# Riemannian Optimization for Registration of Curves in Elastic Shape Analysis

Wen Huang [†¶] K. A. Gallivan [‡] Anuj Srivastava [§] P.-A. Absil [†]

October 14, 2015

**Abstract**

In elastic shape analysis, a representation of a shape is invariant to translation, scaling, rotation and reparameterization, and important problems such as computing the distance and geodesic between two curves, the mean of a set of curves, and other statistical analyses require finding a best rotation and reparameterization between two curves. In this paper, we focus on this key subproblem and study different tools for optimizations on the joint group of rotations and reparameterizations. We develop and analyze a novel Riemannian optimization approach and evaluate its use in shape distance computation and classification using two public data sets. Experiments show significant advantages in computational time and reliability in performance compared to the current state-of-the-art method. A brief version of this paper can be found in [HGSA14].

**Keywords:** Elastic shape; Square root velocity function; Elastic closed curves; Dynamic programming; Riemannian optimization; Riemannian quasi-Newton;

## 1 Introduction

Shape analysis of curves plays an important role in a variety of imaging applications. The basic idea is to isolate contours of objects in images (2D or 3D) and use the shapes of these contours to characterize the original objects. Hence, there is a great interest in tools for shape analysis of planar, closed contours. Many approaches to shape analysis have been proposed in the literature and used to varying degrees of success in applications, e.g., point-based methods, domain-based shape representations and parameterized curve representations. A large majority of past work on statistical shape analysis has been using landmark-based descriptions [DM98]. In this setup, one samples contours with a fixed number of points in a pre-determined way, e.g., using uniform spacing, and the ensuing analysis is based on Euclidean analysis of vectors of landmarks. A consequence of this analysis is that the registration of landmarks—which points on one contour match with which points on the other—is already predetermined. This often results in matching parts across shapes that have different geometrical features. An important solution to this and related problems in shape analysis of contours came in the form of elastic shape analysis which has become increasingly important in recent years due to its superior theoretical basis and empirically demonstrated

[†]Department of Mathematical Engineering, Université catholique de Louvain, B-1348, Louvain-la-Neuve, Belgium.
[‡]Department of Mathematics, Florida State University, Tallahassee FL 32306-4510, USA.
[§]Department of Statistics, Florida State University, Tallahassee FL 32306-4510, USA.
[¶]Corresponding author. E-mail: huwst08@gmail.com.

effectiveness. In elastic shape analysis of contour, the objects of study are parameterized contours and, in pairwise comparisons, one solves for the optimal reparameterizations of contours using an appropriate metric. Figure 1 shows geodesics between two closed curves with and without reparameterization, and using reparameterization clearly gives a more natural transformation between the two curves. More abstractly, elastic shape analysis leads to shape metrics and statistical summaries of shapes that are invariant to the original parameterizations of the contours. The flexibility in parameterizations of curves helps in improving matching of parts across shapes and has the effect of stretching/bending the curves in optimally deforming one into the other—hence, the name elastic shape analysis. Such a framework was first introduced for general 2D curves by Younes [You98] and later studied by several groups. Klassen et al. [KSMJ04] narrowed the focus by studying shape analysis of *closed*, planar curves but did not allow the curves to have arbitrary parameterizations. Younes et al. [YMSM08] focused on elastic analysis of closed curves using complex representations of 2D coordinates of curves. Srivastava et al. [SKJJ11] further extended this analysis to include curves in general Euclidean spaces by introducing a novel mathematical representation called the square root velocity functions (SRVFs).

An important advantage of SRVFs was that their usage transformed a complicated elastic Riemannian metric into the more standard $\mathbb{L}^2$ metric. Naturally, it preserved the isometry property of the elastic metric despite simplification. The isometry property is that if any two curves are reparameterized by the same function, then the resulting distance between them under the elastic metric does not change. Thus, one can define reparameterization (and rotation) orbits of given contours as equivalence classes from the perspective of shape analysis, and induce the $\mathbb{L}^2$ norm from the SRVF representation to a quotient space modulo rotation and reparameterization. This leads to a definition of shape distance between any two curves as the distance between the corresponding orbits in the quotient space. The accurate and efficient computation of distance between shapes of two curves is a fundamental operation in elastic shape analysis, upon which many other important tasks depend. This typically involves solving an optimization problem on the joint space of reparameterizations and rotations. In this paper, we focus on this important subproblem in elastic shape analysis and study different tools for optimizations on the joint group. We develop and analyze a novel optimization approach to solving for optimal reparameterizations and rotations between two curves and evaluate its use in computing the distance between two curves and for classification of closed curves in the plane.

This paper is organized as follows. Section 2 presents the Riemannian framework for shape analysis including the definition of the elastic metric for open and closed curves in $\mathbb{R}^n$. Section 3 presents the algorithm of Srivastava et al. [SKJJ11], the approximations upon which it is based and its core dynamic programming algorithm. The proposed Riemannian approach to the solution of the optimization problem that defines the elastic metric evaluation is derived in Section 4, and a detailed discussion of its implementation using Riemannian optimization algorithms follows in Section 5. Empirical evaluation of the relative efficiency and effectiveness of the methods is presented in Section 6, and our conclusions are given in Section 7.

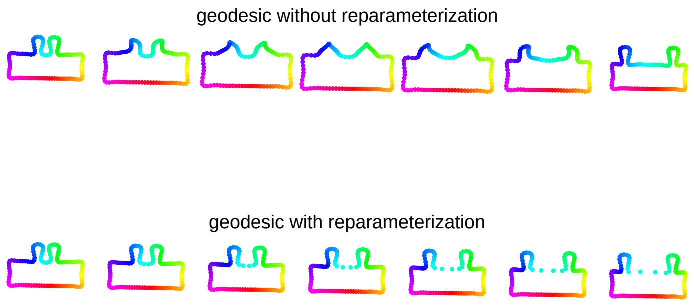
geodesic without reparameterization
geodesic with reparameterization

Figure 1: Geodesics without and with reparameterization are given by the frameworks of landmark-based Kendall's shape analysis [Ken84, DM98] and elastic shape analysis [SKJJ11] respectively.

## 2 Riemannian Framework and Problem Statement

### 2.1 Curve Representation

The derivation of the basic representation of a shape begins with a parameterized curve, i.e., $\beta(t) : \mathbb{D} \to \mathbb{R}^n$, where $\mathbb{D}$ is the domain of the curve — $\mathbb{D} = [0,1]$ for an open curve and $\mathbb{D} = \mathbb{S}^1$, i.e., the unit circle in $\mathbb{R}^2$, for a closed curve — and $\beta$ is a smooth function on $\mathbb{D}$. The shape is taken to be invariant with respect to rescaling, translation, and rotation for inelastic shape analysis, while elastic shape analysis adds invariance with respect to reparameterization. All four invariants must be taken into account when developing a representation that supports efficient and robust computation.

The framework of Srivastava et al. [SKJJ11] uses the square root velocity (SRV) function

$$q(t) = \begin{cases} \frac{\dot{\beta}(t)}{\sqrt{\|\dot{\beta}(t)\|_2}}, & \text{if } \|\dot{\beta}(t)\|_2 \neq 0; \\ 0, & \text{if } \|\dot{\beta}(t)\|_2 = 0. \end{cases}$$

as the basis for elastic analysis of a shape defined by the parameterized curve $\beta(t)$. Observe that $\dot{\beta}(t)$ can be recovered from $q(t)$ by $\dot{\beta}(t) = \|q(t)\|_2 q(t)$. Translation is removed automatically by the use of $\dot{\beta}(t)$ in the definition. Rescaling is removed by the normalization of the length of the curve to 1. Since the length of a curve, $\beta(t)$, is $\int_{\mathbb{D}} \|\dot{\beta}(t)\|_2 dt = \int_{\mathbb{D}} \|q(t)\|_2^2 dt$, the normalization requires that $\int_{\mathbb{D}} \|q(t)\|_2^2 dt = 1$, and the set of all SRV functions is the unit sphere in $\mathbb{L}^2(\mathbb{D}, \mathbb{R}^n)$. This sphere is called the preshape space. For open curves in $\mathbb{R}^n$, the domain is $\mathbb{D} = [0,1]$, and the preshape space

$$l_n^o = \left\{ q \in \mathbb{L}^2([0,1], \mathbb{R}^n) \middle| \int_0^1 \|q(t)\|_2^2 dt = 1 \right\},$$

is the unit sphere of $\mathbb{L}^2([0,1], \mathbb{R}^n)$. For closed curves, the domain is $\mathbb{D} = \mathbb{S}^1$, and the preshape space is

$$l_n^c = \left\{ q \in \mathbb{L}^2(\mathbb{S}^1, \mathbb{R}^n) \middle| \int_{\mathbb{S}^1} \|q(t)\|_2^2 dt = 1, \int_{\mathbb{S}^1} q(t) \|q(t)\|_2 dt = 0 \right\}, \quad (2.1)$$

where $\int_{\mathbb{S}^1} q(t)\|q(t)\|_2 dt = 0$ is the closure condition. Note that the preshape space of closed curves also can be written in a more intuitive expression:

$$l_n^c = \left\{ q \in \mathbb{L}^2([0,1], \mathbb{R}^n) \Big| \int_0^1 \|q(t)\|_2^2 dt = 1, \int_0^1 q(t)\|q(t)\|_2 dt = 0 \right\} \tag{2.2}$$

and the closure condition requires $\beta(0) = \beta(1)$. It is known that $l_n^c$ is a submanifold of $l_n^o$ [SKJJ11, Appendix]. In the later discussions, $l_n^c$ denotes (2.2) rather than (2.1).

Removing rotation and reparameterization is required to define the shape space. This is done by defining an appropriate quotient operation via isometric group actions. This, in turn, defines the distance between curves, the associated optimization problem, and other key tasks such as determining geodesics containing the two curves. Since the approaches taken differ for open and closed curves, they are considered separately below. However, both approaches require the rotation and reparameterization groups, and their actions. In these two definitions, $\Gamma$ and $l_n$ are used to indicate the reparameterization group and preshape space for both open and closed curves.

**Definition 2.1.** *The rotation group for curves in* $\mathbb{R}^n$ *is*

$$\mathrm{SO}(n) = \left\{ O \in \mathbb{R}^{n\times n} | O^T O = I_n, \det(O) = 1 \right\},$$

*and its action is* $\mathrm{SO}(n) \times l_n \to l_n : (O, q) \to Oq$.

**Definition 2.2.** *The reparameterization group for curves in* $\mathbb{R}^n$ *is*

$$\Gamma = \{ \gamma : \mathbb{D} \to \mathbb{D} | \gamma \in \mathcal{D}(\mathbb{D}, \mathbb{D}) \},$$

*and its action is* $l_n \times \Gamma \to l_n : (q, \gamma) \to (q \circ \gamma)\sqrt{\dot{\gamma}}$, *where* $\mathcal{D}(\mathbb{D}, \mathbb{D})$ *is the set of orientation-preserving, absolutely continuous bijections.*

*Specifically, the reparameterization group for open curves* $l_n^o$ *is*

$$\Gamma^o = \left\{ \gamma : [0,1] \to [0,1] | \gamma \text{ is absolutely continuous, } \gamma(0) = 0,\ \gamma(1) = 1, \text{ and } \dot{\gamma}(t) > 0 \text{ almost everywhere.} \right\}.$$

*The reparameterization group for closed curves* $l_n^c$ *is*

$$\Gamma^c = [0,1] \times \Gamma^o,$$

*and its action is therefore* $l_n^c \times \Gamma^c \to l_n^c : (q, (m, \gamma)) \to (\hat{q} \circ \gamma \mod 1))\sqrt{\dot{\gamma}}$, *where* $\hat{q}(t) = (q, m)(t) := q(t + m \mod 1)$.

We denote by $\gamma^{-1}$ the reciprocal (also called inverse) of function $\gamma$. To avoid confusion, we use $\frac{1}{\gamma}$ for the pointwise numerical inverse.

## 2.2 Open Curves in $\mathbb{R}^n$

The preshape space for open curves, $l_n^o$, is a well-known infinite dimensional manifold. The tangent space of $q \in l_n^o$ is

$$\mathrm{T}_q\, l_n^o = \left\{ v \in \mathbb{L}^2([0,1], \mathbb{R}^n) \Big| \int_0^1 q(t)^T v(t) dt = 0 \right\}.$$

The Riemannian metric on $l_n^o$ can be taken as the endowed metric from the embedding space $\mathbb{L}^2([0,1],\mathbb{R}^n)$, i.e.,

$$\langle v_1, v_2\rangle_{l_n^o} = \langle v_1, v_2\rangle_{\mathbb{L}^2} = \int_0^1 v_1(t)^T v_2(t)dt,$$

where $v_1, v_2 \in \mathrm{T}_q\, l_n^o$. The distance function on $l_n^o$ induced by this Riemannian metric is

$$d_{l_n^o}(x,y) = \cos^{-1}\langle x, y\rangle_{\mathbb{L}^2}. \tag{2.3}$$

Observe that $d_{l_n^o}\left((x\circ\gamma)\sqrt{\dot\gamma}, (y\circ\gamma)\sqrt{\dot\gamma}\right) = d_{l_n^o}(x,y)$ for all $\gamma\in\Gamma^o$.

The shape space for open curves is given by the quotient

$$\mathfrak{L}_n^o = \{\overline{[q]} | q\in l_n^o\},$$

where the orbit $\overline{[q]}$ is the closure of the set

$$[q] = \{O(q\circ\gamma)\sqrt{\dot\gamma} | (O,\gamma(t))\in \mathrm{SO}(n)\times\Gamma^o\},$$

and the closure is with respect to the $\mathbb{L}^2$ metric.

Another way to elaborate this is to first introduce a semigroup:

$$\begin{aligned}\Gamma_s^o = \{\gamma : [0,1]\to[0,1] | \gamma(0) = 0, \gamma(1) = 1,\\ \gamma \text{ is an absolutely continuous and non-decreasing function } \}\ .\end{aligned}$$

It is shown in Theorem 2.1 that $\overline{[q]}$ is the orbit of $q$ under the semigroup $\Gamma_s^o$ and rotation group $\mathrm{SO}(n)$ under the assumption that $q^{-1}(0_n)$ has measure zero, where $0_n\in\mathbb{R}^n$ is the zero vector.

**Theorem 2.1.** *Let $[q]_{\mathrm{SO}(n)\times\Gamma_s^o}$ denote the orbit of $q$ under the semigroup $\Gamma_s^o$ and rotation group $\mathrm{SO}(n)$. Then $[q]_{\mathrm{SO}(n)\times\Gamma_s^o}\subseteq\overline{[q]}$. Moreover, if $q^{-1}(0_n)$ has measure zero, then $[q]_{\mathrm{SO}(n)\times\Gamma_s^o} = \overline{[q]}$.*

*Proof.* See the appendix. □

Now we can define a distance between orbits of $\Gamma_s^o$ and $\mathrm{SO}(n)$, $\overline{[q_1]}$ and $\overline{[q_2]}$ as:

$$d_{\mathfrak{L}_n^o}(\overline{[q_1]}, \overline{[q_2]}) = \inf_{\gamma_1,\gamma_2\in\Gamma_s^o, O_1,O_2\in \mathrm{SO}(n)} d_{l_n^o}(O_1(q_1\circ\gamma_1)\sqrt{\dot\gamma_1}, O_2(q_2\circ\gamma_2)\sqrt{\dot\gamma_2})\ .$$

By the definition of closure, elements in $\overline{[q]}$ can be approximated arbitrarily well by elements in $[q]$ with respect to $\mathbb{L}^2$ metric. Therefore, we have for any $\epsilon>0$, there exists a $\gamma^*\in\Gamma^o$ and an $O^*\in\mathrm{SO}(n)$ such that :

$$|d_{\mathfrak{L}_n^o}(\overline{[q_1]}, \overline{[q_2]}) - d_{l_n^o}(q_1, O^*(q_2\circ\gamma^*)\sqrt{\dot\gamma^*})| < \epsilon\ .$$

Note that since $\Gamma^o$ and $\mathrm{SO}(n)$ are isometries, $O$ and $\gamma$ each can be associated with either $q_1$ or $q_2$. While in the definition above both are associated with $q_2$, it is shown below that there are implementation and robustness reasons to associate $O$ with $q_1$ and $\gamma$ with $q_2$.

Our goal is to find such a pair $(O^*,\gamma^*)\in\mathrm{SO}(n)\times\Gamma^o$. Even though this will not be an exact calculation of the shape distance, approximating $d_{\mathfrak{L}_n^o}(\overline{[q_1]}, \overline{[q_2]})$ by

$$d_{l_n^o}(q_1, O(q_2\circ\gamma)\sqrt{\dot\gamma}) = \cos^{-1}\langle q_1, O(q_2\circ\gamma)\sqrt{\dot\gamma}\rangle_{\mathbb{L}^2}, \tag{2.4}$$

evaluated at $(O^*,\gamma^*)$ gives an approximate distance for comparing shapes of curves in practical situations.

## 2.3 Closed Curves in $\mathbb{R}^n$

The preshape space of closed curves, $l_n^c$, is a submanifold of $l_n^o$, and the Riemannian metric inherited from the embedding space is

$$\langle v_1, v_2 \rangle_{l_n^c} = \langle v_1, v_2 \rangle_{\mathbb{L}^2} = \int_{\mathbb{S}^1} v_1(t)^T v_2(t) dt.$$

The shape space for closed curves is

$$\mathfrak{L}_n^c = \{\overline{[q]} | q \in l_n^c\},$$

where the orbit $\overline{[q]}$ is the closure of the set $\{O(q \circ \gamma)\sqrt{\dot{\gamma}} | (O, \gamma(t)) \in \mathrm{SO}(n) \times \Gamma^c\}$.

Proceeding as with open curves, a semigroup $\Gamma_s^c = [0,1] \times \Gamma_s^o$, that is closed under composition, can be defined. It also can be shown in Theorem 2.2 that $\overline{[q]}$ is the orbit of $q$ under the semigroup $\Gamma_s^c$ and rotation group $\mathrm{SO}(n)$.

**Theorem 2.2.** *Let $[q]_{\mathrm{SO}(n)\times\Gamma_s^c}$ denote the orbit of $q$ under the semigroup $\Gamma_s^c = [0,1] \times \Gamma_s^o$ and rotation group $\mathrm{SO}(n)$. Then $[q]_{\mathrm{SO}(n)\times\Gamma_s^o} \subseteq \overline{[q]}$. Moreover, if $q^{-1}(0_n)$ has measure zero and $q$ is absolutely continuous, then $[q]_{\mathrm{SO}(n)\times\Gamma_s^o} = \overline{[q]}$.*

*Proof.* See the appendix. □

The distance between orbits $\overline{[q_1]}$ and $\overline{[q_2]}$ of $q_1$ and $q_2$ under the semigroup $\Gamma_s^c$ is

$$d_{\mathfrak{L}_n^c}(\overline{[q_1]}, \overline{[q_2]}) = \inf_{\gamma_1,\gamma_2 \in \Gamma_s^c, O_1, O_2 \in \mathrm{SO}(n)} d_{l_n^c}(O_1(q_1 \circ \gamma_1)\sqrt{\dot{\gamma}_1}, O_2(q_2 \circ \gamma_2)\sqrt{\dot{\gamma}_2}),$$

and for any $\epsilon > 0$, there exists a $\gamma^* \in \Gamma^c$ and an $O^* \in \mathrm{SO}(n)$ such that:

$$|d_{\mathfrak{L}_n^c}(\overline{[q_1]}, \overline{[q_2]}) - d_{l_n^c}(q_1, O^*(q_2 \circ \gamma^*)\sqrt{\dot{\gamma}^*})| < \epsilon. \tag{2.5}$$

Unlike the case of open curves, there is no known analytical expression of distance on $l_n^c$. Since $l_n^c$ is a submanifold of $l_n^o$, $d_{l_n^o}(q_1, q_2)$ is the extrinsic distance of $q_1, q_2 \in l_n^c$. The approximation

$$\begin{aligned} &\inf_{\gamma_1,\gamma_2 \in \Gamma_s^c, O \in \mathrm{SO}(n)} d_{l_n^c}((q_1 \circ \gamma_1)\sqrt{\dot{\gamma}_1}, O(q_2 \circ \gamma_2)\sqrt{\dot{\gamma}_2}) \\ \approx &\inf_{\gamma_1,\gamma_2 \in \Gamma_s^c, O \in \mathrm{SO}(n)} d_{l_n^o}((q_1 \circ \gamma_1)\sqrt{\dot{\gamma}_1}, O(q_2 \circ \gamma_2)\sqrt{\dot{\gamma}_2}) \end{aligned}$$

is used (see Section 6.4). (In fact, we could have defined $d_{\mathfrak{L}_n^c}(\overline{[q_1]}, \overline{[q_2]}) = \inf_{\gamma_1,\gamma_2 \in \Gamma_s^c, O \in \mathrm{SO}(n)} d_{l_n^o}((q_1 \circ \gamma_1)\sqrt{\dot{\gamma}_1}, O(q_2 \circ \gamma_2)\sqrt{\dot{\gamma}_2})$ right away.) As with open curves, approximating $d_{\mathfrak{L}_n^c}(\overline{[q_1]}, \overline{[q_2]})$ with the extrinsic distance

$$d_{l_n^o}(q_1, O(q_2 \circ \gamma)\sqrt{\dot{\gamma}}) = \cos^{-1} \langle q_1, O(q_2 \circ \gamma)\sqrt{\dot{\gamma}} \rangle_{\mathbb{L}^2} \tag{2.6}$$

evaluated at $(O^*, \gamma^*)$ gives an approximate distance for comparing shapes of curves in practical situations.

## 3 The Coordinate Descent Method

The discussion in Sections 2.2 and 2.3 characterizes the reparameterization problem from the Riemannian manifold point of view but does not suggest an algorithm. Sebastian et al. [SKK03] define an *edit distance* to characterize differences between shapes and develop an algorithm for closed curves with computational complexity $O(N^2 \log N)$, where $N$ is the number of points for representing a curve. It requires the cost function to be invariant to rotation which is clearly not the case for all the cost functions discussed above, i.e., $d_{l_n^o}(q_1, O(q_2 \circ \gamma)\sqrt{\dot\gamma})$ for $O \in \mathrm{SO}(n), \gamma \in \Gamma^o$ or $\gamma \in \Gamma^c$, and their variations (3.1) and (3.3) discussed later.

Srivastava et al. [SKJJ11] developed a method for finding $(\gamma^*, O^*)$ for open and closed curves based on the idea of alternately optimizing on $\mathrm{SO}(n)$ and $\Gamma^o$ or $\Gamma^c$, i.e., a generalized Coordinate Descent method. The simpler open curve problem and algorithm are discussed first followed by the adaptation to closed curves.

### 3.1 The Basic Ingredients

For open curves Srivastava et al. [SKJJ11] use the cost function

$$H^o(O, \gamma) = \int_0^1 \|q_1(t) - O(q_2 \circ \gamma(t))\sqrt{\dot\gamma(t)}\|_2^2 dt, \tag{3.1}$$

that has the same extreme points as the cost function used in (2.4). This is easily seen from

$$\begin{aligned}\int_0^1 \|q_1(t) - O(q_2 \circ \gamma(t))\sqrt{\dot\gamma(t)}\|_2^2 dt =& \langle q_1, q_1\rangle_{\mathbb{L}^2} + \langle q_2, q_2\rangle_{\mathbb{L}^2} - 2\langle q_1, O(q_2 \circ \gamma)\sqrt{\dot\gamma}\rangle_{\mathbb{L}^2} \\ =& 2 - 2\cos(\cos^{-1}(\langle q_1, O(q_2 \circ \gamma)\sqrt{\dot\gamma}\rangle_{\mathbb{L}^2})) \\ =& 2 - 2\cos(d_{l_n^o}(q_1, O(q_2 \circ \gamma)\sqrt{\dot\gamma})).\end{aligned}$$

Note that the cost function (3.1) is not applied to the curve itself but to its corresponding $q$ function. They propose a variant of the general Coordinate Descent method approach given in Algorithm 1.

**Algorithm 1** Coordinate Descent Algorithm for $H^o(O, \gamma)$

```
Input: Initial $\gamma_0$;
1: $k = 0$;
2: Find $O_{k+1} = \arg\min_O H^o(O, \gamma_k)$ using the SVD;
3: Find $\gamma_{k+1} \approx \arg\min_\gamma H^o(O_{k+1}, \gamma)$ using dynamic programming;
4: If termination criterion is satisfied, stop, otherwise, $k = k + 1$ and go to step 2.
```

The minimizer $O_{k+1}$ of $H^o(O, \gamma_k)$ is $O_{k+1} = UV^T$, where $USV^T$ is the singular value decomposition (SVD) of $A = \int_0^1 q_1(t)\tilde q_2(t)^T dt$ and $\tilde q_2(t) = (q_2 \circ \gamma_k(t))\sqrt{\dot\gamma_k(t)}$. The SVD of a generic dense matrix $A \in \mathbb{R}^{n\times n}$ is well-understood and can be computed reliably and efficiently using well-known numerical linear algebra techniques for $n$ up to several hundred, i.e., much larger than typically required for typical shape analysis problems. This is common to both open and closed curve problems. To find approximately the minimizer $\gamma_{k+1}$ of $H^o(O_k, \gamma)$ for open curves, Srivastava et al. [SKJJ11] use dynamic programming (DP) [Ber95].

The approximation arises for this problem because DP works on a grid in $[0,1]\times[0,1]$ rather than the continuous space $\Gamma^o$. Srivastava et al. use $G_N \times G_N$ where $G_N = \{0/N, 1/N, \dots, (N-1)/N, 1\}$. Here, we consider a more general grid, $G_N \times \tilde{G}_N$, where $\tilde{G}_N = \{g_0, g_1, \dots, g_N\}$ is not necessarily uniformly-spaced, $g_0 = 0$, $g_N = 1$, and $g_i < g_j$ if $i < j$. On $G_N \times \tilde{G}_N$, DP uses a partial cost function

$$E(s,t;\gamma) = \int_s^t \|q_1(\tau) - O(q_2 \circ \gamma(\tau))\sqrt{\dot{\gamma}(\tau)}\|_2^2 d\tau$$

and determines a piecewise linear path defined by connecting points in $G_N \times \tilde{G}_N$ moving to the right and up, i.e., $(0,0) = (i_0, j_0) < (i_1, j_1) < (i_2, j_2) < \dots < (i_m, j_m) = (1,1)$ that minimize the cost

$$\sum_{r=0}^{m-1} E(i_r, i_{r+1}; L(i_r, j_r; i_{r+1}, j_{r+1})),$$

where $L(i_r, j_r; i_{r+1}, j_{r+1})$ is the linear function passing though $(i_r, j_r)$ and $(i_{r+1}, j_{r+1})$.

DP uses induction to construct a minimal path. Suppose $S \subseteq G_N \times \tilde{G}_N$ is such that for any $(p,q) \in S$ the globally minimizing path $\gamma^*_{(p,q)}$ from $(0,0)$ to $(p,q)$, and the associated cost function value $W(p,q)$ are known. Let $U_{i,j} := \left\{(p,q) \in G_N \times \tilde{G}_N | 0 \le p < i, 0 \le q < j\right\}$. When $U_{i,j} \subseteq S$, the basic DP step adds $(i,j)$ to $S$ by computing $\gamma^*_{(i,j)}$, the globally minimizing path on $G_N \times \tilde{G}_N$ from $(0,0)$ to $(i,j)$, and the associated cost function value $W(i,j)$. This is done by considering each $(p,q) \in U_{i,j}$, adding the edge between $(p,q)$ and $(i,j)$ to the path $\gamma^*_{(p,q)}$ and determining its cost. Formally, determining $W(i,j)$ and $\gamma^*_{(i,j)}$ is solving

$$\min_{(k,l) \in U_{i,j}} E(k,i; L(k,l;i,j)) + W(k,l), \text{ with } W(0,0) = 0.$$

Eventually, $S = G_N \times \tilde{G}_N$ and a path with minimal cost on $S \subset \Gamma^o$ is given by $\gamma^*_{(1,1)}$.

The complexity of DP as described above is $O(N^5)$ and too high for practical problems. To reduce the complexity, the set $U_{i,j}$ is constrained to

$$\mathcal{N}_{i,j} = \{(k,l) | \max(i-h, 0) \le k < i, \max(j-h, 0) \le l < j\} \subset U_{i,j} \tag{3.2}$$

for some $h$. When the grid $\tilde{G}_N$ is chosen to be uniform, i.e. $\tilde{G}_N = G_N$, the set $\mathcal{N}_{i,j}$ can be further reduced by removing some repeated slopes, e.g., $(i-2, j-2)$ is deleted because $(i-1, j-1)$ exists. Using the set $\mathcal{N}_{i,j}$ rather than $U_{i,j}$ reduces the complexity of DP to $O(N^2)$. However, since the number of slopes considered when adding $(i,j)$ to $S$ is constrained, the minimizer may change and may no longer be a global minimizer on $G_N \times \tilde{G}_N$.

The quality of $\gamma^*_{(1,1)}$ compared to a global minimizer, $\tilde{\gamma}^*_{(1,1)}$ on $\Gamma^o$ is not known analytically nor is the potential further degradation in quality compared to $\tilde{\gamma}^*_{(1,1)}$ that results in replacing $U_{i,j}$ with $\mathcal{N}_{i,j}$.

The cost function defined on $\mathrm{SO}(n) \times \Gamma^c$ for closed curves is

$$H^c(O,\gamma) = \int_{\mathbb{S}^1} \|q_1(t) - O(q_2 \circ \gamma(t))\sqrt{\dot{\gamma}(t)}\|_2^2 dt. \tag{3.3}$$

A DP-based Coordinate Descent algorithm cannot be applied to $H^c(O,\gamma)$ directly since DP requires a grid of a domain that is the cross product of two intervals, e.g., $[0,1] \times [0,1]$ rather than $\mathbb{S}^1 \times \mathbb{S}^1$.

Srivastava et al. [SKJJ11] solve this by applying the open curve DP-based algorithm to a set of open curves, $\{\tilde{\beta}^{(i)},\ 1 \leq i \leq w\}$ derived from the closed curve $\beta$ using $w$ break points, $t_i$, $1 \leq i \leq w$, i.e.,

$$\tilde{\beta}^{(i)}(t) = \begin{cases} \beta(t + t_i), & \text{if } 0 \leq t \leq 1 - t_i; \\ \beta(t - (1 - t_i)), & \text{if } 1 - t_i < t \leq 1. \end{cases}$$

The open curve DP algorithm using $H^o(O, \gamma)$ is applied to each open curve $\tilde{\beta}^{(i)}$ to determine $\gamma^{(i)}$. A $\gamma^{(i)}$ with minimal cost is chosen as the closed curve reparameterization. Since DP is run $w$ times, the complexity for this closed curve algorithm is $O(wN^2)$, and $w$ is usually proportional to $N$, e.g., every second or third point is used as a break point, yielding $O(N^3)$ complexity. Note that this complexity is for only one run of DP in Algorithm 1. A key consideration for closed curve reparameterization is therefore computational complexity versus quality of $\gamma$.

## 3.2 Uniform Grid Coordinate Descent and Some Difficulties

The use of DP on a grid to solve approximately the optimization problem implies that $\gamma$ is represented by a sequence of scalars such that the $i$-th scalar is $\gamma$ at $(i-1)/N$. The curves $\beta_1$ and $\beta_2$ are also represented discretely by a sequence of points in $\mathbb{R}^n$, and values at points other than the discrete set are recovered using an interpolatory parameterized polynomial, e.g., an interpolatory spline of degree 1, 2 or 3. The theoretical descriptions of the optimization algorithms for open and closed curves assume that the operations of rotation and reparameterization preserve the shape of the curves. It is important to maintain this invariant in the context of the discrete representations of $\gamma$, $\beta_1$ and $\beta_2$.

In this section, we consider a uniform grid on both $\beta_1$ and $\beta_2$, and Algorithm 2 and Algorithm 3 are two discrete representation versions of the Coordinate Descent algorithm applied to closed curves based on the cost function (3.3). The open curve discrete versions are easily derived from either. The differences between Algorithm 2 and Algorithm 3 are specifically designed to highlight some crucial implementation decisions and the problems that arise in both implementations. These problems are all overcome by the new Riemannian algorithms we propose in Section 4. A concrete example that illustrates the difficulties of Algorithm 2 and Algorithm 3 is given in Section 6.3.

Note that cost function (3.3) applies the reparameterization, $\gamma$ to $\beta_2$. Also note that in Step 10 of Algorithm 2 interpolation is used when evaluating the reparameterized curve $\beta_2 \circ \gamma$. This implies that the vector of discrete points in $\mathbb{R}^n$ used to represent $\beta_2$ is updated by each reparameterization. If, equivalently from an optimization point of view, $\gamma$ is associated with $\beta_1$ then the vector representation of $\beta_1$ changes. Therefore, when multiple iterations of Coordinate Descent are performed, a problem arises. Since the points upon which the interpolatory parameterized polynomial is based change, the parameterized polynomial changes, and therefore the shape of the curve changes with each reparameterization.

Algorithm 3 overcomes this difficulty by representing $\beta_2$ as a continuous function determined by the interpolatory parameterized polynomial (Step 1) and maintaining it throughout the algorithm. Algorithm 3, however, has a problem that is not seen in Algorithm 2. In Step 11, the expression

$$\bar{\beta}_2^{(\min,k)} = O^{(\min,k)} O_*^{(k)} \left( (\beta_2 \circ \gamma_*^{(k)}) \circ ((\gamma^{(\min,k)} + b_{\min}^{(k)}/N) \mod 1) \right) \tag{3.4}$$

is implicitly used when $H^{(\min,k)} = H^c(O_*^{(k)}, \gamma_*^{(k)})$ is computed and is then explicitly used to compute $\bar{\beta}_2^{(\min,k)}$. The curves $\bar{\beta}_2^{(\min,k)}$ and $\beta_2^{(k+1)}$ are, in theory, the same. However, on the next iteration,

$k+1$, the curve $\beta_2^{(k+1)}$ is explicitly computed using the composition

$$\beta_2^{(k+1)} = O^{(\min,k)} O_*^{(k)} \left( \beta_2 \circ (\gamma_*^{(k)} \circ ((\gamma^{(\min,k)} + b_{\min}^{(k)}/N) \mod 1)) \right). \tag{3.5}$$

Note that associativity has been applied in the composition of functions. This is required given that the interpolatory parameterized polynomial representing $\beta_2$ is maintained for all iterations. The change of order does not matter theoretically in the continuous form but the curves are different in the discrete case. If the cost function value $H^{(\min,k)}$ was computed using the order of composition in (3.5), it may yield a different value than the cost function value used during iteration $k$ to update $\beta_2$ mentioned above. In fact, the cost function value associated with the form (3.5) implicit in iteration $k+1$ may be larger than the cost function value actually computed in iteration $k$ using (3.4). Therefore, we may compute a $\beta_2^{(k+1)}$ that does not decrease the cost function value in practice.

**Algorithm 2** Coordinate Descent Algorithm for $H^c(O,\gamma)$

```
Input: Two closed curves $\beta_1 = \{(\frac{0}{N}, v_0), (\frac{1}{N}, v_1), \dots, (\frac{N-1}{N}, v_{N-1}), (\frac{N}{N}, v_0)\}$ and $\beta_2 = \{(\frac{0}{N}, u_0^{(0)}), (\frac{1}{N}, u_1^{(0)}), \dots, (\frac{N-1}{N}, u_{N-1}^{(0)}), (\frac{N}{N}, u_0^{(0)})\}$ where $u_i^{(0)}, v_i \in \mathbb{R}^n$; a set of break indices $\{b_1, b_2, \dots, b_w\}$;
1: $k = 0$;
2: for $i = 1, 2, \dots, w$ do
3:     Shift $\beta_2^{(k)}$ and get $\tilde{\beta}_2^{(i,k)} = \{(\frac{0}{N}, u_{b_i}^{(k)}), (\frac{1}{N}, u_{b_i+1}^{(k)}), \dots,$
       $(\frac{N-1-b_i}{N}, u_{N-1}^{(k)}), (\frac{N-b_i}{N}, u_0^{(k)}), \dots, (\frac{N}{N}, u_{b_i}^{(k)})\}$;
4:     Compute the rotation $O^{(i,k)}$ based on $\beta_1$ and $\tilde{\beta}_2^{(i,k)}$;
5:     Set $\bar{\beta}_2 = O^{(i,k)} \tilde{\beta}_2^{(i,k)}$;
6:     Compute $\gamma^{(i,k)}$ for $\beta_1$ and $\bar{\beta}_2$ by DP;
7:     Compute cost function $H^{(i,k)}$
8: end for
9: Find $H^{(\min,k)} = \min_{1 \le i \le w} \{H^{(i,k)}\}$ and get the corresponding $O^{(\min,k)}$, $\gamma^{(\min,k)}$ and $\tilde{\beta}_2^{(\min,k)}$;
10: Interpolate $\tilde{\beta}_2^{(\min,k)}$ by a function $F$ (e.g., cubic spline interpolation) and set $\beta_2^{(k+1)} = \{(\frac{0}{N}, F(\gamma^{(\min,k)}(\frac{0}{N}))), (\frac{1}{N}, F(\gamma^{(\min,k)}(\frac{1}{N}))), \dots,$
    $(\frac{N}{N}, F(\gamma^{(\min,k)}(\frac{N}{N})))\}$
11: If a stopping criterion is satisfied, then stop, otherwise $k = k+1$ and goto step 2;
```

## 3.3 Nonuniform Grid Coordinate Descent

The problem in Algorithm 3 with associativity can be addressed by using a nonuniform grid that keeps track of the reassigned parameter values of the $u_i$'s. The details can be found in Algorithm 4.

The example shown in Figures 2 and 3 is used to understand the idea in Algorithm 4. As shown in Figure 2, the $\gamma^{(0)}$ given by DP assigns parameter values to points on $\beta_2$. Therefore, a nonuniform grid on $\beta_2$ can be used to absorb the effect of $\gamma^{(0)}$. In the next iteration as shown in Figure 3, one finds another $\gamma^{(1)}$ which assigns new parameter values to points on $\beta_2$. The accumulated $\gamma_*^{(2)}$ shown in Figure 3 is exactly defined by the values of $(\gamma_*^{(2)})^{-1}$ at $i/N$. In general, the accumulated $\gamma_*^{(k)}$ can be exactly defined by the values of $(\gamma_*^{(k)})^{-1}$ at $i/N$. Therefore, the associativity difficulty

**Algorithm 3** Coordinate Descent Algorithm for $H^c(O, \gamma)$

```
Input: Two closed curves $\beta_1 = \{(\frac{0}{N}, v_0), (\frac{1}{N}, v_1), \dots, (\frac{N-1}{N}, v_{N-1}), (\frac{N}{N}, v_0)\}$ and $\beta_2^{(0)} = \{(\frac{0}{N}, u_0), (\frac{1}{N}, u_1), \dots, (\frac{N-1}{N}, u_{N-1}), (\frac{N}{N}, u_0)\}$ where $u_i, v_i \in \mathbb{R}^n$; a set of break indices $\{b_1, b_2, \dots, b_w\}$; initial $\gamma_*^{(0)} = \{0, 1/N, \dots, 1\}$; $O_*^{(0)} = I_n$;
1: Compute interpolation function $F_{\beta_2}$ for $\beta_2$, e.g., a spline cubic function;
2: $k = 0$;
3: Compute $\beta_2^{(k)}$ by evaluating $F_{\beta_2}$ at $\gamma_*^{(k)}$ and left multiplying by $O_*^{(k)}$;
4: for $i = 1, 2, \dots, w$ do
5:    Shift $\beta_2^{(k)}$ and get $\tilde{\beta}_2^{(i,k)} = \{(\frac{0}{N}, u_{b_i}^{(k)}), (\frac{1}{N}, u_{b_i+1}^{(k)}), \dots, (\frac{N-1-b_i}{N}, u_{N-1}^{(k)}), (\frac{N-b_i}{N}, u_0^{(k)}), \dots, (\frac{N}{N}, u_{b_i}^{(k)})\}$;
6:    Compute the rotation $O^{(i,k)}$ based on $\beta_1$ and $\tilde{\beta}_2^{(i,k)}$;
7:    Set $\bar{\beta}_2 = O^{(i,k)} \tilde{\beta}_2^{(i,k)}$;
8:    Compute $\gamma^{(i,k)}$ for $\beta_1$ and $\bar{\beta}_2$ by DP;
9:    Compute cost function $H^{(i,k)}$;
10: end for
11: Find $H^{(\min,k)} = \min_{1 \le i \le w}\{H^{(i,k)}\}$ and get the corresponding $O^{(\min,k)}$, $\gamma^{(\min,k)}$, $\bar{\beta}_2^{(\min,k)}$ and the shift $b_{\min}^{(k)}$;
12: Set $O_*^{(k+1)} = O^{(\min,k)} O_*^{(k)}$;
13: Interpolate points $\gamma_*^{(k)}$ to get a function, e.g., spline function and evaluate the function at
    $$(\gamma^{(\min,k)} + b_{\min}^{(k)}/N) \mod 1 = \begin{pmatrix} (\gamma^{(\min,k)}(0) + b_{\min}^{(k)}/N) \mod 1 \\ (\gamma^{(\min,k)}(1/N) + b_{\min}^{(k)}/N) \mod 1 \\ \vdots \\ (\gamma^{(\min,k)}(1) + b_{\min}^{(k)}/N) \mod 1 \end{pmatrix}$$
    to get $\gamma_*^{(k+1)}$; (this is the implementation of $\gamma_*^{(k+1)} = \gamma_*^{(k)} \circ \gamma^{(\min,k)}$);
14: If a stopping criterion is satisfied, then stop, otherwise $k = k + 1$ and goto step 3;
```

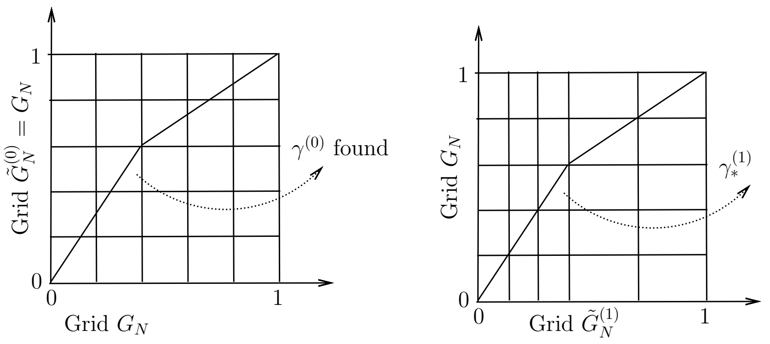

Figure 2: DP in the first iteration of Algorithm 4. Observe that, as annouced in Step 14 of Algorithm 4, $\gamma_*^{(1)}$ can be recovered as the piecewise-linear function that connects the ascending-diagonal points of the grid $\tilde{G}_N^{(1)} \times G_N$.

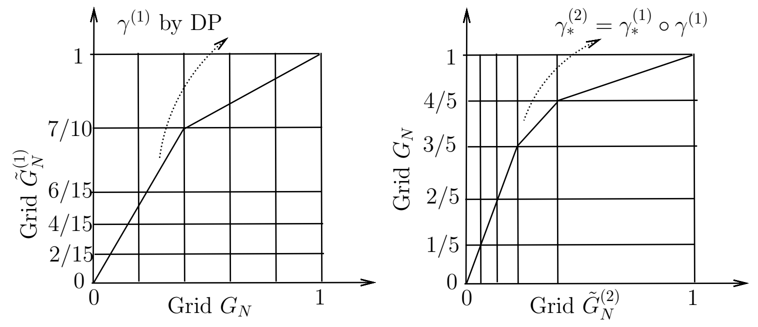

Figure 3: DP in the second iterate of Algorithm 4. Observe again that, as announced in Step 14 of Algorithm 4, $\gamma_*^{(2)}$ can be recovered as the piecewise-linear function that connects the ascending-diagonal points of the grid $\tilde{G}_N^{(2)} \times G_N$.

is avoided. Even though Algorithm 4 avoids the problems in Algorithms 2 and 3, and therefore can reliably perform more than one iteration to determine the reparameterization, this improves only the quality of the distance approximation. The computational complexity required can be substantial and limits the size of problem that can be handled. Specific examples are given in Section 6.9 to demonstrate that Algorithm 4 avoids the difficulties of Algorithms 2 and 3 to produce a more accurate distance estimation.

The experiments in Srivastava et. al. [SKJJ11] simplify the optimization considerably by using only a single iteration of the Coordinate Descent algorithm, denoted CD1. Algorithms 2, 3 and 4 are then identical and avoid both of these problems. If a more accurate optimization is demanded therefore requiring more iterations, as done in Section 6, problems ensue. Note that these problems are not the result of using DP to approximate the optimization problem. Rather, they arise from updating $\gamma$ by composition in the Coordinate Descent approach. The new Riemannian algorithm discussed in Section 4 avoids these difficulties.

**Algorithm 4** Coordinate Descent Algorithm for $H^c(O,\gamma)$

```
Input: Two closed curves $\beta_1$ and $\beta_2$; a set of break indices $\{b_1, b_2, \ldots, b_w\}$; $\tilde{G}_N^{(0)} = G_N$;
1: $k = 0$ and $\gamma_*^{(0)} = \mathrm{id}$;
2: Compute $q$ functions, $q_1 = \{(\frac{0}{N}, v_0), (\frac{1}{N}, v_1), \ldots, (\frac{N-1}{N}, v_{N-1}), (\frac{N}{N}, v_0)\}$ and $q_2 = \{(\frac{0}{N}, u_0), (\frac{1}{N}, u_1), \ldots, (\frac{N-1}{N}, u_{N-1}), (\frac{N}{N}, u_0)\}$, for $\beta_1$ and $\beta_2$, where $u_i, v_i \in \mathbb{R}^n$.
3: Interpolate $q_2$ by a function $F_{q_2}$ (e.g., cubic spline interpolation)
4: for $i = 1, 2, \ldots, w$ do
5:     Shift $\tilde{G}_N^{(k)}$ such that the grid is from 0 to 1 to get
       $\tilde{G}_N^{(i,k)} = \{g_{b_i}^{(k)} - g_{b_i}^{(k)}, g_{b_i+1}^{(k)} - g_{b_i}^{(k)}, \ldots, g_N^{(k)} - g_{b_i}^{(k)}, g_1^{(k)} + 1 - g_{b_i}^{(k)}, \ldots, g_{b_i}^{(k)} + 1 - g_{b_i}^{(k)}\}$;
6:     Shift $q_2$ to get $q_2^{(i,k)} = \{(g_0^{(i,k)}, u_{b_i}^{(k)}), (g_1^{(i,k)}, u_{b_i+1}^{(k)}), \ldots, (g_{N-1-b_i}^{(i,k)}, u_{N-1}^{(k)}), (g_{N-b_i}^{(i,k)}, u_0^{(k)}), \ldots, (g_N^{(i,k)}, u_{b_i}^{(k)})\}$, where $g_m^{(i,k)}$ denote $m$-th point in $\tilde{G}_N^{(i,k)}$;
7:     Resample $q_2$ to get $\hat{q}_2^{(i,k)} = F_{q_2}(\gamma_*^{(k)}(t) + \frac{b_i}{N}\mathrm{mod}1)|_{t \in G_N}$;
8:     Compute the rotation $O^{(i,k)}$ based on $q_1$ and $\hat{q}_2^{(i,k)}$;
9:     Set $\bar{q}_1^{(i,k)} = (O^{(i,k)})^T q_1^{(i,k)}$;
10:    Get a piecewise linear $\gamma^{(i,k)}$ for $\bar{q}_1^{(i,k)}$ and $q_2^{(i,k)}$ by applying DP on the grid $G_N \times \tilde{G}_N^{(i,k)}$, where $L(a, b; c, d)$ in DP is now the linear function passing through $(a/N, g_b^{(i,k)})$ and $(c/N, g_d^{(i,k)})$. Evaluate the corresponding $H^{(i,k)}$;
11: end for
12: Find $H^{(\min,k)} = \min_{1 \le i \le w}\{H^{(i,k)}\}$ and get the corresponding $\gamma^{(\min,k)}$, and $\tilde{G}_N^{(\min,k)}$;
13: Compute the new grid $\tilde{G}_N^{(k+1)}$ by formula $g_i^{(k+1)} = \kappa^{-1}(g_i^{(\min,k)})$, where $\kappa = \gamma^{(\min,k)}$ and $g_m^{(\min,k)}$ is $m$-th point in $\tilde{G}_N^{(\min,k)}$;
14: $\gamma_*^{(k+1)}$ is the piecewise linear function passing through $(g_i^{(k+1)}, i/N)$ shifted so that $\gamma_*^{(k+1)}(0) = \gamma_*^{(k)}(0) + b_{\min}/N$;
15: If a stopping criterion is satisfied, then stop, otherwise $k = k + 1$ and goto step 4;
```

# 4 A Riemannian Optimization Method

Riemannian optimization concerns minimizing (or maximizing) a real-valued function, termed the *cost function*, defined on a Riemannian manifold $\mathcal{M}$. Recent theoretical and algorithmic results and a review of the state-of-the-art can be found in [Bak08, Hua13, HAG15, Qi11, RW12]. In order to make use of Riemannian optimization theory and algorithms in the fundamental elastic shape analysis task of efficiently and effectively computing the distance between two curves, we must define an appropriate cost function on a Riemannian manifold, the Riemannian gradient of the cost function, the tangent space of an element in the manifold, the retraction operation on the manifold, and an appropriate vector transport. The definitions of Riemannian gradient, tangent space, retraction and vector transport are standard and can be found in [O'N83, AMS08].

Several Riemannian optimization algorithms are applicable to distance computation. A representative set is investigated and compared to the DP-based approach of Srivastava et al. for closed curves in this and the following section. Specifically, the chosen state-of-the-art Riemannian algorithms, Riemannian quasi-Newton algorithms given in [Hua13, HAG15, HGA15] including RBFGS, LRBFGS, RTR-SR1, LRTR-SR1 and RSD (see full names in Section 6.1), are applied to the distance problem, and it is shown that a Riemannian approach is more efficient computationally and produces a superior distance computation than the DP-based approach.

## 4.1 Cost Function

Using the Riemannian approach we can handle the closed curve distance problem directly, i.e., breaking the curve into several open curves and taking the minimal solution is avoided. The first step in defining the cost function and associated Riemannian manifold requires reconsidering the representation of $\Gamma^c$ for closed curves

$$\Gamma^c = [0,1] \times \Gamma^o.$$

Note that the interval $[0,1]$ in the group definition removes the need for break points since the offset has been added as a decision variable.

The cost function on the Riemannian manifold $\mathrm{SO}(n) \times \mathbb{R} \times \Gamma^o$ is

$$H(O,m,\gamma) = \int_0^1 \|q_1(t) - O(q_2(\gamma(t) + m \mod 1))\sqrt{\dot\gamma(t)}\|_2^2 dt, \tag{4.1}$$

where we use $\mathbb{R}$ to replace $[0,1]$ due to the use of the $\mod$ operator.

Note that any $\gamma \in \Gamma^o$ and its derivative $\dot\gamma$ satisfy the constraints $\gamma(0) = 0, \gamma(1) = 1$ and $\dot\gamma(s) > 0$ almost everywhere. These are equivalent to $\gamma(0) = 0$, $\int_0^1 \dot\gamma(t)dt = 1$ and $\dot\gamma(s) > 0$ almost everywhere. The positivity constraint on the derivative can be guaranteed by replacing $\dot\gamma$ with an even power function. Let $l^2 = \dot\gamma$, where $l \in \mathbb{L}^2([0,1],\mathbb{R})$. All three constraints are condensed into the constraints $\int_0^1 l^2(t)dt = 1$ and $l^2(s) \neq 0$ almost everywhere. Therefore, $l$ is an element of the sphere, i.e.,

$$l \in \mathcal{L} = \left\{ l \in \mathbb{L}^2([0,1],\mathbb{R}) \Big| \int_0^1 l^2(t)dt = 1 \right\},$$

and $\gamma$ can be recovered by $\int_0^t l^2(s)ds$. It follows that $\sqrt{\dot\gamma} = |l|$ and the cost function becomes

$$\tilde H(O,m,l) = \int_0^1 \left\| q_1(t) - Oq_2\left(\int_0^t l^2(s)ds + m \mod 1\right)|l(t)| \right\|_2^2 dt. \tag{4.2}$$

In order to guarantee $l^2(s) \neq 0$ almost everywhere, we use a barrier function

$$B(\gamma) = \int_0^1 \left(\dot{\gamma}(t) + \frac{1}{\dot{\gamma}(t)}\right) \sqrt{1 + \dot{\gamma}^2(t)} dt = \int_0^1 \left(l^2(t) + \frac{1}{l^2(t)}\right) \sqrt{1 + l^4(t)} dt. \quad (4.3)$$

When some region of $\gamma$ is close to horizontal or vertical, $B(\gamma)$ increases and $\gamma^*$ does not have such a region. Observe that this specific choice of $B$ ensures that $B(\kappa) = B(\gamma)$ since we have

$$\begin{aligned} B(\gamma) &= \int_0^1 \left(\dot{\gamma}(t) + \frac{1}{\dot{\gamma}(t)}\right) \sqrt{1 + \dot{\gamma}^2(t)} dt \\ &= \int_0^1 \left(\dot{\kappa}(\gamma(t)) + \frac{1}{\dot{\kappa}(\gamma(t))}\right) \sqrt{1 + \frac{1}{\dot{\kappa}^2(\gamma(t))}} dt \quad (\text{since } \dot{\gamma}(t) = \tfrac{1}{\dot{\kappa}(\gamma(t))}) \\ &= \int_0^1 \left(\dot{\kappa}(s) + \frac{1}{\dot{\kappa}(s)}\right) \sqrt{1 + \frac{1}{\dot{\kappa}^2(s)}} d\kappa(s) \quad (\text{substitution } \gamma(t) = s) \\ &= B(\kappa), \end{aligned}$$

where $\kappa = \gamma^{-1}$. Thus that (4.4) below remains symmetric with respect to $q_1$ and $q_2$.

Note that if the first iterate in an algorithm satisfies $l_0(t) > 0$, then choosing a suitable step size and using the barrier function keep all iterates $l_i(t) > 0$. Therefore the absolute sign in (4.2) can be ignored. Exploiting the invariance of the norm under rotation, we obtain the final cost function

$$L(O, m, l) = \int_0^1 \left\| O q_1(t) - q_2 \left( \int_0^t l^2(s) ds + m \mod 1 \right) l(t) \right\|_2^2 dt + \omega B(\gamma), \quad (4.4)$$

where $\omega$ is a constant that makes the extra term relatively small. The reason we put $O$ on $q_1(t)$ will be discussed in Section 5.1. Note that the smaller $\omega$ is, the better (4.4) approximates (3.2). However, it is more likely that the solution obtained by Riemannian algorithms contains a region close to being horizontal or vertical.

## 4.2 The Riemannian Manifold

The Riemannian manifold used to define the constraints for the optimization problem associated with the efficient algorithm to compute the distance function for elastic shape analysis is $\mathrm{SO}(n) \times \mathbb{R} \times \mathcal{L}$. The Riemannian gradient of the cost function, the retraction operation on the manifold, and an appropriate vector transport can be constructed by considering each on the components of the product [Hua13, §9.4].

$\mathrm{SO}(n)$ is a well-known Riemannian manifold the structure of which is discussed in the literature [AMS08], and the associated implementation issues are considered in [Hua13, §10.5]. The required Riemannian objects are given in this section, and their derivations can be found in the Appendix.

$\mathcal{L}$ is an infinite dimensional Riemannian manifold. The tangent space of $\mathcal{L}$ at any point is therefore an infinite dimensional linear space with elements that are functions defined on $[0, 1]$. It is well-known that the tangent space $T_l\mathcal{L}$ of $l \in \mathcal{L}$ is

$$\mathrm{T}_l\, \mathcal{L} = \{v \in \mathbb{L}^2([0, 1], \mathbb{R}) | \langle l, v\rangle_{\mathbb{L}^2} = 0\},$$

and the projection onto the tangent space is

$$P_l(v) = v - l \frac{\langle v, l\rangle_{\mathbb{L}^2}}{\langle l, l\rangle_{\mathbb{L}^2}}.$$

Let the metric of $\mathcal{L}$ be endowed from the embedding space $\mathbb{L}^2([0,1],\mathbb{R})$. Retraction is used in updating iterates in a Riemannian algorithm. Vector transport is used in comparing tangent vectors in different tangent spaces and plays an important role in quasi-Newton methods. Specifically, a retraction $R$ is a smooth mapping from the tangent bundle $\mathrm{T}\,\mathcal{M}$ onto $\mathcal{M}$ such that (i) $R(0_x) = x$ for all $x \in \mathcal{M}$ (where $0_x$ denotes the origin of $\mathrm{T}_x\,\mathcal{M}$) and (ii) $\frac{d}{dt}R(t\xi_x)|_{t=0} = \xi_x$ for all $\xi_x \in \mathrm{T}_x\,\mathcal{M}$. The restriction of $R$ to $\mathrm{T}_x\,\mathcal{M}$ is denoted by $R_x$. A vector transport $\mathcal{T} : \mathrm{T}\,\mathcal{M} \oplus \mathrm{T}\,\mathcal{M} \to \mathrm{T}\,\mathcal{M}, (\eta_x, \xi_x) \mapsto \mathcal{T}_{\eta_x}\xi_x$ with associated retraction $R$ is a smooth mapping such that, for all $(x, \eta_x)$ in the domain of $R$ and all $\xi_x \in \mathrm{T}_x\,\mathcal{M}$, it holds that (i) $\mathcal{T}_{\eta_x}\xi_x \in \mathrm{T}_{R(\eta_x)}\,\mathcal{M}$, (ii) $\mathcal{T}_{0_x}\xi_x = \xi_x$, (iii) $\mathcal{T}_{\eta_x}$ is a linear map.

For $\mathcal{L}$, the exponential mapping and parallel translation on $\mathcal{L}$ are well-known, and they are used as retraction and isometric vector transport in Riemannian algorithms.

**Lemma 4.1.** *The exponential mapping and parallel translation on $\mathcal{L}$ are*

$$R_l(v) = \cos(\|v\|_{\mathbb{L}^2})l + \frac{v\sin(\|v\|_{\mathbb{L}^2})}{\|v\|_{\mathbb{L}^2}}$$

*and*

$$\mathcal{T}_u v = v - \frac{2\langle v, \tilde{l}\rangle_{\mathbb{L}^2}}{\|l+\tilde{l}\|^2_{\mathbb{L}^2}}(l+\tilde{l})$$

*respectively, where $u, v, \in \mathrm{T}_l\,\mathcal{L}$ and $\tilde{l} = R_l(u)$.*

For the cost function of interest, an analytical form of the Riemannian gradient can be derived, and it is given in Lemma 4.2.

**Lemma 4.2.** *The Riemannian gradient of the cost function $L(O,m,l)$ in (4.4) is*

$$\operatorname{grad} L(O,m,l) =$$
$$\left(P_O\left(-2\int_0^1 q_2\left(\rho_{l,m}(t)\right) l(t) q_1(t)^T dt\right), -2\int_0^1 \left\langle Oq_1(t), l(t)q_2'\left(\rho_{l,m}(t)\right)\right\rangle_2 dt, P_l(2y(t)l(t) - 2x(t) + 2z(t))\right),$$

*where $P_O U = (U - OU^TO)/2$ is the projection to $\mathrm{T}_O\,\mathrm{SO}(n)$, $\rho_{l,m}(t)$ denotes $\int_0^t l^2(s)ds + m \mod 1$,*

$$x(t) = \left\langle Oq_1(t), q_2\left(\rho_{l,m}(t)\right)\right\rangle_2,$$

*$y(t)$ is any antiderivative of*

$$y'(t) = \left\langle Oq_1(t), 2l(t)q_2'\left(\rho_{l,m}(t)\right)\right\rangle_2,$$

*and $z(t) = \omega l(t)(2 - 1/l^4(t))\sqrt{1+l^4(t)}))$.*

## 5 Implementation Comments

### 5.1 Representation and Cost Function

In practice, all of the curves are represented by a set of points, and therefore, the $q$-function of a curve $\beta(t)$ is also represented by points that are on some smooth function. Since $O$ is an isometry in the cost functions, we can apply it to either $q_1$ or $q_2$. We apply it to $q_1$ because representing $q_1$ does not require the use of an interpolatory function and a vector of points is sufficient, hence, (4.4) is computationally cheaper than (4.2).

In all of the cost functions considered, $q_2$ is composed with some function and, therefore, representing $q_2$ as a set of points is not sufficient. A suitable function must be used. Since the convergence analysis of Riemannian quasi-Newton optimization algorithms requires a $C^2$ cost function, an interpolatory cubic spline of the set of points on $q_2$ is used, but splines of degree 1, i.e., piecewise linear, or degree 2 are also practical.

It should be noted however that there is nothing in the formulation that requires an interpolatory approximation. The discrete points in the representation could be control points for a continuous approximating parameterized curve, e.g., a parameterized B-spline.

Finally, all integrals required by the algorithms are approximated by the Composite Trapezoidal Rule.

## 5.2 Symmetry Considerations

As discussed earlier and seen from (2.6) and (3.3), the optimum, denoted by $\gamma_*$, may not be absolutely continuous and increasing almost everywhere due to a horizontal and/or vertical region and is therefore in the closure of $\Gamma^c$. In order to guarantee the symmetry of the distance function

$$d_{l_n^c}(q_1, O(q_2 \circ \tilde{\gamma})\sqrt{\dot{\tilde{\gamma}}}) = d_{l_n^c}(O^T(q_1 \circ \tilde{\kappa})\sqrt{\dot{\tilde{\kappa}}}, q_2),$$

where $\tilde{\kappa} = \tilde{\gamma}^{-1} \in \Gamma^c$, we must have a symmetric cost function

$$1 - \frac{H^c(O,\tilde{\gamma})}{2} = \int_{\mathbb{S}^1} \langle q_1(t), O(q_2 \circ \tilde{\gamma}(t))\sqrt{\dot{\tilde{\gamma}}(t)} \rangle_2 dt \tag{5.1}$$

$$= \int_{\mathbb{S}^1} \left\langle O^T(q_1 \circ \tilde{\kappa}(t))\sqrt{\dot{\tilde{\kappa}}(t)}, q_2(t) \right\rangle_2 dt. \tag{5.2}$$

If $(O,\tilde{\gamma}) \in \mathrm{SO}(n) \times \Gamma^c$, then the symmetries are guaranteed by the isometry of $\mathrm{SO}(n)$ and $\Gamma^c$. Let $\tilde{\gamma} \in \Gamma^c$ be represented by $(m,\gamma) \in [0,1] \times \Gamma^o$ and $\kappa$ denote $\gamma^{-1}$. If $\gamma$ is not in $\Gamma^o$, then there are some problems. For $\gamma$ containing a flat region, $\gamma(t) = a, \forall t \in [b,c]$, the cost function (5.1) is well defined. However, (5.2) is not due to the non-existence of $\kappa(a)$. One way to guarantee symmetry is to define $\kappa(a)$ to be $b$ or $c$ and $\dot{\kappa}(a)$ be any finite number. In fact, for the purpose of computing the value of the cost function, $\kappa(a)$ can be defined as any finite number since the jump discontinuity of $\kappa$ at $a$ does not change the integral. For $\gamma$ containing a vertical region, $\gamma(a) = [b,c]$, it is not a function. Similarly to the previous idea, we can redefine $\gamma(a), \dot{\gamma}(a)$ to be any finite number and $\kappa$ to satisfy $\kappa(t) = a, \forall t \in [b,c]$ and symmetry is satisfied.

Theoretically, therefore, when $\gamma$ is not in $\Gamma^o$, evaluation and symmetry of the cost function can be handled. In practice, however, numerically evaluating the cost function $H^c(O,\gamma)$ requires a quadrature rule that depends on every point in the discrete set. If $\gamma$ has a vertical or near vertical segment then $\dot{\gamma}$ is infinite or very large and numerical overflow may occur. $\gamma$ containing a flat region does not cause numerical problems when evaluating the cost function. In some versions, e.g., Algorithms 3 and 4, an interpolatory spline is used to represent $\gamma$. If a spline of degree 1, i.e., piecewise linear, is used there is no numerical problem. However, a higher degree spline requires care to guarantee that it is nondecreasing. This is not an issue during the iteration of the new Riemannian algorithm.

These theoretical and practical issues can be avoided for both the Coordinate Descent DP-based Algorithms 3 and 4 and the new Riemannian algorithm. In Section 3.1, the DP algorithm constrains the set of slope choices $\mathcal{N}_{i,j}$ to remain sufficiently far from 0 or $\infty$ and thereby avoids horizontal and vertical regions in $\gamma^*$. In the Riemannian algorithm, the barrier function $B(\gamma)$ in (4.3) is added, and it does not destroy the symmetry of the cost function. There is no explicit lower or upper bound for the slopes of $\gamma$ unlike the approach above for the DP-based algorithm.

## 5.3 Escaping Local Minima

Practical Riemannian optimization methods guarantee convergence to local (possibly not global) minima. There are many approaches to escape from nonglobal local minima when working in Euclidean space. Two standard ones are the MCMC simulated annealing algorithm and the use of multiple runs with different initial conditions. For Riemannian optimization, we can use similar ideas.

We have tested a Riemannian gradient-based MCMC simulated annealing algorithm using a Metropolis-Hastings acceptance test. For sufficiently small "temperature", the algorithm changes to one of the Riemannian quasi-Newton algorithms. The basic idea of this algorithm is to search the domain sufficiently and find a satisfactory minimum. Unfortunately, the dimension of domain $\mathrm{SO}(n) \times \mathbb{R} \times \mathcal{L}$ is infinite, and the dimension of the finite approximation used is large enough so that a sufficiently thorough search was often found to be unacceptably expensive.

A simpler, and in practice effective, choice in this setting is to run the Riemannian quasi-Newton algorithms with multiple initial conditions. Let $(O_0, m_0, l_0) \in \mathrm{SO}(n) \times \mathbb{R} \times \mathcal{L}$ denote the initial iterate. The initial rotation $O_0$ is given by the method of computing the SVD used in Algorithm 1. The initial $l_0$ is given by choosing a small number of points on the curve, $N_s$, and running DP with small $h$, where $h$ is used in (3.2). The motivation is to make use of the global minimization property of DP on a coarse grid with slope constraints and then improve the quality of the solution by Riemannian quasi-Newton algorithms.

The initial value of $m$, denoted $m_0$, can be chosen uniformly spaced or randomly on $[0, 2\pi]$. However, we automatically choose a set of $m_0$'s as well as $N_s$ for Riemannian quasi-Newton algorithms by exploring the structure of the curves. For example, let the curves in Figure 4 be two parts of two closed curves. If the rest of the curves are ignored, there are two minima that correspond to the peak of curve 1 matching the first peak or second peak of curve 2. The two minima can be obtained by using only two $m_0$'s. Suppose the starting point of curve 1 is the point marked with a cross on the graph. The starting point on curve 2 can be any point in the green parts of curve 2. For these two initial conditions, Riemannian algorithms are able to search for the best matching point. Using this idea, if the total change of the angle for some interval along the curve is greater than a specified threshold $T_m$, an $m_0$ is added at the end of the interval. In order to avoid the effect of noise on curves, it is required that the difference between consecutive $m_0$ points is greater than or equal to some positive value $z$. Each of the $m_0$'s produced generates a distinct initial condition for the Riemannian optimization. $N_s$ is taken as a linear function of the total angle variation along the curve 2, i.e., $\tau_1 + \tau_2\theta_T$, where $\theta_T$ denotes the total angle variation along a curve. The Algorithm to generate the set of initial iterates is stated in Algorithm 5.

In practice, when one curve changes direction frequently and the other curve is relatively simple in shape, choosing which curve is used as the basis for the generation of the set of $m_0$'s depends

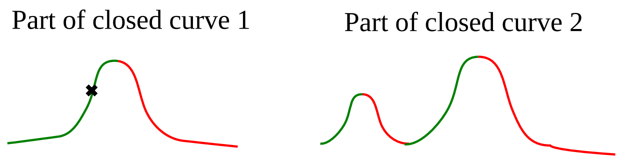

Figure 4: Choosing initial $m$ for Riemannian algorithms.

**Algorithm 5** Generate a set of initial iterates $\{(O_0^{(i)}, m_0^{(i)}, l_0^{(i)})\}$

```
Input: Two $q$ functions of closed curves $q_1 = (v_1, v_2, \dots, v_N, v_1) \in \mathbb{R}^{n\times(N+1)}$ and $q_2 = (u_1, u_2, \dots, u_N, u_1) \in \mathbb{R}^{n\times(N+1)}$; positive integer $z$; positive constant $\tau_1, \tau_2 > 0$, $T_m > 0$.
1:  Define $\theta = 0$, $\tilde{\theta}_T = \theta_T = 0$, $idx = 1$, $u_0 = u_N$ and $v_0 = v_N$;
2:  Initialize all the lists: $ms = \emptyset$, $Os = \emptyset$ and $ls = \emptyset$;
3:  for $i = 1, 2, \dots, N$ do
4:      Compute the angle $\alpha$ between $v_i - v_{i-1}$ and $v_{i+1} - v_i$ and the angle $\tilde{\alpha}$ between $u_i - u_{i-1}$ and $u_{i+1} - u_i$;
5:      $\theta \leftarrow \theta + \alpha$; $\theta_T \leftarrow \theta_T + \alpha$; $\tilde{\theta}_T \leftarrow \tilde{\theta}_T + \tilde{\alpha}$;
6:      if $\theta > \theta_m$ and $i - idx \geq z$ then
7:          Add $i$ to the list $ms$;
8:          $idx \leftarrow i$; $\theta \leftarrow 0$;
9:      end if
10: end for
11: Let $N_s \leftarrow \text{round}(\tau_1 + \tau_2 \max(\theta_T, \tilde{\theta}_T))$;
12: for i = 1, 2, ..., length(ms) do
13:     Define $\tilde{q}_1 = (v_{ms(i)}, v_{ms(i)+1}, \dots, v_N, v_1, \dots, v_{ms(i)}) \in \mathbb{R}^{n\times(N+1)}$ and $\tilde{q}_2 = (u_{ms(i)}, u_{ms(i)+1}, \dots, u_N, v_1, \dots, u_{ms(i)}) \in \mathbb{R}^{n\times(N+1)}$;
14:     Compute $O = UV^T$, where $U, V$ are from the singular value decomposition $USV^T = \tilde{q}_1 \operatorname{diag}(\frac{1}{2N}, \frac{1}{N}, \dots, \frac{1}{N}, \frac{1}{2N}) \tilde{q}_2^T$; (the diagonal matrix $\operatorname{diag}(\frac{1}{2N}, \frac{1}{N}, \dots, \frac{1}{N}, \frac{1}{2N}) \in \mathbb{R}^{(N+1)\times(N+1)}$ defines the composite trapezoidal rule.)
15:     $\tilde{q}_2 \leftarrow O^T \tilde{q}_2$;
16:     Resample $\tilde{q}_1$ and $\tilde{q}_2$ using a cubic interpolatory spline and obtain $N_s$ uniformly-spaced points for each curves. Denote by $\tilde{q}_1^s$ and $\tilde{q}_2^s$ respectively;
17:     Use DP method to obtain a $\gamma^s$ that optimizes $\int_0^1 \|\tilde{q}_1^s - (\tilde{q}_2^s, \gamma)\| dt$;
18:     Compute the initial $l$ from $\gamma^s$;
19:     Add $O$ to the list $Os$ and add $l$ to the list $ls$;
20: end for
21: $\{(Os(i), ms(i), ls(i))\}$ is the set of initial iterates.
```

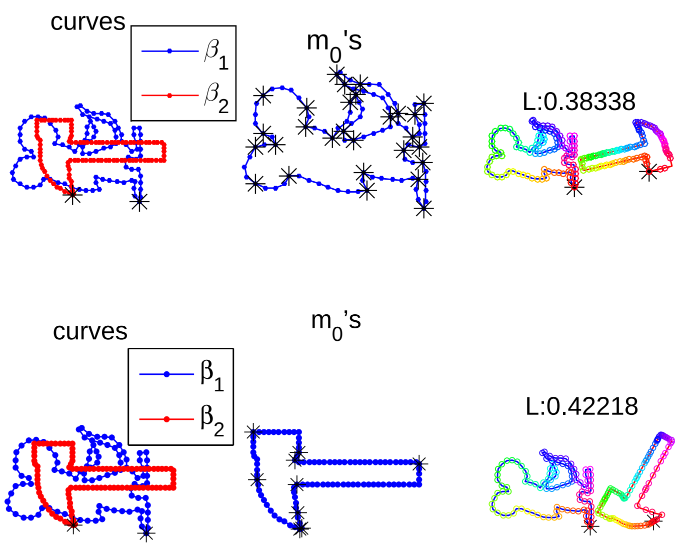

Figure 5: Comparisons for choosing one curve or the other to generate $m_0$'s. The black stars in the left column curves represent the start/end points. The black stars in the middle column represent all $m_0$'s generated by the curves. The colors in the right column curves indicate corresponding points.

on the context of the distance computation. Two curves with significantly different shape are expected to have a large distance. If the application requires an accurate approximation of the large distance then the curve with the more complicated shape should be used to generate the $m_0$'s. If, however, large distances need not be approximated accurately, e.g., when distances are used to determine that the shapes are not in the same class, then the simpler curve should be used to generate the $m_0$'s, and the computational complexity of the optimization is reduced. One example is given in Figure 5. This asymmetry of the resulting distance comes from the asymmetry of the algorithm. This is quite different from DP which often requires a large number of break points to get a satisfactory result in either case above.

## 6 Experiments

### 6.1 Overview of Experiments

The performance of the Riemannian optimization approach and Coordinate Descent methods to computing the elastic distance metric for curves in $\mathbb{R}^2$ is assessed in this section. In Section 6.2, the performances of the Riemannian optimization algorithms, including the Riemannian trust region symmetric rank-one update method (RTR-SR1) [HAG15, Algorithm 1], the limited-memory RTR-SR1 (LRTR-SR1) [HAG15, Algorithm 2], the Riemannian BFGS (RBFGS) [HGA15, Algorithm 1 with $\phi_k \equiv 0$], the limited-memory RBFGS (LRBFGS) [HGA15, Algorithm 2], and the Riemannian steepest descent (RSD) [AMS08, Page 62] are compared to identify the preferred Riemannian

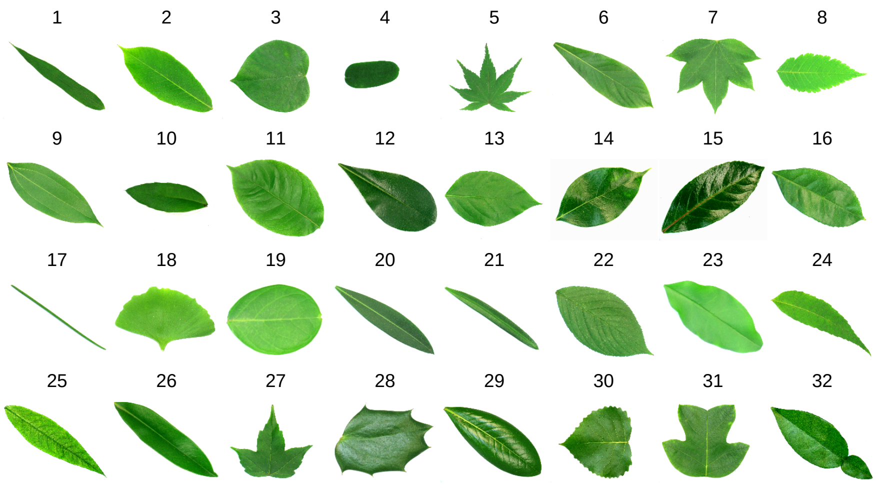

Figure 6: Samples of leaves from the Flavia leaf dataset. One sample per species is illustrated.

method. In Section 6.3, the difficulties that Algorithms 2 and 3 have with multiple coordinate descent iterationsare illustrated. In Section 6.4, the quality of using the extrinsic distance $d_{l_n^o}$ to approximate the intrinsic distance $d_{l_n^c}$ is shown. The preferred Riemannian method is then compared systematically with an adapted version of CD1 (coordinate descent with one single iteration) termed CD1H that consists of doing "one and a half iteration": specifically, we do one iteration of the CD (coordinate descent) method followed by the computation of the best rotation. This is done since empirically the rotation found with $\gamma_0 = id$ is usually not sufficiently close to $O^*$ and updating $O$ based on $\gamma_1$ improves the quality of the rotation. Section 6.5 presents the comparisons of the accuracies of the CD1H and the preferred Riemannian method. Section 6.6 illustrates the influence of representing the curves with different sets of points. Section 6.7 shows more concrete examples, and in Section 6.8, the computational time, the final values of the cost function and the quality of the distances given by the preferred Riemannian method and CD1H are compared. Finally, the problems with complexity and optimization effectiveness for Algorithm 4 are also demonstrated in Section 6.9.

Two public datasets are used in the experiments: the Flavia leaf dataset [WBX+07] and the MPEG-7 dataset [Uni]. The Flavia leaf dataset contains images of 1907 leaves from 32 species. Figure 6 shows an example leaf from each species. MPEG-7 contains 1400 images in 70 clusters each of which contains 20 shapes. Figure 7 shows an example shape from each cluster. The boundary curves of the shapes are extracted using the BWBOUNDARIES function in Matlab. 100 uniformly-spaced points are chosen to represent the shape unless indicated otherwise in the description of the experiments.

## 6.2 The Preferred Riemannian Quasi-Newton Algorithm

The two public datasets were used to compare the performances of several Riemannian optimization algorithms in minimizing the cost function (4.4). For these experiments, the stopping criterion for the Riemannian algorithms requires the relative change of the cost function in two successive iterates to be less than $10^{-3}$ for more than 5 consecutive iterations, and the minimum number of iterations is set to 10. The number of points used to get the third component of the initial condition for the

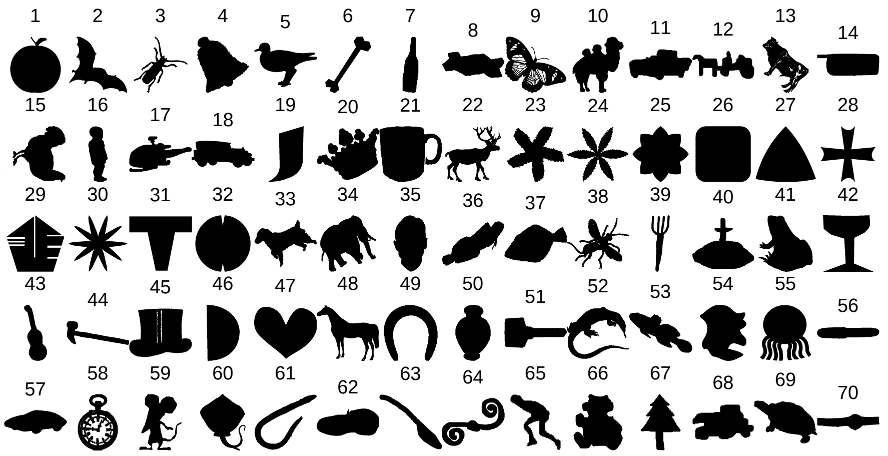

Figure 7: Samples of curves from the MPEG-7 dataset. One sample per cluster is illustrated.

Riemannian algorithms, $N_s$, is set as described in Section 5.3. The weight, $\omega$, in the cost function (4.4) is 1/100 initially and decreases on each iteration by $\omega \leftarrow 0.8\omega$. The more complex curve in the pair of $\beta_1$ and $\beta_2$ , i.e., the curve such that the total change of the angle along the curve is larger, is used for setting $m_0$ and $N_s$. The values $T_m = \pi/2$, $z = 4$, $\tau_1 = 30$ and $\tau_2 = 2/\pi$ are used. These settings of parameters are used for all experiments using Riemannian algorithms.

All codes are written in C++ using BLAS and LAPACK, compiled with g++ and run on a 64 bit Ubuntu system with 3.6 GHz CPU (Intel (R) Core (TM)). The code can be found on `www.math.fsu.edu/~whuang2/papers/RORCESA.htm`.
The output time is the average CPU time of 10 runs with identical parameters. (The times observed had very low variance.)

To find the preferred Riemannian method, all of the methods were run on several sets of randomly chosen pairs of shapes from the two datasets. Table 1 reports $t_{ave}$, the average time to compute the distance between two shapes, and $L_{ave}$, the average cost function value for one of these sets from the Flavia and MPEG-7 shapes. The trends in other sets were similar.

The complexity of RBFGS and RTR-SR1 per iteration is $O(N^2)$ due to the use of a dense matrix vector product. This also implies the $O(N^2)$ space complexity. In contrast, their limited memory versions, LRBFGS and LRTR-SR1, require $O(N)$ complexity. It follows that the computational time of RBFGS and RTR-SR1 per iteration is larger than LRBFGS and LRTR-SR1 respectively. The total computational time further depends on the number of iterations, and it is shown in Table 1 that RBFGS and RTR-SR1 are slightly slower than the LRBFGS and LRTR-SR1 respectively.

All the Riemannian quasi-Newton methods produce similar final function values. Among them, RBFGS and RTR-SR1 perform similarly while LRBFGS gives the smallest final cost function values.

The RSD method has low computational times per iteration due to its relatively low $O(N)$ computational complexity, but it does not result in a competitive final cost function value due to the simplicity of the approach and the large number of iterations required.

In summary, all of the Riemannian algorithms were competitive with CD1H. LRBFGS with its small final cost function and its low computational and storage complexity is chosen as the preferred Riemannian algorithm. It will be used in further comparisons to CD1H from the point of view of quality of shape distance computations.

Table 1: Comparison of Riemannian Methods for representative sets from the Flavia (F) and MPEG-7 (M) datasets: average time per pair ($t_{ave}$) in seconds and average cost function per pair ($L_{ave}$).

| | RBFGS | LRBFGS | RTR-SR1 | LRTR-SR1 | RSD | CD1H |
|---|---|---|---|---|---|---|
| $L_{ave}$(F) | 0.16338 | 0.16182 | 0.16367 | 0.16723 | 0.20665 | 0.22323 |
| $t_{ave}$(F) | 0.08963 | 0.07954 | 0.11603 | 0.10862 | 0.06488 | 0.42895 |
| $L_{ave}$(M) | 0.33214 | 0.31893 | 0.33258 | 0.3418 | 0.48732 | 0.51664 |
| $t_{ave}$(M) | 0.19945 | 0.19318 | 0.24696 | 0.23102 | 0.14373 | 0.42817 |

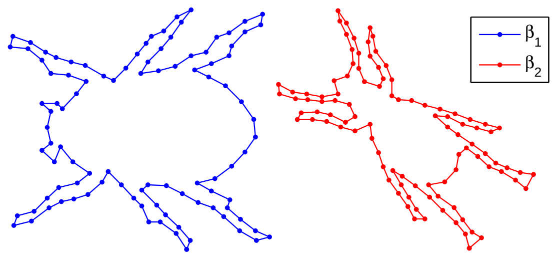

Figure 8: Two shapes from the MPEG-7 dataset. Successive points are connected by straight lines for display purposes.

## 6.3 Examples of Coordinate Descent Difficulties

For the experiments using Coordinate Descent based on DP, the mesh size, $h$ defined in (3.2), is 6, and every 4-th point is chosen as a break point. $\gamma_0$ is the identity, i.e., $\gamma_0(t) = t$ for all experiments using CD1H.

The shapes from MPEG-7 dataset shown in Figure 8 are used to illustrate the difficulties of Algorithm 2 and Algorithm 3, The variation in the shape of curve $\beta_2$ in Algorithm 2 was identified as a serious problem. Figure 9 shows the shape of $\beta_2$ initially and in the first 4 iterations of the algorithm along with the value of the cost function. The change to the shape is clear with many of its details disappearing gradually. Most significantly, the cost function is increasing, and the algorithm is not reliable for optimization.

The potential conflict in Algorithm 3 between the value of the cost function, $H^c$, evaluated during iteration $k$ for $\bar{\beta}_2^{(\min,k)}$ computed using (3.4) and the value of $H^c$ for the theoretically identical curve $\beta^{(k+1)}$ computed using (3.5) is also observed for the illustrative pair of shapes. Table 2 shows the variations of the cost function. The value of $H^c$ in the second row for iteration $k$ should be the same as the value in the first row for iteration $k+1$. Clearly, the values are significantly different. Note also the values in the second row, which are the ones used by the algorithm in optimization decisions, are not decreasing. They in fact increase in subsequent iterations, and the algorithm is unreliable.

The difficulties encountered with Algorithm 2 and Algorithm 3 can be avoided by only performing a single iteration of Coordinate Descent, i.e., CD1. This was done by Srivastava et al. [SKJJ11], and as a result they did not observe the problems. However, the accuracy of the dis-

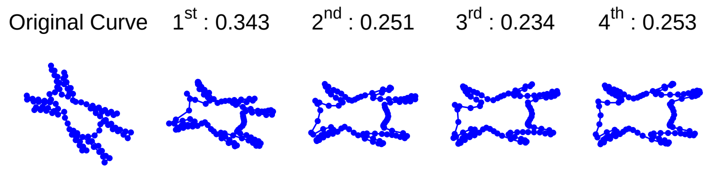

Figure 9: The variation of curve $\beta_2$ during the iteration in Algorithm 2. Curves are shown by connecting consecutive points using straight lines. The cost function values are given in the titles.

Table 2: The variations of the cost function values in Algorithm 3

| iteration ($k$) | 1 | 2 | 3 |
|---|---|---|---|
| $H^c$ for $\beta^{(k)}$ of (3.5) | 1.561523 | 0.365177 | 0.355348 |
| $H^c$ for $\bar{\beta}_2^{(\min,k)}$ of (3.4) | 0.343090 | 0.270597 | 0.284996 |

tance computed using a single iteration is thereby limited by the quality of the choice of the initial reparameterization and rotation.

Figure 10 includes the optimization results for CD1, CD1H, and for the Riemannian algorithm LRBFGS iterating until the cost function value is invariant to three digits. The final cost function LRBFGS is smaller than CD1, and the superior quality of the final rotation from LRBFGS is clearly illustrated. The relationship of LRBFGS and CD1H is discussed in more detail in the following sections.

### 6.4 Extrinsic Distance Versus Intrinsic Distance

In this section, the differences between the intrinsic distance $d_{l_n^c}$ and the extrinsic distance $d_{l_n^o}$ are shown based on 1000 randomly chosen pairs from the Flavia and MPEG-7 data sets. The distance $d_{l_n^c}$ is computed by the path-straightening algorithm [SKJJ11, Section 4.3]. The ratios of $d_{l_n^o}$ and $d_{l_n^c}$ are shown in Figure 11 and all of them are close to 1, i.e., between 1 and 1.007. Therefore, it is acceptable to use $d_{l_n^o}$ to approximate $d_{l_n^c}$ in practical situations. For situations where the intrinsic distance $d_{l_n^c}$ and the associated minimal geodesic are required, e.g., Karcher mean computations, the computational time for the algorithm of [SKJJ11] can be a problem. An improved algorithm based on Riemannian optimization has been developed and used for Karcher means see [YHGA15] and [**?**].

### 6.5 Accuracy of CD1H and LRBFGS

CD1H improves somewhat the quality of the distance approximation of CD1 by improving the rotation while still avoiding the problems of Algorithms 2 and 3 and the potential high computational cost of Algorithm 4 by avoiding additional iterations. However, this results in limitations on the ultimate quality of the distance approximation. In this section and the following sections, we focus on comparing the state-of-the-art CD1H and the LRBFGS method with respect to computational time and final cost function value. For those pairs that do not achieve acceptable cost

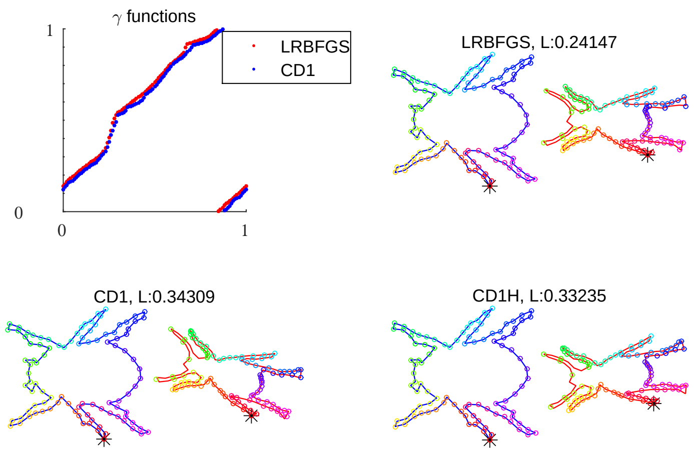

Figure 10: Results for LRBFGS, CD1 and CD1H. The optimal rotation and reparameterization are applied to $\beta_2$. The interpolation points of both curves are kept for display purpose. The colors indicate corresponding points on the two curves. The black stars represents the start/end points of the curves. The title of the matching curves includes the final values of the cost function.

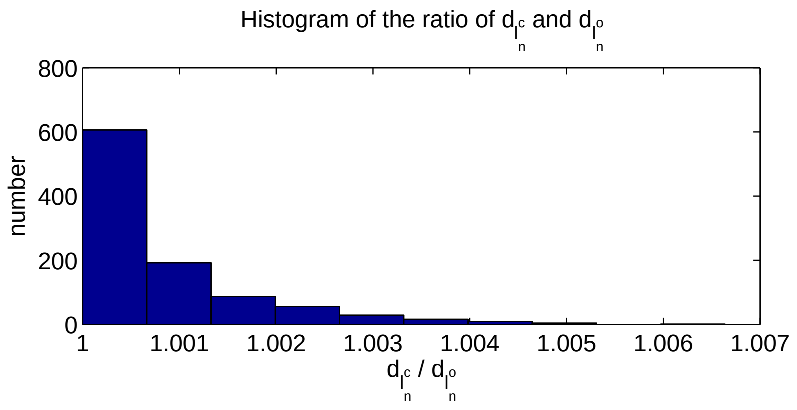

Figure 11: The histogram of the ratio of the intrinsic distance $d_{l_n^c}$ and the extrinsic distance $d_{l_n^o}$.

function values (and therefore distance approximations) we know that Algorithm 4 could improve the approximations. This is discussed in Section 6.9.

For the remainder of the experiments LRBFGS indicates algorithm choice along with using initial iterates given by Algorithm 5 unless indicated otherwise in the description of the experiments.

To show this, a selected curve is compared to a second curve specifically created to be very close to the first using changes that should be detectable by the distance metric and the algorithms. Two methods for specifying these changes are considered to highlight particular difficulties with CD1H compared to LRBFGS. In the comparisons, the first ten curves of the ordered list of shapes defining each cluster of the two data sets, 1020 in total, are used.

In the first set of comparisons, each curve in the set of 1020 is taken as $\beta_2$, in turn, and $\beta_1$ is defined by

$$\beta_1 = O_T(\beta_2 \circ \gamma_T), \tag{6.1}$$

using cubic spline interpolation for $\beta_2$, where $\gamma_T(t) = (t + \sin(2\pi t)/(4\pi))$ and $O_T \in \mathrm{SO}(2)$ is randomly generated. Intuitively the optimal reparameterization should be close to $\gamma_T^{-1}$, and the structures and positions in $\beta_2$ should map to the same structures and positions in $\beta_1$.

Figure 13 shows the histogram of $\mathrm{diff}_\gamma = \int_0^1 |\gamma_{\mathrm{CD1H}}(t) - \gamma_{\mathrm{LRBFGS}}(t)| dt$, where $\gamma_{\mathrm{CD1H}}$ and $\gamma_{\mathrm{LRBFGS}}$ denote the optimal reparameterizations given by CD1H and LRBFGS respectively. The difference of $\gamma$ given by CD1H and LRBFGS satisfies $\mathrm{diff}_\gamma < 0.1$ for most of the samples, and the differences are caused by the accuracies of the solutions. Figure 12(a) shows an example and there is almost no visible difference between the reparameterizations given by CD1H and LRBFGS. However, if one has a closer look, the number of points on the belly of the cow on the right given by CD1H does not exactly match the points on the cow on the left, but the points on the bellies of the two curves resulting from LRBFGS match very well. Additionally, the rotation given by CD1H is also slightly off. While this pair shows that CD1H can produce results close to those of LRBFGS visually, the CD1H results are typically less satisfactory from a structural point of view for these specially created pairs of curves.

For 64 of the 1020 pairs of very close curves, the difference in reparameterization computed by the two algorithms is significant in the sense of $\mathrm{diff}_\gamma > 0.1$. Those 64 samples can be categorized into two classes. The first class contains 39 samples satisfying the property that rotating the samples by an angle $\theta < 2\pi$ gives a shape close to the original one, e.g., the shape in Figure 12(b). For this kind of shape, there are multiple equivalent global minimizers. CD1H and LRBFGS may find different minimizers, and therefore the difference in reparameterization is understandable and not problematic. The second class contains 25 samples for which CD1H cannot give intuitive reparameterizations while LRBFGS can. Figures 12 (c), (d) and (e) are three such examples. These examples demonstrate that the quality of the initial guess at the reparameterization is crucial to finding a good rotation and, similarly, a bad approximation of the rotation can cause significant inaccuracy in the associated reparameterization found using DP.

The second set of experiments on pairs of curves specifically created to be near each other considers the effect of the choice of break points on CD1H and the strategy used with LRBFGS to avoid simple subset selection for break points. The first set of tests above chose $\beta_1$ to be (6.1) which indicates that the correct break point is always used. This can be seen from noting that $\gamma_T(0) = 0$ implies no shifting and the initial break point is always used. In particular it is seen that the accuracy of CD1H may suffer significantly when not every point is available on a curve as a break point.

The same 1020 sample curves are used. $\beta_1$ is chosen to be one of the samples, and $\beta_2$ is created

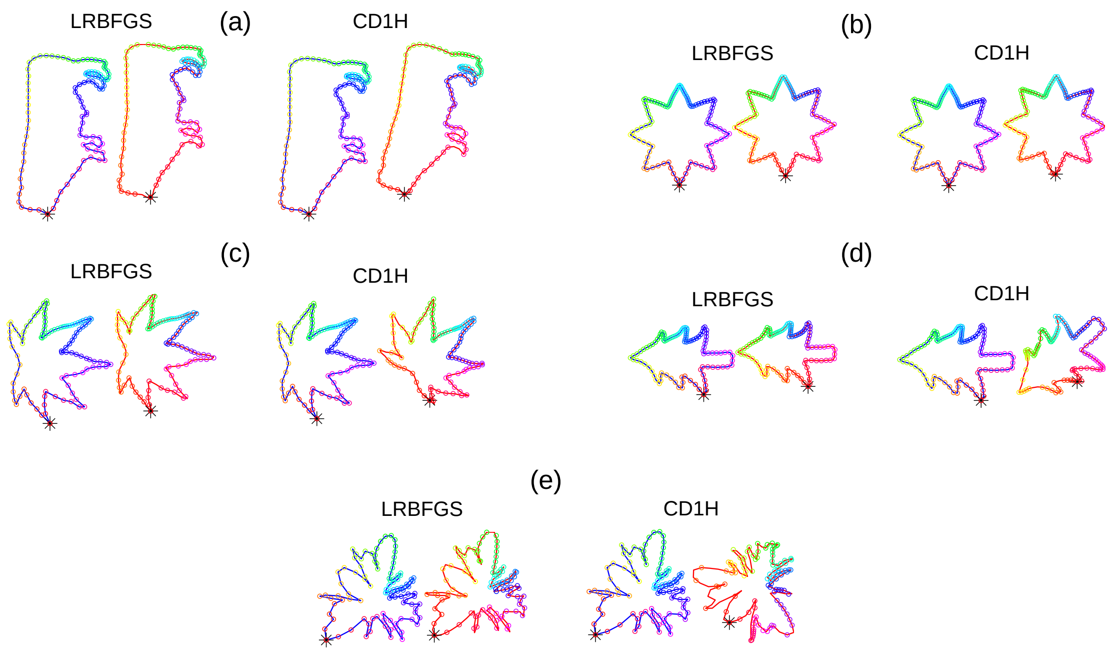

Figure 12: Matching curves given by LRBFGS and CD1H. The optimal rotation and reparameterization are applied for the right curve, i.e., $\beta_2$. The color of points on the two curves represents correspondence between two curves.

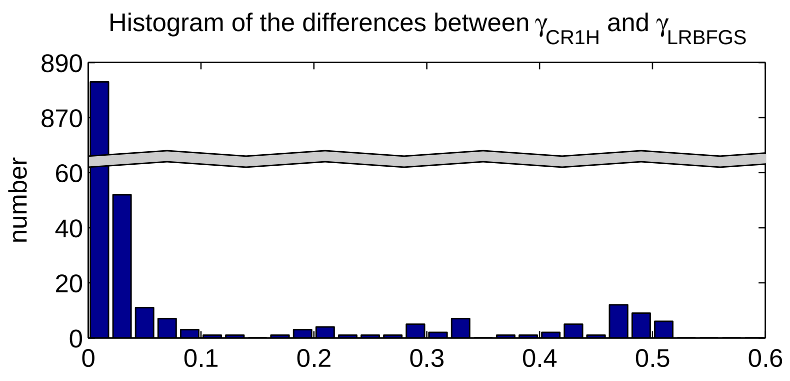

Figure 13: The histogram of $\int_0^1 |\gamma_{\mathrm{CD1H}}(t) - \gamma_{\mathrm{LRBFGS}}(t)| dt$.

Table 3: The results of final cost function values given by LRBFGS (L) and CD1H (C) with multiple $p$ values. $\#_<$ denote the number of final cost function values that is smaller than $10^{-3}$.

| $p$ | 0 | 1 | 2 | 4 | 8 |
|---|---|---|---|---|---|
| $L_{ave}$(L) | $3.28_{-14}$ | $5.04_{-4}$ | $9.71_{-4}$ | $3.38_{-2}$ | $1.04_{-1}$ |
| $\#_<$(L) | 1020 | 900 | 891 | 731 | 444 |
| $L_{ave}$(C) | $3.66_{-14}$ | $2.53_{-2}$ | $9.22_{-2}$ | $2.66_{-1}$ | $4.54_{-1}$ |
| $\#_<$(C) | 1020 | 0 | 0 | 0 | 0 |

by shifting all points on $\beta_1$ by $p$ positions and applying a random rotation. (Examples of shifting 8 points clockwise and counterclockwise are given in the left column of Figure 14.) Note merely shifting $p$ points does not affect, e.g., rotate, the curve in the plane. It simply changes the choice of the initial point in the discrete data representing the curve. For this set of tests, LRBFGS and CD1H only use one initial point and one break point respectively.

Table 3 shows the average of the final cost function values given by CD1H and LRBFGS and the number of those values fewer than $10^{-3}$ with varying values of $p$. When there is no shift on $\beta_2$, the cost functions given by both LRBFGS and CD1H are essentially 0. (The small values are noise due to finite precision.) The average final cost function values given by LRBFGS are smaller than those given by CD1H for all values of $p$. More importantly, even though the shift on $\beta_2$ is not zero, the final cost function values given by LRBFGS are smaller than $10^{-3}$ for many samples, whereas, all of the final cost function values given by CD1H are greater than $10^{-3}$. This demonstrates the power of the inclusion of the location $m$ on the curve as an optimization variable. Essentially, this allows LRBFGS to move in an intelligent manner the initial point for its reparameterization. CD1H, on the other hand, is at the mercy of the initial choice and can only respond by considering many such choices independently thereby increasing the computational time required. Figure 14 is one such example where $\beta_2$ is shifted 8 points clockwise or counterclockwise and LRBFGS adjusts the position of $m$ effectively while the breakpoint used for CD1H is clearly not a good choice and others must be considered.

## 6.6 Points for Representing Curves

The set of boundary points extracted from a high resolution image usually contain a large number of points. As points are removed to reduce time and space complexity important structures may be lost. Figure 15 shows this phenomenon by an example in the MPEG7 database. The boundary set extracted from the image initially contains 5150 points which is too large for computations that involve multiple distance computations, e.g., classification. A method for choosing points must be defined, and the effects on the distance computation considered. The results of different numbers of points for representing the boundary curve are shown. The blue and red points are chosen using different uniformly-spaced points in Figure 15. Table 4 illustrates the cost function values given by CD1H and LRBFGS when $\beta_1$ and $\beta_2$ are the blue and red points respectively. LRBFGS always gives a smaller cost function value than CD1H.

In addition to choosing a sufficient number of points in the proper places to capture appropriately the structure of the curves, the influence of increasing the number of points used on the computational time is much more significant for CD1H than for LRBFGS. The substantial differ-

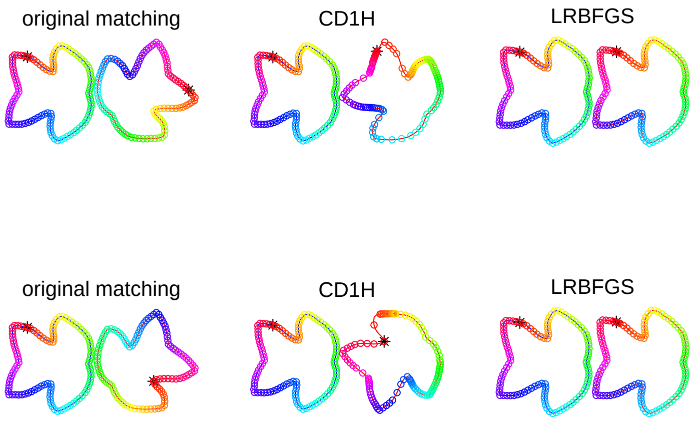

Figure 14: Matching curves given by CD1H and LRBFGS with only one initial point. $\beta_2$ is given by shifting 8 points on $\beta_1$ in clockwise or counterclockwise directions. The black stars represents the start/end points of the curves.

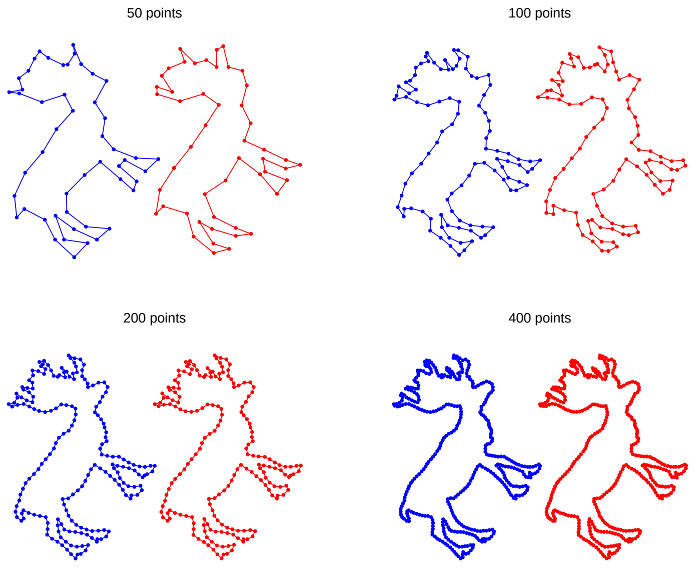

Figure 15: 5150 points extracted from the image in data base. Different numbers of points are chosen for representing the same curve. Red and blue points represent extracting uniformly-spaced points from different initial points.

Table 4: The values of cost function $L$ given by LRBFGS and CD1H when $\beta_1$ and $\beta_2$ are the blue and red curves in Figure 15.

| # of points | 50 | 100 | 200 | 400 |
|---|---|---|---|---|
| L by LRBFGS | $3.09_{-2}$ | $4.23_{-2}$ | $1.18_{-3}$ | $2.67_{-3}$ |
| L by CD1H | $1.27_{-1}$ | $1.29_{-1}$ | $4.27_{-2}$ | $3.03_{-2}$ |

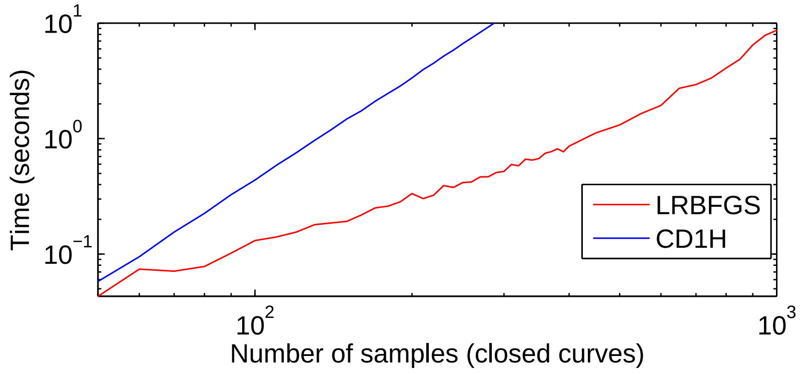

Figure 16: Comparison of complexities of CD1H and LRBFGS.

ence in the complexities of LRBFGS and CD1H is illustrated in Figure 16 for a representative pair of leaf shapes from the 27th species of the Flavia dataset. Boundary curves were extracted with different numbers of points $N$ to test the relationship between $N$ and time costs for LRBFGS and CD1H. LRBFGS needs much smaller computational time than CD1H as the number of points increases. The computational complexity for LRBFGS with DP on a coarse grid is $O(Nkm_s + N_s^2 m_s)$ where $k$ is the number of iterations, $m_s$ is the number of initial conditions. CD1H, due to the DP portion of the computation, contains an $O(N^3)$ term. Note the rise in time for LRBFGS as $N$ increases is not indicative of a nonlinear growth in the number of iterations required but due to the cubic term in the complexity of a very coarse grid, i.e., much smaller than $N$, used by DP to compute the initial reparameterization.

## 6.7 Evaluation with Representative Pairs from Flavia and MPEG-7 Datasets

All previous pairs used nearby pairs constructed by very specific changes from a given image to compare and highlight particular performance characteristics of CD1H and LRBFGS. Figure 17 presents more detailed results for some representative examples from the datasets in which $\beta_1$ and $\beta_2$ are significantly different shapes. The differences between the reparameterizations and rotations computed by LRBFGS and CD1H are clearly shown in those figures. Although a quantitative comparison between the results from the two algorithms is not possible except by the cost function, examining the results shows that LRBFGS produces more reasonable rotations and mappings of structures from one curve to the other.

Since the closed solutions of reparameterization and rotation are unknown for this data, it is unknown whether global minima are obtained. LRBFGS produces smaller cost function values than CD1H for these pairs. Furthermore, LRBFGS gives smaller cost function values for more than 99% pairs of shapes in datasets FLAVIA and MPEG-7 as shown in Section 6.8.

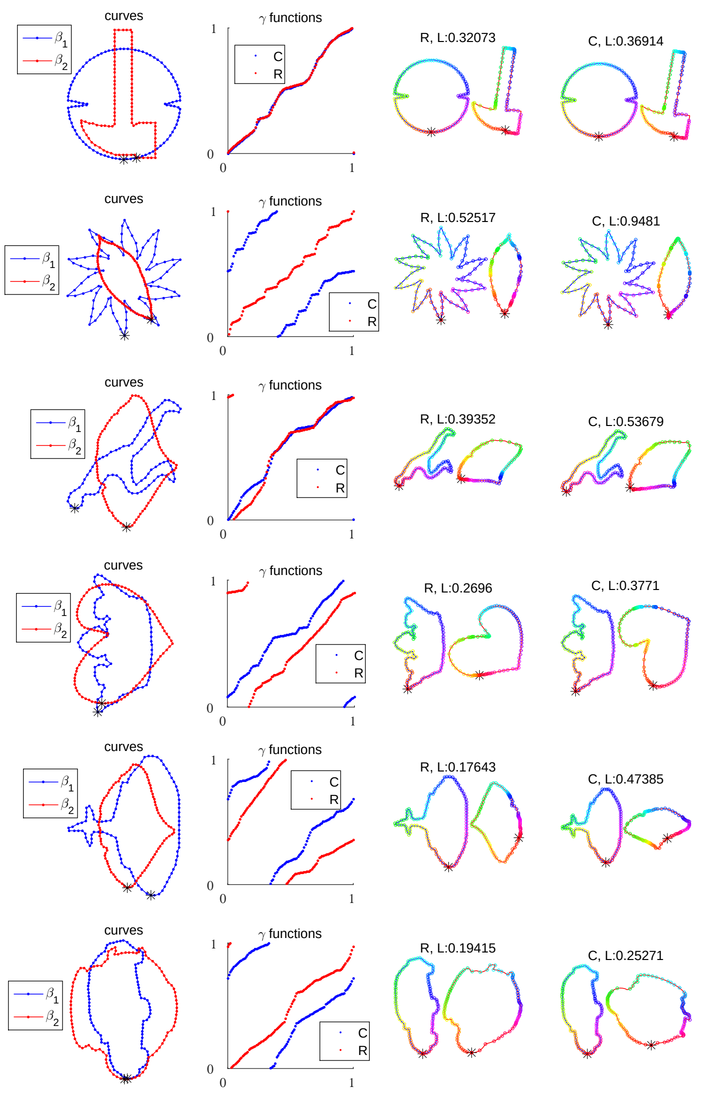

Figure 17: Results of LRBFGS (R) and CD1H (C) for pairs of shapes from Flavia and MPEG-7 datasets. The final rotation and reparameterization are applied to $\beta_2$ to compare with $\beta_1$. The colors of points on the two curves represent correspondence between two curves. The black stars represent the start/end points of the curves. The title of the matching curves includes the final values of the cost function.

## 6.8 Performance Comparison for Flavia and MPEG-7 Datasets

In order to compare the computational cost and efficacy of the preferred Riemannian algorithm, LRBFGS, to those of the current state-of-the-art, CD1H, all pairwise distances in the Flavia and MPEG-7 data sets were computed ($1,819,278$ and $980,700$ pairs respectively) using the testing environment described in Section 6.2. For CD1H, the effect of the number of break points was considered by running each pair with a break point every 2, 4, 8 and 16 points given a fixed initial point, i.e., the sets are nested.

In addition to comparing the computation times and cost function values for the two algorithms, the quality of the distance computations was assessed using the one-nearest-neighbor (1NN) metric of cluster (species) preservation for the MPEG-7 (Flavia) shapes. The 1NN metric, $\mu$, computes the percentage of points whose nearest neighbor is in the same cluster, i.e.,

$$\mu = \frac{1}{n}\sum_{i=1}^{n} C(i), \quad C(i) = \begin{cases} 1 & \text{if point } i \text{ and its} \\ & \text{nearest neighbor are} \\ & \text{in the same cluster;} \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

A very significant improvement in the final value of the cost function achieved by LRBFGS compared to the value achieved by CD1H is observed. For the Flavia dataset, LRBFGS reduces the cost function more than CD1H in 99.16%, 99.45%, 99.68% and 99.83% of the pairs when choosing break points every 2, 4, 8 and 16 points for CD1H respectively. For the MPEG-7 dataset, LRBFGS minimizes the cost function better than CD1H in 99.73%, 99.81%, 99.89% and 99.93% of the pairs when choosing break points every 2, 4, 8 and 16 points for CD1H respectively.

The distribution of the ratio of the cost function value of CD1H to that of LRBFGS is shown in the histograms in Figure 18. Ratios where LRBFGS was more than 4 times better are not included for presentation purposes. The maximum ratios for the Flavia data set (and the number of ratios exceeding 4) were 1184 (1683), 1184 (3348), 1184 (13748) and 1184 (110096) when choosing break points every 2, 4, 8 and 16 points for CD1H respectively. The maximum ratios for the MPEG-7 data set (and the number of ratios exceeding 4) were $6.73*10^{24}$ (564), $6.73*10^{24}$ (1035), $4.76*10^{25}$ (2285) and $1.00*10^{26}$ (5115) when choosing break points every 2, 4, 8 and 16 points for CD1H respectively. The amazingly large ratios beyond 4 occur for pairs of shapes that are fairly close in shape where LRBFGS achieves a very small cost function value. Not only is it clear from this data that, in general, LRBFGS reduces the cost function more than CD1H, but also in the cases when CD1H produces a smaller cost function value it is usually very close to the value produced by LRBFGS.

Of course, if the improvement in the reduction of the cost function requires a very large increase in computation time then the argument in favor of LRBFGS and the other Riemannian methods weakens. The histograms of computation times for CD1H and LRBFGS for the MPEG-7 and Flavia datasets in Figure 19 show that most of the computation time of LRBFGS is smaller than that of CD1H with break points chosen to be every 2 and 4.

A more careful examination of the times indicates an advantage of the Riemannian approach. Specifically, the computation time for a pair of shapes using CD1H is essentially proportional to the number of break points used. There is very little variation between computation times when using the same number of break points as is seen in the CD1H spikes in Figure 19.

Figure 19 also shows that the, much smaller, computation times for LRBFGS have significant variation. Recall that the Riemannian methods automatically select the position and number of

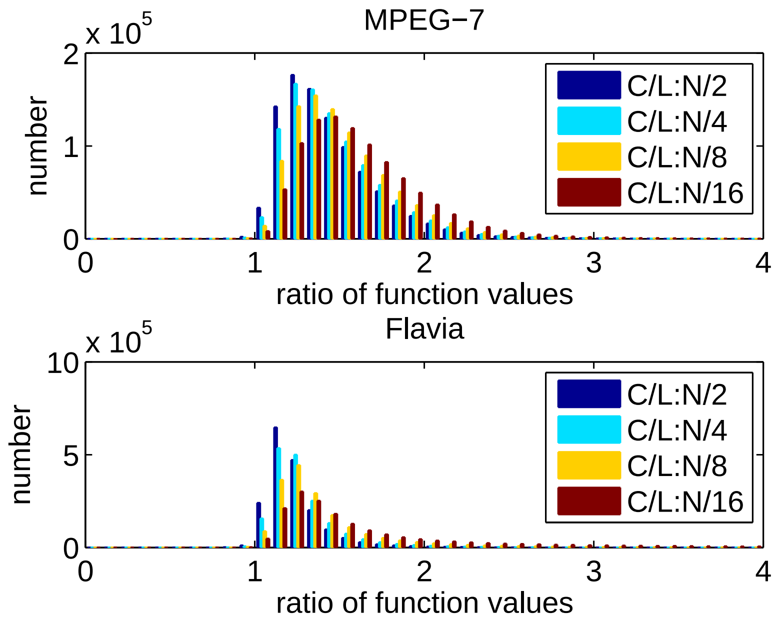

Figure 18: Histograms of ratios of the CD1H cost function value to the LRBFGS cost function value $(C/L)$ for MPEG-7 and Flavia datasets. $N/i, i = 2, 4, 8, 16$ denote the number of break points in CD1H. Bins are $(0, 0.1), \ldots, (0.9, 1.0), \ldots, (3.9, 4.0)$.

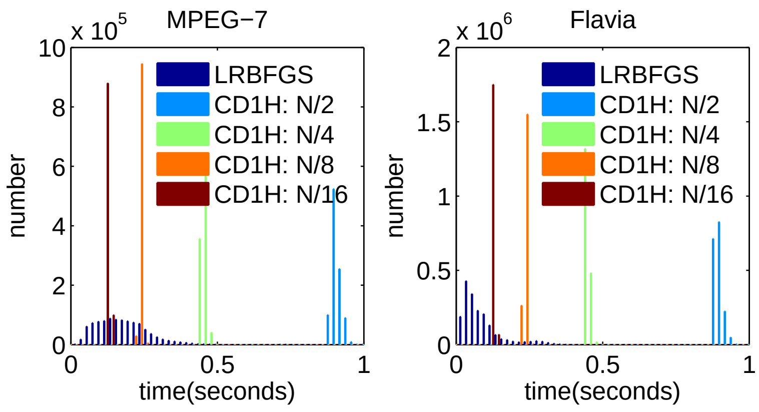

Figure 19: Histograms of computation times of LRBFGS and CD1H for MPEG-7 and Flavia datasets.

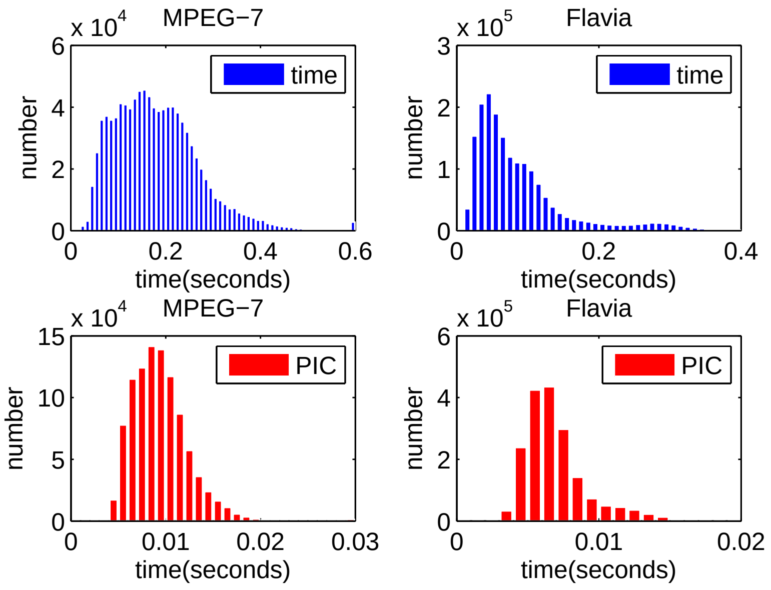

Figure 20: Histograms of computation times of LRBFGS and computation times per initial condition (PIC) of LRBFGS for MPEG-7 and Flavia datasets. The standard deviation of time and PIC for MPEG-7 is 0.090 and 0.003 respectively. The standard deviation of time and PIC for Flavia is 0.067 and 0.002 respectively.

initial conditions used to compute the distance for a pair of shapes. The computation time per initial condition (PIC) for LRBFGS varies only slightly as is shown also in Figure 20, and the computation time for a pair of shapes is essentially proportional to the number of initial conditions used. Since the number of initial conditions is a simple measure of the complexity of one or both of the shapes in the pair, the Riemannian methods have the additional advantage of only requiring a computation time that reflects the difficulty of the problem.

Table 5 shows the average time cost and 1NN metric for both datasets. The trends are as expected given the examples in Figures 6, 7. For the MPEG-7 dataset, the shapes in different clusters are very distinct compared to the significantly greater similarity of shapes in certain pairs of species in the Flavia dataset, e.g., species 1 and 21. Therefore, the $\mu$ values in (6.2) are expected to be higher for MPEG-7 distances since the distinctions are easier to make while lower $\mu$ values are expected for Flavia distances. For CD1H it is expected that $\mu$ values would increase as the number of break points increases. All of these trends are observed in the $\mu$ data.

The comparison of $\mu$ achieved by LRBFGS to those of CD1H shows a clear advantage to LRBFGS. LRBFGS achieves a value of $\mu$ higher than CD1H using the densest set of break points. Not surprisingly, given the distributions of computation times discussed earlier, the average time for LRBFGS is smaller than the average time for even the second sparsest set of CD1H break points ($N/8$).

All of the distances computed so far have used the more complex curve, i.e., the one with greater angular change, to drive the reparameterization. This is the preferred approach to get an accurate approximation of the distance. If the simpler curve is used to drive the reparameterization a reduction in the computational cost is observed for pairs of curves that are sufficiently differen-

Table 5: The average computation time and 1NN metric of LRBFGS and CD1H with break points chosen to be every 2, 4, 8 and 16 points for Flavia (F) and MPEG-7 (M) data sets. Results includes using complex (C) curves and simple (S) curves to generate $m_0$'s.

| | LRBFGS | | CD1H | | | |
|---|---|---|---|---|---|---|
| | (C) | (S) | $N/16$ | $N/8$ | $N/4$ | $N/2$ |
| $t_{ave}$(F) | 0.088 | 0.047 | 0.126 | 0.233 | 0.448 | 0.897 |
| 1NN(F) | 89.51% | 89.04% | 79.55% | 83.01% | 85.95% | 87.52% |
| $t_{ave}$(M) | 0.181 | 0.134 | 0.127 | 0.236 | 0.454 | 0.908 |
| 1NN(M) | 97.79% | 98.07% | 90.29% | 93.86% | 96.07% | 96.79% |

t. However, this results in a less accurate overestimation of the distance and a less satisfactory reparameterization. In some applications, the accuracy of the reparameterization is not the main concern, and accuracy can be usefully traded for lower computational time. Of course, since this is for all pairs in the datasets the distance for pairs of nearby curves do not change significantly. Nevertheless, there is a reduction in the average computational time without a negative effect on the 1NN metric percentage.

### 6.9 Performance of Algorithm 4

The nonuniform grid-based Algorithm 4 does not have the problems of Algorithm 2 and Algorithm 3 described earlier and therefore is a potential alternative coordinate descent method, especially when CD1H does not yield a sufficiently small final cost function value. We next compare Algorithm 4 to LRBFGS to determine if it is competitive. The shapes in Figure 22 show the points of the resampled $\beta_2$ on a uniform grid and the cost function value at each iteration when Algorithm 4 is applied to the shapes in Figure 21. The resulting matching curves, final cost function values and computational time given by LRBFGS and Algorithm 4 are shown in Figure 23.

Algorithm 4 reduces the cost function more than Algorithm 2 and Algorithm 3 therefore improving the distance approximation. However, LRBFGS produces smaller cost function values and uses significantly less computational time. Over the set of experiments, that includes the pairs presented here, Algorithm 4 is never faster than LRBFGS and has computational times that range from 10 to 100 times more than those of LRBFGS. So while Algorithm 4 is more robust than the other implementations of the DP-based approaches, it is simply not competitive with LRBFGS with respect to computational time and effectiveness.

## 7 Conclusion

We have explored the computation of the elastic distance metric for open and closed curves in $\mathbb{R}^n$ and reviewed DP-based coordinate descent algorithms including the state-of-art CD1/CD1H of Srivastava et al., [SKJJ11]. The difficulties for the coordinate descent methods with respect to convergence, robustness and computationaly complexity were identified and an improved coordinate descent method based on a nonuniform grid DP that fixes all of the difficulties was given. As an alternative to the coordinate descent algorithms, we have derived a Riemannian approach to

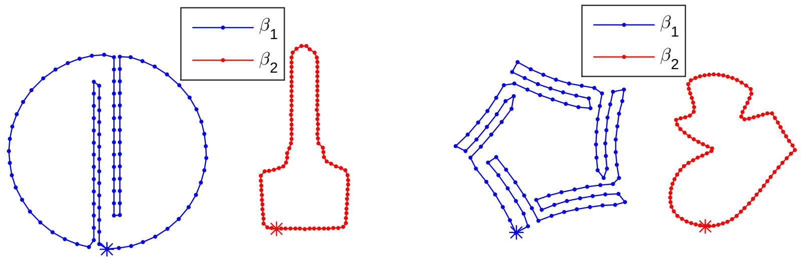

Figure 21: Successive points are connected by straight lines for display purposes.

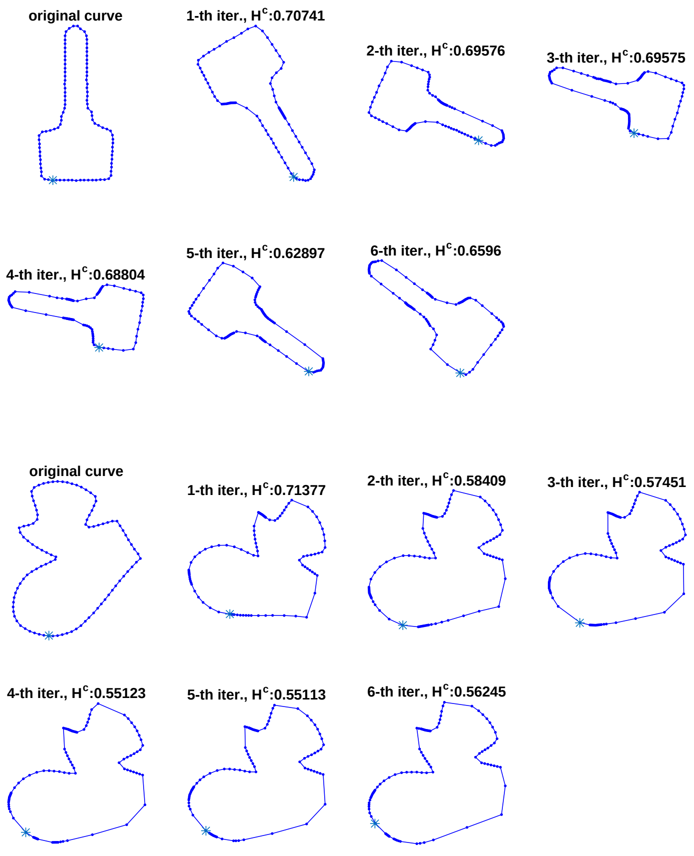
original curve
1-th iter., H$^c$:0.70741
2-th iter., H$^c$:0.69576
3-th iter., H$^c$:0.69575
4-th iter., H$^c$:0.68804
5-th iter., H$^c$:0.62897
6-th iter., H$^c$:0.6596
original curve
1-th iter., H$^c$:0.71377
2-th iter., H$^c$:0.58409
3-th iter., H$^c$:0.57451
4-th iter., H$^c$:0.55123
5-th iter., H$^c$:0.55113
6-th iter., H$^c$:0.56245

Figure 22: For display purposes, the $\beta_2$ is re-sampled such that points on uniformly-spaced are displayed, and successive points are connected by straight lines. The cost function values are given in the titles.

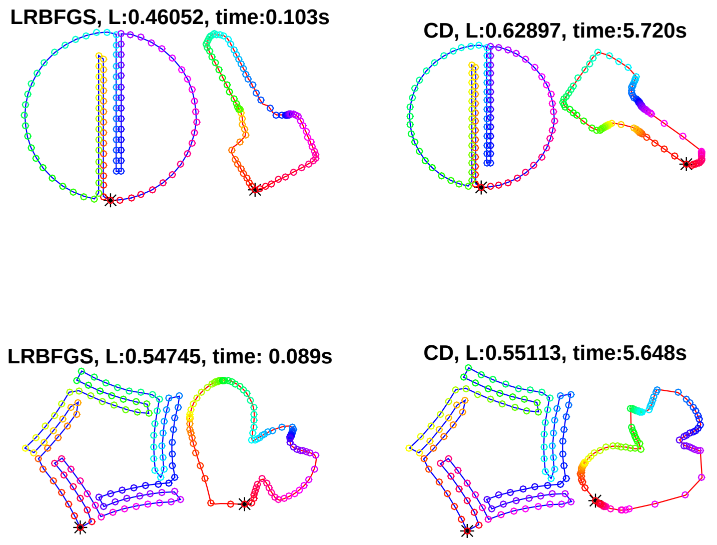

Figure 23: Matching curves obtained by LRBFGS and Algorithm 4. The color of points on the two curves represents correspondence between two curves. The cost function values and computational time are given in the titles.

computing the elastic distance metric and developed an efficient implementation using various Riemannian optimization algorithms.

Empirical comparisons of the Riemannian approach using LRBFGS with CD1H using shapes from the MPEG-7 and Flavia datasets were performed. The results highlight the difficulties and inaccuracies associated with CD1H and demonstrate that the Riemannian approach produces more accurate distance estimates in significantly less time since the computational time required adapts appropriately to the complexity of the shapes compared. The nonuniform grid coordinate descent method Algorithm 4 was verified to improve the final cost function value compared to CD1H but only at a significant computational cost compared to CD1H and therefore also the Riemannian approach.

The efficiency and efficacy of the Riemannian approach to computing the elastic distance metric promises to improve substantially shape analysis computations that are based upon distance, e.g., the Karcher mean of a set of shapes, geodesic paths between shapes, and inferences on shapes. These improvements will be demonstrated in future work.

## 8 Acknowledgements

The authors are grateful to the careful reading and helpful comments and suggestions of three anonymous referees, especially, for the suggestion by one referee to include the discussion of the nonuniform-grid Coordinate Descent algorithm.

## References

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds.* Princeton University Press, Princeton, NJ, 2008.

[Bak08] C. G. Baker. *Riemannian manifold trust-region methods with applications to eigenproblems.* PhD thesis, Florida State University, Department of Computational Science, 2008.

[Ber95] D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 1995.

[DM98] I. L. Dryden and K. V. Mardia. *Statistical shape analysis.* Wiley, 1998.

[HAG15] W. Huang, P.-A. Absil, and K. A. Gallivan. A Riemannian symmetric rank-one trust-region method. *Mathematical Programming*, 150(2):179–216, February 2015.

[HGA15] Wen Huang, K. A. Gallivan, and P.-A. Absil. A Broyden Class of Quasi-Newton Methods for Riemannian Optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015.

[HGSA14] W. Huang, K. A. Gallivan, A. Srivastava, and P.-A. Absil. Riemannian optimization for elastic shape analysis. In *Proceedings of the 21st Internaltional Symposium on Mathematical Theory of Networks and Systems (MTNS 2014)*, 2014.

[Hua13] W. Huang. *Optimization algorithms on Riemannian manifolds with applications.* PhD thesis, Florida State University, Department of Mathematics, 2013.

[Ken84] D. G. Kendall. Shape manifolds, Procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, March 1984.

[KSMJ04] E. Klassen, 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*, 26(3):372–83, March 2004. doi:10.1109/TPAMI.2004.1262333.

[LRK15] S. Lahiri, D. Robinson, and E. Klassen. Precise matching of PL curves in $\mathbb{R}^n$ in the square root velocity framework. pages 1–41, January 2015. arxiv:1501.00577.

[O'N83] B. O'Neill. *Semi-Riemannian geometry.* Academic Press Incorporated [Harcourt Brace Jovanovich Publishers], 1983.

[Qi11] C. Qi. *Numerical optimization methods on Riemannian manifolds.* PhD thesis, Florida State University, Department of Mathematics, 2011.

[RW12] W. Ring and B. Wirth. Optimization methods on Riemannian manifolds and their application to shape space. *SIAM Journal on Optimization*, 22(2):596–627, January 2012. doi:10.1137/11082885X.

[SKJJ11] A. Srivastava, E. Klassen, S. H. Joshi, and I. H. Jermyn. Shape analysis of elastic curves in Euclidean spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1415–1428, September 2011. doi:10.1109/TPAMI.2010.184.

[SKK03] T. B. Sebastian, P. N. Klein, and B. B. Kimia. On aligning curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):116–125, January 2003. doi:10.1109/TPAMI.2003.1159951.

[Uni] Temple University. Shape similarity research project. www.dabi.temple.edu/ shape/MPEG7/dataset.html.

[WBX+07] S. G. Wu, F. S. Bao, E. Y. Xu, Y.-X. Wang, Y.-F. Chang, and Q.-L. Xiang. A leaf recognition algorithm for plant classification using probabilistic neural network. *2007 IEEE International Symposium on Signal Processing and Information Technology*, pages 11–16, 2007. arXiv:0707.4289v1.

[YHGA15] Y. You, W. Huang, K. A. Gallivan, and P.-A. Absil. A Riemannian Approach for Computing Geodesics in Elastic Shape Analysis. In *3rd IEEE Global Conference on Signal and Information Processing*, December 2015. to appear.

[YMSM08] L. Younes, P. Michor, J. Shah, and D. Mumford. A metric on shape space with explicit geodesics. *Rendiconti Lincei - Matematica e Applicazioni*, 9(1):25–57, 2008. doi:10.4171/RLM/506.

[You98] L. Younes. Computable elastic distances between shapes. *SIAM Journal on Applied Mathematics*, 58(2):565–586, April 1998. doi:10.1137/S0036139995287685.

## A Proofs

Proof of Theorem 2.1:

*Proof.* By definition, for any $\tilde{q} \in [q]_{\mathrm{SO}(n)\times\Gamma_s^o}$, there exist $\tilde{O} \in \mathrm{SO}(n)$ and $\tilde{\gamma} \in \Gamma_s^o$ such that $\tilde{q} = \tilde{O}(q, \tilde{\gamma})$. It follows from [LRK15, Lemma 11] that there exists a sequence $\{\gamma_i\} \subset \Gamma^o$ such that $(q, \gamma_i) \to (q, \tilde{\gamma})$ with respect to the $\mathbb{L}^2$ metric. Since $\mathrm{SO}(n)$ is isometric for $\mathbb{L}^2$, we have $\tilde{O}(q, \gamma_i) \to \tilde{O}(q, \tilde{\gamma})$, which implies that $\tilde{q} \in \overline{[q]}$. Therefore, we obtain $[q]_{\mathrm{SO}(n)\times\Gamma_s^o} \subseteq \overline{[q]}$. We finished proving the first statement.

For any $v \in \overline{[q]}$, there is a sequence of $\{O_i\}$ and $\{\gamma_i\}$ such that $O_i(q, \gamma_i) \to v$ with respect to the $\mathbb{L}^2$ metric. Since $\mathrm{SO}(n)$ is a compact set, there exists a convergent subsequence of $\{O_i\}$, i.e., $\{O_{i_j}\} \subseteq \{O_i\}$ and $O_{i_j} \to \tilde{O}$ with respect to 2-norm. Let $\tilde{q}$ denote $\tilde{O}^T v$. It follows that $O_{i_j}(q, \gamma_{i_j}) \to \tilde{O}\tilde{q}$. We have

$$\begin{aligned} &\|O_{i_j}(q, \gamma_{i_j}) - \tilde{O}\tilde{q}\|_{\mathbb{L}^2} = \|O_{i_j}(q, \gamma_{i_j}) - \tilde{O}(q, \gamma_{i_j}) + \tilde{O}(q, \gamma_{i_j}) - \tilde{O}\tilde{q}\|_{\mathbb{L}^2} \\ \geq\ &\|\tilde{O}(q, \gamma_{i_j}) - \tilde{O}\tilde{q}\|_{\mathbb{L}^2} - \|O_{i_j}(q, \gamma_{i_j}) - \tilde{O}(q, \gamma_{i_j})\|_{\mathbb{L}^2} = \|(q, \gamma_{i_j}) - \tilde{q}\|_{\mathbb{L}^2} - \|O_{i_j} - \tilde{O}\|_2 \|q\|_{\mathbb{L}^2}. \end{aligned}$$

It follows that

$$\|O_{i_j} - \tilde{O}\|_2 \|q\|_{\mathbb{L}^2} + \|O_{i_j}(q, \gamma_{i_j}) - \tilde{O}\tilde{q}\|_{\mathbb{L}^2} \geq \|(q, \gamma_{i_j}) - \tilde{q}\|_{\mathbb{L}^2}, \tag{A.1}$$

which implies $(q, \gamma_{i_j}) \to \tilde{q}$. Since $q^{-1}(0_n)$ has measure zero, it follows from [LRK15, Corollary 3] that there exists $\tilde{\gamma} \in \Gamma_s^o$ such that $\tilde{q} = (q, \tilde{\gamma})$. Therefore, $v = \tilde{O}(q, \tilde{\gamma}) \in [q]_{\mathrm{SO}(n)\times\Gamma_s^o}$, which implies $\overline{[q]} \subseteq [q]_{\mathrm{SO}(n)\times\Gamma_s^o}$. □

Proof of Theorem 2.2:

*Proof.* By definition, for any $\tilde{q} \in [q]_{\mathrm{SO}(n)\times\Gamma_s^c}$, there exist $\tilde{O} \in \mathrm{SO}(n)$ and $(\tilde{m}, \tilde{\gamma}) \in \Gamma_s^c$ such that $\tilde{q} = \tilde{O}((q, \tilde{m}), \tilde{\gamma})$. It follows from [LRK15, Lemma 11] that there exists a sequence $\{\gamma_i\} \subset \Gamma^o$ such that $((q, \tilde{m}), \gamma_i) \to ((q, \tilde{m}), \tilde{\gamma})$ with respect to the $\mathbb{L}^2$ metric. Since $\mathrm{SO}(n)$ is isometric for $\mathbb{L}^2$, we have $\tilde{O}((q, \tilde{m}), \gamma_i) \to \tilde{O}((q, \tilde{m}), \tilde{\gamma})$, which implies that $\tilde{q} \in \overline{[q]}$. Therefore, we obtain $[q]_{\mathrm{SO}(n)\times\Gamma_s^c} \subseteq \overline{[q]}$. We finished proving the first statement.

For any $v \in \overline{[q]}$, there is a sequence of $\{O_i\}$, $\{m_i\}$ and $\{\gamma_i\}$ such that $O_i((q, m_i), \gamma_i) \to v$ with respect to the $\mathbb{L}^2$ metric. Since $\mathrm{SO}(n)$ and $[0, 1]$ are compact sets, there exists a convergent subsequence of $\{O_i\}$ and $m_i$, i.e., $\{O_{i_j}\} \subseteq \{O_i\}$ and $O_{i_j} \to \tilde{O}$ with respect to 2-norm and $\{m_{i_j}\} \subseteq \{m_i\}$ and $m_{i_j} \to \tilde{m}$. Let $\tilde{q}$ denote $\tilde{O}^T(v, -\tilde{m})$. It follows that $O_{i_j}((q, m_{i_j}), \gamma_{i_j}) \to \tilde{O}(\tilde{q}, \tilde{m})$.

Proceeding as (A.1), we have

$$\|O_{i_j} - \tilde{O}\|_2 \|(q, m_{i_j})\|_{\mathbb{L}^2} + \|O_{i_j}((q, m_{i_j}), \gamma_{i_j}) - \tilde{O}(\tilde{q}, \tilde{m})\|_{\mathbb{L}^2} \geq \|((q, m_{i_j}), \gamma_{i_j}) - (\tilde{q}, \tilde{m})\|_{\mathbb{L}^2}. \tag{A.2}$$

It holds that

$$\begin{aligned}\|((q, m_{i_j}), \gamma_{i_j}) - (\tilde{q}, \tilde{m})\|_{\mathbb{L}^2} &= \|((q, m_{i_j}), \gamma_{i_j}) - ((q, \tilde{m}), \gamma_{i_j}) + ((q, \tilde{m}), \gamma_{i_j}) - (\tilde{q}, \tilde{m})\|_{\mathbb{L}^2} \\ &\geq \|((q, \tilde{m}), \gamma_{i_j}) - (\tilde{q}, \tilde{m})\|_{\mathbb{L}^2} - \|(q, m_{i_j}) - (q, \tilde{m})\|_{\mathbb{L}^2}. \end{aligned} \tag{A.3}$$

Since $q$ is absolutely continuous, $m_{i_j} \to \tilde{m}$ implies $\|(q, m_{i_j}) - (q, \tilde{m})\|_{\mathbb{L}^2} \to 0$. Therefore, using (A.2) and (A.3) yields $((q, \tilde{m}), \gamma_{i_j}) \to (\tilde{q}, \tilde{m})$. Since $q^{-1}(0_n)$ has measure zero, $(q, \tilde{m})^{-1}(0_n)$ also has measure zero. It follows from [LRK15, Corollary 3] that there exists $\tilde{\gamma} \in \Gamma_s^o$ such that $(\tilde{q}, \tilde{m}) = ((q, \tilde{m}), \tilde{\gamma})$. Therefore, $v = \tilde{O}((q, \tilde{m}), \tilde{\gamma}) \in [q]_{\mathrm{SO}(n)\times\Gamma_s^c}$, which implies $\overline{[q]} \subseteq [q]_{\mathrm{SO}(n)\times\Gamma_s^c}$. □

Proof of Lemma 4.2:

*Proof.* The cost function $L(O, m, l)$ is equal to

$$2 - 2\int_0^1 \mathrm{trace}(q_2\left(\rho_{l,m}(t)\right) l(t) q_1(t)^T O^T) dt + \omega \int_0^1 \left(l^2(t) + \frac{1}{l^2(t)}\right)\sqrt{1 + l^4(t)}dt.$$

Consider the cost function defined on the embedding manifold

$$\begin{aligned}&\bar{L}(O, m, l) : \mathbb{R}^{n\times n} \times \mathbb{R} \times \mathbb{L}^2 \to \mathbb{R} : (O, m, l) \mapsto \\ &2 - 2\int_0^1 \mathrm{trace}(q_2\left(\rho_{l,m}(t)\right) l(t) q_1(t)^T O^T) dt + \omega \int_0^1 \left(l^2(t) + \frac{1}{l^2(t)}\right)\sqrt{1 + l^4(t)}dt.\end{aligned}$$

The gradient for the variable $O$ is

$$\nabla_O \bar{L}(O, m, l) = -2\int_0^1 q_2\left(\rho_{l,m}(t)\right) l(t) q_1(t)^T dt \in \mathbb{R}^{n\times n}.$$

The gradient for the variable $m$ is

$$\nabla_m \bar{L}(O, m, l) = -2\int_0^1 \langle O q_1(t), l(t) q_2'\left(\rho_{l,m}(t)\right)\rangle_2 dt.$$

The gradient for the variable $l$ is not easy to compute directly. First, consider the directional derivative along $v \in \mathrm{T}_l\, \mathcal{L}$.

$$\begin{aligned}\mathrm{D}_l\, \bar{L}(O,m,l)[v] =& -2\int_0^1 \Big\langle Oq_1(t), v(t)q_2\left(\rho_{l,m}(t)\right) + 2l(t)q_2'\left(\rho_{l,m}(t)\right)\int_0^t l(s)v(s)ds\Big\rangle_2 dt \\ &+ \omega\int_0^1 2v(t)l(t)(2-1/l^4(t))\sqrt{1+l^4(t)}dt.\end{aligned}$$

Simplifying, we have

$$\begin{aligned}\mathrm{D}_l\, \bar{L}(O,m,l)[v] =& -2\int_0^1 \langle Oq_1(t), q_2\left(\rho_{l,m}(t)\right)\rangle_2 v(t)dt \\ &-2\int_0^1 \langle Oq_1(t), 2l(t)q_2'\left(\rho_{l,m}(t)\right)\rangle_2 \int_0^t l^3(s)v(s)dsdt \\ &+2\omega\int_0^1 l(t)(2-1/l^4(t))\sqrt{1+l^4(t)}v(t)dt.\end{aligned}$$

If

$$\begin{aligned}x(t) &= \langle Oq_1(t), q_2\left(\rho_{l,m}(t)\right)\rangle_2 \\ y'(t) &= \langle Oq_1(t), 2l(t)q_2'\left(\rho_{l,m}(t)\right)\rangle_2 \\ z(t) &= \omega l(t)(2-1/l^4(t))\sqrt{1+l^4(t)},\end{aligned}$$

then

$$\begin{aligned}\mathrm{D}_l\, \bar{L}(O,m,l)[v] - \langle 2z(t), v(t)\rangle_{\mathbb{L}^2} &= -2\int_0^1 x(t)v(t)dt - 2\int_0^1 y'(t)\int_0^t l(s)v(s)dsdt \\ &= -2\int_0^1 x(t)v(t)dt - 2\Big(y(t)\int_0^t l(s)v(s)ds|_0^1 - \int_0^1 y(t)l(t)v(t)dt\Big) \text{ (integration by parts)} \\ &= -2\int_0^1 x(t)v(t)dt + 2\int_0^1 y(t)l(t)v(t)dt \quad (\text{by } v\in \mathrm{T}_l\,\mathcal{L}) \\ &= \int_0^1 (2y(t)l(t)-2x(t))v(t)dt = \langle 2yl-2x, v\rangle_{\mathbb{L}^2}.\end{aligned}$$

Since the gradient is the vector that satisfies

$$D_l\bar{L}(O,m,l)[v] = \langle \nabla_l \bar{L}(O,m,l), v(t)\rangle_{\mathbb{L}^2},$$

we obtain

$$\nabla_l \bar{L}(O,m,l) = 2y(t)l(t) - 2x(t) + 2z(t).$$

Finally, the Riemannian gradient is given by projecting each component of $\bar{L}(O,m,l)$ to its associated manifold. □