{\dot{f}(t)}{\sqrt{|\dot{f}(t)|}} & \text{if } \dot{f}(t) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

with inverse

$$\mathcal{Q}^{-1}(q)(t) = \int_0^t q(s)|q(s)|\, ds$$

As before, any translate (vertical shift) of f will have the same image under $\mathcal{Q}$.

If f has unit arc length, then its SRVF $q = \mathcal{Q}(f)$ will have unit norm in $\mathbb{L}^2(I,\mathbb{R})$. In fact, the SRVF map $\mathcal{Q}$ sends the class of all such functions bijectively onto the unit sphere in $\mathbb{L}^2(I,\mathbb{R})$. Let $\mathcal{A} \subset \mathcal{AC}_0$ denote the set of all functions in $\mathcal{AC}_0$ with unit arc length, and let $\mathcal{C} \subset \mathbb{L}^2(I,\mathbb{R})$ denote the unit sphere in $\mathbb{L}^2(I,\mathbb{R})$.

$\mathcal{C}$ is a set of shape descriptors which is invariant under uniform scaling and translation (vertical shift). For one-dimensional curves, the rotation group $\mathrm{SO}(1,\mathbb{R})$ is trivial, so the only shape-preserving transformation remaining to be dealt with is reparametrization. Identifying functions which differ only by a reparametrization is our next task.

2.1.2 The Reparametrization Group

In order to address this issue, we must decide exactly what we mean by a reparametrization. For the moment, we will take as our set of allowable reparametrizations the set Γ_0 of all increasing, absolutely continuous homeomorphisms $\gamma : I \to I$ which have absolutely continuous inverses. We will see in a moment that this is not *exactly* the right set of functions, but it seems like a reasonable starting point. It is easy to verify that Γ_0 is a group under composition. As in Chapter 1, we let Γ_0 act on $\mathbb{L}^2(I,\mathbb{R})$ from the right as follows: for $q \in \mathbb{L}^2(I,\mathbb{R})$ and $\gamma \in \Gamma_0$, the action of γ on q is defined as

$$q * \gamma = \sqrt{\dot{\gamma}}(q \circ \gamma)$$

As we saw in Chapter 1, the action is defined in this way so that acting on an SRVF is equivalent to reparametrizing the corresponding function. That is, the action is defined so that the following equation holds for any $f \in \mathcal{AC}_0$ and any $\gamma \in \Gamma_0$:

$$\mathcal{Q}(f) * \gamma = \mathcal{Q}(f \circ \gamma)$$

Let $q\Gamma_0$ denote the orbit of $q \in \mathbb{L}^2(I,\mathbb{R})$ under this action. Recall that our action is isometric. Consequently, if $q \in \mathcal{C}$, then $q\Gamma_0 \subset \mathcal{C}$. Therefore, we have a well-defined action of Γ_0 on $\mathcal{C}$.

2.1.3 Shape Space and Distance Function

The set of all orbits $q\Gamma_0$ is a partition of $\mathcal{C}$ which provides a fully-invariant set of shape descriptors for $\mathcal{AC}_0$. That is, if f_1 and f_2 are two elements of $\mathcal{AC}_0$ which differ only by a translation, uniform scaling, or reparametrization by an element of Γ_0, then the SRVFs of f_1 and f_2 will belong to the same orbit in $\mathcal{C}$.

Since the underlying space is a sphere, it is natural to define the distance between two orbits as the smallest possible great-circle distance between any pair of orbit representatives:

$$d(q_1\Gamma_0, q_2\Gamma_0) = \inf_{\gamma_1,\gamma_2\in\Gamma_0} \cos^{-1}(\langle q_1 * \gamma_1, q_2 * \gamma_2\rangle)$$

Unfortunately, this turns out not to be a proper metric:

Claim 1. *There exist distinct orbits $q_1\Gamma_0$ and $q_2\Gamma_0$ such that $d(q_1\Gamma_0, q_2\Gamma_0) = 0$. Therefore, our distance function is degenerate.*

Proof. Define q_1 and q_2 as follows:

$$q_1(t) = 1 \text{ for all } t \qquad\qquad q_2(t) = \begin{cases} 0 & \text{if } t < \frac{1}{2} \\ \sqrt{2} & \text{if } t \geq \frac{1}{2} \end{cases}$$

Clearly q_1 and q_2 have unit $\mathbb{L}^2$-norm, and so are elements of $\mathcal{C}$. Also note that for any $\gamma \in \Gamma_0$

$$(q_1 * \gamma)(t) = \sqrt{\dot{\gamma}(t)}q_1(\gamma(t)) = \sqrt{\dot{\gamma}(t)}$$

Since γ is injective, $\dot{\gamma}$ cannot vanish on the interval $(0, \frac{1}{2})$, so we see that $q_1 * \gamma \neq q_2$. Therefore, $q_2 \notin q_1\Gamma_0$, and the orbits $q_1\Gamma_0$ and $q_2\Gamma_0$ are distinct. We will show that the distance between these two orbits, as defined above, is zero. For $n = 2, 3, 4, \ldots$, define $\gamma_n \in \Gamma_0$ as the piecewise-linear function passing through the points $(0, 0)$, $(\frac{1}{2}, \frac{1}{n})$, and $(1, 1)$. Then

$$(q_1 * \gamma_n)(t) = \sqrt{\dot{\gamma_n}(t)} = \begin{cases} \sqrt{\frac{2}{n}} & \text{if } t < \frac{1}{2} \\ \sqrt{\frac{2(n-1)}{n}} & \text{if } t > \frac{1}{2} \end{cases}$$

so we have

$$\begin{aligned} \langle q_1 * \gamma_n, q_2\rangle &= \int_0^{1/2} \sqrt{\frac{2}{n}} \cdot 0 \; dt + \int_{\frac{1}{2}}^1 \sqrt{\frac{2(n-1)}{n}}\sqrt{2} \; dt \\ &\longrightarrow \int_{\frac{1}{2}}^1 \sqrt{2}\sqrt{2} \; dt = 1 \end{aligned}$$

as $n \longrightarrow \infty$. Therefore, $d(q_1\Gamma_0, q_2\Gamma_0) = 0$. □

In order to obtain a metric which is not degenerate, we pass to the set of $\mathbb{L}^2$ *closures* of Γ_0-orbits. For $q \in \mathcal{C}$, we let $[q]$ denote the $\mathbb{L}^2$ closure of $q\Gamma_0$. Note that, by continuity of the $\mathbb{L}^2$ norm, the limit of any sequence of points with unit norm must also have unit norm. Therefore, we have $[q] \subset \mathcal{C}$ for each closed-up orbit $[q]$.

2.1.4 Extending the Reparametrization Group

It will be useful to express the closed-up orbits $[q]$ as "orbits" under the action of a larger class of functions which contains Γ_0. Let Γ denote the set of all absolutely continuous, *non-decreasing*, surjective functions $\gamma : I \to I$. Γ is a semigroup under function composition: Γ is closed under composition (*c.f.* Bogachev [4], Exercise 5.8.59), the operation is associative, and there is an identity element. However, not every function has an inverse since these functions may not be injective. Clearly, $\Gamma_0 \subset \Gamma$, and since elements of Γ are absolutely continuous (and hence differentiable a.e.), we can let Γ act on $\mathcal{C}$ in the same way as Γ_0. Again, we will use $q\Gamma$ to denote the orbit under this action. Since Γ is not a group, many of the usual facts about group actions do not apply here. However, we can say some useful things about the action of Γ on $\mathcal{C}$.

First, a definition: a function $q \in \mathcal{C}$ is in **standard form** if the interval I can be expressed as the union of disjoint measurable sets A and B, so that

$$q(t) = \begin{cases} 1 & \text{if } t \in A \\ -1 & \text{if } t \in B \end{cases}$$

Klassen [19] showed that whenever $q \in \mathcal{C}$ is in standard form, we have $q\Gamma = [q]$. In practice, we reparametrize all functions by arc length as a preprocessing step, so that all of our SRVFs start out in standard form. Therefore, for our purposes here, Γ is exactly the right set of reparametrization functions.

2.1.5 Step Functions and Matchings

Having identified the full set Γ of reparametrization functions, we now turn to the computational problem of finding an optimal matching between two 1-D curves. A strategy for curves in $\mathbb{R}^n$ was given in Chapter 1. However, 1-D functions have a particularly simple structure. In the remainder of this chapter, we will take advantage of this structure to develop a combinatorial algorithm for elastic matching. First, we restrict our attention to a certain class of "tame" functions:

Definition 3. *A function $q : I \to \mathbb{R}$ is a **step function** if there exists a finite partition $0 = t_0 < t_1 < \ldots < t_n = 1$ of $[0, 1]$ such that q is constant on each open interval (t_{i-1}, t_i).*

Note that we are already requiring all of our functions to be absolutely continuous, and absolutely continuous functions have constant-speed parametrizations (*c.f.* Stein and Shakarchi [35]). A 1-D function that is constant-speed parametrized has a piecewise-constant SRVF, so the only additional restriction arising when we limit ourselves to step functions is that we now insist that all of our SRVFs are piecewise constant *and* have only finitely-many change points. Moreover, it can be shown that the step functions are dense in $\mathbb{L}^2$ (*c.f.* Klassen *et al.* [19]). Restricting to step functions does not limit the applicability of the method in practice, since the datasets that we analyze have only finitely-many points.

Now, let $\mathcal{S}_{st}$ denote the set of all closed-up orbits $[q]$ which contain at least one step function. This restricted set of closed-up orbits will serve as our shape space. Since elements of $\mathcal{S}_{st}$ in general contain SRVFs which are *not* step functions, it is natural to ask whether, given two orbits $[q_1], [q_2] \in \mathcal{S}_{st}$, it is always possible to find step function representatives in these orbits that realize the minimum distance between the orbits. It turns out that this is the case; we will show this by developing an algorithm to find such representatives and proving that they are optimal.

Another important fact concerns geodesics: if q_1 and q_2 are step functions in $\mathcal{C}$, then all SRVFs that lie on the great circle path from q_1 to q_2 in $\mathcal{C}$ are step functions. Indeed, if $0 = t_0^1 < t_1^1 < \ldots < t_{n_1}^1 = 1$ and $0 = t_0^2 < t_1^2 < \ldots < t_{n_2}^2 = 1$ are the changepoint partitions for q_1 and q_2, respectively, we can take the union $0 = t_0 < t_1 < \ldots < t_n$ of these two changepoint partitions. Then both q_1 and q_2 are constant on all subintervals of this refined partition. The great circle path from q_1 to q_2 can be parametrized as

$$\Psi_s(t) = \frac{1}{\sin(\theta)} \left((\sin(\theta(1-s)))q_1(t) + \sin(\theta s)q_2(t)\right)$$

and for any fixed value of the geodesic parameter s, we have a function which is constant on all subintervals of the refined partition $t_0 < t_1 < \ldots < t_n$.

We now explain exactly what we mean by a *matching* between two elements of our (restricted) shape space:

Definition 4. *A* ***matching*** *between $[q_1], [q_2] \in \mathcal{S}_{st}$ is a choice of orbit representatives $w \in [q_1]$ and $v \in [q_2]$ which are step functions.*

Our goal is to find an **optimal matching** between $[q_1]$ and $[q_2]$; that is, a choice of step function representatives which realize the minimum distance between the orbits. In the next section, we take a closer look at optimal matchings, and establish some necessary conditions for optimality.

2.2 Necessary Conditions for an Optimal Match

This section establishes necessary conditions for an optimal match, which will eventually leave us with a finite search space. Let q_1 be a step function with changepoint partition $0 = t_0^1 < t_1^1 < t_2^1 < \ldots < t_{n_1}^1$, and let q_2 be a step function with changepoint partition $0 = t_0^2 < t_1^2 < t_2^2 < \ldots < t_{n_2}^2$. Let $0 = t_0 < t_1 < \ldots < t_n$ be the union of the partitions for q_1 and q_2, so that on each subinterval of this new partition, both functions are constant. At this point we introduce a notational convention to describe the form of the matching between q_1 and q_2. For $i = 1, 2$ and $j = 1, 2, \ldots, n$, we let

$$s_j^i = \begin{cases} \text{`U'} & \text{if } q_i > 0 \text{ on } (t_{j-1}, t_j) \\ \text{`0'} & \text{if } q_i = 0 \text{ on } (t_{j-1}, t_j) \\ \text{`D'} & \text{if } q_i < 0 \text{ on } (t_{j-1}, t_j) \end{cases}$$

Then the structure of the matching can be represented by the $2 \times n$ matrix with top row (s^1) and bottom row (s^2). Each column of this matrix represents the type of the matching between q_1 and q_2 on the corresponding subinterval. For example, suppose q_1 and q_2 are defined as

$$q_1(t) = \begin{cases} 1 & \text{if } t \in (0, 1/2) \\ -1 & \text{if } t \in (1/2, 1) \end{cases} \qquad q_2(t) = \begin{cases} 1 & \text{if } t \in (0, 1/3) \\ -1 & \text{if } t \in (1/3, 2/3) \\ 1 & \text{if } t \in (2/3, 1) \end{cases}$$

After taking the union of the changepoints and adding in the necessary duplicate function values, we have the following representations of q_1 and q_2:

$$q_1(t) = \begin{cases} 1 & \text{if } t \in (0, 1/3) \\ 1 & \text{if } t \in (1/3, 1/2) \\ -1 & \text{if } t \in (1/2, 2/3) \\ -1 & \text{if } t \in (2/3, 1) \end{cases} \qquad q_2(t) = \begin{cases} 1 & \text{if } t \in (0, 1/3) \\ -1 & \text{if } t \in (1/3, 1/2) \\ -1 & \text{if } t \in (1/2, 2/3) \\ 1 & \text{if } t \in (2/3, 1) \end{cases}$$

The form of the matching is then

$$\begin{bmatrix} \text{UUDD} \\ \text{UDDU} \end{bmatrix}$$

We now identify several patterns that cannot occur in an optimal matching. We will need a result due to Klassen [19]:

Proposition 4. *Let $0 \leq a < b \leq 1$, and suppose q and w are two elements of $\mathcal{C}$ satisfying the following three properties:*

1. *For all $t \in (a,b)$, either $q(t) \geq 0$ and $w(t) \geq 0$, or $q(t) \leq 0$ and $w(t) \leq 0$.*
2. *$\int_a^b q(t)^2 dt = \int_a^b w(t)^2 dt$*
3. *For all $t \notin (a,b)$, $q(t) = w(t)$.*

Then $[q] = [w]$.

The first observation is that it is never optimal to match positive segments to negative segments:

Lemma 1. *No optimal matching contains a column of the form $\begin{bmatrix} U \\ D \end{bmatrix}$ or $\begin{bmatrix} D \\ U \end{bmatrix}$.*

Proof. Suppose the pattern $\begin{bmatrix} \text{U} \\ \text{D} \end{bmatrix}$ appears on the subinterval (t_{i-1}, t_i) for some i, so that on this subinterval, $q_1(t) > 0$ and $q_2(t) < 0$. Let y_1 and y_2 denote the values of q_1 and q_2 on this interval, respectively, and let $t_m = \frac{t_{i-1}+t_i}{2}$ be the midpoint of the interval. Define the functions $\hat{q}_1$ and $\hat{q}_2$ as

$$\hat{q}_1(t) = \begin{cases} q_1(t) & \text{if } \ t \in (0, t_{i-1}) \\ \sqrt{2}y_1 & \text{if } \ t \in (t_{i-1}, t_m) \\ 0 & \text{if } \ t \in (t_m, t_i) \\ q_1(t) & \text{if } \ t \in (t_i, 1] \end{cases} \qquad \hat{q}_2(t) = \begin{cases} q_2(t) & \text{if } \ t \in (0, t_{i-1}) \\ 0 & \text{if } \ t \in (t_{i-1}, t_m) \\ \sqrt{2}y_2 & \text{if } \ t \in (t_m, t_i) \\ q_2(t) & \text{if } \ t \in (t_i, 1] \end{cases}$$

Then it follows by Proposition 4 that $\hat{q}_1 \in [q_1]$, since

$$\begin{aligned} \int_{t_{i-1}}^{t_i} (\hat{q}_1(t))^2 dt &= \int_{t_{i-1}}^{t_m} (\sqrt{2}y_1)^2 dt = 2(t_m - t_{i-1})y_1^2 = (t_i - t_{i-1})y_1^2 \\ &= \int_{t_{i-1}}^{t_i} (y_1)^2 dt = \int_{t_{i-1}}^{t_i} (q_1(t))^2 dt \end{aligned}$$

Similarly, $\hat{q}_2 \in [q_2]$. Now,

$$\int_{t_{i-1}}^{t_i} q_1(t) q_2(t) dt = (t_i - t_{i-1}) y_1 y_2 < 0$$

whereas

$$\int_{t_{i-1}}^{t_i} \hat{q}_1(t)\hat{q}_2(t) dt = \int_{t_{i-1}}^{t_m} (\sqrt{2}y_1)^2 \cdot 0 \ dt + \int_{t_m}^{t_i} 0 \cdot (\sqrt{2}y_2)^2 \ dt = 0$$

We have traded a negative contribution to the inner product for a contribution of zero, thereby increasing the inner product and decreasing the distance. Therefore, the original matching between q_1 and q_2 was not optimal. Occurrences of $\begin{bmatrix}\mathrm{D}\\ \mathrm{U}\end{bmatrix}$ are handled in the same way. □

This last lemma shows that if q_1 and q_2 are ever matched so that one function is positive and the other is negative on some subinterval, then it is always possible to reparametrize the functions so that that interval has a contribution of zero to the overall inner product. This illustrates a somewhat curious result, which actually holds for any $[q_1], [q_2] \in \mathcal{S}$ (not just step functions):

Corollary 1. *If $[q_1], [q_2] \in \mathcal{S}$, then there exist $v \in [q_1]$ and $w \in [q_2]$ such that $\langle v, w\rangle \geq 0$. Hence, $d([q_1], [q_2]) \leq \frac{\pi}{2}$.*

Proof. Assume that q_1 and q_2 are in standard form (see Section 2.1.4), so that $q_1\Gamma = [q_1]$ and $q_2\Gamma = [q_2]$. Define $\gamma_1 \in \Gamma$ and $\gamma_2 \in \Gamma$ as

$$\gamma_1(t) = \begin{cases} 2t & \text{if } t \leq \frac{1}{2} \\ 1 & \text{otherwise} \end{cases} \qquad\qquad \gamma_2(t) = \begin{cases} 0 & \text{if } t \leq \frac{1}{2} \\ 2(t - \frac{1}{2}) & \text{otherwise} \end{cases}$$

Then $q_1 * \gamma_1$ is zero on $(\frac{1}{2}, 1)$ and $q_2 * \gamma_2$ is zero on $(0, \frac{1}{2})$, so $\langle q_1 * \gamma_1, q_2 * \gamma_2\rangle = 0$, and $d(q_1 * \gamma_1, q_2 * \gamma_2) = \frac{\pi}{2}$. Hence, $d([q_1], [q_2]) \leq \frac{\pi}{2}$. □

The next lemma justifies certain types of reductions that may be performed on a matching in order to obtain a simpler matching without decreasing the inner product.

Lemma 2. *Let q_1 and q_2 be step functions which are constant on the subintervals of the partition $0 = t_0 < t_1 < \ldots < t_n = 1$. Fix a number i between 1 and $n-1$, inclusive, and consider the two-column pattern formed by the matching on the intervals (t_{i-1}, t_i) and (t_i, t_{i+1})*

$$\begin{bmatrix} \cdots & ** & \cdots \\ & ** & \end{bmatrix}$$

where the asterisks are placeholders which may represent 'U', '0', or 'D'. Unless both a 'U' and a 'D' appear in the pattern, it is possible to find reparametrizations $\hat{q}_1 \in [q_1]$ and $\hat{q}_2 \in [q_2]$ such that

1. *$\hat{q}_1$ and $\hat{q}_2$ are constant on (t_{i-1}, t_{i+1}),*

2. *$\hat{q}_1$ and $\hat{q}_2$ agree with q_1 and q_2, respectively, outside of the interval (t_{i-1}, t_{i+1}), and*

3. *$\langle \hat{q}_1, \hat{q}_2\rangle \geq \langle q_1, q_2\rangle$*

Proof. Suppose that on the interval (t_{i-1}, t_i), $q_1(t) = y_{11}$ and $q_2(t) = y_{21}$, and that on the interval (t_i, t_{i+1}), $q_1(t) = y_{12}$ and $q_2(t) = y_{22}$. Let $w_1 = t_i - t_{i-1}$ and $w_2 = t_{i+1} - t_i$ denote the widths of the two subintervals, and let $w = t_{i+1} - t_{i-1}$ denote the width of their union. For each of the patterns, we will construct new functions $\hat{q}_1$ and $\hat{q}_2$ which agree with q_1 and q_2, respectively, outside of these two intervals, but are constant on (t_{i-1}, t_{i+1}). We will let $\hat{y}_1$ and $\hat{y}_2$ denote the values of $\hat{q}_1$ and $\hat{q}_2$, respectively, on this interval.

Suppose that only 'U' and '0' appear in the two adjacent columns. This leaves $2^4 = 16$ possible cases to consider. The specific values for $\hat{y}_1$ and $\hat{y}_2$ are given for each case in table 2.2. In each case, it can be routinely verified that $\hat{q}_1 \in [q_1]$ and $\hat{q}_2 \in [q_2]$ (using Proposition 4), and that $\langle \hat{q}_1, \hat{q}_2 \rangle \geq \langle q_1, q_2 \rangle$.

In the last case, $\int_{t_{i-1}}^{t_{i+1}} \hat{q}_1(t)\hat{q}_2(t)\ dt = w\hat{y}_1\hat{y}_2$ is as large as possible. To see this, note that for any choice of $\hat{q}_1 \in [q_1]$ and $\hat{q}_2 \in [q_2]$, we must have

$$\int_{t_{i-1}}^{t_{i+1}} (\hat{q}_1(t))^2\ dt = \int_{t_{i-1}}^{t_{i+1}} (q_1(t))^2\ dt = w_1(y_{11})^2 + w_2(y_{12})^2$$

and

$$\int_{t_{i-1}}^{t_{i+1}} (\hat{q}_2(t))^2\ dt = \int_{t_{i-1}}^{t_{i+1}} (q_2(t))^2\ dt = w_1(y_{21})^2 + w_2(y_{22})^2$$

Therefore, it follows from the Cauchy-Schwarz inequality that

$$\begin{aligned}\int_{t_{i-1}}^{t_{i+1}} \hat{q}_1(t)\hat{q}_2(t)\ dt &\leq \left(\int_{t_{i-1}}^{t_{i+1}} (\hat{q}_1(t))^2\ dt\right)^{1/2} \left(\int_{t_{i-1}}^{t_{i+1}} (\hat{q}_2(t))^2\ dt\right)^{1/2} \\ &= \sqrt{w_1(y_{11})^2 + w_2(y_{12})^2}\sqrt{w_1(y_{21})^2 + w_2(y_{22})^2}\end{aligned}$$

and this last value is the inner product attained by our choice of $\hat{y}_1$ and $\hat{y}_2$ in the table above.

The case where the list contains only 'D' and '0' follows similarly, and is omitted. □

Next, we prove a necessary condition which will be used in several results that follow:

Lemma 3. *Let q_1 and q_2 be step functions in $\mathcal{C}$ such that either $q_2 \geq 0$ on (a, b), or $q_2 \leq 0$ on (a, b). Write (a, b) as a union of disjoint sets A and B, where $q_1q_2 > 0$ (i.e. both functions are strictly positive, or both are strictly negative) on A, and $q_1q_2 \leq 0$ on B, and suppose that A has positive measure. Then there exists $\hat{q}_2 \in [q_2]$ such that*

1. *$\hat{q}_2(t) = q_2(t)$ for all $t \notin (a, b)$,*

2. *the ratio $r = \hat{q}_2(t)/q_1(t)$ is constant on A,*

3. *$\langle q_1, \hat{q}_2 \rangle \geq \langle q_1, q_2 \rangle$, and*

4. *$\hat{q}_2(t) = 0$ for $t \in B$*

Further, if $\hat{q}_2$ and q_2 are not equal (a.e.), then the inequality in item (3) will be strict.

Proof. Suppose that $q_2 \geq 0$ on (a, b); the case where $q_2 \leq 0$ on (a, b) is similar. Then we have the following upper bound for the contribution of (a, b) to the overall inner product of

Table 2.1: The possible configurations for Lemma 2.

Old Pattern	$\hat{y}_1$	$\hat{y}_2$	**New Pattern**	**Effect on Inner Product**
$\begin{bmatrix} 00 \\ 00 \end{bmatrix}$	0	0	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	Still 0
$\begin{bmatrix} 00 \\ 0U \end{bmatrix}$	0	$\sqrt{\frac{w_2}{w}} y_{22}$	$\begin{bmatrix} 0 \\ U \end{bmatrix}$	Still 0
$\begin{bmatrix} 00 \\ U0 \end{bmatrix}$	0	$\sqrt{\frac{w_1}{w}} y_{21}$	$\begin{bmatrix} 0 \\ U \end{bmatrix}$	Still 0
$\begin{bmatrix} 00 \\ UU \end{bmatrix}$	0	$\sqrt{\frac{w_1(y_{21})^2+w_2(y_{22})^2}{w}}$	$\begin{bmatrix} 0 \\ U \end{bmatrix}$	Still 0
$\begin{bmatrix} 0U \\ 00 \end{bmatrix}$	$\sqrt{\frac{w_2}{w}} y_{12}$	0	$\begin{bmatrix} U \\ 0 \end{bmatrix}$	Still 0
$\begin{bmatrix} 0U \\ 0U \end{bmatrix}$	$\sqrt{\frac{w_2}{w}} y_{12}$	$\sqrt{\frac{w_2}{w}} y_{22}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Same
$\begin{bmatrix} 0U \\ U0 \end{bmatrix}$	$\sqrt{\frac{w_2}{w}} y_{12}$	$\sqrt{\frac{w_1}{w}} y_{21}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Increased
$\begin{bmatrix} 0U \\ UU \end{bmatrix}$	$\sqrt{\frac{w_2}{w}} y_{12}$	$\sqrt{\frac{w_1(y_{21})^2+w_2(y_{22})^2}{w}}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Increased
$\begin{bmatrix} U0 \\ 00 \end{bmatrix}$	$\sqrt{\frac{w_1}{w}} y_{11}$	0	$\begin{bmatrix} U \\ 0 \end{bmatrix}$	Still 0
$\begin{bmatrix} U0 \\ 0U \end{bmatrix}$	$\sqrt{\frac{w_1}{w}} y_{11}$	$\sqrt{\frac{w_2}{w}} y_{22}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Increased
$\begin{bmatrix} U0 \\ U0 \end{bmatrix}$	$\sqrt{\frac{w_1}{w}} y_{11}$	$\sqrt{\frac{w_1}{w}} y_{21}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Same
$\begin{bmatrix} U0 \\ UU \end{bmatrix}$	$\sqrt{\frac{w_1}{w}} y_{11}$	$\sqrt{\frac{w_1(y_{21})^2+w_2(y_{22})^2}{w}}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Increased
$\begin{bmatrix} UU \\ 00 \end{bmatrix}$	$\sqrt{\frac{w_1(y_{11})^2+w_2(y_{12})^2}{w}}$	0	$\begin{bmatrix} U \\ 0 \end{bmatrix}$	Still 0
$\begin{bmatrix} UU \\ 0U \end{bmatrix}$	$\sqrt{\frac{w_1(y_{11})^2+w_2(y_{12})^2}{w}}$	$\sqrt{\frac{w_2}{w}} y_{22}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Increased
$\begin{bmatrix} UU \\ U0 \end{bmatrix}$	$\sqrt{\frac{w_1(y_{11})^2+w_2(y_{12})^2}{w}}$	$\sqrt{\frac{w_1}{w}} y_{21}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Increased
$\begin{bmatrix} UU \\ UU \end{bmatrix}$	$\sqrt{\frac{w_1(y_{11})^2+w_2(y_{12})^2}{w}}$	$\sqrt{\frac{w_1(y_{21})^2+w_2(y_{22})^2}{w}}$	$\begin{bmatrix} U \\ U \end{bmatrix}$	Maximal (see below)

q_1 and q_2:

$$\begin{aligned}
\int_a^b q_1(t)q_2(t)\ dt &= \int_A q_1(t)q_2(t)\ dt + \int_B q_1(t)q_2(t)\ dt \\
&\leq \int_A q_1(t)q_2(t)\ dt && \text{since } \int_B q_1(t)q_2(t)\ dt \leq 0 \\
&\leq \left(\int_A q_1(t)^2\ dt\right)^{1/2} \left(\int_A q_2(t)^2\ dt\right)^{1/2} && \text{by Cauchy-Schwarz} \\
&\leq \left(\int_A q_1(t)^2\ dt\right)^{1/2} \left(\int_a^b q_2(t)^2\ dt\right)^{1/2}
\end{aligned}$$

Now we define our ratio r as

$$r = \left(\frac{\int_a^b q_2^2(t)\ dt}{\int_A q_1^2(t)\ dt}\right)^{1/2}$$

(note the denominator is non-zero since A has positive measure and $q_1 > 0$ on A). We then define the function $\hat{q}_2 \in [q_2]$ as

$$\hat{q}_2(t) = \begin{cases} q_2(t) & \text{if } t \notin (a,b) \\ rq_1(t) & \text{if } t \in A \\ 0 & \text{if } t \in B \end{cases}$$

Then $\hat{q}_2 \geq 0$ on (a,b), and it is easily checked that $\hat{q}_2 \in [q_2]$ using Proposition 4:

$$\int_a^b \hat{q}_2^2(t)\ dt = \int_A r^2 q_1^2(t)\ dt = \frac{\int_a^b q_2^2(t)\ dt}{\int_A q_1^2(t)\ dt} \int_A q_1^2(t)\ dt = \int_a^b q_2^2(t)\ dt$$

Finally, we have

$$\begin{aligned}
\int_a^b q_1(t)\hat{q}_2(t)\ dt &= \int_A rq_1^2(t)\ dt \\
&= \left(\frac{\int_a^b q_2^2(t)\ dt}{\int_A q_1^2(t)\ dt}\right)^{1/2} \int_A q_1^2(t)\ dt \\
&= \left(\int_a^b q_2^2(t)\ dt\right)^{1/2} \left(\int_A q_1^2(t)\ dt\right)^{1/2}
\end{aligned}$$

which is the upper bound obtained above.

It remains to show that, unless q_2 and $\hat{q}_2$ are equal, the inequality in item (3) will be strict. First, suppose that q_2 is nonzero on part of B. Since

$$\int_a^b q_2^2(t)\ dt = \int\limits_A q_2^2(t)\ dt + \int\limits_B q_2^2(t)\ dt$$

it follows that

$$\int\limits_A q_2^2(t)\ dt < \int_a^b\ q_2^2(t)\ dt$$

Therefore, we have

$$\begin{aligned}\int_a^b\ q_1(t)q_2(t)\ dt &= \int_A\ q_1(t)q_2(t)\ dt + \int_B\ q_1(t)q_2(t)\ dt \\ &\leq \int_A\ q_1(t)q_2(t)\ dt && \text{since } \int_B\ q_1(t)q_2(t)\ dt \leq 0 \\ &\leq \left(\int_A\ q_1(t)^2\ dt\right)^{1/2}\left(\int_A\ q_2(t)^2\ dt\right)^{1/2} && \text{by Cauchy-Schwarz} \\ &< \left(\int_A\ q_1(t)^2\ dt\right)^{1/2}\left(\int_a^b\ q_2(t)^2\ dt\right)^{1/2}\end{aligned}$$

(note the strict inequality on the last line), which shows that the matching between q_1 and $\hat{q}_2$ is strictly better than that between q_1 and q_2.

Now consider the set of all orbit representatives $v \in [q_2]$ which agree with q_2 outside of (a, b) and are zero on B. We claim that out of all these functions, $\hat{q}_2$ gives the highest inner product with q_1. To see this, restrict all of these functions v to the set A, so that we have a collection of functions in $\mathbb{L}^2(A, \mathbb{R})$. All of these functions must have the same norm, so they all lie on the same sphere in $\mathbb{L}^2$. Since $\hat{q}_2$ is the projection of q_1 onto this sphere, it follows that $\hat{q}_2$ has the highest inner product with q_1 out of all points on the sphere. □

Lemma 4. *If the pattern* $\begin{bmatrix} ** \\ 00 \end{bmatrix}$ *or* $\begin{bmatrix} 00 \\ ** \end{bmatrix}$ *appears in a matching, then we can find either a matching yielding a higher inner product, or a matching with fewer columns which yields the same inner product.*

Proof. We prove the statement for $\begin{bmatrix} ** \\ 00 \end{bmatrix}$; the argument for $\begin{bmatrix} 00 \\ ** \end{bmatrix}$ is completely analogous. First, the top row must contain both a 'U' and a 'D' (else the pattern can be reduced using Lemma 2), so the pattern is either $\begin{bmatrix} UD \\ 00 \end{bmatrix}$ or $\begin{bmatrix} DU \\ 00 \end{bmatrix}$.

Assume the pattern is $\begin{bmatrix} UD \\ 00 \end{bmatrix}$ – the argument for the other pattern, $\begin{bmatrix} DU \\ 00 \end{bmatrix}$, follows similarly. Since $\|q_2\| = 1$, there must be nonzero entries somewhere in the bottom row. If there is a nonzero entry in the second row that occurs to the left of the pattern, then we have four possible forms. Each one of these can be improved via a reparametrization of q_2. The four cases, along with the new forms which yield a higher inner product than the original matching, are summarized in table 2.2. The reductions for the cases where the bottom row contains a non-zero entry to the right of the pattern are similar, and are omitted.

□

Table 2.2: The possible configurations for Lemma 4.

Pattern	**New Pattern**	**Notes**
$\begin{bmatrix}0 & * & \cdots & * & U & D\\ U & 0 & \cdots & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & * & \cdots & * & U & D\\ 0 & 0 & \cdots & 0 & U & 0\end{bmatrix}$	
$\begin{bmatrix}U & * & \cdots & * & U & D\\ U & 0 & \cdots & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}U & * & \cdots & * & U & D\\ U & 0 & \cdots & 0 & U & 0\end{bmatrix}$	Apply Lemma 3
$\begin{bmatrix}0 & * & \cdots & * & U & D\\ D & 0 & \cdots & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & * & \cdots & * & U & D\\ 0 & 0 & \cdots & 0 & 0 & D\end{bmatrix}$	
$\begin{bmatrix}D & * & \cdots & * & U & D\\ D & 0 & \cdots & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}D & * & \cdots & * & U & D\\ D & 0 & \cdots & 0 & 0 & D\end{bmatrix}$	Apply Lemma 3

Lemma 5. *If either of the patterns* $\begin{bmatrix}D\\ 0\end{bmatrix}$ *or* $\begin{bmatrix}0\\ D\end{bmatrix}$ *appears as part of an optimal matching, and the pattern does not appear as the first or last column, then the pattern appears as part of the three-column pattern* $\begin{bmatrix}UDU\\ U0U\end{bmatrix}$ *or* $\begin{bmatrix}U0U\\ UDU\end{bmatrix}$*. Similarly, if either of the patterns* $\begin{bmatrix}U\\ 0\end{bmatrix}$ *or* $\begin{bmatrix}0\\ U\end{bmatrix}$ *appears as part of an optimal matching, and the pattern does not appear as the first or last column, then the pattern appears as part of the three-column pattern* $\begin{bmatrix}DUD\\ D0D\end{bmatrix}$ *or* $\begin{bmatrix}D0D\\ DUD\end{bmatrix}$*.*

Proof. Suppose that $\begin{bmatrix}\text{D}\\ 0\end{bmatrix}$ appears as part of the pattern $\begin{bmatrix}s_{11}\ \text{D}\ s_{12}\\ s_{21}\ 0\ s_{22}\end{bmatrix}$ where $s_{11}, s_{12}, s_{21}, s_{22} \in$ {‘U’, ‘0’, ‘D’}. We first show that s_{21} and s_{22} must both be ‘U’. If s_{21} is ‘D’, then s_{11} cannot be ‘U’, so the first two columns form one of the two patterns $\begin{bmatrix}\text{DD}\\ \text{D0}\end{bmatrix}$ or $\begin{bmatrix}\text{0D}\\ \text{D0}\end{bmatrix}$, both of which can be reduced using Lemma 2. If s_{21} is ‘0’, then we have the pattern $\begin{bmatrix}\text{*D}\\ \text{00}\end{bmatrix}$, which is reducible by Lemma 4. Therefore, s_{21} must be ‘U’, and by a similar argument, s_{22} must be ‘U’.

Now consider the entries in the top row, which must be either ‘0’ or ‘U’ by Lemma 1. If both entries are ‘0’, then we make the reduction

$$\begin{bmatrix}\text{0D0}\\ \text{U0U}\end{bmatrix} \longrightarrow \begin{bmatrix}\text{0D}\\ \text{U0}\end{bmatrix}$$

at no cost to the inner product. If one entry is ‘U’ and the other is ‘0’, then Lemma 3 applies. Therefore, both entries must be ‘U’, as desired. The argument for $\begin{bmatrix}0\\ \text{D}\end{bmatrix}$ is similar and is omitted.

To prove the statement for the patterns $\begin{bmatrix}\text{U}\\ 0\end{bmatrix}$ and $\begin{bmatrix}0\\ \text{U}\end{bmatrix}$, repeat the argument above with the roles of ‘U’ and ‘D’ reversed. □

Lemma 5 is particularly useful because we can use it to establish the basic pieces out of which optimal matchings are built.

Definition 5. *An* ***up-segment*** *in a matching is a pattern consisting of* $\begin{bmatrix} U \\ U \end{bmatrix}$*, followed by zero or more instances of* $\begin{bmatrix} 0U \\ DU \end{bmatrix}$ *or* $\begin{bmatrix} DU \\ 0U \end{bmatrix}$*. A* ***down-segment*** *in a matching is a pattern consisting of* $\begin{bmatrix} D \\ D \end{bmatrix}$*, followed by zero or more instances of* $\begin{bmatrix} 0D \\ UD \end{bmatrix}$ *or* $\begin{bmatrix} UD \\ 0D \end{bmatrix}$*.*

Theorem 2. *Any optimal matching consists of the following components, in the following order:*

1. *Optionally, one column of the form* $\begin{bmatrix} U \\ 0 \end{bmatrix}$*,* $\begin{bmatrix} 0 \\ U \end{bmatrix}$*,* $\begin{bmatrix} D \\ 0 \end{bmatrix}$*, or* $\begin{bmatrix} 0 \\ D \end{bmatrix}$*, then*
2. *An alternating sequence of zero or more up-segments and down-segments, and finally*
3. *Optionally one column of the form* $\begin{bmatrix} U \\ 0 \end{bmatrix}$*,* $\begin{bmatrix} 0 \\ U \end{bmatrix}$*,* $\begin{bmatrix} D \\ 0 \end{bmatrix}$*, or* $\begin{bmatrix} 0 \\ D \end{bmatrix}$

Proof. The statement is trivial for matchings with two columns or fewer, so assume that the matching has at least three columns. If the first column is not $\begin{bmatrix} U \\ U \end{bmatrix}$ or $\begin{bmatrix} D \\ D \end{bmatrix}$, then the second column must be, since interior half-zero forms must appear surrounded by pairs of $\begin{bmatrix} U \\ U \end{bmatrix}$ columns or pairs of $\begin{bmatrix} D \\ D \end{bmatrix}$ columns (by Lemma 5). Now we consider what possible forms could follow, say, a $\begin{bmatrix} U \\ U \end{bmatrix}$ column. These are:

1. Nothing (i.e., the column appears at the end of the matching),
2. A $\begin{bmatrix} D \\ 0 \end{bmatrix}$ or $\begin{bmatrix} 0 \\ D \end{bmatrix}$ column which is the last column in the matching,
3. A $\begin{bmatrix} D \\ 0 \end{bmatrix}$ or $\begin{bmatrix} 0 \\ D \end{bmatrix}$ column, followed by another $\begin{bmatrix} U \\ U \end{bmatrix}$ column, or
4. A $\begin{bmatrix} D \\ D \end{bmatrix}$ column.

In light of Lemmas 2 and 5, these are the only possibilities. In cases 1 and 2, the up-segment ends, as does the matching. In case 3, the up-segment continues, and in case 4, the up-segment ends and a down-segment begins.

A similar consideration shows that a $\begin{bmatrix} D \\ D \end{bmatrix}$ column begins a down-segment which continues until either the end of the matching, or the beginning of an up-segment, is encountered. □

Lemma 6. *Let q_1 and q_2 be step functions representing an optimal matching between their respective orbits, and assume that q_1 and q_2 have a common changepoint partition $0 = t_1 < t_2 < \ldots < t_n = 1$. Suppose the matching from t_a to t_b forms an up-segment, so that $b - a$ is odd, and the matching form is*

- $\begin{bmatrix} U \\ U \end{bmatrix}$ *on (t_a, t_{a+1}), (t_{a+2}, t_{a+3}), $\ldots$, and (t_{b-1}, t_b), and*
- $\begin{bmatrix} D \\ 0 \end{bmatrix}$ *or* $\begin{bmatrix} 0 \\ D \end{bmatrix}$ *on (t_{a+1}, t_{a+2}), (t_{a+3}, t_{a+4}), $\ldots$, and (t_{b-2}, t_{b-1}).*

Then the ratio $q_2(t)/q_1(t)$ is the same for all intervals on which both q_1 and q_2 are positive. Similarly, in a down-segment, the ratio $q_2(t)/q_1(t)$ must be the same for all intervals on which both q_1 and q_2 are negative.

Proof. If the up-segment or down-segment consists of a single column, there is nothing to show. Otherwise, apply Lemma 3 to the intervals (t_a, t_{a+3}), (t_{a+2}, t_{a+5}), $\ldots$, (t_{b-5}, t_{b-2}), and (t_{b-3}, t_b), and note that if there is more than one such interval, then the ratios must agree on the overlaps. □

In fact, we can do better: looking at the proof of Lemma 3, we see that the ratio $q_2(t)/q_1(t)$ on the intervals where the signs agree is simply the square root of the ratio of the total rise (or fall) of the functions. That is, if we write $A = (t_a, t_{a+1}) \cup (t_{a+2}, t_{a+3}) \cup \ldots \cup (t_{b-1}, t_b)$, then our ratio can be written as

$$\frac{q_2(t)}{q_1(t)} \equiv \left(\frac{\int_A \ (q_2(t))^2 \ dt}{\int_A \ (q_1(t))^2 \ dt} \right)^{1/2} \quad \text{for } t \in A \tag{2.1}$$

2.3 The Matching Algorithm

In the previous section, we started by assuming that we were handed an optimal matching between two functions, and then deduced several properties that any such matching must have. Ultimately, we showed that any optimal matching must consist of an alternating sequence of up-segments and down-segments, with an optional half-zero column at the beginning and end. We also showed that, on any up-segment (resp. down-segment), the ratio $q_1(t)/q_2(t)$ must be constant on the intervals where both functions are positive (resp. negative). Now, any up-segment must begin with a valley-to-valley match and end with a peak-to-peak match. That is, if the up-segment begins at t_a and ends at t_b, then both f_1 and f_2 have a valley at t_a and a peak at t_b. Similarly, any down-segment must begin with a peak-to-peak matching and must end with a valley-to-valley matching. So by Theorem 2, any optimal matching corresponds to an alternating sequence of peak-to-peak matchings and valley-to-valley matchings.

In this section, we will shift our focus to the problem of actually finding an optimal matching. We will start with two functions $f_1, f_2 \in \mathcal{A}$ which are arc length parametrized and have finitely-many turning points. The corresponding SRVFs $q_1, q_2 \in \mathcal{C}$ are step functions which are unit-speed (i.e. taking only values 1 and -1). In general, these SRVFs will not constitute an optimal matching. We must find reparametrizations $\gamma_1, \gamma_2 \in \Gamma$ which maximize the inner product

$$\langle q_1 * \gamma_1, q_2 * \gamma_2 \rangle = \int_0^1 \sqrt{\dot{\gamma_1}(t)} q_1(\gamma_1(t)) \sqrt{\dot{\gamma_2}(t)} q_2(\gamma_2(t)) \; dt$$

Let $t_0^1 = 0 < t_1^1 < \ldots < t_{n_1}^1 = 1$ and $t_0^2 = 0 < t_1^2 < \ldots < t_{n_2}^2 = 1$ be the changepoint partitions for q_1 and q_2, respectively. Then for $i = 1, 2$, the values t_j^i correspond to the peaks and valleys of f_i. Suppose that a, b, c, and d are indices such that $a < b$, $c < d$, t_a^1 and t_c^2 correspond to valleys on their respective functions, and t_b^1 and t_d^2 correspond to peaks. Then matching t_a^1 to t_c^2 and t_b^1 to t_d^2 defines an up-segment. Similarly, if t_a^1 and t_c^2 correspond to peaks and t_b^1 and t_d^2 correspond to valleys, then matching t_a^1 to t_c^2 and t_b^1 to t_d^2 defines a down-segment.

We will see in a moment that, for any up-segment or down-segment, the corresponding contribution to the inner product between the reparametrized SRVFs can be easily determined, as can the corresponding portions of γ_1 and γ_2 that realize the matching for the up- or down-segment. Once we have settled on an alternating sequence of up-segments and down-segments, the rest of the matching is determined (up to a common reparametrization of both functions, which has no effect on the registration or the inner product). Therefore, our matching problem boils down to an optimization problem where the search space is the set of all possible alternating sequences of peak-to-peak and valley-to-valley matchings. We will develop an algorithm to solve this problem using the technique of dynamic programming. Before presenting the algorithm, we need to do one last bit of preliminary work.

2.3.1 The Contribution of an Up- or Down-Segment to the Inner Product

As above, let $t_0^1 = 0 < t_1^1 < \ldots < t_{n_1}^1 = 1$ and $t_0^2 = 0 < t_1^2 < \ldots < t_{n_2}^2 = 1$ be the respective changepoint partitions for our unit-speed SRVFs q_1 and q_2. Suppose that $\tilde{q_1} = q_1 * \gamma_1$ and $\tilde{q_2} = q_2 * \gamma_2$ are reparametrized SRVFs constituting an optimal matching, and suppose that $\tilde{q_1}$ and $\tilde{q_2}$ are step functions sharing the common changepoint partition $t_0 = 0 < t_1 < \ldots < t_n = 1$. Suppose that $\tilde{q_1}$ and $\tilde{q_2}$ form an up-segment on the interval (t_a, t_{a+l}) for some odd integer $l \geq 1$, and let $\tilde{A}$ be the union of the intervals on which both functions are positive:

$$\tilde{A} = (t_a, t_{a+1}) \cup (t_{a+2}, t_{a+3}) \cup \ldots \cup (t_{a+l-1}, t_{a+l})$$

Let $A_1 = \gamma_1(\tilde{A})$ and let $A_2 = \gamma_2(\tilde{A})$, and note that these are the sets in the up-segment on which q_1 and q_2 are positive. Further, since q_1 and q_2 are unit-speed SRVFs, it is easy to see that the measures $|A_1|$ and $|A_2|$ of these sets are equal to the total rise of f_1 and f_2 (respectively) for the up-segment. Now, using the Cauchy-Schwarz inequality and a simple change of variable, we get the following upper bound for the contribution of the up-segment to the overall inner product between the matched functions:

$$\begin{aligned}
\int_{t_a}^{t_{a+l}} \tilde{q_1}(t)\tilde{q_2}(t)dt &= \int_{\tilde{A}} \tilde{q_1}(t)\tilde{q_2}(t)dt \\
&\leq \left(\int_{\tilde{A}} (\tilde{q_1}(t))^2\ dt\right)^{1/2} \left(\int_{\tilde{A}} (\tilde{q_2}(t))^2\ dt\right)^{1/2} \\
&= \left(\int_{\tilde{A}} (\sqrt{\dot{\gamma_1}(t)}q_1(\gamma_1(t)))^2\ dt\right)^{1/2} \left(\int_{\tilde{A}} (\sqrt{\dot{\gamma_2}(t)}q_2(\gamma_2(t)))^2\ dt\right)^{1/2} \\
&= \left(\int_{A_1} (q_1(\gamma_1))^2\ d\gamma_1\right)^{1/2} \left(\int_{A_2} (q_2(\gamma_2))^2\ d\gamma_2\right)^{1/2} \\
&= \left(\int_{A_1} d\gamma_1\right)^{1/2} \left(\int_{A_2} d\gamma_2\right)^{1/2} = \sqrt{|A_1|}\sqrt{|A_2|}
\end{aligned}$$

In the next section, we will give an algorithm for constructing functions γ_1 and γ_2 which achieve this inner product. It can similarly be shown that for a down-segment, the maximum contribution to the overall inner product is equal to $\sqrt{|A_1|}\sqrt{|A_2|}$, where A_1 and A_2 are the sets on which q_1 and q_2 are negative, and $|A_1|$ and $|A_2|$ are the total fall in f_1 and f_2 in the down-segment. Therefore, the optimal inner product contribution for an up- or down-segment can be found very efficiently.

2.3.2 The Matching Graph

Our optimization problem can be naturally formulated as a maximum-weight path problem in a certain weighted, directed, bipartite graph, which we introduce now. As above, assume that f_1 and f_2 are arc-length parametrized, so that their SRVFs q_1 and q_2 are step functions with values in $\{-1, 1\}$. Let $t_0^1 = 0 < t_1^1 < \ldots < t_{n_1}^1 = 1$ and $t_0^2 = 0 < t_1^2 < \ldots < t_{n_2}^2 = 1$ be the changepoint partitions for q_1 and q_2, respectively. Then each of these parameter values corresponds to either a peak or a valley in the respective function. A legal assignment between peaks and valleys can be represented by a sequence $(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m) \subseteq \{0, 1, \ldots, n_1\} \times \{0, 1, \ldots, n_2\}$ satisfying the following:

1. For $k = 2, 3, \ldots, m$, $i_k > i_{k-1}$ and $j_k > j_{k-1}$.
2. For $k = 1, 2, \ldots, m$, $t_{i_k}^1$ and $t_{j_k}^2$ both correspond to peaks, or they both correspond to valleys.
3. If $t_{i_k}^1$ and $t_{j_k}^2$ correspond to peaks, then the next pair will correspond to valleys, and vice versa.
4. For i_1 and j_1, one of the following is true:
    - $i_1 = 0$ and $j_1 = 0$
    - $i_1 = 1$ and $j_1 = 0$
    - $i_1 = 0$ and $j_1 = 1$
5. For i_m and j_m, one of the following is true:

- $i_m = n_1$ and $j_m = n_2$
- $i_m = n_1 - 1$ and $j_m = n_2$
- $i_m = n_1$ and $j_m = n_2 - 1$

Conditions (4) and (5) reflect the fact that in an optimal matching, there is at most one leading half-zero column and at most one trailing half-zero column. The set of all possible sequences is naturally represented as a directed bipartite graph G with a vertex for each peak-to-peak match and a vertex for each valley-to-valley match. G will have a vertex for every pair of indices (i, j) such that t^1_i and t^2_j are either both peaks, or both valleys. G has a directed edge from (i, j) to (k, l) iff $i < k$, $j < l$, and one of the vertices represents a peak-peak matching while the other represents a valley-valley matching. Therefore, the edges of G correspond exactly to the possible up-segments and down-segments that could appear in a matching satisfying the necessary conditions for optimality covered in the previous section. We will also assign a weight to each edge, which is equal to the contribution of the corresponding up- or down-segment to the total inner product of the matching, which was derived above. Figure 2.3.2 shows an example of a legal path through the matching graph, corresponding to an alternating sequence of up-segments and down-segments. The total inner product of the matching represented by the path is equal to the sum of the weights of the edges in the path.

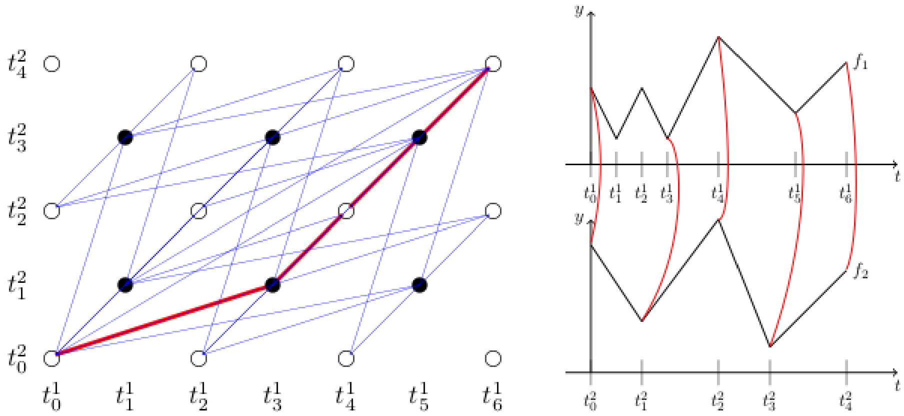

Figure 2.1: A path through the matching graph for 1-D functions. The graph is shown on the left, with the path in red. The path represents a correspondence between the peaks and valleys of two functions f_1 and f_2 (shown on right). Solid dots in the graph represent valley-to-valley matches, and hollow circles represent peak-to-peak matches. Edges in the graph correspond to up-segments and down-segments. The weight of each edge is equal to the contribution of the corresponding up- or down-segment to the inner product. f_1 and f_2 are shown to the right. The red arcs indicate the peak-to-peak and valley-to-valley matchings between f_1 and f_2.

In the example above, both functions begin with peaks, and both functions end with peaks. All legal paths through the graph must start at the bottom-left vertex and end at the top-right vertex. However, if one function begins with a peak and the other begins with a valley, then the matching must begin with one of the half-zero forms:

$$\begin{bmatrix} U \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ U \end{bmatrix}, \begin{bmatrix} D \\ 0 \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} 0 \\ D \end{bmatrix}$$

In this case, paths can either start at the vertex $(0,1)$ (corresponding to a matching beginning with $\begin{bmatrix} 0 \\ U \end{bmatrix}$ or $\begin{bmatrix} 0 \\ D \end{bmatrix}$), or at the vertex $(1,0)$ (corresponding to a matching beginning with $\begin{bmatrix} U \\ 0 \end{bmatrix}$ or $\begin{bmatrix} D \\ 0 \end{bmatrix}$). We will see shortly that having two possible starting points is inconvenient, because our algorithm is basically a single-source shortest-path algorithm. We can avoid this problem by introducing a new vertex v_s to the graph and adding zero-weight edges from v_s to $(0,1)$ and from v_s to $(1,0)$. Our algorithm will then produce a path which starts at v_s and passes through exactly one of the real start vertices $(0,1)$ and $(1,0)$, so we can recover the real path simply by discarding v_s from the front of the path.

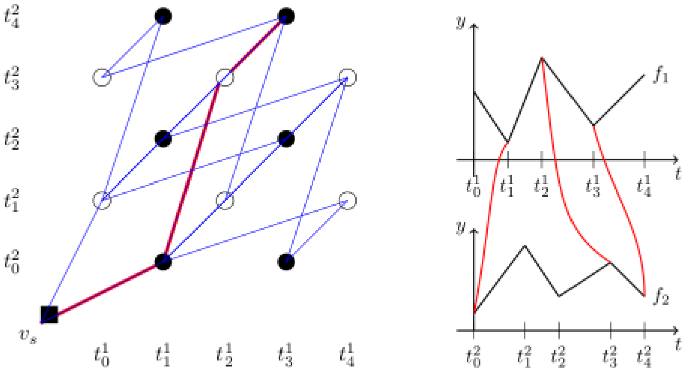

Figure 2.2: A path through the matching graph for two 1-D functions, in the case where one function begins with a peak and the other begins with a valley. The matching graph is shown on the left, and the functions f_1 and f_2 are shown on the right. Parameters for f_1 correspond to columns in the graph, and parameters for f_2 correspond to rows. f_1 begins and ends with a peak (hollow circle), while f_2 begins and ends with a valley (solid dot), so a dummy starting vertex v_s has been added. A path representing a matching between f_1 and f_2 has been shown in red. The resulting correspondence between the peaks and valleys of f_1 and f_2 is indicated by the red arcs on the right.

2.3.3 Finding an Optimal Path Through the Matching Graph

The algorithm for finding an optimal path through the matching graph (representing a sequence of peak-to-peak and valley-to-valley matchings yielding the highest possible inner product) is a straightforward application of dynamic programming. For each vertex v in our graph G, let $D[v]$ denote the weight of the path from the start vertex to v which has the highest total weight out of all such paths, and let $P[v]$ denote the predecessor of v along that path. The key idea that enables us to apply dynamic programming is that, since the total weight of a path is equal to the sum of the weights of its edges, the maximum-weight path problem has optimal substructure, and D can be given recursively by the following **dynamic programming functional equation** (DPFE):

$$D[v] = \max_{p \in \text{Preds}(v)} D[p] + W[p, v] \tag{2.2}$$

where Preds(v) is the set of all vertices p from which there is a directed edge e_{pv} from p to v in the graph, and $W[p, v]$ is the weight of the edge from p to v. Recall that there will be an edge from $p = (i, j)$ to $v = (k, l)$ whenever $i < k$, $j < l$, and $k - i$ is odd (so that one vertex represents a peak-to-peak matching and the other represents a valley-to-valley matching). Therefore, when searching over the predecessors of v as in equation 2.2, we only need to consider vertices appearing strictly below and to the left of v in the grid.

This predictable structure makes it possible for us to traverse the graph in a very simple way. We start by setting $D[(0, 0)] = 0$ (or $D[v_s] = 0$ if a dummy starting vertex was added), and then we compute the remaining values in a bottom-up flood-fill pattern. We work through the grid one row at a time, filling in the bottom row of the grid from left to right, then moving to the next row up, and so on. When we visit a vertex v, the values of $D[p]$ have already been computed for every possible predecessor p of v. At each iteration, the predecessor vertex p of v which maximizes the expression in equation 2.2 is stored as $P[v]$.

We should note that there are other orderings that will work just a well as the bottom-to-top left-to-right ordering that we use. For example, one could also fill in the grid one column at a time, starting with the leftmost column and filling it in from bottom to top, then moving on to the next column to the right, and so on. Another possibility would be to use a diagonal wavefront pattern. Whether one pattern has any advantage over the others depends on the implementation.

After D and P have been computed, we recover the path by starting at the top-right corner of the grid and tracing the path back to the starting vertex using P. If there are two possible ending vertices, as will be the case when one function ends with a peak and the other ends with a valley as in Figure 2.3.2, then we select the one with the higher score and trace back starting from there.

See Algorithm 2.1 for the details on computing D and P using dynamic programming, and see Algorithm 2.2 for details on recovering the optimal path given D and P. In Algorithm 2.1, c_s and c_t index changepoint parameters on q_1, while r_s and r_t index changepoint parameters on q_2 (c stands for "column", r stands for "row", s stands for "source" and t stands for "target"). We ensure that on each iteration, (c_s, r_s) and (c_t, r_t) are vertices in G, and that one of them represents a peak-to-peak matching and the other represents a valley-to-valley matching. Note that the latter is true if and only if c_s and c_t have opposite parity (i.e. one is even and the other is odd) and r_s and r_t have opposite parity.

Algorithm 2.1 Calculate D and P using dynamic programming

```
inputs
  The matching graph $G$
outputs
  The maximum path weight $D[v]$ and predecessor $P[v]$ for all vertices $v$
/** Initialize **/
if the start vertex is $(0, 0)$ then
  Set $D[(0, 0)] = 0$
else
  Set $D[(1, 0)] = 0$ and $D[(0, 1)] = 0$
  Set $P[(1, 0)] = v_s$ and $P[(0, 1)] = v_s$
end if
/** Compute $D$ and $P$ from the bottom up, working left-to-right across the rows **/
for $r_t = 1$ to $n_2$ do
  /** Select starting column for target vertex **/
  if $(1, r_t)$ is a vertex in $G$ then
    Set $\alpha_c = 1$
  else
    Set $\alpha_c = 2$
  end if
  for $c_t = \alpha_c$ to $n_1$ step 2 do
    /** Search over all possible predecessors, as in equation 2.2 **/
    Set $D[(c_t, r_t)] = 0$
    for $r_s = (1 - (r_t \mod 2))$ to $(r_t - 1)$ step 2 do
      for $c_s = (1 - (c_t \mod 2))$ to $(c_t - 1)$ step 2 do
        Set $D_{\text{cur}} = D[(c_s, r_s)] + W[(c_s, r_s), (c_t, r_t)]$
        if $D_{\text{cur}} > D[(c_t, r_t)]$ then
          Set $D[(c_t, r_t)] = D_{\text{cur}}$
          Set $P[(c_t, r_t)] = (c_s, r_s)$
        end if
      end for
    end for
  end for
end for
```

Algorithm 2.2 Reconstruct the optimal path through the matching graph

```
inputs
  The predecessor map $P$
outputs
  The maximum-weight path $\Pi = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$ defined by $P$
/** Select the final vertex of the path **/
if $(n_1, n_2)$ is a vertex in $G$ then
  Set $v = (n_1, n_2)$
else if $D[(n_1 - 1, n_2)] > D[(n_1, n_2 - 1)]$ then
  Set $v = (n_1 - 1, n_2)$
else
  Set $v = (n_1, n_2 - 1)$
end if
Set $\Pi = \{v\}$
/** Trace the path back one step at a time until we reach the starting vertex **/
while $v$ is not equal to the start vertex do
  Set $v = P[v]$
  Insert $v$ at the beginning of $\Pi$
end while
```

2.3.4 Finding γ_1 and γ_2 which Realize the Optimal Matching

At this point, we have a method for finding a maximum-weight path Π through the matching graph. The path itself defines the best possible sequence of up-segments and down-segments, and the total weight of the path gives the maximum possible inner product attainable between orbit representatives of $[q_1]$ and $[q_2]$. If we are only interested in computing the shape distance between our two functions, we can stop here. However, for many applications, we need to find a pair of reparametrization functions $\gamma_1, \gamma_2 \in \Gamma$ which actually realize the optimal matching when applied to the unit-speed SRVFs q_1 and q_2. We next develop an algorithm to translate the matching path Π into such a pair of reparametrizations.

First, we should point out that there is not a unique solution to this problem: if $\gamma_1, \gamma_2 \in \Gamma$ is a pair of reparametrizations realizing an optimal matching between our functions, then $\gamma_1 \circ \psi$ and $\gamma_2 \circ \psi$ also give us an optimal matching, for any piecewise-linear $\psi \in \Gamma$. This is due to the fact that Γ acts by isometries on $\mathcal{C}$. Taking this idea a bit further, suppose that $0 = t_0 < t_1 < \ldots < t_n = 1$ is the common changepoint partition for $\tilde{q_1}$ and $\tilde{q_2}$ (as above, $\tilde{q_1}$ and $\tilde{q_2}$ are the SRVFs comprising our optimal matching). If $0 = u_0 < u_1 < \ldots < u_n = 1$ is any other partition with the same number of points, we can easily construct a piecewise-linear reparametrization $\psi \in \Gamma$ sending $u_i \mapsto t_i$ for $i = 0, 1, \ldots, n$. Then $\tilde{q_1} * \psi$ and $\tilde{q_2} * \psi$ is another optimal matching, but its changepoint partition is now $\{u_i\}$ instead of $\{t_i\}$. This shows that the changepoint partition for an optimal matching carries no meaningful information about the matching (aside from its length).

Now, suppose we have found a pair of optimal reparametrizations $\gamma_1, \gamma_2 \in \Gamma$, so that the SRVFs comprising the optimal matching can be written as $\tilde{q_1} = q_1 * \gamma_1$ and $\tilde{q_2} = q_2 * \gamma_2$. As above, let $0 = t_0 < t_1 < \ldots < t_n = 1$ be the changepoint partition for $\tilde{q_1}$ and $\tilde{q_2}$. Since

q_1 and q_2 are unit-speed SRVFs, it follows that for any $t \in [0,1]$, we have

$$|\tilde{q_i}(t)| = |(q_i * \gamma_i)(t)| = |\sqrt{\dot{\gamma_i}(t)}q_i(\gamma_i(t))| = \sqrt{\dot{\gamma_i}(t)}$$

for $i = 1, 2$. Since $\tilde{q_1}(t)$ and $\tilde{q_2}(t)$ are constant on each subinterval of the partition $\{t_i\}$, it follows that $\dot{\gamma_1}(t)$ and $\dot{\gamma_2}(t)$ are also constant on those subintervals. Therefore, γ_1 and γ_2 are piecewise-linear functions with $\{t_i\}$ as their common changepoint partition.

Above we noticed that choosing a different changepoint partition for $\tilde{q_1}$ and $\tilde{q_2}$ is equivalent to applying the same piecewise-linear reparametrization to both functions. Since the changepoints of $\tilde{q_1}$ and $\tilde{q_2}$ are also the changepoints of γ_1 and γ_2, this implies that *the changepoint parameters of γ_1 and γ_2 are completely arbitrary.* As we are building γ_1 and γ_2, we can choose any parition $\{t_i\}$ we like for the changepoint parameter values, so long as the partition has the correct number of points, and so long as we output the correct lists of function values $\{\gamma_1(t_i)\}$ and $\{\gamma_2(t_i)\}$. For lack of a "right" way to choose a changepoint partition, we simply use a uniformly-spaced partition of $[0,1]$. For the remainder of this section, we for the most part ignore the changepoint partition and focus on producing the lists of function values for γ_1 and γ_2.

Let $\Pi = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$ be the matching path produced by Algorithms 2.1 and 2.2. As above, let $\{t_i^1\}$ and $\{t_i^2\}$ denote the changepoint parameters of the original unit-speed SRVFs q_1 and q_2. Algorithm 2.3 produces the lists of function values for γ_1 and γ_2.

Most of the work of building γ_1 and γ_2 is done in Algorithm 2.4, which builds the pieces of γ_1 and γ_2 corresponding to a single up-segment or down-segment. Here is a short argument for its correctness. Consider the main loop in that algorithm, and suppose for concreteness that we are building an up-segment; down-segments work in the same way. In each iteration prior to the last, we output the γ_1 and γ_2 values corresponding to one of the patterns

$$\begin{bmatrix} UD \\ U0 \end{bmatrix} \qquad \text{or} \qquad \begin{bmatrix} U0 \\ UD \end{bmatrix}$$

For example, lines 17 and 18 create the pattern $\begin{bmatrix} UD \\ U0 \end{bmatrix}$. Suppose for the purposes of this discussion that we have chosen a changepoint partition for γ_1 and γ_2 (as noted above, the choice is arbitrary), and that our two-column pattern takes place on the intervals (u_1, u_2) and (u_2, u_3). Then we have

$$\begin{array}{lll} \gamma_1(u_1) = \tau_1 & \gamma_1(u_2) = t^1_{\alpha_1+1} & \gamma_1(u_3) = t^1_{\alpha_1+2} \\ \gamma_2(u_1) = \tau_2 & \gamma_2(u_2) = \lambda & \gamma_2(u_3) = \lambda \end{array}$$

For the first column, realized on the interval (u_1, u_2), we append $t^1_{\alpha_1+1}$ to X_1 and append λ to X_2, which results in both γ_1 and γ_2 having positive slope on the first interval of the pattern. Since q_1 is positive on $(\tau_1, t^1_{\alpha_1+1})$ and q_2 is positive on (τ_2, λ), this means that the reparametrized SRVFs $\tilde{q_1}$ and $\tilde{q_2}$ will be positive on (u_1, u_2). Further, looking at line 16, λ is chosen so that the slopes of γ_1 and γ_2 will have the correct ratio on (u_1, u_2). Therefore, $\tilde{q_1}$ and $\tilde{q_2}$ will also have the correct ratio on (u_1, u_2).

Algorithm 2.3 Build the lists of function values for γ_1 and γ_2

```
inputs
  The matching path $\Pi = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$
  Changepoint partitions $\{t_i^1\}$ and $\{t_i^2\}$ for unit speed SRVFs $q_1$ and $q_2$
outputs
  Function values $X_1 = \{\gamma_1(t_i)\}$ and $X_2 = \{\gamma_2(t_i)\}$ for $i = 1, 2, \ldots, n$
/** Initialize **/
Set $X_1 = \{0\}$ and $X_2 = \{0\}$
/** Deal with leading half-zero column, if present **/
if $(i_1, j_1) = (0, 1)$ then
  Set $X_1 = X_1 * \{0\}$ and set $X_2 = X_2 * \{t_1^2\}$
else if $(i_1, j_1) = (1, 0)$ then
  Set $X_1 = X_1 * \{t_1^1\}$ and set $X_2 = X_2 * \{0\}$
end if
/** Alternating sequence of up-segments and down-segments **/
for $k = 1$ to $m - 1$ do
  Append the values for the up- or down-segment defined by $(i_k, j_k)$ and $(i_{k+1}, j_{k+1})$ to
  $X_1$ and $X_2$, using Algorithm 2.4.
end for
/** Deal with trailing half-zero column, if present **/
if $(i_m, j_m) = (n_1, n_2 - 1)$ or $(i_m, j_m) = (n_1 - 1, n_2)$ then
  Set $X_1 = X_1 * \{1\}$ and set $X_2 = X_2 * \{1\}$
end if
```

Now, for the second interval of the pattern, we append $t^1_{\alpha_1+2}$ to X_1 and we append a second copy of λ to X_2. q_1 is negative on $(t^1_{\alpha_1+1}, t^1_{\alpha_1+2})$, and the two consecutive instances of λ create a horizontal segment in γ_2. Therefore, on (u_2, u_3), $\tilde{q}_1 < 0$ and $\tilde{q}_2 = 0$, resulting in the pattern $\begin{bmatrix} D \\ 0 \end{bmatrix}$ in the second column.

Lines 23-28 build the pattern $\begin{bmatrix} U0 \\ UD \end{bmatrix}$ in the same way that lines 16-21 build the pattern $\begin{bmatrix} UD \\ U0 \end{bmatrix}$. Finally, in the last iteration, line 10 adds the final values $t^1_{c_t}$ to X_1 and $t^2_{r_t}$ to X_2, and we are done.

2.4 Karcher Means

When dealing with a collection of functions, we are often interested in computing some sort of an “average” function. For example, Kurtek *et al.* [21] recently presented a method for signal estimation using the square root velocity framework. Computing the mean of the observations is a key step in their method.

In a linear space V, the average of a collection of points $v_1, v_2, \ldots, v_k \in V$ can be defined simply as

$$\bar{v} = \frac{1}{k} \sum_{i=1}^{k} x_k$$

However, our preshape space $\mathcal{C}$, and certainly our shape space $\mathcal{S}_{st}$, are nonlinear spaces, so finding the average of a collection of points in these spaces is much more difficult. Before going any further, we need to decide exactly what we mean by “average” in this case. Perhaps the most common notion of “average” in a nonlinear metric space is the Karcher mean, which is sometimes called a Fréchet mean (*c.f.* Karcher [16], Krakowski *et al.* [20], Rentmeesters and Absil [28]). Let $[q_1], [q_2], \ldots, [q_k] \in \mathcal{S}_{st}$ be a collection of points in the shape space. The Karcher mean of this collection, denoted $[\mu] \in \mathcal{S}_{st}$, is defined as

$$[\mu] = \underset{[q] \in \mathcal{S}_{st}}{\operatorname{argmin}} \sum_{i=1}^{k} d([q_i], [q])^2 \tag{2.3}$$

where the shape space distance $d(\cdot, \cdot)$ is given by

$$d([p], [q]) = \min_{u \in [p], v \in [q]} \cos^{-1}(\langle u, v \rangle)$$

2.4.1 Computing the Karcher Mean

We now present a method for computing the Karcher mean in $\mathcal{S}_{st}$. Our approach is basically the same as the method outlined in [33]; the only real difference is that we are using the 1-D function matching algorithms presented here, rather than the more general matching algorithms in [33], in the steps that require alignment to the current mean estimate. The underlying space of $\mathcal{S}_{st}$ is the infinite-dimensional sphere $\mathcal{C} \subset \mathbb{L}^2(I, \mathbb{R})$, so this approach is

Algorithm 2.4 Build the pieces of γ_1 and γ_2 corresponding to an up- or down-segment

```
inputs
  Vertices $(c_s, r_s)$ and $(c_t, r_t)$ defining the up- or down-segment
  Changepoint partitions $\{t^1_i\}$ and $\{t^2_i\}$ for the unit-speed SRVFs $q_1$ and $q_2$
outputs
  Sequences $X_1$ and $X_2$, containing the function values of $\gamma_1$ and $\gamma_2$ for the segment.
  $X_1$ and $X_2$ are meant to be appended to the end of the main lists of function values by
  the caller. Therefore, $X_1$ will not contain the first value $t^1_{c_s}$, and $X_2$ will not contain
  the first value $t^2_{r_s}$.
Set $v_1 = \sum_{i=0}^{\frac{c_t-c_s-1}{2}} (t^1_{c_s+2i+1} - t^1_{c_s+2i})$, $v_2 = \sum_{i=0}^{\frac{r_t-r_s-1}{2}} (t^2_{r_s+2i+1} - t^2_{r_s+2i})$, $R = \frac{v_2}{v_1}$
Set $\alpha_1 = c_s$, $\alpha_2 = r_s$          /** $\alpha_i$ = current index in $\{t^i_j\}$ **/
Set $\tau_1 = t^1_{c_s}$, $\tau_2 = t^2_{r_s}$          /** $\tau_i$ = last value output in $X_i$ **/
Set $u_1 = 0$, $u_2 = 0$          /** $u_i$ = rise so far in $f_i$ for segment **/
Set $X_1 = \{\}$, $X_2 = \{\}$
while $\alpha_1 < c_t$ and $\alpha_2 < r_t$ do
  if $\alpha_1 = c_t - 1$ and $\alpha_2 = r_t - 1$ then
    Set $X_1 = X_1 * \{t^1_{c_t}\}$, $X_2 = X_2 * \{t^2_{r_t}\}$ /** Finished **/
    Set $\alpha_1 = c_t$, $\alpha_2 = r_t$
  else
    /** $p_i$ = percent of total rise in $f_i$ up to $t^i_{\alpha_i+1}$ **/
    Set $p_1 = \frac{u_1+(t^1_{\alpha_1+1}-\tau_1)}{v_1}$, $p_2 = \frac{u_2+(t^2_{\alpha_2+1}-\tau_2)}{v_2}$
    if $p_1 < p_2$ then
      Set $\lambda = \tau_2 + R(t^1_{\alpha_1+1} - \tau_1)$
      Set $X_1 = X_1 * \{t^1_{\alpha_1+1}, t^1_{\alpha_1+2}\}$
      Set $X_2 = X_2 * \{\lambda, \lambda\}$
      Set $u_1 = u_1 + (t^1_{\alpha_1+1} - \tau_1)$, $u_2 = u_2 + (\lambda - \tau_2)$
      Set $\tau_1 = t^1_{\alpha_1+2}$, $\tau_2 = \lambda$
      Set $\alpha_1 = \alpha_1 + 2$
    else
      Set $\lambda = \tau_1 + \frac{1}{R}(t^2_{\alpha_2+1} - \tau_2)$
      Set $X_1 = X_1 * \{\lambda, \lambda\}$
      Set $X_2 = X_2 * \{t^2_{\alpha_2+1}, t^2_{\alpha_2+2}\}$
      Set $u_1 = u_1 + (\lambda - \tau_1)$, $u_2 = u_2 + (t^2_{\alpha_2+1} - \tau_2)$
      Set $\tau_1 = \lambda, \tau_2 = t^2_{\alpha_2+2}$
      Set $\alpha_2 = \alpha_2 + 2$
    end if
  end if
end while
```

similar to Karcher mean algorithms for collections of points on a finite-dimensional sphere $\mathbb{S}^n$ (*c.f.* Krakowski *et al.* [20]).

Suppose we have our collection of points $[q_1], [q_2], \ldots, [q_k] \in \mathcal{S}_{st}$, and suppose that $[q] \in \mathcal{S}_{st}$ is our current estimate of the mean $[\mu]$. We will see in a moment that it is possible to choose representatives from these orbits in such a way that all of the representatives from the orbits $[q_i]$ are simultaneously at minimal distance from the representative of $[q]$:

Proposition 5. *There exist SRVFs $p \in [q], p_1 \in [q_1], p_2 \in [q_2], \ldots, p_k \in [q_k]$ such that, for $i = 1, 2, \ldots, k$,*

$$\cos^{-1}(\langle p, p_i \rangle) = d([q], [q_i])$$

Let $p, p_1, p_2, \ldots, p_k \in \mathcal{C}$ be such a set of representatives. For $i = 1, 2, \ldots, k$, the **shooting vector** for p_i, denoted $v_i \in T_p\mathcal{C}$, is given by

$$v_i = (p_i - \langle p, p_i \rangle\, p) \cdot \frac{\theta}{\sin\theta} \tag{2.4}$$

This vector is tangent to the great circle path from p to p_i in $\mathcal{C}$; its length is equal to $\theta = \cos^{-1}(\langle p, p_i \rangle)$, the great circle distance from p to p_i. The negative of the gradient of our energy function is then given by

$$v = \frac{1}{k} \sum_{i=1}^{k} v_i \tag{2.5}$$

Algorithm 2.5 outlines the gradient descent used to find a local minimum of the energy function in equation 2.3. The remainder of this section deals with the problem of finding orbit representatives as in proposition 5.

2.4.2 Groupwise Alignment to the Mean

In order to simplify the expressions that follow, we introduce a couple of new symbols. The first represents the cumulative rise (resp. fall) in a function over the first part of an up-segment (resp. down-segment):

Definition 6. *Let $q \in \mathcal{C}$ be a unit-speed SRVF with changepoint partition $t_0 = 0 < t_1 < t_2 < \ldots < t_n = 1$, and let $a, b \in \{0, 1, \ldots, n\}$ with $a < b$ and $b - a$ odd. For $t \in [t_a, t_b]$, define $\rho_q(a, b, t)$ as*

$$\rho_q(a, b, t) = meas\left(\left(\bigcup_{i=0}^{\frac{b-a-1}{2}} (t_{a+2i+1} - t_{a+2i})\right) \cap (t_a, t)\right)$$

where meas$(\cdot)$ is Lebesgue measure. The second definition is the following:

Definition 7. *Let $q_1, q_2 \in \mathcal{C}$ be unit-speed SRVFs with changepoint partitions $0 = t_0^1 < t_1^1 < \ldots < t_{n_1}^1 = 1$ and $0 = t_0^2 < t_1^2 < \ldots < t_{n_2}^2 = 1$, respectively. Let v_1 and v_2 be vertices defining an edge in the matching graph. If v_1 is not the dummy vertex v_s, then the edge from v_1 to v_2 corresponds to an up- or down-segment in the matching. In this case, write*

Algorithm 2.5 Karcher mean in $\mathcal{S}_{st}$

```
inputs
  A collection of points $[q_1], [q_2], \ldots, [q_k] \in \mathcal{S}_{st}$
outputs
  The Karcher mean $[\mu] \in \mathcal{S}_{st}$ of the points $[q_i]$
/** Initialize **/
Choose a point $[q] \in \mathcal{S}_{st}$ as the initial mean estimate
loop
  Find orbit representatives $p, p_1, p_2, \ldots, p_k$ using Algorithm 2.6.
  Compute $v$, the negative of the gradient of the energy function, as in equation 2.5.
  if $\|v\| < \varepsilon$ then
    Stop and return the current mean estimate $[q]$.
  else /** Update $q$ **/
    Choose a small step size $\alpha$
    Set $u = q + \alpha \cdot v$.
    Set $q = \frac{u}{\|u\|}$.
  end if
end loop
```

$v_1 = (c_s, r_s)$ and $v_2 = (c_t, r_t)$. For $t \in [t^1_{c_s}, t^1_{c_t}]$, we define $\overrightarrow{\sigma}_{q_1,q_2}[(c_s, r_s), (c_t, r_t), t]$ to be the smallest number s in $[t^2_{r_s}, t^2_{r_t}]$ such that

$$\frac{\rho_{q_2}(r_s, r_t, s)}{\rho_{q_2}(r_s, r_t, r_t)} = \frac{\rho_{q_1}(c_s, c_t, t)}{\rho_{q_1}(c_s, c_t, c_t)}$$

Similarly, for $t \in [t^2_{r_s}, t^2_{r_t}]$, we define $\overleftarrow{\sigma}_{q_1,q_2}[(c_s, r_s), (c_t, r_t), t]$ to be the smallest number s in $[t^1_{c_s}, t^1_{c_t}]$ such that

$$\frac{\rho_{q_1}(c_s, c_t, s)}{\rho_{q_1}(c_s, c_t, c_t)} = \frac{\rho_{q_1}(r_s, r_t, t)}{\rho_{q_2}(r_s, r_t, r_t)}$$

In words, $\overrightarrow{\sigma}$ and $\overleftarrow{\sigma}$ translate an argument t of one function to an argument s of the other function which corresponds to the same percentage of the total rise (or fall) for the up- or down-segment.

If v_1 is the dummy vertex v_s, then the edge from v_1 to v_2 corresponds to a leading half-zero column in the matching, and v_2 is either $(0, 1)$ or $(1, 0)$. In this case, we define $\overrightarrow{\sigma}_{q_1,q_2}[v_s, (0, 1), \cdot]$ and $\overrightarrow{\sigma}_{q_1,q_2}[v_s, (1, 0), \cdot]$ as follows:

$$\overrightarrow{\sigma}_{q_1,q_2}[v_s, (0, 1), t] = \begin{cases} t^2_1 & \textit{if } t = 0 \\ \infty & \textit{if } t > 0 \end{cases} \qquad \overrightarrow{\sigma}_{q_1,q_2}[v_s, (1, 0), t] = \begin{cases} 0 & \textit{if } t \le t^1_1 \\ \infty & \textit{if } t > t^1_1 \end{cases}$$

Similarly, $\overleftarrow{\sigma}_{q_1,q_2}[v_s, (0, 1), \cdot]$ and $\overleftarrow{\sigma}_{q_1,q_2}[v_s, (1, 0), \cdot]$ are defined as:

$$\overleftarrow{\sigma}_{q_1,q_2}[v_s, (0, 1), t] = \begin{cases} 0 & \textit{if } t \le t^2_1 \\ \infty & \textit{if } t > t^2_1 \end{cases} \qquad \overleftarrow{\sigma}_{q_1,q_2}[v_s, (1, 0), t] = \begin{cases} t^1_1 & \textit{if } t = 0 \\ \infty & \textit{if } t > 0 \end{cases}$$

The groupwise alignment algorithm is given in Algorithm 2.6. The basic idea is that we build the reparametrizations $\gamma_1, \gamma_2, \ldots, \gamma_k, \eta \in \Gamma$ by stepping through the SRVFs in parallel, outputting one new function value for each of the γ_i's and η at each step. On each step, we look at all of the SRVFs to find the peak or valley which should be visited first. Then for each of the γ_i's and η, we output the parameter value which corresponds to that peak or valley. The maps $\overrightarrow{\sigma}$ and $\overleftarrow{\sigma}$ are used to translate parameter values between functions, so that the ratios of the slopes of the γ_i's to η on the relevant intervals (and therefore the ratios of the reparametrized q_i's to μ) are correct, as per Lemma 6.

2.5 Experimental Results

2.5.1 Pairwise Alignment of Randomly-Generated Functions

As a basic test of our method, we aligned pairs of randomly-generated functions using our new method as well as the general-purpose dynamic programming algorithm typically used in the SRV framework (see Section 3.4 for a description of the general-purpose algorithm). Figure 2.5.1 shows the resulting alignments for one such pair of functions.

We performed this comparison on 1000 pairs of randomly-generated functions. The number of sample points for each of these functions was randomly chosen to be between 20 and 70 points, inclusive. For the general-purpose dynamic programming algorithm, we used a 6×6 neighborhood, so that the slopes of admissible segments were between $\frac{1}{6}$ and 6.

For these 1000 trials, the matchings produced by the new method yielded shape distances that were lower by about 30 percent on average, and were never higher than the distances produced by the general-purpose algorithm. The average running time for the new method was about 4 ms per call, and the average running time for the general-purpose algorithm was about 10 ms per call.

We remark that using a 6×6 neighborhood in the general-purpose matching algorithm sacrifices accuracy for speed. However, after repeating the experiment with a 17×17 neighborhood, the distance results were not appreciably different, but the running time of the general-purpose method was much higher.

2.5.2 Groupwise Alignment of Time Series Data

Following Kurtek *et al.* [21], we used our methods to compute the Karcher mean of a collection of time series data taken from the Berkeley growth study [36]. The original dataset gives the heights of 39 boys and 54 girls at 31 different ages, between 1 and 18 years, inclusive. As in [21], we actually work with a centered difference approximation of the first derivative of the data, which better illustrates the growth patterns. We analyzed the growth rates for the boys and the girls separately. For each group of growth rate functions $f_1, f_2, \ldots, f_n$, we found normalizing reparametrizations $\alpha_1, \alpha_2, \ldots, \alpha_n$ such that each function $f_i \circ \alpha_i$ is a piecewise-linear function with constant speed. We then found the unit-norm, unit-speed SRVFs $q_1, q_2, \ldots, q_n$ for each f_i, and computed their Karcher mean μ according to Algorithm 2.5. Then we used the groupwise alignment method in Algorithm 2.6 to compute functions $\gamma_\mu, \gamma_1, \gamma_2, \ldots, \gamma_n \in \Gamma$ such that for $i = 1, 2, \ldots, n$, $q_i * \gamma_i$

Algorithm 2.6 Groupwise alignment of the q_i to the current mean estimate μ

```
  inputs
    $\{t_i^1\}_{i=0}^{n_1}$,$\{t_i^2\}_{i=0}^{n_2}$,. . .,$\{t_i^k\}_{i=0}^{n_k}$,$\{t_i^\mu\}_{i=0}^{n_\mu}$ : changepoint partitions for the unit-speed SRVFs.
    Matching paths $\Pi^l = \{(i_1^l, j_1^l), (i_2^l, j_2^l), \ldots, (i_{m_l}^l, j_{m_l}^l)\}$, for $l = 1, 2, \ldots, k$.
  outputs
    Lists $X_1, X_2, \ldots, X_l$, and $Y$ of values for the piecewise-linear reparametrizations
    $\gamma_1, \gamma_2, \ldots, \gamma_k, \eta \in \Gamma$ which simultaneously maximize $\langle \mu * \eta, q_i * \gamma_i \rangle$ for $i = 1, 2, \ldots, k$.
  For $l = 1, 2, \ldots, k$, set $X_l = \{0\}$, $\alpha_l = 0$, $\beta_l = 0$, $\tau_l = 0$
  Set $Y = \{0\}$, $\alpha_\mu = 0$, $\tau_\mu = 0$
  while $\alpha_\mu < n_\mu$ do
    for $l = 1$ to $k$ do
      Set $c_l = \overleftarrow{\sigma}_{\mu,q_l}[\Pi^l_{\beta_l}, \Pi^l_{\beta_l+1}, t^l_{\alpha_l+1}]$ if $\beta_l < m_l$, or $c_l = 1$ otherwise
    end for
    Set $c = \min\{c_1, c_2, \ldots, c_k, t^\mu_{\alpha_\mu+1}\}$
    if $c = t^\mu_{\alpha_\mu+1}$ then
      Set $\tau_\mu = t^\mu_{\alpha_\mu+1}$, set $Y = Y * \{t^\mu_{\alpha_\mu+1}\}$, and set $\alpha_\mu = \alpha_\mu + 1$
      for $l = 1$ to $k$ do
        Let $(c_t, r_t)$ denote the index pair in $\Pi^l_{\beta_l+1}$
        if $\alpha_\mu = c_t$ then
          Set $\tau_l = t^l_{r_t}$, set $\alpha_l = r_t$, and set $\beta_l = \beta_l + 1$
        else
          Set $\tau_l = \overrightarrow{\sigma}_{\mu,q_l}[\Pi^l_{\beta_l}, \Pi^l_{\beta_l+1}, \tau_\mu]$
        end if
        Set $X_l = X_l * \{\tau_l\}$
      end for
    else
      Set $\tau_\mu = c$, and set $Y = Y * \{c\}$
      for $l = 1$ to $k$ do
        if $\beta_l = m_l$ then
          Set $\tau_l = t^l_{\alpha_l}$
        else if $c = \overleftarrow{\sigma}_{\mu,q_l}[\Pi^l_{\beta_l}, \Pi^l_{\beta_l+1}, t^l_{\alpha_l+1}]$ then
          Set $\tau_l = t^l_{\alpha_l+1}$ and set $\alpha_l = \alpha_l + 1$
          if $\alpha_l = j^l_{\beta_l+1}$ then
            Set $\beta_l = \beta_l + 1$
          end if
        else
          Set $\tau_l = \overrightarrow{\sigma}_{\mu,q_l}[\Pi^l_{\beta_l}, \Pi^l_{\beta_l+1}, c]$
        end if
        Set $X_l = X_l * \{\tau_l\}$
      end for
    end if
  end while
  if $\alpha_l < n_l$ for any $l$ then
    Set $X_l = X_l * \{1\}$ for $l = 1, 2, \ldots, k$, and set $Y = Y * \{1\}$
  end if
```

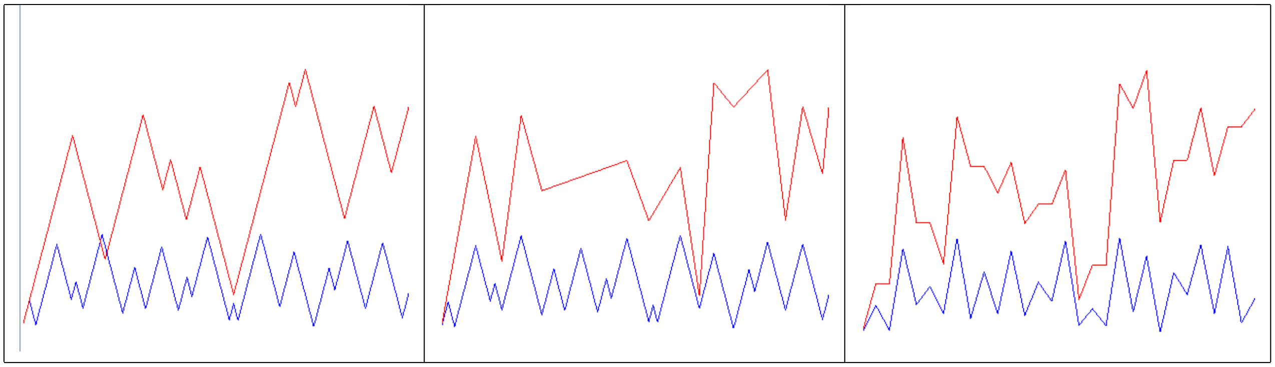

Figure 2.3: A comparison between the new 1-D function matching and the old elastic matching. The pane at the far left shows two randomly-generated functions. In the center pane is the alignment produced by the dynamic programming elastic matching algorithm typically used in the SRV framework; this alignment yielded a shape distance of approximately 0.737. At the far right is the alignment produced by the specialized 1-D function matching algorithms described in this chapter; for this alignment, the shape distance was 0.470.

is optimally aligned to $\mu * \gamma_\mu$. The second column of Figure 2.4 shows the piecewise-linear functions $f_i \circ (\alpha_i \circ \gamma_i)$ corresponding to the groupwise-aligned SRVFs.

For display purposes, we applied an additional reparametrization, again following [21], which brings back the features of the growth functions which are lost when we normalize to constant-speed parametrizations. Specifically, we considered the functions $\tilde{\gamma}_i = \alpha_i \circ \gamma_i$, representing the reparametrizations which, when applied to the original functions, will yield the groupwise-aligned piecewise-linear functions shown in the center column. The basic idea here is to center the $\tilde{\gamma}_i$ by finding the average $\bar{\gamma}$ of these functions, and then composing each $\tilde{\gamma}_i$ on $\bar{\gamma}^{-1}$. The average function $\bar{\gamma}$ is found by taking the SRVF of the $\tilde{\gamma}_i$, which are just points in $\mathcal{C}$, and using gradient descent to find the Karcher mean. This gradient descent is identical to the one described in Algorithm 2.5, except that here we are not optimizing over reparametrizations, so the groupwise alignment step is not needed. The third column of Figure 2.4 shows the result of applying these centered reparametrizations to the original functions.

It should be pointed out that the average function $\bar{\gamma}$ is usually not injective. To get around this problem, we slightly modify the function so that it is always strictly increasing. Then the inverse of this function is applied to the $\tilde{\gamma}_i$.

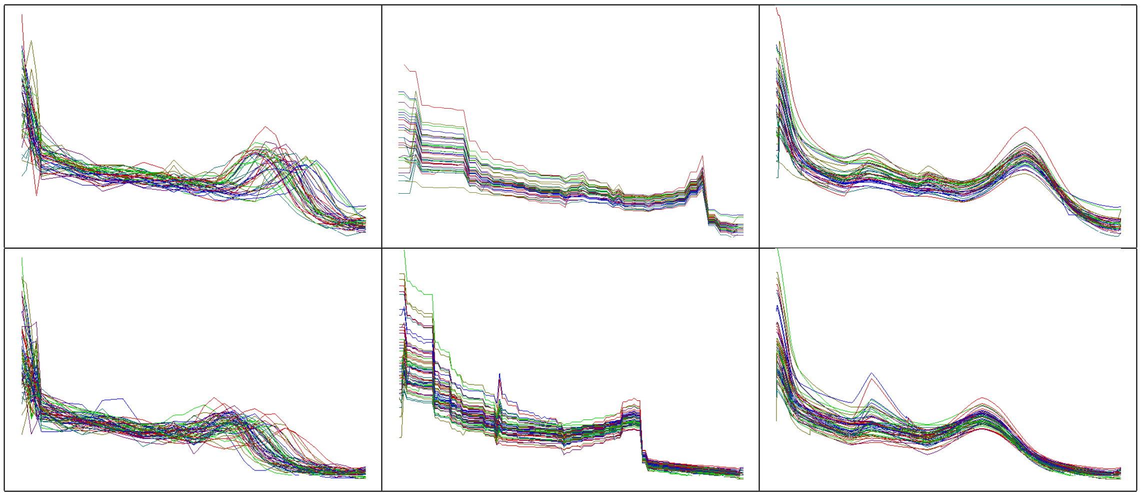

Figure 2.4: Groupwise alignment results on the Berkeley growth study data. The top row shows growth rate functions for 39 boys, and the growth rates for 54 girls are on the bottom row. The original growth rate functions are shown in the left column. In the center are the piecewise-linear functions optimally aligned to the Karcher mean. The right column shows the functions aligned for display purposes, as described above. The groupwise alignment accentuates the peaks common to all of the observations; these peaks represent growth spurts.

CHAPTER 3

PARTIAL MATCHING IN THE SQUARE ROOT VELOCITY FRAMEWORK

In this chapter we develop a method of finding optimal partial correspondences between elastic curves in $\mathbb{R}^n$, working within the square root velocity (SRV) framework. At its heart, our method is an extension of the dynamic programming algorithm for *full* elastic matching in the SRV framework.

3.1 Overview and Prior Work

The partial matching problem arises in many applications. For example, if we are searching a 2-dimensional image for occurrences of a model shape, it is often the case that matching objects in the image are partially hidden behind other objects. These partially-occluded objects will usually not resemble the model shape as a whole, so a means of detecting partial similarities is useful. Partial matching is also applicable to protein structural analysis, since chains which are globally dissimilar may still share similar domains.

Bronstein *et al.* investigated the partial matching problem for both planar shapes [6] and surfaces [7]. These authors view the partial matching problem as a multi-objective optimization problem: we want to match parts of the shapes which are very similar (*i.e.* we want to minimize the shape distance of the matched parts), while at the same time, we want to match as large a proportion of the two shapes as possible (*i.e.* we want to minimize the *partiality* of the match). Unless the two shapes are similar as a whole, it is impossible to simultaneously minimize the shape distance and the partiality, so the partial matching problem fits naturally into a Paretian framework. We follow this approach in Section 3.3.

Several approaches to the partial matching problem have been proposed which represent each shape as a sequence of feature points, introduce some sort of a distance measure on the space of all possible feature point values, and then use dynamic programming techniques to find optimal partial matches between sequences. Chen *et al.* [9] used an adaptation of the Smith-Waterman protein sequence alignment algorithm [32] to find partial correspondences between planar shapes, which are represented as sequences of feature vectors. Their method is quite flexible, in that many different representations may be used. In the paper, they demonstrate the method with a descriptor derived from the curvature and chord distance, but the authors' implementation also includes support for the shape context descriptor of

Belongie *et al.* [2].

Donoser *et al.* [10] introduced a partial matching algorithm for closed planar curves in which each curve is represented by a matrix derived from the angles formed by certain chords across the shape. Once the matrix is computed for each shape, the integral image transform is used to efficiently find matching pairs of submatrices, which represent matching pairs of subcurves. Sebastian *et al.* [29] use a 1-dimensional representation for planar shapes called a *shock graph*. Briefly, the medial axis of a shape can be interpreted as the locus of shocks (or singularities) in the course of waves propagating inward from the boundary of the shape. This interpretation induces a flow on each segment of the medial axis, and the resulting directed graph is called the shock graph. Under this representation, one shape can be deformed into another using a combination of bending/stretching (i.e. deforming edges in the graph), as well as splicing and contracting (i.e. adding or removing edges altogether).

As mentioned above, partial matching also has applications to structural alignment of proteins and RNA molecules. This is a very large area of research, and many specialized algorithms have been proposed to solve the problem, such as DALI [14] and CE [31] (for proteins) and ARTS [11] and SARA [8] (for RNAs). These methods typically use biochemical information in addition to the shape of the backbone when producing their alignments, so they can be expected to outperform a general-purpose algorithm which measures similarity based only on the shape of the backbone. While augmenting the square root velocity framework to take residue or nucleotide sequence into account has been shown to improve the performance of the SRV framework in these applications (*c.f.* Liu *et al.* [24], Laborde *et al.* [22]), we will not explore that avenue here.

3.2 Partial Matches in the Square Root Velocity Framework

3.2.1 Modifications to the Framework

In order to use the SRV framework for partial matching, it is convenient to make a few minor changes to the framework. First, we drop the restriction that all curves must have unit arc length. Our space of admissible curves then becomes the set of all absolutely-continuous functions $\beta : [0,1] \to \mathbb{R}^n$ such that $\beta(0) = 0$; as in Chapter 1, we let $\mathcal{AC}_0$ denote this space. The SRV map $\mathcal{Q} : \mathcal{AC}_0 \to \mathbb{L}^2([0,1], \mathbb{R}^n)$ is defined (up to a set of measure zero) exactly as in Chapter 1:

$$\mathcal{Q}(\beta)(t) = \begin{cases} \frac{\dot\beta(t)}{\sqrt{\|\dot\beta(t)\|}} & \text{if } \dot\beta(t) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

However, this time the image of $\mathcal{Q}$ is not just the unit sphere, but all of $\mathbb{L}^2([0,1], \mathbb{R}^n)$. Consequently, geodesics in the new preshape space are just straight lines, and distances can be measured using the $\mathbb{L}^2$ distance. We will let $\mathcal{C}$ denote the preshape space:

$$\mathcal{C} = \mathbb{L}^2([0,1], \mathbb{R}^n)$$

The groups $\mathrm{SO(n,\mathbb{R})}$ and Γ act on $\mathcal{C}$ in the same way as in Chapter 1: for $q \in \mathcal{C}$, $A \in \mathrm{SO(n,\mathbb{R})}$, and $\gamma \in \Gamma$, we have

$$(A, q) = Aq \qquad\qquad (q, \gamma) = \sqrt{\dot\gamma}(q \circ \gamma) \tag{3.2}$$

As before, these actions are isometric with respect to the $\mathbb{L}^2$ metric. For $q \in \mathcal{C}$, we let $[q]$ denote the *closure* of the orbit of q under the two actions:

$$[q] = \mathrm{cl}\{\mathrm{A}\sqrt{\dot{\gamma}}(\mathrm{q} \circ \gamma) | \mathrm{A} \in \mathrm{SO(n, \mathbb{R})}, \gamma \in \Gamma\} \tag{3.3}$$

The set of all such closed-up orbits is our shape space, denoted $\mathcal{S}$:

$$\mathcal{S} = \{[q] | q \in \mathcal{C}\} \tag{3.4}$$

The distance between two orbits $[q_1], [q_2] \in \mathcal{S}$ is defined as

$$d([q_1], [q_2]) = \inf \left\| q_1 - A\sqrt{\dot{\gamma}(t)} q_2(\gamma(t)) \right\| \tag{3.5}$$

where the inf runs over all possible choices of $A \in \mathrm{SO(n, \mathbb{R})}$ and $\gamma \in \Gamma$.

3.2.2 Subcurves and Partial Matches

Now suppose that $\beta : [0, 1] \to \mathbb{R}^n$ is a curve in $\mathcal{AC}_0$, and choose a subinterval $[a, b] \subset [0, 1]$. Restricting β to the interval $[a, b]$ results in a curve $\beta|_{[a,b]}$ which is a piece of the original curve, and which will in general have a shorter arc length. It is for this reason that we dropped the restriction that our curves have unit arc length: if $\eta : [0, 1] \to [a, b]$ is any increasing diffeomorphism, then we can parametrize this "piece" of β on the interval $[0, 1]$:

$$(\beta|_{[a,b]}) \circ \eta : [0, 1] \to \mathbb{R}^n$$

and after a translation, the resulting function will be an element of $\mathcal{AC}_0$ in its own right. Therefore, the shape distance between $\beta|_{[a,b]}$ and any other element of $\mathcal{AC}_0$ can be defined as the distance from $(\beta|_{[a,b]}) \circ \eta$ to that curve. Note that our choice of η does not matter, since any other choice would have resulted in a different element of the same Γ-orbit. Also note that, for the purposes of computing the shape distance, whether we translate $(\beta|_{[a,b]}) \circ \eta$ back to the origin or not does not matter, since both the translated and untranslated versions will have the same image under $\mathcal{Q}$. Finally, since the unit-length requirement has been dropped, the subcurve does not have to be rescaled in order to fit into our framework. The advantages of this will become clearer when we develop our partial matching algorithm in Section 3.5.

A partial match between two curves $\beta_1, \beta_2 \in \mathcal{AC}_0$ is defined by a choice of two subintervals $[a, b]$ and $[c, d]$. It is important to note that, if either β_1 or β_2 is reparametrized, then the corresponding interval $[a, b]$ or $[c, d]$ will, in general, parametrize a different piece of the original curve. We adopt the convention that, for a partial match, the defining subintervals are taken relative to the constant-speed parametrizations of β_1 and β_2.

3.3 Measuring the Quality of a Partial Match

Before developing an algorithm to find optimal partial matches, we first need to decide exacly what it means for a partial match to be optimal. The quality of a partial match between $\beta_1|_{[a,b]}$ and $\beta_2|_{[c,d]}$ is determined by two aspects. The first is the *partiality*, denoted $\lambda(a, b, c, d)$, which measures the portions of β_1 and β_2 which are not included in the match.

As noted above, we are assuming that β_1 and β_2 are constant-speed parametrizations, so that the lenghts of $[a, b]$ and $[c, d]$ are proportional to the arc lengths of the corresponding subcurves. Therefore, we define the partiality as

$$\lambda(a, b, c, d) = 2 - ((b - a) + (d - c)) \tag{3.6}$$

A partiality of 0 indicates a match of the whole curves, and a partiality of 2 indicates an empty match. The second aspect is the *shape distance* (or dissimilarity) between the two subcurves, which we defined above. We will use the symbol $\varepsilon(a, b, c, d)$ to represent the shape distance for the partial match between $\beta_1|_{[a,b]}$ and $\beta_2|_{[c,d]}$.

If we are given a choice between two partial matches with the same partiality, and one of the matches has a lower shape distance, then we prefer that match over the other one. Likewise, if we are given a choice between two partial matches having the same shape distance, and one of the matches has a lower partiality, then we prefer that match over the other one because matches involving larger portions of the curves are likely (although not always) more significant than matches involving smaller portions of the curves.

In the SRV framework, we can make the shape distance as small as we like by considering sufficiently small subcurves. In fact, a shape distance of zero can always be obtained by choosing an empty match; however, this is usually not the desired solution. On the other hand, unless the shapes of β_1 and β_2 are fully similar, the partial match with a partiality of 0 (*i.e.* the full match) will not have the lowest possible shape distance.

Consequently, it is usually impossible to simultaneously minimize the partiality and the shape distance. As observed by Bronstein *et al.* [6], the partial matching problem fits naturally into a Paretian framework, in which all partial matches that are not dominated in both partiality and shape distance by any other match are considered *Pareto-optimal*, and the set of Pareto-optimal partial matches is called the *Pareto frontier*. In terms of λ and ε, the definition of Pareto-optimal is as follows:

Definition 8. *The partial match defined by $[a, b]$ and $[c, d]$ is* ***Pareto-optimal*** *if, for all other choices of intervals $[a', b']$ and $[c', d']$, we have either $\lambda(a, b, c, d) \le \lambda(a', b', c', d')$ or $\varepsilon(a, b, c, d) \le \varepsilon(a', b', c', d')$.*

3.3.1 The Pareto Frontier

We now establish a couple of useful facts about the Pareto frontier. First, for each possible partiality $l \in [0, 2]$, define $\mathcal{P}(l)$ as follows:

$$\mathcal{P}(l) = \inf\{\varepsilon(a, b, c, d) | \lambda(a, b, c, d) = l\} \tag{3.7}$$

Here is our first fact:

Lemma 7. *The function $\mathcal{P}$ is non-increasing: for all partiality measures l_1 and l_2 with $l_1 < l_2$, we have $\mathcal{P}(l_1) \ge \mathcal{P}(l_2)$.*

Proof. Suppose we have a match $[a, b]$,$[c, d]$ with $\lambda(a, b, c, d) = l_1$. Let $\eta_1 : [0, 1] \to [a, b]$ and $\eta_2 : [0, 1] \to [c, d]$ be increasing diffeomorphisms, and let α_1 and α_2 be any elements of the orbits of $\beta_1 \circ \eta_1$ and $\beta_2 \circ \eta_2$, respectively, under the actions of $\mathrm{SO}(\mathrm{n}, \mathbb{R})$ and Γ. Choose $z \in [0, 1]$ such that $2 - ((\eta_1(z) - a) + (\eta_2(z) - c)) = l_2$. Then

$$\int_0^1 \|\mathcal{Q}(\alpha_1)(t) - \mathcal{Q}(\alpha_2)(t)\|^2 \ dt \geq \int_0^z \|\mathcal{Q}(\alpha_1)(t) - \mathcal{Q}(\alpha_2)(t)\|^2 \ dt$$

It is straightforward to see that this second integral is the preshape distance corresponding to a match of partiality l_2, which is a restriction of our original match of partiality l_1. Therefore, the minimal shape distance for partiality l_2 can be no larger than the minimal shape distance for partiality l_1. □

Our second fact is a useful criterion for Pareto optimality.

Lemma 8. *A match of partiality l is Pareto-optimal if and only if*

1. *the shape distance of the match is $\mathcal{P}(l)$, and*
2. *for all $l' < l$, $\mathcal{P}(l') > \mathcal{P}(l)$ (note the strict inequality).*

Proof. To see that both of these conditions are necessary for a Pareto-optimal match, note that if condition (1) fails for a partial match, then that match is not Pareto-optimal since there are partial matches having the same partiality and lower shape distances. Similarly if condition (2) fails for a match, then that match cannot be Pareto-optimal because there are longer matches having the same shape distance. On the other hand, if a partial match satisfies both conditions, then all matches having lower shape distance must have higher partiality (by condition (1) and the fact that $\mathcal{P}(l)$ is non-increasing), and all matches with lower partiality have strictly higher shape distance (by condition (2)). □

3.3.2 The High-Level Partial Matching Algorithm

Because of discretization, we have only finitely-many possible match lengths in practice, so the Pareto-optimal matches can be found by identifying, for each possible match length l, the set of partial matches of that length having the minimal shape distance $\mathcal{P}(l)$. After discarding matches which fail to satisfy condition (2) above, we are left with the set of Pareto-optimal partial matches. The high-level algorithm for partial matching is described in Algorithm 3.1. The majority of the work lies in computing the shape distance for every possible match. Since we are using an elastic curve representation, we must find optimal elastic matchings for every possible pair of subcurves. The main goal of this chapter is to develop a method of doing this efficiently.

3.4 Full Elastic Matching using Dynamic Programming

In this section, we give a brief summary of the *full* elastic matching algorithm used in the square root velocity framework, which is described in Mio *et al.* [27] and in Srivastava and Klassen [33]. We examine this method in detail because in Section 3.5, we will extend it to an algorithm for finding optimal elastic matches for every possible partial match between β_1 and β_2. Throughout this section, we assume that the scale and rotational orientation of

Algorithm 3.1 High-level algorithm for identifying the Pareto frontier

```
[Initialize] For each partiality $l$, set $\mathcal{P}(l) = \infty$ and set $\mathcal{M}(l) = \{\}$.
for each partial match $[a, b]$, $[c, d]$ do
  Set $l = \lambda(a, b, c, d)$
  Compute $r = \varepsilon(a, b, c, d)$
  if $r \leq \mathcal{P}(l)$ then
    Set $\mathcal{P}(l) = r$
    Add the tuple $(a, b, c, d, r)$ to the set $\mathcal{M}(l)$
    for each $(a', b', c', d', r')$ in $M(l)$ with $r' > \mathcal{P}(l)$ do
      Remove $(a', b', c', d', r')$ from $\mathcal{M}(l)$.
    end for
  end if
end for
```

β_1 and β_2 are fixed. Again, assume that both curves are constant-speed parametrizations. Let $q_1 = \mathcal{Q}(\beta_1)$ and $q_2 = \mathcal{Q}(\beta_2)$.

Recall that our elastic matching problem is that of finding

$$\hat{\gamma} = \underset{\gamma \in \Gamma}{\operatorname{arginf}} \int_0^1 \left\| q_1(t) - \sqrt{\dot{\gamma}(t)} q_2(\gamma(t)) \right\|^2 dt$$

At this point, we define an energy function for increasing diffeomorphisms $\eta : [a, b] \to [c, d]$ between subintervals $[a, b], [c, d] \subseteq [0, 1]$:

$$E[\eta] = \int_a^b \left\| q_1(t) - \sqrt{\dot{\eta}(t)} q_2(\eta(t)) \right\|^2 dt \tag{3.8}$$

Such functions η will be the "building blocks" out of which we will piece together full reparametrization functions $\gamma \in \Gamma$. In terms of the energy function above, our elastic matching problem can be written as

$$\hat{\gamma} = \underset{\gamma \in \Gamma}{\operatorname{arginf}}\ E[\gamma] \tag{3.9}$$

3.4.1 Discretizing the Problem

The elastic matching problem can be discretized and then solved using dynamic programming. We first choose two partitions $0 = t_1^1 < t_2^1 < \ldots < t_{N_1}^1 = 1$ and $0 = t_1^2 < t_2^2 < \ldots < t_{N_2}^2 = 1$ of the interval $[0, 1]$, and then consider the grid on the unit square $[0, 1] \times [0, 1]$ consisting of all points of the form (t_i^1, t_j^2). Diffeomorphisms will be approximated by piecewise-linear paths from $(0, 0)$ to $(1, 1)$ whose segments begin and end on gridpoints and have positive slopes, as shown in Figure 3.1.

For each grid point $p = (t_i^1, t_j^2)$, we have a set of *predecessors* of p, consisting of all grid points $p' = (t_k^1, t_l^2)$ such that

1. $k < i$ and $l < j$,

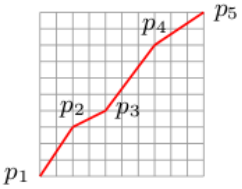

Figure 3.1: An increasing, piecewise-linear path through the grid, approximating a diffeomorphism. The path passes through grid points $p_1, \ldots, p_m$, and e_k denotes the line segment from p_k to p_{k+1}. The energy of the path is equal to the sum of the energies of the segments e_k.

2. $\gcd(i-k, j-l) = 1$, and
3. $(i-k), (j-l) \leq B_{dp}$, where B_{dp} is user-specified

In this case, we will call the line segment from p' to p an *admissible segment*. We will concatenate admissible segments together to form *admissible paths* through the grid from $(0,0)$ to $(1,1)$; these paths approximate diffeomorphisms $\gamma \in \Gamma$. The first condition in the list above ensures that all paths have positive, finite slope. The second condition is motivated by the observation that if $\gcd(i-k, j-l) > 1$, then the segment from p_2 to p_1 must pass through another grid point. Therefore, if we exclude this segment from our collection of basic building blocks, we can still recover it as the concatenation of two smaller segments. The third condition places constraints on the slopes which an admissible segment is allowed to have. This is done to reduce the time complexity of the elastic matching algorithm: we will see later that the algorithm's running time with the slope constraints is $O(N_1 N_2)$, but without them the running time is $O((N_1 N_2)^2)$.

We will represent an admissible path by the sequence of gridpoints through which it passes; for example, $(p_1, p_2, \ldots, p_m)$ represents a path that starts at gridpoint p_1, passes through $p_2, \ldots, p_{m-1}$ as intermediate points, and ends at p_m. The discretized version of our elastic matching problem is that of finding an admissible path with minimal *energy*, which we define now.

3.4.2 The Energy of a Path

Each admissible segment from $p_1 = (t_i^1, t_j^2)$ to $p_2 = (t_k^1, t_l^2)$ in the grid is the graph of a linear function $\eta : [t_i^1, t_k^1] \to [t_j^2, t_l^2]$. The energy of this segment is just the energy of η, as given in equation 3.8; slightly abusing notation, we denote this by $E[p_1, p_2]$. The energy of an admissible path is defined as the sum of the energies in the segments making up the path.

3.4.3 The Dynamic Programming Functional Equation

Now, consider an admissible path passing through gridpoints $(p_1, \ldots, p_m)$. Suppose that our path is optimal (*i.e.* out of all paths from p_1 to p_m, ours has minimal energy). Then if we break the path into subpaths $(p_1, \dots, p_i)$ and $(p_i, \dots, p_m)$, these subpaths must also be optimal: if there was a better path, say, from p_1 to p_i, then concatenating this path with $(p_i, \dots, p_m)$ would yield a path from p_1 to p_m with lower energy than our optimal path, a contradiction. This shows that the elastic matching problem has optimal substructure. If $D[p]$ denotes the energy of an optimal path from $(0, 0)$ to p, then the dynamic programming functional equation for our elastic matching problem is

$$D[p] = \min_{p'} D[p'] + E[e_{p'p}] \tag{3.10}$$

where p' ranges over all predecessors of p and $e_{p'p}$ is the segment from p' to p. $D[(1, 1)]$ then is our discrete approximation of the minimum energy achievable by any $\gamma \in \Gamma$. To compute $D[(1, 1)]$, we compute $D[p]$ for all gridpoints p, starting at $(0, 0)$ and working left-to-right across each row from the bottom up. As each value $D[p]$ is computed, it is stored in a lookup table and used to compute subsequent values. We also keep track of the best predecessor of each gridpoint, so that the optimal path can be reconstructed by tracing backwards from $(1, 1)$ to $(0, 0)$ after all values of D have been computed. The procedure is summarized in Algorithm 3.2.

Algorithm 3.2 Compute the optimal path through the elastic matching graph using dynamic programming

```
inputs
  Grid partitions $\{t^1_1, t^1_2, \dots, t^1_{n_1}\}$ and $\{t^2_1, t^2_2, \dots, t^2_{n_2}\}$
  Edge weights $E[p_1, p_2]$ for all gridpoints $p_1$ and $p_2$
outputs
  The minimum-energy path $\Pi$ from $(0, 0)$ to $(1, 1)$ in the matching graph, and
  $D[(1, 1)]$, the minimum energy
[Initialize] Set $D[(0, 0)] = 0$, and set $D[p] = \infty$ for all other gridpoints $p$
for $r_t = 1$ to $n_2$ do
  for $c_t = 1$ to $n_1$ do
    Let $p_t$ denote the gridpoint $(t^1_{c_t}, t^2_{r_t})$
    for each predecessor $p_s$ of $p_t$ do
      if $D[p_s] + E[p_s, p_t] < D[p_t]$ then
        Set $D[p_t] = D[p_s] + E[p_s, p_t]$
        Set $P[p_t] = p_s$
      end if
    end for
  end for
end for
[Initialize the path] Set $\Pi = \{(1, 1)\}$
Set $p = (1, 1)$
while $p \neq (0, 0)$ do
  Set $p = P[p]$
  Insert $p$ at the beginning of $\Pi$
end while
```

3.5 Elastic Partial Matching using Dynamic Programming

3.5.1 Computing the Shape Distances of all Partial Matches

We now consider the task of computing the elastic shape distances for all partial matches. At this stage, we are still keeping the rotational alignments of both curves fixed. The key observation providing the link between the *full* elastic matching algorithm and the *partial* matching problem is that any rectangle in the elastic matching grid corresponds to a partial match. A path across such a rectangle, from the lower-left corner to the upper-right corner, determines a particular elastic matching of the corresponding subcurves. Therefore, an optimal elastic matching of two subcurves is just a minimum-energy path through the corresponding rectangle in the grid (see Figure 3.2).

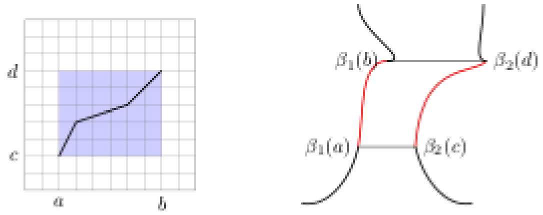

Figure 3.2: A rectangle in the elastic matching grid, corresponding to a pair of subcurves. The rectangle from (a, c) to (b, d) corresponds to the subcurves $\beta_1|_{[a,b]}$ and $\beta_2|_{[c,d]}$. A path from (a, c) to (b, d) determines an elastic matching of these subcurves. If such a path has minimal energy, then that path determines an optimal elastic matching between the subcurves.

At this point, it is natural to express our matching problems in the language of graph theory. We can think of our elastic matching grid as a weighted, directed graph with a vertex for every gridpoint and an edge for every admissible segment. Then the full elastic matching problem is just the single-source shortest path problem, and the partial matching problem is the *all-pairs* shortest path problem. To solve this problem, we use the well-known Floyd-Warshall algorithm.

3.5.2 The Floyd-Warshall Algorithm

Let G be a weighted graph with vertex set $\mathcal{V} = \{v_1, \dots, v_n\}$ and edge set $\mathcal{E} = \{e_1, \dots, e_m\}$. Let $W(e_i) \in \mathbb{R}$ denote the weight of the edge e_i. Assume that G has no parallel edges, and that no cycle in G has negative weight. Now, for integers $i, j \in \{1, 2, \dots, n\}$ and $k \in \{0, 1, \dots, n\}$, we call a path from v_i to v_j ***k*-minimal** if it has the lowest weight attainable by any path from v_i to v_j which only uses vertices $v_1, v_2, \dots, v_k$ as intermediate vertices. The 0-minimal paths are just the single-edge paths. Define $F[i, j, k]$ as the weight of a k-minimal path from v_i to v_j (or set $F[i, j, k] = \infty$ if there are no k-minimal paths from v_i to v_j).

Suppose that for some $k < n$, we know $F[i, j, k]$ for all $i, j \in \{1, 2, \ldots, n\}$. For a particular pair of vertices v_i and v_j, let $\alpha_{i,j,k+1}$ be any $(k+1)$-minimal path from v_i to v_j. There are two possibilities for the path $\alpha_{i,j,k+1}$: either $\alpha_{i,j,k+1}$ actually uses v_{k+1} as an intermediate vertex, or it does not. If $\alpha_{i,j,k+1}$ does not use v_{k+1} as an intermediate vertex, then $\alpha_{i,j,k+1}$ is actually k-minimal, so we have $F[i, j, k+1] = F[i, j, k]$. Otherwise, $\alpha_{i,j,k+1}$ is the concatenation of paths $\alpha_{i,k+1,k}$ from v_i to v_{k+1} and $\alpha_{k+1,j,k}$ from v_{k+1} to v_j. Since our graph has no negative-weight cycles, we can assume that neither $\alpha_{i,k+1,k}$ nor $\alpha_{k+1,j,k}$ uses v_{k+1} as an intermediate vertex. Further, it is easy to see that both of these paths must be k-minimal (else we could find a $(k+1)$-minimal path from v_i to v_j with lower weight than $\alpha_{i,j,k+1}$). Therefore in this situation, we have $F[i, j, k+1] = F[i, k+1, k] + F[k+1, j, k]$. The dynamic programming functional equation is

$$F[i, j, k] = \min(F[i, j, k-1], F[i, k, k-1] + F[k, j, k-1]).$$

An outline of the algorithm is as follows. The values of $F[i, j, k]$ will be computed and stored in an $n \times n$ array A. $A[i, j]$ is initialized to the weight of the edge from v_i to v_j (or ∞ if there is no edge from v_i to v_j). For each value of k from 1 to n, we initially have $A[i, j] = F[i, j, k-1]$ for all i, j. We then compute $F[i, j, k]$ by considering all pairs of vertices v_i and v_j, and setting

$$A[i, j] = \min(A[i, j], A[i, k] + A[k, j]) \tag{3.11}$$

Whether or not any k-minimal path from v_i to v_j uses v_k as an intermediate vertex, $A[i, j]$ will be equal to $F[i, j, k]$. Finally, after the iteration with $k = n$, we have $A[i, j] = F[i, j, n]$ for all i, j, which is the weight of the shortest path from v_i to v_j. Clearly, the running time is $O(n^3)$, which is remarkable since there are n^2 shortest paths being computed. See Algorithm 3.3 for a summary of the algorithm.

Notice that, during the Floyd-Warshall algorithm, the matrix A is modified in-place; however, this does not affect the computation. To see this, notice that during the k^{th} pass, entries that lie in the k^{th} row or the k^{th} column do not change when we update them using equation 3.11 (this is a consequence of our assumption that G has no negative cycles, so that the minimum path weight from any vertex to itself is zero). As for the other entries $A[i, j]$, the update depends on the current value $A[i, j]$ as well as $A[i, k]$ and $A[k, j]$. All of these three entries have the same value as they had at the beginning of the k^{th} pass. Therefore, the Floyd-Warshall algorithm does not require additional storage.

3.5.3 Open Curve Partial Matching with Fixed Rotation

Now, if the rotational alignment of the curves remains fixed, then finding the Pareto frontier is simply a matter of running the Floyd-Warshall algorithm on the partial matching graph to compute the shape distances, and then examining for each partial match (a, b, c, d) the partiality $\lambda(a, b, c, d)$ and shape distance $\varepsilon(a, b, c, d)$ to see if the partial match is Pareto-optimal.

3.5.4 Optimizing Over Rotations

In most applications, we need to optimize over rotations as well as reparametrizations. This presents a challenge for our method: changing the rotational alignment of the curves

Algorithm 3.3 The Floyd-Warshall algorithm

```
/** Initialize **/
Initialize $A$ so that $A[i,j]$ is equal to the weight of the edge from $v_i$ to $v_j$ (or $\infty$ if there is no edge from $v_i$ to $v_j$).
/** Main Loop **/
for $k = 1$ to $n$ do
  for $i = 1$ to $n$ do
    for $j = 1$ to $n$ do
      Set $A[i,j] = \min(A[i,j], A[i,k] + A[k,j])$
    end for
  end for
end for
```

results in a change in the weights of the matching graph which is not easy to predict, and if the curves are not properly aligned, then we will miss good partial matches. Our strategy for coping with this problem is as follows. If rotation invariance is required, then we select a representative set of rotations such that, for each subcurve pair, there is at least one rotation in our set that, when applied to β_2, aligns the subcurves well enough to approximate the rotation-invariant shape distance. The entire partial matching procedure for fixed rotational alignment, as outlined above, is then carried out for each of these rotations.

To build our collection of representative rotations, we do the following. For each partial match (a, b, c, d), we first find the rotation $A \in \mathrm{SO}(\mathrm{n}, \mathbb{R})$ which, when applied to β_2, optimally aligns these subcurves. Next, we find the matrix B in our collection which is closest to A (where we compute the distance between two elements of $\mathrm{SO}(\mathrm{n}, \mathbb{R})$ by considering them as points in $\mathbb{R}^{n^2}$ and taking the Euclidean distance). If the distance is above a user-specified threshold, then we add A to our collection; otherwise, we do not. Since this requires a large number of nearest-neighbor searches, we store the representative rotations in a $k - d$ tree as the collection is being built.

This strategy is quite effective for planar curves, and still effective for curves in $\mathbb{R}^3$, although the number of rotations necessary to achieve good results can be quite large (several hundred) in the latter case.

3.6 Selecting Matches from the Pareto Frontier

While the Pareto frontier is useful, and is in a way the complete solution to our partial matching problem, in most applications we need to select a single match, or in some cases a handful of matches, as the "best". In this section, we present two strategies for making such a choice.

3.6.1 Strategy 1: Minimize the Salukwadze Distance

This strategy is taken from Bronstein *et al.* [6]; the idea is to place some sort of a metric on the Pareto frontier, and find the point on the frontier with minimal distance from the ideal point $(0, 0)$ under this metric. In our experiments, we have used metrics of the form

$$\Phi_w(\lambda, \varepsilon) = \lambda + w\varepsilon \tag{3.12}$$

where λ is the partiality, ε is the shape distance, and w is a constant which determines the relative weight given to λ and ε; this constant must be chosen experimentally. The partial match (or matches – there may be more than one) corresponding to the point on the Pareto frontier which minimizes Φ_w are returned as the result. Also, the minimal value of Φ_w can be used as an overall dissimilarity measure between the two curves, which can be used for clustering or retrieval applications.

3.6.2 Strategy 2: Find the Knee of the Pareto Curve

This strategy is motivated by the following line of reasoning: suppose that we are matching two curves which are partially similar, but not globally similar. For very high values of the partiality λ (*i.e.* very short matches), we can find partial matches between similar parts of the two curves. As the partiality decreases, our matches between similar parts of the curves grow into longer matches between similar parts of the curves. However, since the curves are not globally similar, there exists a partiality below which any partial match necessarily involves matching dissimilar pieces of the curves, resulting in a more rapid increase in the shape distance as the partiality approaches 0. In our experience, the Pareto curve $\mathcal{P}(l)$ tends to be approximately linear for the highest values of l, but there is usually a point $l = l_0$ where the linearity breaks down and the graph quickly gets steeper.

We turn this into a match selection strategy as follows. Suppose that the possible values of the partiality are $0 = l_1 < l_2 < \ldots < l_M = 2$. We first choose a small threshold τ. Then for $i = M - 1$ down to 1, we find the linear regression of the points $(l_i, \mathcal{P}(l_i))$, $(l_{i+1}, \mathcal{P}(l_{i+1}))$, $\ldots$, $(l_M, \mathcal{P}(l_M))$, and then compute the sum of squared residuals (using only the points that were involved in the regression). If the sum of squared residuals is greater than our threshold τ, then we stop and return l_{i+1} as the partiality of the knee point. If the sum of squared residuals is never above the threshold, then we return 0 as the partiality of the knee point.

Once we have found l_0, the partiality of the knee of the Pareto curve, we consider all Pareto-optimal partial matches which have partiality l_0 or greater to be “good”. Matches with partiality less than l_0 are considered to be “bad” since these likely match dissimilar pieces of the curve. Depending on the specific application, we may simply return the largest “good” match, or we may select several of the “good” matches. Like the Salukwadze distance, the partiality of the knee point can be used as an overall dissimilarity measure for clustering or retrieval.

3.7 Experimental Results

3.7.1 Qualitative Results

In this section we present some qualitative partial matching results on several different datasets. The first of these is a small collection of images of plant leaves provided by Wu *et al.* [37] as part of the Flavia project. We created partially-occluded copies of these images, and then extracted the boundaries using the `potrace` utility [30]. We then used our partial

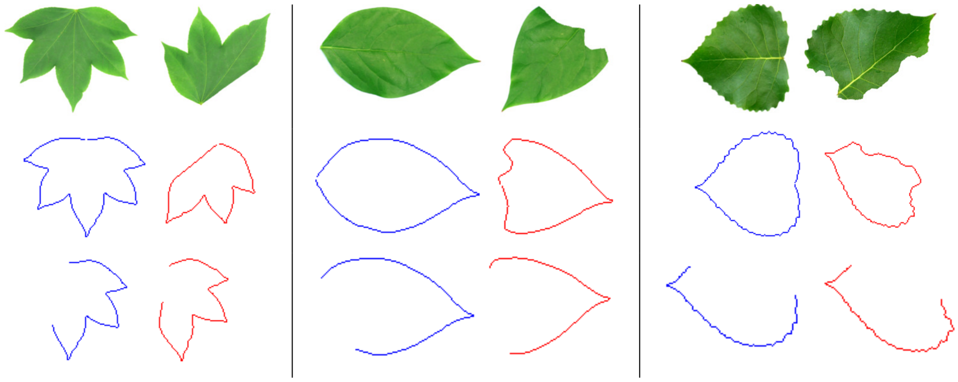

Figure 3.3: Partial matches of an object with a partially-occluded version of the same object. The original images are shown on the top row. The second row shows the full boundary curves, and the match selected by our algorithm is shown on the bottom row.

matching program to identify the largest "good" match between the original boundary curve and the boundary of the partially-occluded leaf image, using the method described in Section 3.6.2 with a regression error threshold of $\tau = 0.003$. For this experiment, the matching grid size was 30×30, and the strategy outlined in Section 3.5.4 was used to approximate the rotation-invariant shape distance for each partial match; this resulted in 13-14 rotations on average. Figure 3.3 shows the results for three cases.

For the next example, we used the UNIPEN ICROW '03 dataset [13], which contains digitized handwriting samples from multiple authors. The partial matching program was used to find occurrences of a three-character substring in six larger words. For this experiment, all samples were taken from the same writer. We used a 40×40 matching grid, and we used the strategy outlined in Section 3.6.2 with regression error threshold $\tau = 0.003$ to select a single match. Since all of the handwriting samples in the dataset were already horizontally aligned, we did not optimize over rotations for this example. The results are shown in Figure 3.4.

3.7.2 Classification of Partially-Occluded Planar Shapes

For the sake of completeness, we describe the results of an experiment in which we used our partial matching algorithm to build a simple classifier, which was then run against a collection of 99 planar shapes from the Kimia database [29]. The 99 shapes represent 9 different categories of objects, and there are 11 shapes in each category. The boundaries of these shapes were extracted using the `potrace` utility [30]. Here, the matching grid dimensions were 25×25, and we optimized over rotations as described in Section 3.5.4. We used the strategy described in Section 3.6.2, with a regression error threshold of $\tau = 0.003$, to select a single match, then used the partiality of the match as a dissimilarity score for the pair of shapes as a whole. After computing this distance for each pair of shapes, we performed a leave-one-out nearest-neighbor classification. Out of the 99 class assignments,

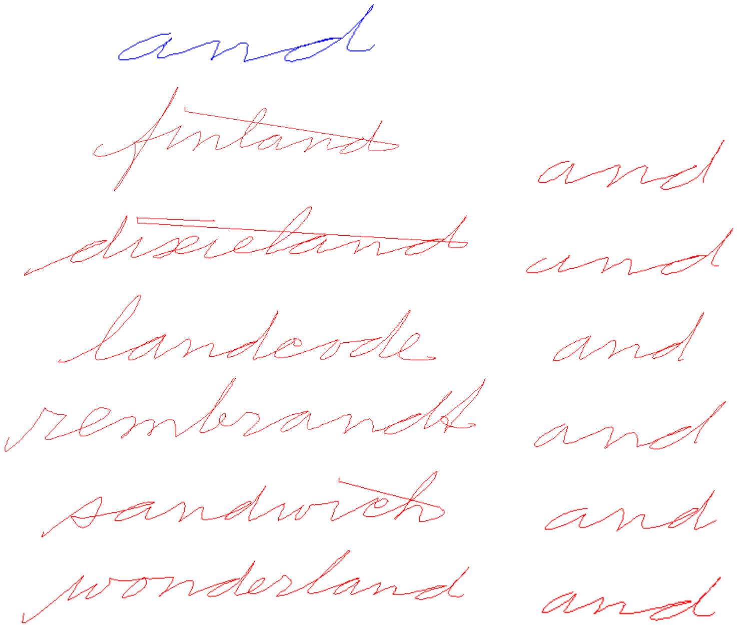

Figure 3.4: A common substring identified in several cursive handwriting samples. Top row: model curve, representing a commonly-occuring substring. Each of the bottom six rows shows a full curve (left) and the subcurve identified by our program as best matching the model (right).

96 were correct.

For comparison, we repeated the same experiment, this time using the full elastic shape distance as described in Algorithm 1.1 for the classification. Out of the 99 class assignments, 91 were correct. However, the running time for the classification using the full elastic shape distance was only 18 minutes, while the running time for the classification using partial matching was well over an hour.

CHAPTER 4

LIBSRVF: AN OPEN SOURCE LIBRARY FOR ELASTIC CURVE SHAPE ANALYSIS

libsrvf is a collection of code implementing the open-curve portions of the square root velocity framework, as well as the algorithms described in Chapters 2 and 3. In this chapter, we give a brief overview of the library, and discuss a few of the implementation issues and pitfalls that we encountered in writing it.

4.1 Project Overview

The library is written in `C++`, runs on Linux, and is freely available under the terms of the GNU General Public License. At the time of this writing, the library was hosted on GitHub (https://github.com/fsu-ssamg/libsrvf).

libsrvf consists of a main package which provides all of the basic classes and algorithms implementing the SRV framework. There is also an optional graphics package which provides very basic plotting support using FLTK and OpenGL. In addition to the native `C++` interface, libsrvf includes a high-level interface that enables it to be used from within Matlab or GNU Octave.

libsrvf uses the GNU Scientific Library (www.gnu.org/software/gsl) for some linear algebra routines; this dependency may be removed in the future. If the plotting features are enabled, then FLTK (www.fltk.org) must be installed, as well as OpenGL.

4.1.1 Algorithms

libsrvf provides the following algorithms:

- Conversion between original function and SRVF representation
- Interpolation; evaluation of PLFs and SRVFs
- Distances and inner products in $\mathbb{L}^2$; great circle distance
- Optimal rotational alignment for curves in $\mathbb{R}^n$
- Full elastic matching using dynamic programming

- Full shape distance calculation for open curves in $\mathbb{R}^n$
- Great-circle and straight-line geodesics in $\mathbb{L}^2$
- Karcher mean for open curves in $\mathbb{R}^n$
- Elastic partial matching for open curves in $\mathbb{R}^n$
- Specialized elastic matching for 1-D functions
- Specialized Karcher mean and groupwise alignment for 1-D functions

4.2 Usage Examples

4.2.1 Compute the Shape Distance Between Two Open Curves

Suppose that `beta1.csv` and `beta2.csv` are CSV datafiles, each one containing a matrix of sample points representing an open curve in $\mathbb{R}^3$. Suppose that the CSV files are actually space-delimited, and that the sample point matrices are in point-per-column order (i.e. the first row contains the x-coordinates of all points, the second contains the y-coordinates, the third row contains the z-coordinates, and each point is stored in a single column). The code in Figure 4.1 will compute the open-curve shape distance between these two curves, as described in Algorithm 1.1, and write it to standard output.

4.2.2 Groupwise Alignment of a Collection of Real-Valued Functions

In this example, we have a collection of time series data stored in the file `data.csv`. The file contains a 1-row matrix for each observation, and the data points were sampled at uniform time intervals. The following code uses the specialized 1-D functional data analysis routines described in Algorithms 2.5 and 2.6 to compute the elastic Karcher mean of these functions and then reparametrize all of the functions to best match the mean. Finally, a plot of the results is displayed using one of the convenience functions in plothelper.h.

4.2.3 Elastic Partial Matching

The following example uses the partial matching routines in the namespace `srvf::pmatch` to compute the Pareto frontier for a pair of elastic open curves in $\mathbb{R}^n$. For this example, assume that `F1` and `F2` are instances of `srvf::Plf` that have been created as in the last two examples.

4.3 Implementation Details

Although the SRV framework models curves as continuous functions $[0, 1] \to \mathbb{R}^n$, in practice we are given a finite sequence $p_1, p_2, \ldots, p_k \in \mathbb{R}^n$ of points as input. In order to transform this finite sequence of points into a continuous function, one assigns parameter values $0 = t_1 < t_2 < \cdots < t_k = 1$ to the points p_i, and then fits a continuous curve to these data points. There are many ways to do this, and in the past our own programs have used

```
// Load the sample point matrices
std::ifstream ifs1("beta1.csv"), ifs2("beta2.csv");
std::vector<srvf::Matrix> samps1_data = srvf::io::load_csv(ifs1, ' ', '\n');
std::vector<srvf::Matrix> samps2_data = srvf::io::load_csv(ifs2, ' ', '\n');
ifs1.close(); ifs2.close();

// Initialize piecewise-linear functions from the sample points.
// F1 and F2 will be assigned uniformly-spaced parameter values
// from 0 to 1.
srvf::Pointset samps1(samps1_data[0], srvf::Pointset::POINT_PER_ROW);
srvf::Pointset samps2(samps2_data[0], srvf::Pointset::POINT_PER_ROW);
srvf::Plf F1(samps1);
srvf::Plf F2(samps2);

// Compute the square root velocity functions of F1 and F2, and project
// them onto the preshape space.
srvf::Srvf Q1 = srvf::plf_to_srvf(F1);
srvf::Srvf Q2 = srvf::plf_to_srvf(F2);
Q1.scale_to_unit_norm();
Q2.scale_to_unit_norm();

// Compute the shape distance and output it.
// By default, the function shape_distance() will do just one
// rotate-reparametrize-rotate optimization round.
double d = srvf::opencurves::shape_distance(Q1,Q2);
std::cout << "Shape distance is " << d << std::endl;
```

Figure 4.1: Code sample to load two curves from file and compute their shape distance

piecewise-cubic Hermite splines, cubic B-splines, Bézier curves, and Akima interpolation, with fairly good results. However, libsrvf simply uses piecewise-linear interpolating splines, for the following reasons.

First, piecewise-linear splines are simple, so they make life easier in several ways when working in the SRV framework. Most importantly, the derivative of a piecewise-linear function is piecewise-constant, which means that SRVFs are piecewise-constant. This in turn enables us to compute integrals such as

$$\int_0^1 \|q_1(t) - q_2(t)\| \; dt$$

exactly, without the need for sophisticated quadrature routines.

The second reason libsrvf uses piecewise-linear interpolation is that, if we have no additional information about what happens in between the points p_i, a piecewise-linear interpolation is just as faithful to the data that we are given as any other interpolation. The fact that the derivative does not exist at the joints does not pose a problem, since our curves are only assumed to be absolutely continuous.

```
// Load the data from file
std::ifstream ifs("data.csv");
std::vector<srvf::Matrix> samps_data = srvf::io::load_csv(ifs,' ');
ifs.close();

std::vector<srvf::Plf> Fs;        // Original PLFs
std::vector<srvf::Plf> Fsr;       // Aligned PLFs
std::vector<srvf::Srvf> Qs;       // Unit-speed, unit-norm SRVFs
size_t nfuncs = samps_data.size();

// Create a Plf instance for each observation, and compute its SRVF
for (size_t i=0; i<nfuncs; ++i)
{
  srvf::Pointset samps(samps_data[i],srvf::Pointset::POINT_PER_COLUMN);
  Fs.push_back(srvf::Plf(samps));
  Qs.push_back(srvf::plf_to_srvf(Fs[i]));
}

// Colors for the functions in Fs (will be cycled through automatically)
std::vector<srvf::plot::Color> colors;
colors.push_back(srvf::plot::Color(1.0,0.0,0.0));  // red
colors.push_back(srvf::plot::Color(0.0,1.0,0.0));  // green
colors.push_back(srvf::plot::Color(0.0,0.0,1.0));  // blue

// Compute the Karcher mean, and groupwise-align all SRVFs to it
// Stop when the gradient norm drops below 0.0005, or after 30 iterations
srvf::Srvf Mu = srvf::functions::karcher_mean(Qs, 0.0005, 30);
Mu = srvf::constant_speed_param(Mu);
std::vector<srvf::Plf> GE=srvf::functions::groupwise_optimal_reparam(Mu, Qs);

// Display the results
for (size_t i=0; i<Qs.size(); ++i)
{
  Fsr.push_back(composition(Fs[i], GE[i]));
}
srvf::Plf FMu = srvf_to_plf(Mu);
Fsr.push_back(composition(FMu, GE.back()));
plot_1d_plfs(Fsr,colors,0,0,600,400,"Groupwise alignment example");
```

Figure 4.2: Code sample to compute groupwise alignment of a collection of 1-D functions

```
// Compute the SRVFs of F1 and F2
srvf::Srvf Q1 = srvf::plf_to_srvf(F1);
srvf::Srvf Q2 = srvf::plf_to_srvf(F2);

// Find the Pareto−optimal partial matches.
// By default, find_matches() will use the changepoint parameters
// of Q1 and Q2 for the matching grid parameters, and it will also
// optimize over rotations.
srvf::pmatch::ParetoSet S = srvf::pmatch::find_matches(Q1, Q2);

// Print the results
std::cout << ”# Salukwadze dist: ”
          << S.salukwadze_dist(1.0)
          << std::endl;

std::cout << ”# Pareto set:” << std::endl;
for (size_t i=0; i<S.nbuckets(); ++i)
{
  for (size_t j=0; j<S[i].size(); ++j)
  {
    std::cout << S[i][j].a << ” ” << S[i][j].b
              << ” ” << S[i][j].c << ” ” << S[i][j].d
              << S[i][j].dist << std::endl;
  }
}
```

Figure 4.3: Code sample to find the Pareto frontier and Salukwadze distance for a pair of open curves

BIBLIOGRAPHY

[1] Robert B. Ash and Catherine A. Doléans-Dade. *Probability and Measure Theory.* Academic Press, 2000.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002 (PAMI 2002)*, 2002.

[3] H Blum. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 1967.

[4] Vladimir I. Bogachev. *Measure Theory, Volumes I-II.* Springer, 2006.

[5] F.L. Bookstein. Landmark methods for forms without landmarks: morphometrics of group differences in outline shape. *Medical Image Analysis*, 1(3):225–43, April 1997.

[6] Alexander M. Bronstein, Michael M. Bronstein, Alfred M. Bruckstein, and Ron Kimmel. Analysis of two-dimensional non-rigid shapes. *International Journal of Computer Vision*, 78(1):67–88, 2008.

[7] Alexander M. Bronstein, Michael M. Bronstein, Alfred M. Bruckstein, and Ron Kimmel. Partial similarity of objects, or how to compare a centaur to a horse. *International Journal of Computer Vision*, 84(2):163–183, 2009.

[8] Emidio Capriotti and Marc A. Marti-Renom. Rna structure alignment by a unit-vector approach. *Bioinformatics*, 24(16):I112–I118, AUG 15 2008.

[9] Longbin Chen, Rogerio Ferris, and Matthew Turk. Efficient partial shape matching using smith-waterman algorithm. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008 (CVPR 2008)*, 2008.

[10] Michael Donoser, Hayko Riemenschneider, and Horst Bischof. Efficient partial shape matching of outer contours. In *ACCV (1)*, pages 281–292, 2009.

[11] O. Dror, R. Nussinov, and H. J. Wolfson. Arts: alignment of rna tertiary structures. *Bioinformatics*, 21(Suppl. 2):47–53, 2005.

[12] I.L. Dryden and K.V. Mardia. *Statistical Shape Analysis.* John Wiley & Sons, 1998.

[13] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Proceedings of the 12th International Conference on Pattern Recognition (ICPR'94)*, pages 29–33, October 1994.

[14] L. Holm and C. Sander. Mapping the protein universe. *Science*, 273:595–603, 1996.

[15] Shantanu Joshi, Eric Klassen, Anuj Srivastava, and Ian Jermyn. A novel representation for riemannian analysis of elastic curves in $\mathbb{R}^n$. *IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

[16] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30:509–541, 1977.

[17] David G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, 1984.

[18] E. Klassen, A. A. Srivastava, W. Mio, and S.H. Joshi. Analysis of planar shapes using geodesic paths on shape spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(3):372–383, March 2004.

[19] Eric Klassen and Daniel Robinson. Analysis of functions under reparametrization. Technical report, work in progress.

[20] KA Krakowski, K Huper, and JH Manton. On the computation of the karcher mean on spheres and special orthogonal groups. *Proceedings of the Workshop on Robotics and Mathematic (RoboMat'07)*, 2007.

[21] Sebastian Kurtek, Wei Wu, and Anuj Srivastava. Signal estimation under random time warpings and its applications in nonlinear signal alignments. In *Neural Information Processing Systems (NIPS)*, 2011.

[22] Jose Laborde, Daniel Robinson, Anuj Srivastava, Eric Klassen, and Jinfeng Zhang. Rna alignment in the joint sequence-structure space using elastic shape analysis. *Nucleic Acids Research*, in review.

[23] Huiling Le and David G. Kendall. The riemannian structure of euclidean shape spaces: A novel environment for statistics. *The Annals of Statistics*, 21(3):1225–1271, 1993.

[24] Wei Liu, Anuj Srivastava, and Jinfeng Zhang. A mathematical framework for protein structure comparison. *PLoS Comput Biol*, 7(2), 02 2011.

[25] Peter W. Michor and David Mumford. An overview of the riemannian metrics on spaces of curves using the hamiltonian approach. *Applied and Computational Harmonic Analysis*, 23:74–113, 2007.

[26] W. Mio, J.C. Bowers, and X. Liu. Shape of elastic strings in euclidean space. *International Journal of Computer Vision*, 82(1):96–112, 2009.

[27] W. Mio, A. Srivastava, and S. H. Joshi. On shape of plane elastic curves. *Intl. Journal of Computer Vision*, 73(3):307–324, 2007.

[28] Quentin Rentmeesters and P.A. Absil. Algorithm comparison for karcher mean computation of rotation matrices and diffusion tensors. In *19th European Signal Processing Conference (EUSIPCO 2011)*, 2011.

[29] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Recognition of shapes by editing shock graphs. In *ICCV*, pages 755–762, 2001.

[30] Peter Selinger. Potrace. *http://potrace.sourceforge.net*.

[31] I.N. Shindyalov and P.E. Bourne. Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. *Protein Engineering*, 11(9):739–747, 1998.

[32] TF Smith and MS Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147:195–197, 1981.

[33] Anuj Srivastava and Eric Klassen. *Statistical Analysis of Shapes of Curves and Surfaces*. Springer, work in progress.

[34] Anuj Srivastava, Eric Klassen, Shantanu Joshi, and Ian Jermyn. Shape analysis of elastic curves in euclidean spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1415–1428, July 2011.

[35] Elias M. Stein and Rami Shakarchi. *Real Analysis: Measure Theory, Integration, And Hilbert Spaces*. Princeton University Press, 2005.

[36] R. D. Tuddenham and M. M. Snyder. Physical growth of california boys and girls from birth to age 18. *University of California Publications in Child Development*, 1:183–364, 1954.

[37] Stephen Gang Wu, Forrest Sheng Bao, Eric You Xu, Yu-Xuan Wang, Yi-Fan Chang, and Qiao-Liang Xiang. A leaf recognition algorithm for plant classification using probabilistic neural network. *IEEE 7th International Symposium on Signal Processing and Information Technology*, Dec. 2007.

[38] Laurent Younes, Peter W. Michor, Jayant Shah, and David Mumford. A metric on shape spaces with explicit geodesics. Technical report, 2007.

BIOGRAPHICAL SKETCH

Daniel Robinson was born in Fletcher, North Carolina, and is the youngest of three children. Before coming to Florida State University, he attended the University of North Carolina at Asheville, where he earned a bachelor's degree in computer science and mathematics. Prior to that, he worked as a computer repair technician, as a residential maintenance assistant, as a construction worker, and (briefly) as a salesman. Aside from mathematics and shape analysis, his interests include artisan baking, foreign languages, music, and anything related to computer programming.