LEARNING FROM LANDMARKS, CURVES, SURFACES, AND SHAPES IN GEOMSTATS

LUÍS F. PEREIRA, ALICE LE BRIGANT, ADELE MYERS, EMMANUEL HARTMAN, AMIL KHAN, MALIK TUERKOEN, TREY DOLD, MENGYANG GU, PABLO SUÁREZ-SERRATO, AND NINA MIOLANE

Abstract. We introduce the `shape` module of the Python package Geomstats to analyze shapes of objects represented as landmarks, curves and surfaces across fields of natural sciences and engineering. The `shape` module first implements widely used shape spaces, such as the Kendall shape space, as well as elastic spaces of discrete curves and surfaces. The `shape` module further implements the abstract mathematical structures of group actions, fiber bundles, quotient spaces and associated Riemannian metrics which allow users to build their own shape spaces. The Riemannian geometry tools enable users to compare, average, interpolate between shapes inside a given shape space. These essential operations can then be leveraged to perform statistics and machine learning on shape data. We present the object-oriented implementation of the `shape` module along with illustrative examples and show how it can be used to perform statistics and machine learning on shape spaces.

1. Introduction

Geomstats [41] is an open-source Python package for statistics and learning from data that belong to *manifolds*, *i.e.*, to nonlinear generalizations of vector spaces.

Analyzing data on manifolds. Manifolds arise in many applications. Hyperspheres model directional data in molecular and protein biology [27, 20, 63]. Hyperbolic spaces have recently gained interest to represent graph and hierarchical data across applications of computer vision [14, 38]. The manifold of symmetric positive-definite (SPD) matrices characterizes data from diffusion tensor imaging (DTI) [48, 65, 47] and functional magnetic resonance imaging (fMRI) [54, 16]. The manifolds implemented in Geomstats come equipped with mathematical structures, such as Riemannian metrics, that allow users to process data belonging to them. For example, users can compute a distance between two data points on a manifold. Geomstats also provides statistical learning algorithms that are compatible with the manifold structures, *i.e.*, that can be applied to data belonging to any of the implemented manifolds. These algorithms are typically generalizations of traditional estimation, clustering, dimension reduction, classification and regression methods to nonlinear manifolds. For example, one algorithm called geodesic regression is the generalization of the traditional linear regression, but for data belonging to manifolds.

Analyzing shape data on shape spaces. Shapes are a type of complex data that belong to nonlinear generalizations of vector spaces, called *shape spaces*, including manifolds. Shape

data are ubiquitous across fields of science and engineering. In molecular biology, the relationship between the shapes and functions of proteins is an active area of research. The statistical analysis of protein shapes, *e.g.*, protein misfolding, helps understand illnesses such as Parkinson's disease [24]. Integrating advanced imaging techniques, such as cryogenic electron microscopy (cryo-EM), with image reconstruction tools, made it possible to determine versatile 3D structures of protein at near-atomic resolution [32]. In evolutionary biology, the shape of monkey skulls, in combination with ecological and biogeographic data, is analyzed by paleontologists, gaining insights into evolutionary changes covering multiple geographies [13]. In the medical realm, before an operation, orthopaedic surgeons analyze the shape of bones and then plan the surgery accordingly [9]. In computer vision, shape analysis of biological structures, for instance, seen as elements within shape manifolds, has gained traction for a long time [11, 64]. In recent years, machine learning tools, such as U-Net [50], have demonstrated their competitive performance in image segmentation, which can provide massive high-quality shapes of different objects, such as cells [59], tissues and organism [52]. The enhanced image segmentation methods hold great promise for probing disease progression such as fibrosis [34], by integrating cell shape, orientation and dynamics into the analysis [61, 36]. Similarly, in neuroimaging, researchers leverage the shape analysis of brain structure morphology, registered in MRI scans, advancing our comprehension of pathologies such as Alzheimer's disease [35]. Thus, tools for processing and analyzing shape data can advance a broad range of scientific and engineering fields.

Implementing shape spaces in Geomstats. To process and analyze shape data, we introduce the `shape` module of the Python package Geomstats. Shape spaces in the `shape` module are implemented with the same design that drives the implementation of existing manifolds in Geomstats. Consequently, users can process and analyze shape data in the same way that they process and analyze manifold data. Specifically, the proposed `shape` module focuses on mathematical models of shapes where these are defined as the features of an *object* that are invariant under certain *transformations* [12, 53]. In other words, the shape of a set of landmarks, the shape of a curve, or the shape of a surface are defined as the remainder after we have filtered out the position and the orientation of the object—or more generally after we have filtered out any transformations that do not change the shape of the object. Mathematically, this framework represents objects as elements of a fiber bundle equipped with the action of a group of transformations and shapes as elements of a quotient space that removes the group's action.

Examples of shapes and shape spaces that fall into this mathematical framework include the Kendall shape spaces [26], quotient spaces from Procrustes analysis [11], and the elastic geometry of discrete curves [45, 39, 55] and surfaces [23, 2].

Related Works. There are two main approaches to shape analysis: the extrinsic approach, where a shape is mapped to another by deforming the ambient space [4], and the intrinsic approach, where the deformations are defined on the shapes themselves. Computational tools

for the first approach have been proposed, see e.g. [17] and the Python package Scikit-Shapes[1]. Here, we focus on the second, intrinsic approach. While implementations of some tools are available in C, Python and Matlab [12, 62, 49, 2, 23], to the best of our knowledge, there exists no wide-ranging open source Python implementation of intrinsic shape analysis, *i.e.*, no implementation that tackles intrinsic metrics on shape spaces of landmarks sets, curves, and surfaces in a consistent mathematical framework.

Contributions. This paper presents the `shape` module of Geomstats. Leveraging the fact that shape spaces bear crucial similarities thanks to their common quotient space structure, the module implements the differential geometry of object shapes and shape spaces, in particular: their group actions, fiber bundles, Riemannian and quotient structures. The implementation of object spaces and shape spaces is compatible with the main statistical learning algorithms of Geomstats' existing `learning` module. Thus, the proposed `shape` module unlocks the capacity to run learning algorithms on object data and shape data. As in the rest of Geomstats, the implementation of the `shape` modules is object-oriented and extensively unit-tested. All operations are vectorized for batch computation and support is provided for different execution backends — namely NumPy [21], Autograd [37], and PyTorch [46].

Outline. The rest of the paper is organized as follows. Section 2 reviews the mathematics of and implementation of the main package Geomstats. Then, Section 3 introduces the structure of `shape` module, *i.e.*, the abstract Python classes used to define objects and their shapes. Section 4 details the geometries of the concrete object and shape spaces implemented in the module, specifically objects and shapes of landmarks, curves and surfaces. This section includes code illustrations and examples of real-world use cases in the literature. Altogether, the proposed `shape` module represents the first comprehensive implementation of mathematical models of objects and their shapes in Python.

2. Background: Differential Geometry in Geomstats

In this section, we review the mathematics and design of the main package Geomstats required to understand the mathematics and design of the `shape` module. The package Geomstats implements concepts of differential geometry and Riemannian geometry following an object-oriented structure. Abstract Python classes represent high-level mathematical concepts, such as `Manifold` (Subsection 2.1), `Connection` (Subsection 2.2) and `RiemannianMetric` (Subsection 2.3). Their child Python classes then implement tangible manifolds, such as `Hypersphere`.

Here we give an informal introduction to Riemannian geometry. For more details, we refer the interested reader to standard textbooks such as [10, 31] and to [19] for its original implementation in Geomstats.

[1] `https://github.com/scikit-shapes/scikit-shapes`

2.1. **Manifold.** A manifold is a space that locally resembles a vector space, without necessarily having its global flat structure. The simplest examples of manifolds are Euclidean spaces $\mathbb{R}^d$, or more generally vector spaces in finite dimensions, open sets of vector spaces and level sets of functions. A d-dimensional manifold M admits a *tangent space* T_pM at each point $p \in M$ that is a d-dimensional vector space. The set of all tangent spaces to the manifold is called the tangent bundle and denotes TM. A manifold with a smooth differential structure is called a smooth manifold.

2.1.1. *Manifold in Geomstats.* The `Manifold` abstract Python class implements the structure of a *manifold*. Inheritance of Python classes allows us to specialize implementations. For example, `VectorSpace`, `OpenSet` and `LevelSet` are abstract classes inheriting from the parent class `Manifold`. Additionally, composition of Python classes allows us to combine already-existing structures. For example, `ProductManifold` is a Python class that represents a manifold created as the product of two or more existing manifolds. Similarly, `NFoldManifold` is a class that represents a manifold created as the product of a base manifold repeated n times.

The parent class `Manifold` and its child classes contain methods that allow users to process data points on manifolds. For example, a user can verify that a given data point indeed belongs to the manifold via the `belongs()` method and that a given input is a tangent vector to the manifold at a given base point via `is_tangent()`. Additionally, a user can generate random data points and tangent vectors to the manifold with the methods `random_point()` and `random_tangent_vec()` respectively. The latter two methods will be particularly relevant for the unit-testing framework.

2.2. **Connection.** An *affine connection* is a mathematical structure that allows us to define the generalization of straight lines, addition and subtraction to nonlinear manifolds, respectively called geodesics, exponential map and logarithm map. To this end, a connection allows us to take derivatives of vector fields, *i.e.*, mappings $V : M \to TM$ that associate to each point p a tangent vector $V(p) \in T_pM$. The derivative induced by the connection ∇ is referred to as *covariant derivative*. In a local coordinate system along the manifold, the coefficients of the connection are called the Christoffel symbols. Mathematical details on connections, covariant derivatives, and Christoffel symbols can be found in [10].

2.2.1. *Geodesics.* Consider $t \mapsto \gamma(t)$ a curve on M, parameterized by time t. Its velocity $t \mapsto \dot{\gamma}(t)$ is a vector field along γ, *i.e.*, $\dot{\gamma}(t) \in T_{\gamma(t)}M$ for all t. The acceleration of a curve is, by definition, the covariant derivative of this velocity field with respect to the affine connection ∇. A curve γ of zero acceleration

$$\nabla_{\dot{\gamma}}\dot{\gamma} = 0, \tag{1}$$

is called a ∇-*geodesic*. Geodesics are the manifolds counterparts of vector spaces' straight lines, *i.e.*, their second derivative vanishes (in the sense of connections). Equation (1) becomes a system of ordinary differential equations (ODEs) that can be solved to find geodesics.

2.2.2. *Exponential and logarithm maps.* Existence results for solutions of ODEs allow us to define geodesics starting at a point p with velocity $v \in T_pM$ for times t in a neighborhood of zero, or equivalently for all time $t \in [0, 1]$ but for tangent vectors v of small norm. The *exponential map* at $p \in M$ associates to any $v \in T_pM$ of sufficiently small norm the end point $\gamma(1)$ of a geodesic γ starting from p with velocity v:

$$\exp_p(v) = \gamma(1), \quad \text{where} \begin{cases} \gamma \text{ is a geodesic,} \\ \gamma(0) = p,\ \dot{\gamma}(0) = v. \end{cases}$$

The map $\exp_p$ is a diffeomorphism in a neighborhood of 0 in T_pM, and its inverse $\log_p \equiv \exp_p^{-1}$ defines the *logarithm map*. The logarithm map then associates to any point q the velocity $v \in T_pM$ necessary to get to q when departing from p:

$$\log_p(q) = v \quad \text{where} \quad \exp_p(v) = q.$$

The exponential and logarithm maps can be seen as generalizations of the Euclidean addition and subtraction to nonlinear manifolds. Indeed, the exponential map *adds* a tangent vector to a point which outputs a point. Similarly, the logarithm map *subtracts* two points and outputs a tangent vector. Geodesics can in turn be expressed in terms of the exponential map: if a geodesic γ verifies $\gamma(0) = p$ and $\dot{\gamma}(0) = v$, then $\gamma(t) = \exp_p(tv)$.

2.2.3. *Connection in Geomstats.* The `Connection` class implements the structure of an *affine connection*. This class first contains the `christoffels()` method that implements the Christoffel symbols defining the connection. Just as a connection allows us to define geodesic equation and geodesics, the `Connection` class contains the `geodesic_equation()` method implementing Equation (1), as well as the `geodesic()` method computing geodesics either from initial conditions $\gamma(0)$, $\dot{\gamma}(0)$ or from boundary conditions $\gamma(0)$, $\gamma(1)$. Similarly, we find the `exp()` and `log()` methods for exponential and logarithm maps, respectively. We refer to [19] for additional details on the `Connection` class.

2.3. **Riemannian metric.** A *Riemannian metric* is a collection of inner products $(\langle \cdot, \cdot \rangle_p)_{p \in M}$ defined on the tangent spaces of a manifold M, that depend on the base point $p \in M$ and vary smoothly with respect to it.

2.3.1. *Levi-Civita Connection.* Given a Riemannian metric there exists a unique affine connection, called the *Levi-Civita connection*, which is the only affine connection that is symmetric and compatible with the metric, *i.e.*, that verifies

$$UV - VU = \nabla_U V - \nabla_V U$$
$$U\langle V, W \rangle = \langle \nabla_U V, W \rangle + \langle V, \nabla_U W \rangle$$

for all vector fields U, V, W. The geodesics of a Riemannian manifold are those of its Levi-Civita connection.

2.3.2. *Geodesic Distance.* The *geodesic distance* induced by the Riemannian metric is defined as the length of the shortest curve joining two points $p, q \in M$. Here, the length of a (piecewise) smooth curve $\gamma : [0, 1] \to M$ is computed by integrating the norm of its velocity using the norm induced by the Riemannian metric:

$$d(p, q) = \inf_{\gamma;\gamma(0)=p,\gamma(1)=q} L(\gamma), \quad \text{where} \quad L(\gamma) = \int_0^1 ||\dot{\gamma}(t)||_{\gamma(t)} dt.$$

In a Riemannian manifold, geodesics extend another property of straight lines: they are locally length-minimizing. In a geodesically complete manifold, any pair of points can be linked by a minimizing geodesic, not necessarily unique, and the distance between them can be computed using the logarithm map, written for all p, q in M as follows:

$$d(p, q) = || \log_p(q)||_p.$$

2.3.3. *Riemannian metric in Geomstats.* The abstract class `RiemannianMetric` implements the structure of a Riemannian metric. It is a child class of `Connection` and inherits all its methods, including `geodesic()`, `exp()` and `log()`. The class `RiemannianMetric` overwrites the `Connection` class' method `christoffels()` and computes the Christoffel symbols using derivatives of the metric by using automatic differentiation. The geodesics, by the compatibility property, have velocity of constant norm, *i.e.*, are parametrized by arc length. The `dist()` method implements the geodesic distance induced by the Riemannian metric.

Any `Manifold` can be equipped with a Riemannian metric through a `equip_with_metric()` method, *i.e.*, it sets `metric` as an attribute of `Manifold`. In other words, `equip_with_metric()` transforms a smooth manifold M into a Riemannian manifold (M, g). This method is particularly important since a manifold may be equipped with different metrics.

3. The Shape Module of Geomstats

We can now introduce the `shape` module of Geomstats. The architecture of the module follows an object-oriented design, consistent with the design of the main package Geomstats. Abstract Python classes represent high-level mathematical concepts, such as as `FiberBundle` (Subsection 3.1), `GroupAction` (Subsection 3.2), and `QuotientMetric` (Subsection 3.3) — summarized in Figure 1. In this section, we review the differential geometry of shape spaces which motivates the architecture of the `shape` module. The module is available in the three backends of Geomstats: NumPy, Autograd and PyTorch. Additionally, the code is extensively unit-tested, documented and included in the continuous integration pipeline and documentation website of the Geomstats library.

3.1. **Fiber Bundles.**

In shape analysis, a space of objects can be represented through the mathematical structure of a fiber bundle, which we introduce here.

Let M, B and F be smooth manifolds and suppose that $\pi : M \to B$ is smooth. The triple (π, M, B) is called *fiber bundle with total space M, base space B and fiber F* if:

i) π is surjective;

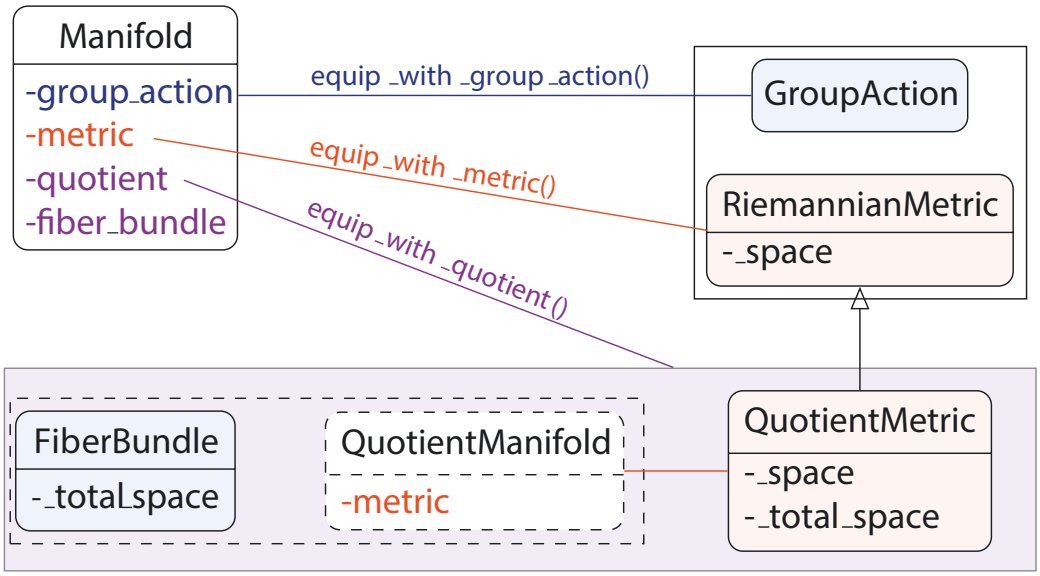

FIGURE 1. **Abstract Python classes of the `shape` module in Geomstats.** A manifold can be equipped with a group action and a Riemannian metric through the methods `equip_with_group_action()` and `equip_with_metric()`. When those are compatible, a quotient structure can be put on the manifold through the method `equip_with_quotient()`. The quotient structure consists of a fiber bundle, and a quotient manifold equipped with a quotient metric, which inherits from the class `RiemannianMetric` (arrow represents inheritance). `QuotientManifold` is dashed because it may be the same class (but different instance) as `Manifold` when points are represented via their representatives. The variables listed under each Python class represent the main attributes of an object instantiated from this class.

ii) there exists an open cover $(U_i)_{i\in I}$ of B and diffeomorphisms

$$h_i : \pi^{-1}(U_i) \to U_i \times F$$

such that $h_i(\pi^{-1}(p)) = \{p\} \times F$.

Informally, the total space M of a fiber bundle *locally* looks like a product of the base B with the fiber F. In the context of shape analysis, the total space M will typically represent a space of objects, whereas the space B will represent a space of these objects' shapes. The map π represents the extraction of a shape from an object, and $\pi^{-1}(p)$ represents all objects that have shape p.

3.1.1. *Fiber bundles in the `shape` module.* The Python class `FiberBundle` implements the differential geometry of fiber bundles in the `shape` module — see Figure 1. In particular, it implements methods that transform points from the base space B to the total space M,

Method	Description
`riemannian_submersion()`	$M \to B$
`lift()`	$B \to M$
`tangent_riemannian_submersion()`	$T_pM \to T_{\pi(p)}B$
`horizontal_lift()`	$T_{\pi(p)}B \to T_pM$
`horizontal_projection()`	$T_pM \to H_p$
`vertical_projection()`	$T_pM \to V_p$
`align()`	$G \cdot p \to G \cdot p$

TABLE 1. **Main methods for the Python class `FiberBundle`.** Abbreviations: M: total space, B: base space, π: Riemannian submersion, T_pM: tangent space at $p \in M$, H_p: horizontal subspace at $p \in M$, V_p: vertical tangent space at $p \in M$, G: group acting on M.

and vice-versa. These methods are summarized in Table 1, and are discussed in the next subsections in more details.

3.2. **Group Actions.** In shape analysis, objects can be transformed by translations, rotations, and other geometric transformations. These ideas are mathematically formulated through the concepts of groups and group actions — which we introduce now. A group is a pair $(G,\ \circ)$ where G is a set and $\circ : G \times G \mapsto G$ is an associative multiplication with an identity element $\mathrm{id} \in G$ and such that any $\phi \in G$ has an inverse $\phi^{-1} \in G$. A Lie group $(G, \circ)$ is a group that is also equipped with a smooth manifold structure, such that both the group action $\circ : G \times G \to G$ and map $G \to G$, $\phi \mapsto \phi^{-1}$ are smooth.

A Lie group $(G, \circ)$ *acts* on a smooth manifold M, if there exists a smooth map $\cdot : G \times M \to M$ such that

$$\phi_1 \cdot (\phi_2 \cdot p) = (\phi_1 \circ \phi_2) \cdot p \quad \text{for all } \phi_1, \phi_2 \in G, \text{ and } p \in M.$$

The Lie group G is said to act *freely* on M if $\phi \cdot p \neq p$ for all $\phi \neq \mathrm{id} \in G$ and $p \in M$ and is said to act *properly* on M if for any compact $K \subset M$ the set

$$G_K = \{\phi \in G \,|\, \phi K \cap K \neq \emptyset\}$$

is relatively compact in G (i.e. $\overline{G}_K \subset G$ is compact), where ϕK is the image of the map $\phi : K \to G$, $p \mapsto \phi \cdot p$.

3.2.1. *Group actions in the `shape` module.* The Python class `LieGroup` of Geomstats implements the mathematical structure of Lie groups. This class contains for example the method `compose()` that represents the composition operation $\circ$ of the group. In the `shape` module specifically, the Python class `GroupAction` implements the action of a group element ϕ on a point p of the manifold — see Figure 1. Analogously to equipping a manifold with a metric, `equip_with_group_action()` method allows to endow a `Manifold` with a group action.

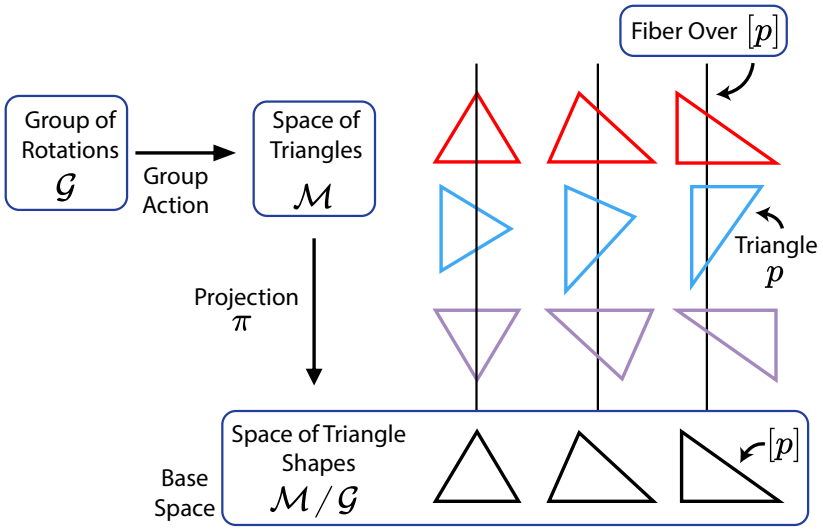

FIGURE 2. **Example of fiber bundle induced by a group action.** The group of rotations G acting on the space of triangles M induces a fiber bundle with total space M, base space M/G and fiber G. The fiber over $[p]$ is the orbit $p \cdot G$ composed of all triangles that can be obtained by rotating p. They all project on the same shape $[p] \in M/G$.

3.3. **Quotient Space.** In shape analysis, the shapes of objects can be represented by points in a quotient space, defined next. Consider a group G that acts on a point p of a manifold M: this will reach a set of points $G \cdot p$, called the *orbit* of $p \in M$ through the action of G. This orbit defines an equivalence class, that we denote $[p]$. By definition, the set of equivalence classes is called a *quotient space* and denoted M/G. A group G acting on M defines a fiber bundle with total space M, base space M/G, and fiber G. Each orbit $G \cdot p$ is homeomorphic to G, and is therefore called the fiber over $[p]$.

For example, as shown in Figure 2, M can be a set of objects such as a set of triangles and G can be the group of rotations and translations. In this scenario, p is a triangle, and $[p]$ represents all triangles that are obtained from p by rotations and translations. Intuitively, $[p]$ defines a notion of shape: the shape of a triangle p is defined as all the triangles that have the same shape as p.

3.3.1. *Quotient Space in the* `shape` *module.* In order to compute with shape data, we need to compute with points in a quotient space. Points in a quotient space M/G are equivalence classes. In general, there is no obvious way to explicitly implement an equivalence class in the computer. Consequently, we choose to (implicitly) represent an equivalence class $[p]$ of M/G as an element $p \in [p]$ of the total space M. In doing so, p is called the *representative* of the class. All quotient spaces presented in the next subsection represent points via representatives in the total space M.

When points on the quotient space are represented via representatives, we can use the fundamental operations of Riemannian geometry of the total space to compute in the quotient space.

3.4. **Quotient Metric.** We need a notion of compatibility between the group action and the metric structure in order to define a distance on the quotient space. We say that G *acts isometrically on a Riemannian manifold* (M, g) if for the action $\cdot : G \times M \to M$, one has that

$$d(p, q) = d(\phi \cdot p, \phi \cdot q) \quad \text{for } p, q \in M \quad \text{and } \phi \in G.$$

Equivalently, we say that the metric is G-invariant, or that the metric and the group action are compatible. When the Riemannian metric and the group action by G are compatible, we obtain a definition of distance on the quotient space M/G :

$$d_{M/G}([p], [q]) = \inf_{\phi \in G} d_M(p, \phi \cdot q). \tag{2}$$

In our example with triangles, the distance between the shape $[p]$ of the triangle p and the shape $[q]$ of the triangle q is the minimum of the distances in object space between the triangle p and all possible rotations and translations of the triangle q.

If G acts freely and properly on M, the distance (2) is induced by a Riemannian metric on the quotient space, such that the projection π is a Riemannian submersion, *i.e.*, an orthogonal projection along the fibers. The space $V_p := \ker d\pi_p$ is called the *vertical subspace* of T_pM and is composed of vectors of T_pM that are tangent to the fibers; following such vectors amounts to staying in the same equivalence class. In the context of shape analysis, vertical vectors correspond to infinitesimal perturbations of landmarks, curves or surfaces that do not change the shape. The orthogonal complement of the vertical subspace in T_pM is called the *horizontal subspace* and denoted H_p.

3.4.1. *Quotient Metric in the* `shape` *module.* The Python class `QuotientMetric` inherits from `RiemannianMetric`, overriding most of the methods with general quotient space-specific implementations that take advantage of the methods available in `FiberBundle` and the total space metric.

When a total space M is equipped with a metric and a group action, we can use the method `equip_with_quotient()` to create a quotient space (see Figure 1). This method instantiates a base space equipped with a quotient metric and equips the total space with a fiber bundle. A quotient registry maps total spaces, metrics and group actions to base spaces, fiber bundles and quotient metrics. The function `register_quotient()` allows users to register their own quotient structures.

3.5. **Alignment.** The definition of distance in the quotient space M/G in Eq. (2) relies on a notion of alignment. Indeed, in this equation, we pick the best representative of an orbit $G \cdot q = [q]$ with respect to the point p, *i.e.*, we pick the point on the orbit $[q]$ for which the

(total space) distance to p is minimized. Computing a distance between two points in M/G relies on our ability to run an alignment procedure.

On the one hand, the alignment procedure is straightforward when the group G acting on the manifold is finite-dimensional. In this case, an explicit representation of the group elements exists. Consequently, the alignment can be solved by a gradient-based minimization of the (squared) distance to p over the group. To this aim, we can leverage the automatic differentiation capability of Geomstats; though closed-form solutions are implemented for particular cases (*e.g.*, rotations acting on matrices).

On the other hand, the alignment procedure is very challenging when the group is infinite-dimensional. Such an infinite-dimensional group arises in the context of the shape spaces of curves and surfaces in the next section: the group of reparametrizations. A first solution consists in discretizing the group [28, 60] and/or the action of the group [39]. However, this approach is limited in scope (*e.g.*, it requires surfaces to be of genus-zero in the case of shapes of surfaces) or leads to computationally expensive algorithms (*e.g.*, dynamic programming-based algorithm to align discrete curves). There exists an alternative solution that avoids having an explicit parameterization of the group. In this approach, we can formulate the geodesic boundary value problem on the quotient space as a relaxed numerical optimization problem involving an orbit membership enforcing term $\Gamma\left(\gamma(1), q\right)$ [1], *i.e.*, the geodesic γ between a point p and the (approximate) orbit of a point q is the path resulting from

$$\underset{\gamma \in \mathcal{P}_p}{\operatorname{argmin}} E(\gamma) + \lambda \Gamma\left(\gamma(1), q\right) \tag{3}$$

where $\mathcal{P}_p$ is the space of paths starting at p, and $E(\gamma)$ is the Riemannian energy of path γ. This approach is closely related to the inexact formulation of the LDDMM (Large Diffeomorphic Deformation Metric Mapping) framework, where the relaxation term is known as fidelity or data attachment term [7]. It has been introduced in the setting of discrete curves [1], and later extended to the setting of discrete surfaces [2]. This formulation admits a symmetrized version:

$$\underset{\gamma \in \mathcal{P}}{\operatorname{argmin}} E(\gamma) + \lambda_0 \Gamma\left(\gamma(0), p\right) + \lambda_1 \Gamma\left(\gamma(1), q\right) \tag{4}$$

3.5.1. *Alignment in the* `shape` *module.* The numerical procedure associated with the alignment approach is implemented in the `align()` method of the `FiberBundle` class, see Table 1. An abstract Python class `AlignerAlgorithm` can be passed to `FiberBundle` to implement `align()` in the general case, when no closed-form exists. If there exists a closed-form solution to the alignment problem, then it is implemented by overriding the `align()` method of `FiberBundle`.

Together with the the horizontal and vertical projection of a tangent vector, the alignment procedure is central to all computations in the quotient space. We already saw how the distance between two points amounts to aligning them and computing the distance using the total space metric. Additionally, the exponential map of a tangent vector at a base point amounts to horizontally projecting the tangent vector and shooting using the total space

metric. Similarly, the logarithm map of a point at a base point amounts to aligning point to base point and taking the logarithm map of the aligned point at base point using the total space metric.

3.6. **Remark: Combining Group Actions.** Central to shape spaces is the ability to easily combine group actions, as the notion of "shape" is application-dependent. For example, we may want to combine the action of rotations and the action of reparametrizations when we consider shapes of curves and surfaces in the next section.

Assume we have k groups acting on a manifold, and we know how to align points with respect to each individual group action. A general alignment procedure consists of a k-step iterative process, where at each step the alignment with respect to the action $0 \leq i \leq k-1$ is performed [57].

Combination of group actions is achieved through the use of a tuple of group actions. In this case, an alternating alignment algorithm is used by default (`AlternatingAligner`).

4. Landmarks, Curves, Surfaces and Their Shapes in the `shape` Module

This section details the concrete object spaces and shape spaces that we implement in the `shape` module, specifically: landmark sets, curves, surfaces spaces and their corresponding shape spaces. As such, this section also provides a comprehensive review of shape analysis. Each subsection further showcases code snippets using each shape space to demonstrate the diversity of use cases of the proposed `shape` module.

4.1. Landmark Sets.

4.1.1. *Pre-shape space.* Consider the space of k landmarks of $\mathbb{R}^d$. A data point of this space is $p \in M(k,d)$ where $M(k,d)$ is the space of real $k \times d$ matrices. Changing the position and size of a landmark set p do not change its shape. After removing translations ($p_i - \bar{p}$, where $\bar{p}$ is the barycenter of p) and scaling (by dividing by the Frobenius norm here denoted by $||\cdot||$), we obtain the *pre-shape space* (see, *e.g.*, [12, 18]) defined by

$$\mathcal{S}(k,d) = \left\{ p \in M(k,d) \mid \sum_{i=1}^{k} p_i = 0,\ ||p|| = 1 \right\}.$$

The Frobenius metric on $M(k,d)$ induces a metric on $\mathcal{S}(k,d)$ by the embedding $i : \mathcal{S}(k,d) \to M(k,d)$, known as the spherical Procrustes metric. This metric makes $\mathcal{S}(k,d)$ isometric to the round sphere $\mathbb{S}^{d(k-1)-1} \subset \mathbb{R}^{d(k-1)}$. The left side of Figure 3 illustrates the operation of centering and resizing that converts an original landmark set of $k = 3$ landmarks in $d = 2$ dimensions into an element of the pre-shape space.

4.1.2. *Kendall shape space.* Rotating a set of landmarks does not change its shape. Accordingly, the *Kendall shape space*, denoted by $\Sigma(k,d)$, is the quotient space induced by the action of the group of rotations $\mathrm{SO}(d)$ on $\mathcal{S}(k,d)$:

$$\Sigma(k,d) = \mathcal{S}(k,d)/\mathrm{SO}(d). \tag{5}$$

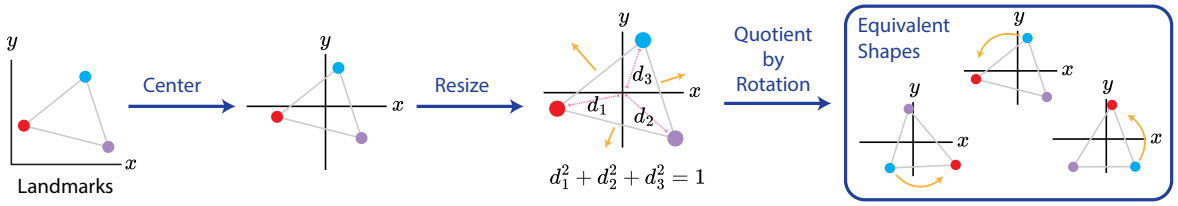

FIGURE 3. **Landmark sets.** Consider an object defined as a set of 3 landmarks in 2D (left). We transform this object into a shape, *i.e.*, into a point in Kendall shape space via the following operations: center the landmark set by placing its barycenter at the origin, resize the landmark set so that its 'size' equals 1, and then quotient by the group of rotations such that the difference in rotation will not contribute to distance between points in Kendall shape space. "Equivalent shapes" in the figure all correspond to the same point in Kendall shape space

.

The quotient metric on $\Sigma(k, d)$ induced from this group action and quotient structure is the so-called *Kendall metric*. Note that this action is smooth, proper but not free when $d \geq 3$, which is why the quotient space has singularities (see, *e.g.*, [12, 18]). In other words, this space is a Riemannian manifold when $d = 2$ (complex projective space) and a stratified space when $d \geq 3$ [12]. Away from singularities, the canonical projection $\pi : \mathcal{S}(k, d) \to \Sigma(k, d)$ is a Riemannian submersion. The full preprocessing evolution which takes unprocessed landmarks to a point in Kendall shape space is shown in Fig. 3, where the right side corresponds to the quotient by rotations.

The fiber bundle structure associated with Kendall shape spaces is well known. In particular, the vertical and horizontal subspaces can be computed explicitly. The vertical subspace at $p \in \mathcal{S}(k, d)$ is given by

$$V_p = \{pA^T \mid A \in \text{Skew}(d)\},$$

where $\text{Skew}(d)$ denotes the space of skew-symmetric $d \times d$ matrices. The horizontal subspace is given by

$$\begin{aligned} H_p &= \left\{w \in T_p\mathcal{S}(k, d) \mid \text{Tr}\left(Ap^T w\right) = 0, \forall A \in \text{Skew}(d)\right\} \\ &= \left\{w \in T_p\mathcal{S}(k, d) \mid p^T w \in \text{Sym}(d)\right\}, \end{aligned}$$

where $\text{Sym}(d)$ is the space of symmetric $d \times d$ matrices. The vertical component of a tangent vector can be computed as $\text{Ver}_p(w) = pA^T$ [44], where A solves the Sylvester equation:

$$Ap^T p + p^T pA = w^T p - p^T w.$$

Given a geodesic γ in $\mathcal{S}(k, d)$ such that $\dot{\gamma}(0)$ is a horizontal vector, then $\dot{\gamma}(t)$ is horizontal for all t, and in particular, $\pi \circ \gamma$ is a geodesic of $\Sigma(k, d)$. We leverage these facts in our implementation of the Kendall shape spaces in the `shape` module.

4.1.3. *Aligning landmark sets.* Algorithms to align landmark sets are central to perform computations on the Kendall shape space. Here, we align landmark sets using the finite-dimensional group of rotations $SO(d)$. This approach is related to Procrustes analysis literature, where landmark sets are aligned using different groups of transformations such as general invertible affine transformations. The alignment strategy implemented in the `shape` module to find optimal rotation aligning a given set of landmarks onto another one uses a traditional approach leveraging SVD.

4.1.4. *Landmark sets and their shapes in the `shape` module.* The `Landmarks` class implements the space $M(k, d)$ of matrices representing k landmarks in d dimensions, using a product manifold $M = \mathbb{R}^d \times \cdots \times \mathbb{R}^d$ in the Python class `Landmarks` which inherits from the `NFoldManifold` class. To instantiate a space of triangles in 2D, users can employ the following code:

```
from geomstats.geometry.euclidean import Euclidean
from geomstats.geometry.landmarks import Landmarks

euclidean = Euclidean(dim=2)
landmarks_space = Landmarks(k_landmarks=3, ambient_manifold=euclidean)
```

The `PreShapeSpace` class, which inherits from the abstract class `LevelSet`, implements the pre-shape space $\mathcal{S}(k, d)$ (see Figure 4 for details). Points are represented by $k \times d$ matrices, *i.e.*, we scale centered landmark coordinates X_C by the Frobenius norm $(x_C/||x_C||)$ to obtain valid pre-shape space points. Alternatively, *e.g.*, Helmertized landmark coordinates, where points are represented by $(k-1) \times d$ matrices, could have been used [12]. The `PreShapeMetric` class implements the spherical Procrustes metric. These implementations take advantage of the round sphere S^{dk-1} implementation, as points only need to be flattened to directly apply it.

The `KendallShapeMetric` class inherits from the class `QuotientMetric`. Similarly, the `PreShapeBundle` class inherits from the `FiberBundle` class. Both classes implement the quotient structures required for the Kendall shape space. We note that computations related to the affine connection, such as the so-called parallel transport, on this space have been implemented by the algorithm introduced in [18].

In the `shape` module, a pre-shape space $\mathcal{S}(k, d)$ of triangles in 2D, *i.e.*, with $k = 3$ and $d = 2$, is instantiated with the following code:

```
from geomstats.geometry.pre_shape import PreShapeSpace

preshape = PreShapeSpace(k_landmarks=3, ambient_dim=2)
```

Next, in order to create a Kendall shape space, we equip the pre-shape space with the action of the group of rotations $SO(d)$ which we quotient out. The resulting quotient space is the Kendall shape space.

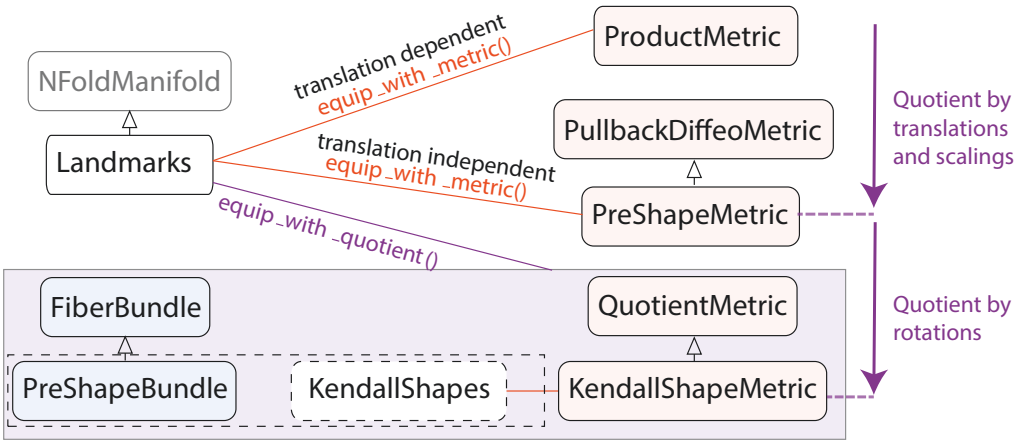

FIGURE 4. **Landmark sets and their shapes in the `shape` module of Geomstats.** The Python class `Landmarks` (left) represents a space of objects, a pre-shape space, or a Kendall shape space depending on the Riemannian metric that equips it: with the product Euclidean metric, the pre-shape metric, or the Kendall shape metric, respectively (right). Each of these metrics quotients specific group actions: translations, scalings, and rotations, as shown by the purple arrows (right). We indicate the abstract Python classes from which each class inherits, with empty black arrows representing inheritance.

```
preshape.equip_with_group_action("rotations")
preshape.equip_with_quotient()

kendall_shape = preshape.quotient
kendall_metric = kendall_shape.metric
```

We observe that the Kendall shape space comes equipped with its Kendall shape metric. The user can access essential computations of Riemannian geometry, such as `kendall_metric.exp()`, `kendall_metric.log()` and `kendall_metric.geodesic()`.

4.1.5. *Use case: Regression on landmark set shapes.* We show how to perform machine learning on landmark sets shapes implemented in the `shape` module using a regression method available in the main Geomstats package. Specifically, we demonstrate the use of geodesic regression [15] to estimate the shape of a rat's skull as a function of time. In this example, we use Vilmann's rat calvaria (skulls excluding the lower jaw) from X-ray images [58] (see Figure 5).

The goal of this regression is to estimate the shape of a rat skull of a 4.5 day old rat, based on 18 rat skulls of ages 7, 14, 21, 30, 40, 60, 90, and 150 days. More formally, our task is to estimate parameters $\theta \in \Sigma(k, d)$, $\phi \in T_\theta\Sigma(k, d)$ such that the difference between the geodesic

value $f_\theta(X)$ for each input X and the given value in the data set y is minimal (with respect to the Riemannian distance function). Here, we set

$$f_\theta(X) = \exp_\theta(X\phi) \in \Sigma(k, d), \tag{6}$$

where X is the time in days. Once we know the parameters θ, ϕ, we can then estimate the shape of a rat's skull for different times X by just plugging in X into f_θ. Below, we show how this task can be done conveniently using the implementation of landmark sets from the `shape` module and the learning algorithm from Geomstats. Figure 5 shows the plot of the resulting estimate. The Kendall shape space `kendall_space` can be created as above, by specifying 8 landmarks.

```
import geomstats.backend as gs
from geomstats.learning.geodesic_regression import GeodesicRegression

# Instantiate estimator
gr = GeodesicRegression(
    space=kendall_shape,
    center_X=False,
    method="riemannian",
    initialization="warm_start")

# Fit regression: X: times; y: array of skulls
gr.fit(X, y)

estimated_skull = gr.predict(gs.array([4.5]))
```

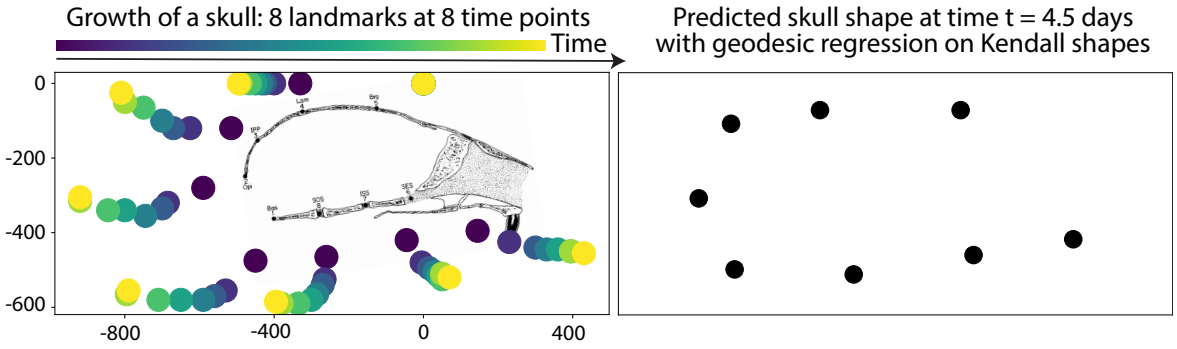

FIGURE 5. **Prediction of rat's skull shapes using the `shape` module of Geomstats**. Each rat skull is represented by a set of $k = 8$ landmarks in $d = 2$ dimensions. Left: Skull shape of rats at 8 time points, at ages of 7, 14, 21, 30, 40, 60, 90, and 150 days (different times shown by different color) [58]. A anatomical drawing of a rat skull is shown under the scatter plot for reference. Right: Predicted shape of the rat skull after 4.5 days after geodesic regression on Kendall shape space.

4.1.6. *Opportunities and challenges when computing with landmarks sets.* Representing objects as sets of landmarks, and object shapes as elements of a Kendall shape space, is appealing for several reasons. First, this representation is compact, in the sense that it has a very low memory footprint. In other words, only a few landmarks can distill the essence of a shape. This leads to fast computations and machine learning algorithms that typically require smaller datasets. Second, the differential geometry of the Kendall shape space has been extensively studied [29, 12]. Consequently, it is possible to derive important insights on the behavior of statistical and machine learning algorithms through analytical computations on this shape space. For example, researchers have leveraged Riemannian geometry to study the statistical properties of the mean shape in spaces of landmark shapes including Kendall shape spaces. In particular, it has been observed that for shape spaces without a quotient by the action of scaling, there is an asymptotic bias on the computation of the mean shape. Even with an infinite number of data points, the true mean shape cannot be recovered exactly such that researchers need to resort to bias correction techniques [40].

Yet, describing objects as landmark sets and their shapes has some drawbacks. First, it requires that the landmarks are correctly positioned next to points of semantic interest: in our example above, next to the actual location of anatomical landmarks on a rat's skull shape. Next, beyond the example of triangles in 2D and triangles in 3D, Kendall shape spaces might be harder to leverage for interpretation purposes.

4.2. **Curves.**

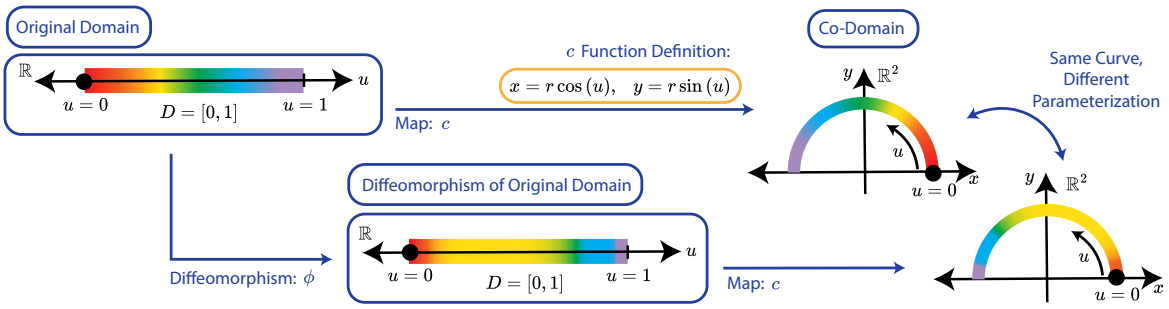

FIGURE 6. **Curves.** Consider a planar curve represented by a function $c : [0, 1] \to \mathbb{R}^2$ which maps a parameter $u \in [0, 1]$ to points in $\mathbb{R}^2$: for example, c is a half-circle (top row). The curve's parametrization can be changed by applying a diffeomorphism ϕ to the domain $D = [0, 1]$ before mapping to $\mathbb{R}^2$ (bottom row). This reparametrization does not change the shape of the curve, which is still a half circle.

4.2.1. *Spaces of curves.* Consider the (infinite-dimensional) manifold $\mathrm{Imm}\left(D, \mathbb{R}^d\right)$, whose points are immersions with domain D either the interval $I = [0, 1]$ or the circle S^1 [3]:

$$\mathrm{Imm}\left(D, \mathbb{R}^d\right) = \left\{c \in C^\infty\left(D, \mathbb{R}^d\right) : c'(u) \neq 0 \ \forall u \in D\right\}.$$

Each immersion c represents a regular smooth parameterized curve in a Euclidean space $\mathbb{R}^d$, denoted $c : D \to \mathbb{R}^d$. The curve c is open if D is the interval $[0, 1]$; closed if D is the circle S^1. An example of open curve in $\mathbb{R}^2$ is shown in Figure 6.

In shape analysis, one is not interested in the parametrized curve itself, but only in its features after quotienting out the action of shape-preserving transformations: translations, rotations and reparametrizations. To deal with translations, we restrict to the space of parameterized curves starting at the origin in Euclidean space

$$\mathrm{Imm}_0\left(D, \mathbb{R}^d\right) = \mathrm{Imm}\left(D, \mathbb{R}^d\right)/\mathbb{R}^d. \tag{7}$$

Just like for landmarks, the group of rotations $SO(d)$ acts by left multiplication on this space of curves starting at the origin. As for reparametrizations, they are represented by elements of an appropriate group of diffeomorphisms $\mathcal{D}(D)$ and act by right composition on $\mathrm{Imm}_0\left(D, \mathbb{R}^d\right)$:

$$\mathrm{Imm}_0\left(D, M\right) \times \mathcal{D}(D) \to \mathrm{Imm}_0\left(D, \mathbb{R}^d\right), \quad (c, \varphi) \mapsto c \circ \varphi.$$

This action merely changes the parametrization of the curve but not its actual shape, as shown in Figure 6.

We then define a Riemannian metric on $\mathrm{Imm}_0\left(D, \mathbb{R}^d\right)$ that is invariant with respect to these shape preserving transformations. A popular family of metrics with this property is given by the elastic metrics [39, 3], defined for scalars a, b as

$$G_c^{a,b}(h, k) = \int_D a^2 \langle D_s h^\perp, D_s k^\perp \rangle + b^2 \langle D_s h^\top, D_s k^\top \rangle ds. \tag{8}$$

Here, h, k are tangent vectors to the manifold of curves at a point c, *i.e.*, vector fields along c that represent infinitesimal deformations of the curve; $D_s = \frac{1}{|c'|}\frac{d}{du}$, $ds = |c'|\, du$ represent differentiation and integration with respect to the arc length parameter s; $\bullet^\top$ and $\bullet^\perp$ denote projection onto the normal and tangential parts of a tangent vector, *i.e.*, $D_s h^\top = \left\langle D_s h, \frac{c'}{|c'|} \right\rangle \frac{c'}{|c'|}$, where $\langle \cdot, \cdot \rangle$ is the Euclidean inner product. The coefficients a and b in the $G^{a,b}$ metric indicate the extent to which bending and stretching the curve are respectively penalized.

Besides their reparametrization-invariant property and interpretation from linear elasticity theory (see [3]), elastic metrics are appealing due to the existence of closed-form solutions for geodesics. For example, [3] shows that the square root velocity (SRV) transform R [56] defined as:

$$R : \mathrm{Imm}_0\left([0, 1], \mathbb{R}^d\right) \to C^\infty\left([0, 1], \mathbb{R}^d \backslash \{0\}\right) \tag{9}$$

$$c \mapsto \frac{c'}{\sqrt{||c'||}},$$

is an isometry between the manifold $\mathrm{Imm}_0([0, 1], \mathbb{R}^d)$ of Euclidean curves starting at the origin equipped with the elastic metric $G^{a,b}$, and the space $C^\infty([0, 1], \mathbb{R}^d \backslash \{0\})$ equipped with a multiple $4b^2 G^{L^2_\lambda}$ of a conic Riemannian metric defined in [3]. Geodesics are known in

closed form for the metric G^2_λ, which allows us to compute geodesics in the original space of curves $\mathrm{Imm}_0\left([0,1],\mathbb{R}^d\right)$. Indeed, we convert an initial point and initial tangent vector in $\mathrm{Imm}_0\left([0,1],\mathbb{R}^d\right)$ into a point and tangent vector in $C^\infty\left([0,1],\mathbb{R}^d\backslash\{0\}\right)$ using the transform R and its differential, compute the geodesic there, and use the inverse transform R^{-1} to obtain the corresponding geodesic in $\mathrm{Imm}_0\left([0,1],\mathbb{R}^d\right)$. In particular, when $a=1, b=1/2$, $G^{L^2_\lambda}$ reduces to a standard L^2 metric on $C^\infty\left(I,\mathbb{R}^d\backslash\{0\}\right)$, *i.e.*, a metric resulting from endowing the space with the L^2-inner product. The metric $G^{1,1/2}$ is known as the square root velocity (SRV) metric [56].

4.2.2. *Spaces of curve shapes.* Recall that we have defined shapes to be what is left after factoring out certain transformations. Depending on the application, we can thus define the space of curve shapes to be the quotient of $\mathrm{Imm}_0\left(D,\mathbb{R}^d\right)$ under the action of either reparametrizations $\mathcal{D}(D)$, rotations $\mathrm{SO}(d)$ or both $\mathcal{D}(D)\times\mathrm{SO}(d)$):

$$\begin{aligned}\mathcal{Q}_{\text{reparam.}} &= \mathrm{Imm}_0\left(D,\mathbb{R}^d\right)/\mathcal{D}(D)\\ \mathcal{Q}_{\text{rotations}} &= \mathrm{Imm}_0\left(D,\mathbb{R}^d\right)/\mathrm{SO}(d)\\ \mathcal{Q} &= \mathrm{Imm}_0\left(D,\mathbb{R}^d\right)/(\mathcal{D}(D)\times\mathrm{SO}(d)).\end{aligned}$$

The elastic metric $G^{a,b}$ on $\mathrm{Imm}_0\left(D,\mathbb{R}^d\right)$ is invariant under both actions, *i.e.*:

$$G^{a,b}_c(h,k) = G^{a,b}_{R(c\circ\varphi)}(R(h\circ\varphi),R(k\circ\varphi)) \quad \forall\phi\in\mathcal{D}(D),\, R\in\mathrm{SO}(d),$$

and induces a Riemannian metric and a quotient distance (2) on all possible quotient spaces, as described in Section 3.4.

4.2.3. *Aligning curves.* Algorithms to align curves are central to perform computations on the respective quotient spaces.

Aligning curves in rotations. As with landmark sets, the optimal rotation that aligns one curve onto another can be found by a Procrustes approach, specifically through the singular value decomposition of a quantity involving only the SRV representations of the two curves.

Aligning curves in reparametrizations. Alignment is particularly challenging when quotienting out reparametrizations due to the infinite-dimensional nature of the reparametrization group. The alignment strategies currently implemented in the `shape` module to find optimal reparametrizations include a dynamic programming-based algorithm [56], and an iterative horizontal alignment algorithm [30] — which we recall below.

The dynamic programming-based algorithm [56] searches for a monotonically increasing diffeomorphism γ in a uniform $n\times n$-grid that, for the SRV metric, minimizes the integrand

$$q_1(t)\cdot q_2(\gamma(t))\dot\gamma(t)^{\frac{1}{2}}$$

over the domain, where q_i is the SRV transform of c_i. The computation of the integral is done recursively. Assuming the integral is zero at $(0,0)$, it sequentially computes its value for each point of the grid. To limit the computational cost, a numerical parameter s is usually

used to limit the degree of looking-back. Specifically, the solution at node (i, j) is found by considering only the subgrid $(i-s, j-s) \times (i-1, j-1)$ instead of the full grid (such parameter restricts the admissible slopes).

The iterative horizontal alignment algorithm [30] finds an horizontal geodesic by successively iterating between two steps: i) construct the geodesic between two curves c_1, c_2; ii) compute the horizontal part of the geodesic. In other words, it successively finds a better representative of the curve c_2 with respect to c_1, *i.e.*, a curve in the fiber of c_2 that is closer (with respect to the total space metric) to c_1. The key ingredient for this procedure is the computation of the horizontal part of the geodesic, which requires solving a linear system (see [30] for details). This algorithm is agnostic to the metric on the total space.

Aligning curves in rotations and reparametrizations. Finding the optimal pair of rotation and reparametrization is approximated by a two-step iterative process: i) find the optimal rotation while fixing the parametrization; ii) find the optimal reparametrization while fixing the rotation[57].

4.2.4. *Curves and their shapes in the* `shape` *module.* In the *shape* module, we represent spaces of continuous curves and their shapes by discrete curves with k sampling points, as done in [30]. All the quantities of interest can be obtained from this discrete curves representation by standard numerical methods, such as finite differences and numerical integration. We note that points are considered to be uniformly sampled with respect to their parameterization.

The (finite-dimensional) discrete curves space with k sampling points is defined as a product manifold $M = \mathbb{R}^d \times \cdots \times \mathbb{R}^d$. It is implemented in the Python class `DiscreteCurves`, which inherits from `NFoldManifold`. By default, it is equipped with the standard discrete L^2 metric, implemented in the Python class `L2CurvesMetric`, which inherits from the class `NFoldMetric`. For example, a space of discrete curves in 2D with 200 sampling points can be instantiated with the following code:

```
from geomstats.geometry.discrete_curves import DiscreteCurves

curves = DiscreteCurves(k_sampling_points=200, ambient_dim=2, starting_at_origin=False)
```

Removing translations is achieved by considering curves that start at the origin. In practice, we subtract the initial point and omit it from the representation.

```
curves = DiscreteCurves(k_sampling_points=200, ambient_dim=2, starting_at_origin=True)
```

This space can be equipped with the elastic metric $G^{a,b}$ and in particular the SRV metric $G^{1,\frac{1}{2}}$, respectively implemented in the classes `ElasticMetric` and `SRVMetric`. Taking advantage of the isometry (9), we have implemented these metrics as children of the class `PullbackDiffeoMetric`, which implements the general structure of pullback metrics by diffeomorphisms in Geomstats. Similarly, the operations related to the SRV transform (9) are

encoded in the Python class `SRVTransform` which inherits from the abstract class `Diffeo` that implements the structure of diffeomorphisms in Geomstats.

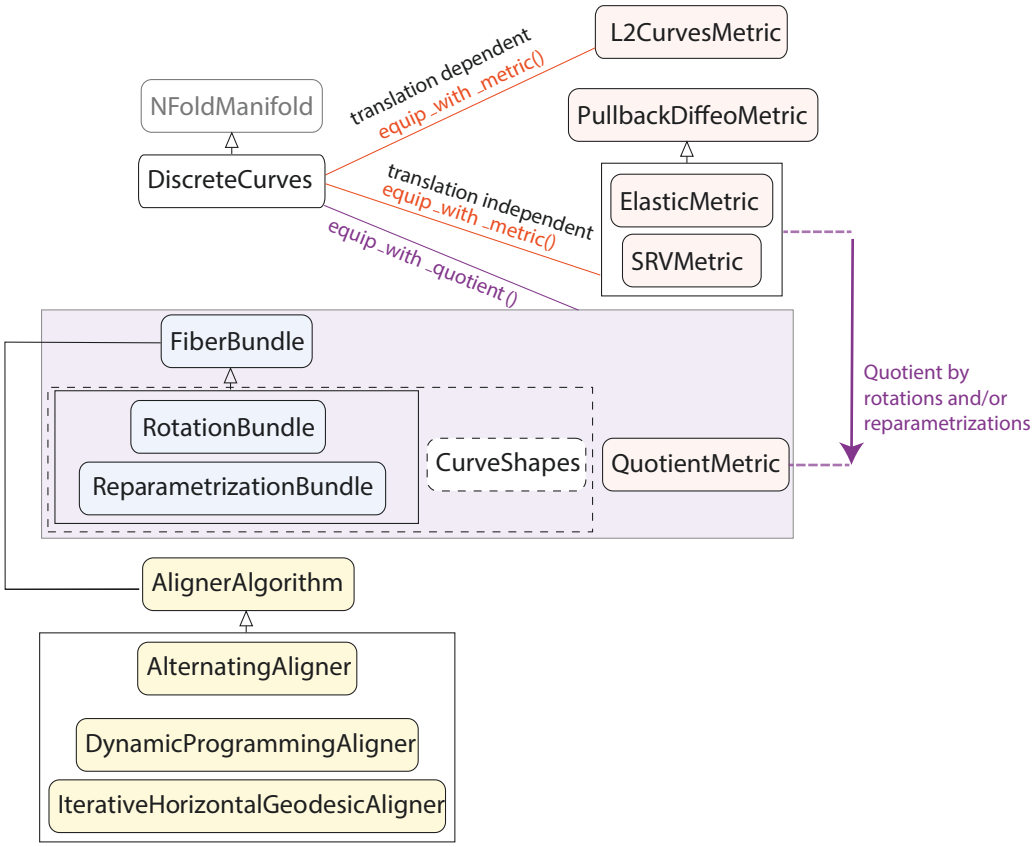

FIGURE 7. **Curves and their shapes in the `shape` module of Geomstats.** The Python class `DiscreteCurves` (left) represents a space of curves, a space of curves that start at the origin, or a space of curve shapes where either rotation or reparametrization or both have been quotiented, depending on the Riemannian metric that equips it: with the L_2 Euclidean metric, the elastic or the Square-Root-Velocity (SRV) metric, or a quotient elastic or SRV metric where the quotient is shown by the purple arrow, respectively (right). Each of these metrics quotients specific group actions: translations, rotations and/or reparametrizations. We note that, while the elastic and SRV metrics do not depend on translations, they are *not* built as quotients of the translation-dependent L_2 Euclidean metric. We indicate the abstract Python classes from which each class inherits, with empty black arrows representing inheritance. The alignment algorithms that perform quotients by rotations and reparametrizations are indicated in yellow.

Next, in order to create a space of curve shapes, we equip the space of curves with the action of the group of rotations $SO(d)$, and/or of the group of reparametrizations, which we quotient out. The quotient space is the space of curve shapes.

```
discrete_curves.equip_with_group_action(("rotations", "reparametrizations"))
discrete_curves.equip_with_quotient()

curve_shapes = discrete_curves.quotient
curve_shapes_metric = curve_shapes.metric
```

Recall that when a manifold is equipped with a metric and a group action, the method `equip_with_quotient()` does two things.

First, it creates a quotient space equipped with a quotient metric. Therefore, the shape space comes equipped with a Riemannian metric, which is the one induced by the metric on the space of curves; by default, these are the SRV metric and the quotient SRV metric. The user can access essential computations of Riemannian geometry, such as `curve_shapes_metric.exp()`, `curve_shapes_metric.log()` and `curve_shapes_metric.geodesic()`.

Second, it equips the total space with a fiber bundle. Two children of the class `FiberBundle` are implemented: `ReparametrizationBundle` and `RotationBundle`. The reparametrization alignment algorithms `DynamicProgrammingAligner` and `IterativeHorizontalGeodesicAligner` are implemented as children of `AlignerAlgorithm`, and can be passed by composition to an instance of `ReparametrizationBundle`. The general structure and code provided by the Python class `QuotientMetric` performs the rest of the computations.

4.2.5. *Use case: Statistics on curve shapes.* We show how to perform computations and calculate summary statistics on curve shapes implemented in the `shape` module using algorithms available in the main Geomstats package. Specifically, we study a data set of mouse *Osteosarcoma* imaged cells (AXCFP2019), see Figure 8. Each curve is represented by 200 uniformly-spaced sampling points. To showcase how cells can be compared in Geomstats, we start by showing how to compute the geodesic between two cells with respect to the SRV metric (equipped by default), see Figure 8 (D).

```
from geomstats.geometry.discrete_curves import DiscreteCurves

# Instantiate space
space = DiscreteCurves(k_sampling_points=200, ambient_dim=2, starting_at_origin=True)

# Compute geodesic and evaluate it at midpoint
geod_func = space.metric.geodesic(initial_point=cell_start, end_point=cell_end)
geod_func(0.5)
```

The mean of 20 of such cells is showed in Figure 8 (C). The following code snippet shows how it can be computed in Geomstats.

```python
from geomstats.learning.frechet_mean import FrechetMean

# Instantiate and fit estimator
mean = FrechetMean(space)
mean.fit(cell_shapes)

# Access mean shape
mean_estimate = mean.estimate_
```

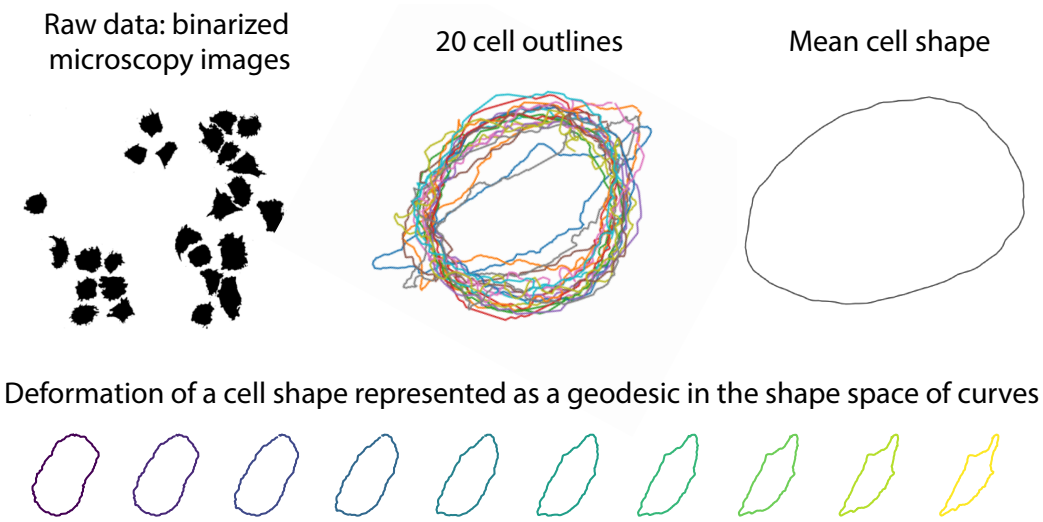

FIGURE 8. **Statistics on cancer cell shapes with the `shape` module of Geomstats.** Top-left: Binarized image obtained from microscopy and containing a set of cancer cells. Top-middle: 20 cells from the data set. Top-right: Frechét mean of the 20 cells shown on the left. Bottom: Geodesic between two cancer cells, going from a less aggressive cancer to a more aggressive cancer, showing the corresponding characteristic deformation of the cell outline.

4.2.6. *Opportunities and challenges when computing with curves.* Computing with curves and their shapes is appealing for objects that do not have well-defined, semantic landmarks, but rather contours. For example, a cell border typically does not have landmarks, yet it has a well-defined contour. The differential geometry theory of spaces of curves and curve spaces is well-defined and relatively mature. However, and perhaps surprisingly, there exist several outstanding challenges when it comes to their numerical implementations. First, while the theory for closed curves is well-defined, as a submanifold of the space of open curves, the implementation of closed curves is often ad-hoc, resorting to various techniques to close the curves. Second, the algorithms performing alignment in reparametrization can be inaccurate and slow. Third, while the theory prescribes to quotient the joint action of rotations and reparametrization, in practice researchers quotient each sequentially. This approach seems

justified by the fact that the results are often only minimally affected. Lastly, while the theory proposes elastic Riemannian metric on curve spaces for any real scalar a, b, we observe in practice that larger or finer values of a, b or their ratios can deteriorate the quality of the numerical results, making the choice of the parameters particularly challenging.

Meanwhile, this produces exciting research directions. An open research direction is to investigate whether changing parameters a, b in computations and machine learning algorithms can lead to different, interpretable results upon extracting knowledge from curves in natural sciences. A related research direction also performs Riemannian metric learning, *i.e.*, learns the parameters a, b that best describe a trajectory of curve shapes as a geodesic [42].

4.3. **Surfaces.**

4.3.1. *Spaces of surfaces.* Spaces of surfaces are defined analogously to spaces of curves introduced in the previous subsection. The space of regular smooth parameterized immersed surfaces in $\mathbb{R}^3$ can be identified with the (infinite-dimensional) manifold:

$$\operatorname{Imm}\left(D, \mathbb{R}^3\right) = \left\{q \in C^\infty\left(D, \mathbb{R}^3\right) : q'(u) \neq 0 \ \forall u \in D\right\}, \tag{10}$$

where D is a compact 2-dimensional space of parameters, for example $D = [0, 2\pi] \times [0, \frac{\pi}{2}]$. This is shown in Fig. 9, adapted from [43].

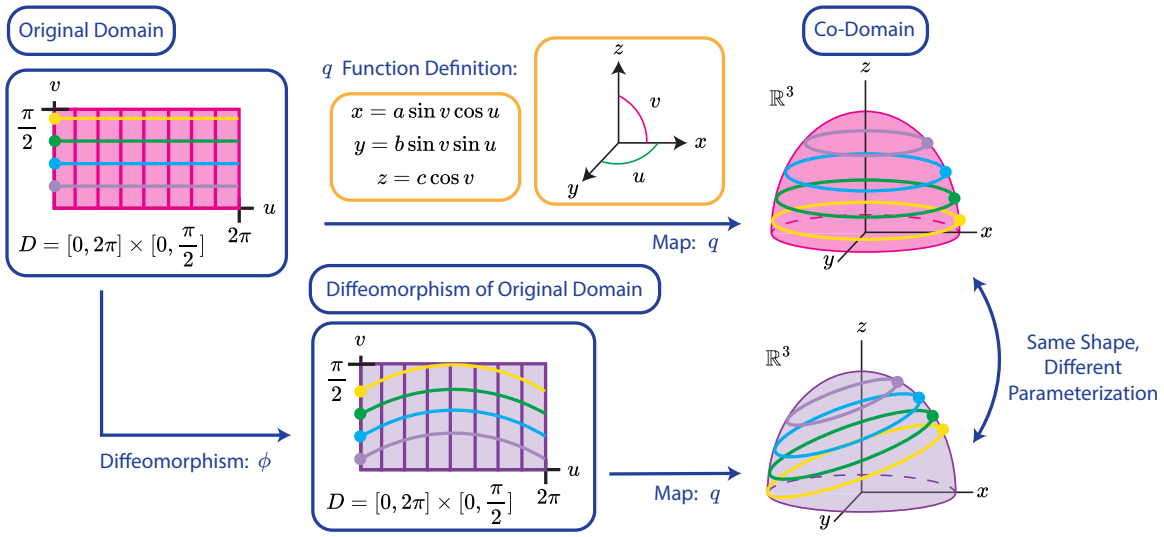

FIGURE 9. **Surfaces**. Consider a surface represented by a function $q : D \to \mathbb{R}^3$ that maps parameters $(u, v) \in D$ to points in 3D space $q(u, v) \in \mathbb{R}^3$, forming a half sphere (top row). The surface's parameterization can be changed by applying a diffeomorphism ϕ to the domain D before mapping to $\mathbb{R}^3$ (bottom row). This reparametrization does not change the shape of the surface, which is still a half sphere. Figure and caption adapted from [43].

A second-order family of Sobolev metrics can be defined on $\mathrm{Imm}(D, \mathbb{R}^3)$ [23]:

$$\begin{aligned} G_q(h,k) = \int_D & a_0\langle h,k\rangle + a_1 g_q^{-1}\left(dh_m, dk_m\right) \\ & + b_1 g_q^{-1}\left(dh_+, dk_+\right) + c_1 g_q^{-1}\left(dh_\perp, dk_\perp\right) \\ & + d_1 g_q^{-1}\left(dh_0, dk_0\right) + a_2\langle \Delta_q h, \Delta_q h\rangle \,\mathrm{vol}_q, \end{aligned} \tag{11}$$

where $q \in \mathrm{Imm}(D, \mathbb{R}^3)$, $h, k \in C^\infty\left(D, \mathbb{R}^3\right)$ are tangent vectors to the manifold of surfaces. In practice, h is a vector field along the surface, which has tangential and normal components with respect to the surface. The differential dh can be interpreted as a vector-valued one form (*i.e.*, a map from the tangent bundle TD to $\mathbb{R}^3$) and g_q refers to the Riemannian metric *of the surface* q, *i.e.*, the pullback of the Euclidean metric of the 3D ambient space onto the 2D surface defined by q [23]. Lastly, vol_q is the surface area measure of the smooth immersion q. Each choice of parameters $(a_0, a_1, b_1, c_1, d_1, a_2)$ gives a different Riemannian metric from the family of metrics.

This metric results from the weighted combination of a zeroth-order term:

$$G_q^{\mathrm{0th}}(h,k) = \int_D \langle h,k\rangle \,\mathrm{vol}_q,$$

with a first-order term:

$$G_q^{\mathrm{1st}}(h,k) = \int_D g_q^{-1}(dh, dk)\,\mathrm{vol}_q,$$

and a second-order term:

$$G_q^{\mathrm{2nd}}(h,k) = \int_D \langle \Delta_q h, \Delta_q h\rangle \,\mathrm{vol}_q\,.$$

The zeroth-order term $G_q^{\mathrm{0th}}(h,k)$ is the L^2 metric on $q \in \mathrm{Imm}(D, \mathbb{R}^3)$, which is degenerate [23]. The first-order term $G_q^{\mathrm{1st}}(h,k)$ is introduced to avoid this degeneracy and can be expanded, as shown in Equation (11), by the decomposition of dh into physically-interpretable components [60, 23]: a shearing term dh_m, a scaling term dh_+, and a bending term $dh_\perp$, and a less interpretable dh_0 term. This physical interpretation is the reason first-order Sobolev metrics on $\mathrm{Imm}(D, \mathbb{R}^3)$ are known as elastic metrics. The second-order term $G_q^{\mathrm{2nd}}(h,k)$ is introduced due to empirical evidence that first-order terms are still too weak to prevent geodesics from leaving the space of immersions which leads to instability in numerical schemes [23] and involves the computation of the Laplacian Δ_q.

4.3.2. *Spaces of surface shapes.* We recall that second-order Sobolev metrics $G_q(h,k)$ on $\mathrm{Imm}(D, \mathbb{R}^3)$ are invariant under the action of the reparametrization $\mathcal{D}(D)$, rotation $\mathrm{SO}(3)$, and translation $\mathbb{R}^3$ groups [23]: $G_q(h,k) = G_{R(q\circ\varphi)+\tau}(R(h\circ\varphi), R(k\circ\varphi))$ where $\phi \in \mathcal{D}(D)$, $R \in \mathrm{SO}(3)$, and $\tau \in \mathbb{R}^3$. Thus, we can define spaces of surface shapes by considering quotient structures on $\mathrm{Imm}(D, \mathbb{R}^3)$ coming from any of those invariances. Here, we are interested on the quotient of reparametrizations $\mathcal{D}(D)$:

$$\mathcal{Q}_{\mathrm{reparam.}} = \mathrm{Imm}(D, \mathbb{R}^3)/\mathcal{D}(D).$$

The reparametrization-invariance of $G_q(h,k)$ induces a quotient metric on $\mathcal{Q}_{\text{reparam.}}$ the spaces of surface shapes in $\mathbb{R}^3$.

4.3.3. *Aligning surfaces in reparametrization.* As with discrete curves, the action of the (infinite-dimensional) reparametrization group on discrete surfaces is particularly challenging to implement. Following [23], we perform alignment by solving the geodesic boundary value problem in the quotient space of unparametrized discrete surfaces. Specifically, we follow their relaxed optimization formulation which uses a discrepancy term given by kernel metrics on (oriented) varifold representations of surfaces (see [25] for details on varifolds).

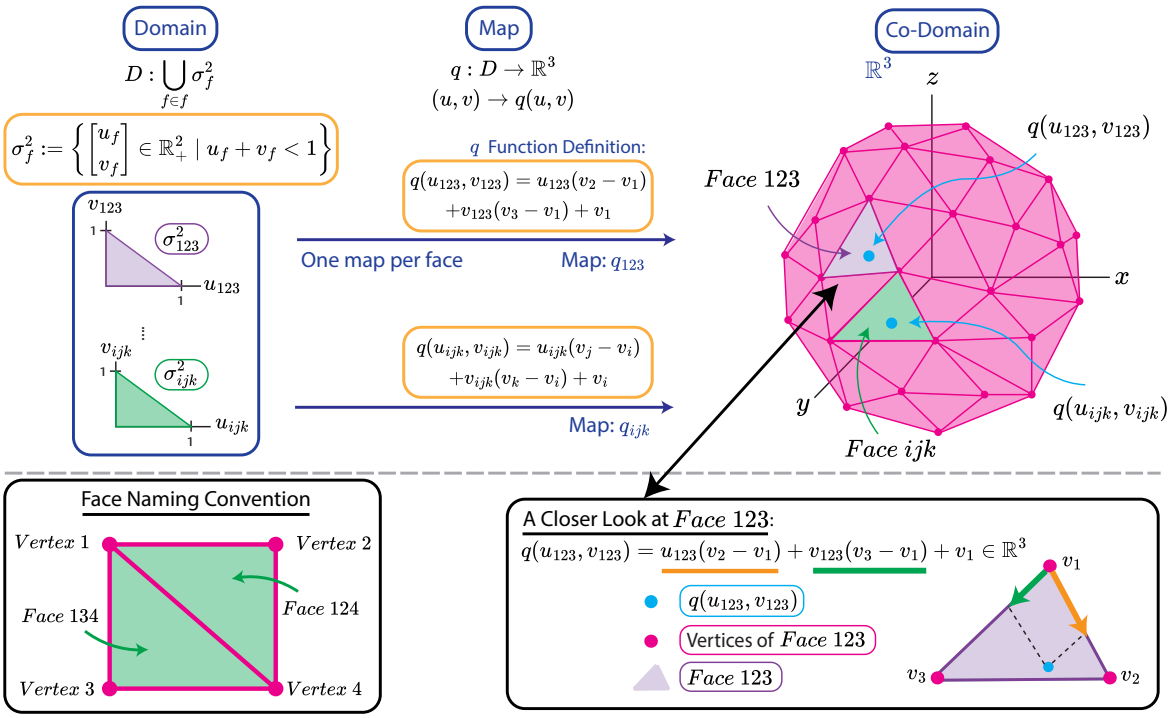

FIGURE 10. **Discrete Surfaces.** A discrete surface can be represented by a map $q : D \to \mathbb{R}^3$. Each "sub-domain" in D corresponds to a different face on the discrete surface in $\mathbb{R}^3$. We can change the parameterization of the discrete surface by changing the number of sub-domains in D, or by changing the vertices corresponding to one or more of the sub-domains.

4.3.4. *Surfaces and their shapes in the* `shape` *module.* In the *shape* module, we follow [23] and represent surfaces by triangle meshes with fixed combinatorial structure, *i.e.*, meshes with fixed number of vertices V and connectivity (set of edges E and faces F). We call these meshes the discrete surfaces. Let $\mathfrak{M}$ represent the set of such triangle meshes. As shown in Fig. 10, a mesh $q \in \mathfrak{M}$ is fully defined by the position of its vertices and can be seen as a piecewise linear surface.

In the `shape` module, the Python class `DiscreteSurfaces` inherits from the `Manifold` class and implements the space of discrete surfaces (see Figure 11 for details). A space of surfaces with 4 vertices and two triangular faces can be instantiated as:

```
from geomstats.geometry.discrete_surfaces import DiscreteSurfaces

faces = gs.array([[0, 1, 2], [1, 2, 3]])
discrete_surfaces = DiscreteSurfaces(faces=faces)
```

In order to use Riemannian geometry to compute with discrete surfaces, the discrete counterparts of several smooth quantities, including the Riemannian metric, need to be defined [23]. In the field of discrete differential geometry [8], we may discretize tangent vectors on the vertices V, first-order terms on the faces F, the Laplacian on the dual cell [5], and the surface area measure both at vertices and at faces. With these discretization choices, the discrete counterpart of the Sobolev metric of Equation (11) is found in [23]:

$$\begin{aligned} G_q(h,k) = &\sum_{v\in V} a_0\langle h,k\rangle \,\mathrm{vol}_v \\ &+\sum_{f\in F}\Big(a_1 g_f^{-1}\left(dh_m, dk_m\right) + b_1 g_f^{-1}\left(dh_+, dk_+\right) \\ &+c_1 g_f^{-1}\left(dh_\perp, dk_\perp\right) + d_1 g_f^{-1}\left(dh_0, dk_0\right)\Big)\,\mathrm{vol}_f \\ &+\sum_{v\in V} a_2\left\langle \Delta_q h, \Delta_q k\right\rangle \mathrm{vol}_v\,. \end{aligned}$$

We refer the reader to [22] for detailed expressions for each term. These computations are implemented in the `ElasticMetric` class, which inherits from the `RiemannianMetric` class. The class `DiscreteSurfaces` and can be equipped with an elastic metric object.

Next, we equip the space of surfaces with the action of the group of reparametrizations, which we quotient out. The quotient space is the space of surface shapes.

```
discrete_surfaces.equip_with_group_action("reparametrizations")
discrete_surfaces.equip_with_quotient()

surface_shapes = discrete_surfaces.quotient
surface_shapes_metric = surface_shapes.metric
```

The shape space comes equipped with a Riemannian metric, which by default is the quotient elastic metric. The user can access essential computations of Riemannian geometry, such as `surface_shapes_metric.exp()`, `surface_shapes_metric.log()` and `surface_shapes_metric.geodesic()`. The quotient structure relies on the alignment procedure discussed above, where the class `VarifoldMetric` implements (oriented) varifold distances within the framework of discrete surfaces [23] by leveraging the fast kernel operations implementation available in Keops [6].

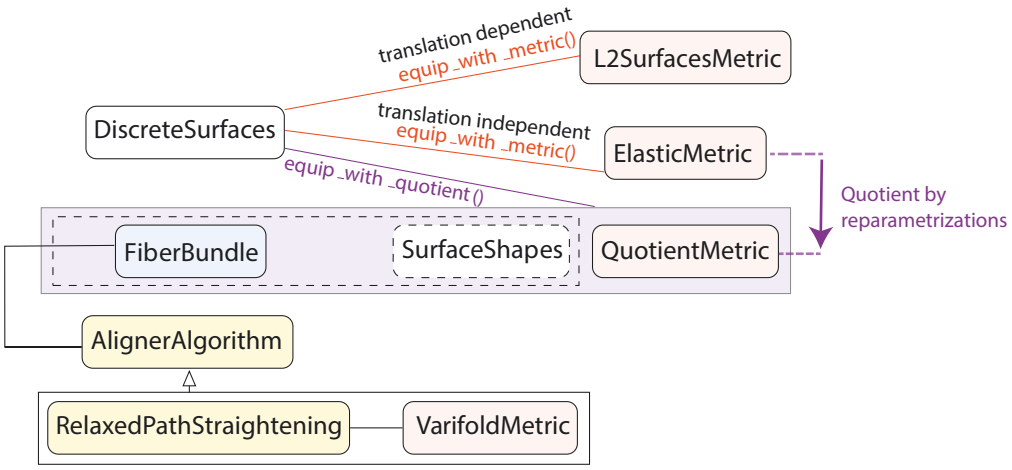

FIGURE 11. **Surfaces and their shapes in the `shape` module of Geomstats.** The Python class `DiscreteSurfaces` (left) represents a space of surfaces, a space of surfaces whose barycenter is located at the origin, or a space of surface shapes where reparametrization has been quotiented, depending on the Riemannian metric that equips it: with the L_2 Euclidean metric, the elastic metric, or a quotient elastic metric where the quotient is shown by the purple arrow, respectively (right). Each of these metrics quotients specific group actions: translations and reparametrizations. We note that, while the elastic metric does not depend on translations, it is *not* built as quotient of the translation-dependent L_2 Euclidean metric. We indicate the abstract Python classes from which each class inherits, with empty black arrows representing inheritance. The alignment algorithm that performs quotient by reparametrizations is indicated in yellow.

Numerical approaches to compute geodesics. Given the absence of closed-form solutions for most of the metric-related quantities, `ElasticMetric` heavily relies on numerical methods, which we describe here. For example, there are no closed-form expressions for the exponential and logarithm maps, which thus need to be computed numerically from the geodesic equation. The geodesic equation for $(\mathrm{Imm}_0(D, \mathbb{R}^3), G_q(h, k))$ is a non-linear, second-order in time, fourth-order in space, partial differential equation [23]. To avoid solving it directly, we find geodesics on $(\mathfrak{M}, G_q(h, k))$ through algorithms that rely on numerical optimization, usually limited memory BFGS algorithm [33] combined with automatic differentiation for gradient computation [46, 37]. These methods find the optimal location of the vertices of a path of discrete surfaces, represented as a path of triangle meshes [51].

First, we present the numerical approach to solve the geodesic boundary value (BVP) problem, *i.e.*, the problem of finding the geodesic in surface space connecting two surfaces. This

approach is called *path straightening*. A piecewise-linear path of N uniformly-sampled triangle meshes $q = (q(t_i))_{i \in J}$, for $J = \{0, \cdots, N-1\}$ and $t_i = i/(N-1)$, can be "straightened" into a geodesic by minimizing the Riemannian energy, given in discrete form by:

$$E(q) = \frac{1}{2N} \sum_{i=0}^{N-1} G_{q(t_i)}(\dot{q}(t_i), \dot{q}(t_i)). \tag{12}$$

Here, $\dot{q}(t_i)$ is obtained by, *e.g.*, forward finite differences: $\dot{q}(t_i) = (N-1)(q(t_{i+1}) - q(t_i))$. Given a discrete geodesic q, the tangent vector h that shoots from $q(t_0)$ to $q(t_{N-1})$ is found through: $h = (N-1)(q(t_1) - q(t_0))$. The path-straightening algorithm is implemented via the class `PathStraightening`, which is an attribute of the class `ElasticMetric`.

Second, we present the numerical approach to solve the geodesic initial value (IVP) problem, *i.e.*, the problem of finding the geodesic starting at an initial surface $q(t_0)$ with initial velocity h. Here, strategy consists in finding the midpoint $q(t_i)$ of a geodesic segment assuming known endpoints $q(t_{i-1}), q(t_{i+1})$ [51, 23]. This leads to the following numerical optimization problem at each time step [23]:

$$q(t_{i+1}) = \underset{\bar{q}(t_{i+1})}{\operatorname{argmin}} \, ||F\left(\bar{q}(t_{i+1}); q(t_i), q(t_{i-1})\right)||_2^2, \tag{13}$$

where $F\left(q(t_{i+1}); q(t_i), q(t_{i-1})\right) = 0$ denotes the following system of equations [23]:

$$\begin{aligned} &2G_{q(t_{i-1})}(q(t_i) - q(t_{i-1}), B_i) - 2G_{q(t_i)}(q(t_{i+1}) - q(t_i), B_i) \\ &+ D_{q(t_i)}G(q(t_{i+1}) - q(t_i), q(t_{i+1}) - q(t_i))_i = 0, \end{aligned}$$

where B_i is the i-th basis vector of $\mathbb{R}^{3n}$. The process starts by setting $q(t_1) = q(t_0) + h/(N-1)$, and terminates with the result of the exponential map $q(t_{N-1})$. The numerical approach to solve the geodesic initial value problem is implemented into the class `DiscreteSurfacesExpSolver`, which is an attribute of the class `ElasticMetric`.

4.3.5. *Use case: Geodesic between surfaces.* We show how to perform computations on surface shapes implemented in the `shape` module using algorithms available in the main Geomstats package. Specifically, we show how to compute geodesics between meshes with known point correspondences (see Figure 12).

```
from geomstats.geometry.discrete_surfaces import DiscreteSurfaces

# Instantiate space
space = DiscreteSurfaces(faces)

# Compute geodesic
geod_func = space.metric.geodesic(point_a, end_point=point_b)

# Get (chosen) geodesic points
time = gs.linspace(0.0, 1.0, num=6)
geod_points = geod_func(time)
```

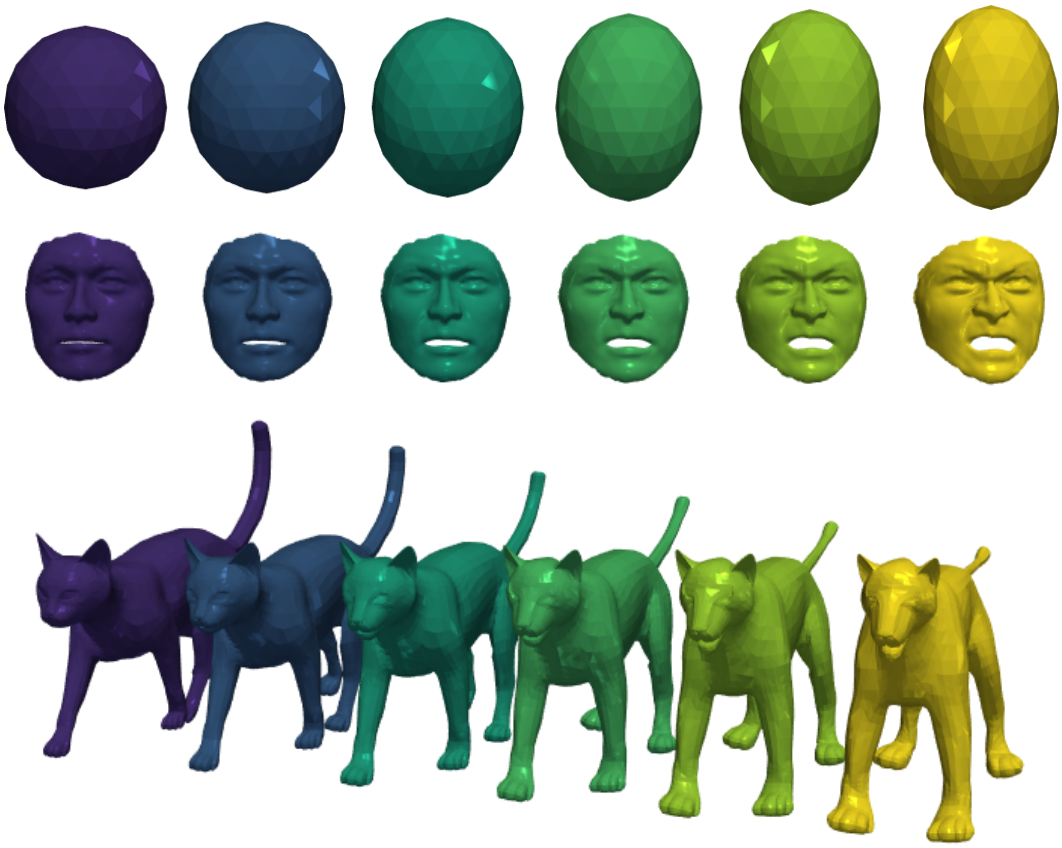

FIGURE 12. **Deformations of 3D shapes with the `shape` module of Geomstats.** Geodesics between meshes in the space of discrete surfaces equipped with an elastic metric, with all elastic parameters equal to 1. Top: a sphere deforms into an ellipsoid. Middle: a neutral face deforms into an angry face. Bottom: a cat deforms into a lion. The mesh data of the last two geodesics come from H2 Surface Match Github Repository.

The geodesics shown in Figure 12 match the geodesics obtained in the original paper [23], showing that the Geomstats equivalent geodesic computation is working as intended. Yet, the computation in Geomstats improved the original implementation in that the computation of the inner product defining the metric is substantially faster.

4.3.6. *Opportunities and challenges when computing with surfaces.* While the differential geometric framework for computations with curves and their shapes is relatively mature, the corresponding framework to compute with surfaces and their shapes is newer. For example, the families of elastic Riemannian metric presented in this paper have only been published a year ago [23]. Accordingly, both the theory and its numerical implementation contain outstanding challenges and interesting opportunities for research in the field.

First, computations with surfaces rely on numerical algorithms requiring the solution of high-dimensional optimization problems, being inherently slow. This is the case with the computation of the Riemannian logarithm and exponential maps, even when restricted to parameterized surfaces. Dedicated research to improve the speed and accuracy of these approaches would definitely unlock a wider range of applications for the differential geometric branch of shape analysis with surfaces. Second, the quotient by reparametrizations contains its set of challenges: the question of how to determine whether two triangle meshes represent the same shape is still ill-defined. (Oriented) varifold distances appear to provide an answer in [23], but it comes with the additional challenge of tweaking kernels and corresponding parameters. Research investigating the stability of these approaches with respect to the hyperparameters would help advance the field. Besides, the associated high-dimensional optimization problems are even more challenging than their parameterized counterparts. Third, and interestingly, quotienting rotations on the surfaces remains a surprisingly complicated operation when performed jointly with the quotient by reparametrizations. This is due to the fact that there is no definite way of choosing the center of the 3D rotation acting on the surface. One could think of using the barycenter of the surface's vertices. However, reparametrizing the surface changes the position of these points and therefore changes the position of their barycenter. Future theoretical and numerical research should investigate how to perform the joint quotient of rotations and reparameterizations on surfaces.

Conclusion

In this paper, we presented a Python implementation of the differential geometry of shape analysis in the module `shape` integrated in the software Geomstats. We introduced the architecture of the module and showed that it contains the essential building blocks to perform statistics and machine learning on shape data, through several code snippets. We hope that our implementation will inspire researchers to use, and contribute to, shape analysis with the `Geomstats` library.

Acknowledgements

This research was partially funded by the National Science Foundation Materials Research Science and Engineering Center (MRSEC) at UC Santa Barbara (NSF DMR-2308708, Data Expert Group and IRG-2). L.F.P. was also partially funded by the MUR under the grant PRIN 2022 project "GEOPRIDE Geometric primitive fitting on 3D data for geometric analysis and 3D shapes". This work was also partially funded by the NSF CAREER 2240158. P.S.S. is grateful to the Max-Planck-Institute for Mathematics in Bonn for its hospitality and financial support.

References

[1] Martin Bauer, Martins Bruveris, Nicolas Charon, and Jakob Møller-Andersen. A relaxed approach for curve matching with elastic metrics, September 2018. arXiv:1803.10893 [math].

[2] Martin Bauer, Nicolas Charon, Philipp Harms, and Hsi-Wei Hsieh. A numerical framework for elastic surface matching, comparison, and interpolation. *International Journal of Computer Vision*, 129(8):2425–2444, 2021.

[3] Martin Bauer, Nicolas Charon, Eric Klassen, Sebastian Kurtek, Tom Needham, and Thomas Pierron. Elastic Metrics on Spaces of Euclidean Curves: Theory and Algorithms. *Journal of Nonlinear Science*, 34(3):56, April 2024.

[4] M Faisal Beg, Michael I Miller, Alain Trouvé, and Laurent Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International journal of computer vision*, 61:139–157, 2005.

[5] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy. *Polygon Mesh Processing*. CRC Press, October 2010.

[6] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the GPU, with autodiff, without memory overflows. *The Journal of Machine Learning Research*, 22(1):74:3457–74:3462, January 2021.

[7] Nicolas Charon, Benjamin Charlier, Joan Glaunès, Pietro Gori, and Pierre Roussillon. Fidelity metrics between curves and surfaces: currents, varifolds, and normal cycles. In Xavier Pennec, Stefan Sommer, and Tom Fletcher, editors, *Riemannian Geometric Statistics in Medical Image Analysis*, pages 441–477. Academic Press, January 2020.

[8] Keenan Crane. Discrete differential geometry: An applied introduction. *Notices of the AMS, Communication*, pages 1153–1159, 2018.

[9] Hugo Darmanté, Benoit Bugnas, Régis Bernard de Dompsure, Laurent Barresi, Nina Miolane, Xavier Pennec, Fernand Peretti, and Nicolas Bronsard. Analyse biométrique de l'anneau pelvien en 3 dimensions – à propos de 100 scanners. *Revue de Chirurgie Orthopédique et Traumatologique*, 100, 11 2014.

[10] Manfredo Perdigao Do Carmo. *Riemannian geometry*, volume 6. Springer, 1992.

[11] Ian Dryden and Kanti Mardia. *Statistical shape analysis, with Applications in R*. John Wiley & Sons, New York, 1998.

[12] Ian Dryden and Kantilal Mardia. *Statistical Shape Analysis, with Applications in R. Second Edition.* John Wiley and Sons, Chichester, 2016.

[13] Ashraf M.T. Elewa. *Morphometrics for Nonmorphometricians*. Number 57 in Lecture Notes in Earth Sciences. Springer Berlin, Heidelberg, Boca Raton, Florida, USA, 2010.

[14] Pengfei Fang, Mehrtash Harandi, Trung Le, and Dinh Phung. Hyperbolic geometry in computer vision: A survey, 2023.

[15] Thomas Fletcher. Geodesic regression and the theory of least squares on riemannian manifolds. *International Journal of Computer Vision*, 105(2):171–185, nov 2012.

[16] Simeon G, Piella G, Camara O, and Pareto D. Riemannian geometry of functional connectivity matrices for multi-site attention-deficit/hyperactivity disorder data harmonization. *Front Neuroinform*, 16(1):769274, 2022.

[17] Barbara Gris, Stanley Durrleman, and Alain Trouvé. A sub-riemannian modular framework for diffeomorphism-based analysis of shape ensembles. *SIAM Journal on Imaging Sciences*, 11(1):802–833, 2018.

[18] Nicolas Guigui, Elodie Maignant, Alain Trouvé, and Xavier Pennec. Parallel Transport on Kendall Shape Spaces. In *GSI 2021 - 5th conference on Geometric Science of Information*, volume 12829 of *Lecture Notes in Computer Science*, pages 103–110, Paris, France, July 2021. Springer.

[19] Nicolas Guigui, Nina Miolane, and Xavier Pennec. Introduction to riemannian geometry and geometric statistics: from basic theory to implementation with geomstats. (in press)., 2023.

[20] Thomas Hamelryck, John T Kent, and Anders Krogh. Sampling realistic protein conformations using local structural bias. *PLOS Computational Biology*, 2(9):1–13, 09 2006.

[21] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. Publisher: Nature Publishing Group.

[22] Emmanuel Hartman, Emery Pierson, Martin Bauer, Mohamed Daoudi, and Nicolas Charon. Basis restricted elastic shape analysis on the space of unregistered surfaces, November 2023. arXiv:2311.04382 [cs, math].

[23] Emmanuel Hartman, Yashil Sukurdeep, Eric Klassen, Nicolas Charon, and Martin Bauer. Elastic shape analysis of surfaces with second-order sobolev metrics: a comprehensive numerical framework. *International Journal of Computer Vision*, 131(5):1183–1209, 2023.

[24] Li Jia-Yi, Englund Elisabet, Holton Janice L, Soulet Denis, Hagell Peter, Lees Andrew J, Lashley Tammaryn, Quinn Niall P, Rehncrona Stig, Björklund Anders, Widner Håkan, Revesz Tamas, Lindvall Olle, and Brundin Patrik. Lewy bodies in grafted neurons in subjects with parkinson's disease suggest host-to-graft disease propagation. *Nat Med*, 14(5):501–3, 2008.

[25] Irene Kaltenmark, Benjamin Charlier, and Nicolas Charon. A General Framework for Curve and Surface Comparison and Registration With Oriented Varifolds. pages 3346–3355, 2017.

[26] David G. Kendall. Shape Manifolds, Procrustean Metrics, and Complex Projective Spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, 03 1984.

[27] John Kent and Thomas Hamelryck. Using the Fisher-Bingham distribution in stochastic models for protein structure. *Quantitative Biology, Shape Analysis, and Wavelets*, 24(1):57–60, 2005.

[28] Hamid Laga, Qian Xie, Ian H. Jermyn, and Anuj Srivastava. Numerical Inversion of SRNF Maps for Elastic Shape Analysis of Genus-Zero Surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2451–2464, December 2017. arXiv:1610.04531 [cs].

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

[30] Alice Le Brigant. A Discrete Framework to Find the Optimal Matching Between Manifold-Valued Curves. *Journal of Mathematical Imaging and Vision*, 61(1):40–70, January 2019.

[31] John M Lee. *Introduction to Riemannian manifolds*, volume 2. Springer, 2018.

[32] Xueming Li, Paul Mooney, Shawn Zheng, Christopher R Booth, Michael B Braunfeld, Sander Gubbens, David A Agard, and Yifan Cheng. Electron counting and beam-induced motion correction enable near-atomic-resolution single-particle cryo-em. *Nature methods*, 10(6):584–590, 2013.

[33] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989.

[34] Yi Long, Yudi Niu, Kaini Liang, and Yanan Du. Mechanical communication in fibrosis progression. *Trends in Cell Biology*, 32(1):70–90, 2022.

[35] Marco Lorenzi, Nicholas Ayache, Giovanni B. Frisoni, and Xavier Pennec. Mapping the effects of aβ1 - 42 levels on the longitudinal changes in healthy aging: Hierarchical modeling based on stationary velocity fields. In Gabor Fichtinger, Anne Martel, and Terry Peters, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2011*, pages 663–670, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[36] Yimin Luo, Mengyang Gu, Minwook Park, Xinyi Fang, Younghoon Kwon, Juan Manuel Urueña, Javier Read de Alaniz, Matthew E Helgeson, Cristina M Marchetti, and Megan T Valentine. Molecular-scale substrate anisotropy, crowding and division drive collective behaviours in cell monolayers. *Journal of the Royal Society Interface*, 20(204):20230160, 2023.

[37] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML workshop*, volume 238, 2015. Issue: 5.

[38] Pascal Mettes, Mina Ghadimi Atigh, Martin Keller-Ressel, Jeffrey Gu, and Serena Yeung. Hyperbolic deep learning in computer vision: A survey, 2023.

[39] Washington Mio, Anuj Srivastava, and Shantanu Joshi. On shape of plane elastic curves. *International Journal of Computer Vision*, 73(3):307–324, 2007.

[40] N. Miolane, S. Holmes, and X. Pennec. Template shape estimation: Correcting an asymptotic bias. *SIAM Journal on Imaging Sciences*, 10(2), 2017.

[41] Nina Miolane, Nicolas Guigui, Alice Le Brigant, Johan Mathe, Benjamin Hou, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, et al. Geomstats: a python package for riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9, 2020.

[42] Adele Myers and Nina Miolane. Regression-based elastic metric learning on shape spaces of cell curves. In *NeurIPS 2022 Workshop on Learning Meaningful Representations of Life*, 2022.

[43] Adele Myers, Caitlin Taylor, Emily Jacobs, and Nina Miolane. Geodesic regression characterizes 3d shape changes in the female brain during menstruation. pages 2534–2543, 10 2023.

[44] Esfandiar Nava-Yazdani, Hans-Christian Hege, T. J. Sullivan, and Christoph von Tycowicz. Geodesic Analysis in Kendall's Shape Space with Epidemiological Applications. *Journal of Mathematical Imaging and Vision*, 62(4):549–559, May 2020.

[45] Tom Needham and Sebastian Kurtek. Simplifying transforms for general elastic metrics on the space of plane curves. *SIAM Journal on Imaging Sciences*, 13(1):445–473, 2020.

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 2019.

[47] Xavier Pennec. 3 - manifold-valued image processing with spd matrices. In Xavier Pennec, Stefan Sommer, and Tom Fletcher, editors, *Riemannian Geometric Statistics in Medical Image Analysis*, pages 75–134. Academic Press, 2020.

[48] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian Framework for Tensor Computing. *International Journal of Computer Vision*, 66(1):41–66, 1 2006.

[49] D. Robinson, S. Lahiri, E. Klassen, and M. Bruveris. LibSRVF, 2018.

[50] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[51] Martin Rumpf and Benedikt Wirth. Variational time discretization of geodesic calculus. *IMA Journal of Numerical Analysis*, 35(3):1011–1046, July 2015. Conference Name: IMA Journal of Numerical Analysis.

[52] Nahian Siddique, Sidike Paheding, Colin P Elkin, and Vijay Devabhaktuni. U-net and its variants for medical image segmentation: A review of theory and applications. *Ieee Access*, 9:82031–82057, 2021.

[53] Christopher G Small. *The statistical theory of shape*. Springer series in statistics. Springer, New York, NY, 1996 edition, December 2012.

[54] Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: A structural description of the human brain. *PLoS Computational Biology*, 1(4):0245–0251, 2005.

[55] Anuj Srivastava, Ian Jermyn, and Shantanu Joshi. Riemannian analysis of probability density functions with applications in vision. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

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

[57] Anuj Srivastava and Eric P. Klassen. *Functional and Shape Data Analysis*. Springer, October 2016. Google-Books-ID: 0cMwDQAAQBAJ.

[58] D Stoyan. Bookstein, f. l.: Morphometric tools for landmark data. geometry and biology. cambridge university press, Cambridge/New York/Port Chester/Melbourne/Sydney 1991, XVIII, 435 s., £ 50.00; $ 89.95. *Biom. J.*, 35(4):512–512, 1993.

[59] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nature methods*, 18(1):100–106, 2021.

[60] Zhe Su, Martin Bauer, Stephen C. Preston, Hamid Laga, and Eric Klassen. Shape Analysis of Surfaces Using General Elastic Metrics. *Journal of Mathematical Imaging and Vision*, 62(8):1087–1106, October 2020.

[61] Marisa Tisler, Samuel Alkmin, Hsin-Yu Chang, Jon Leet, Ksenija Bernau, Nathan Sandbo, and Paul J Campagnola. Analysis of fibroblast migration dynamics in idiopathic pulmonary fibrosis using image-based scaffolds of the lung extracellular matrix. *American Journal of Physiology-Lung Cellular and Molecular Physiology*, 318(2):L276–L286, 2020.

[62] J.D. Tucker. Fdasrsf. `https://github.com/jdtuck/fdasrsf_python`.

[63] Amelia Villegas-Morcillo, Victoria Sanchez, and Angel M. Gomez. Foldhsphere: deep hyperspherical embeddings for protein fold recognition. *BMC Bioinformatics*, 22(1):490, 2021.

[64] Laurent Younes. Spaces and manifolds of shapes in computer vision: An overview. *Image and Vision Computing*, 30(6-7):389–397, 2012.

[65] Ying Yuan, Hongtu Zhu, Weili Lin, and James Stephen Marron. Local polynomial regression for symmetric positive definite matrices. *Journal of the Royal Statistical Society Series B*, 74(4):697–719, 2012.