

---

# RDA-INR: RIEMANNIAN DIFFEOMORPHIC AUTOENCODING VIA IMPLICIT NEURAL REPRESENTATIONS

---

A PREPRINT

 **Sven Dummer**

Mathematics of Imaging & AI (MIA)  
Department of Applied Mathematics  
University of Twente  
Drienerlolaan 5, Enschede 7522 NB  
the Netherlands  
s.c.dummer@utwente.nl

 **Nicola Strisciuglio**

Data Management & Biometrics (DMB)  
Department of Computer Science  
University of Twente  
Drienerlolaan 5, Enschede 7522 NB  
the Netherlands  
n.strisciuglio@utwente.nl

 **Christoph Brune**

Mathematics of Imaging & AI (MIA)  
Department of Applied Mathematics  
University of Twente  
Drienerlolaan 5, Enschede 7522 NB  
the Netherlands  
c.brune@utwente.nl

## ABSTRACT

Diffeomorphic registration frameworks such as Large Deformation Diffeomorphic Metric Mapping (LDDMM) are used in computer graphics and the medical domain for atlas building, statistical latent modeling, and pairwise and groupwise registration. In recent years, researchers have developed neural network-based approaches regarding diffeomorphic registration to improve the accuracy and computational efficiency of traditional methods. In this work, we focus on a limitation of neural network-based atlas building and statistical latent modeling methods, namely that they either are (i) resolution dependent or (ii) disregard any data/problem-specific geometry needed for proper mean-variance analysis. In particular, we overcome this limitation by designing a novel encoder based on resolution-independent implicit neural representations. The encoder achieves resolution invariance for LDDMM-based statistical latent modeling. Additionally, the encoder adds LDDMM Riemannian geometry to resolution-independent deep learning models for statistical latent modeling. We showcase that the Riemannian geometry aspect improves latent modeling and is required for a proper mean-variance analysis. Furthermore, to showcase the benefit of resolution independence for LDDMM-based data variability modeling, we show that our approach outperforms another neural network-based LDDMM latent code model. Our work paves a way to more research into how Riemannian geometry, shape/image analysis, and deep learning can be combined.

**Keywords** shape space · Riemannian geometry · principal geodesic analysis · LDDMM · diffeomorphic registration · latent space · implicit neural representations

## 1 Introduction

Shape and image registration are important tools for analyzing object similarities and differences. To register two images  $I$  and  $J$ , a function  $f(x)$  is constructed such that point  $f(x)$  in image  $J$  corresponds to  $x$  in image  $I$ . For instance, when comparing two human brain images, one wants to match points on the same subpart of the brain. When instead matching two shapes  $\mathcal{S}$  and  $\mathcal{T}$ , a function  $f(x)$  should assign to each point  $x$  on  $\mathcal{S}$  a point  $f(x)$  on  $\mathcal{T}$  in a meaningful way. For example, when comparing two human shapes, one wants to match points on the arm of one human

with points on the same arm of the other human. These image and shape registrations have, for instance, applications in the medical domain. As the human anatomy varies from person to person, registration algorithms transform these anatomies into a single coordinate frame. Once the anatomies are registered, it is possible to compare them and establish correspondences between them.

Various registration frameworks are available, among which diffeomorphic registration frameworks such as Stationary Velocity fields (SVF) [22, 23, 24, 25] and Large Deformation Diffeomorphic Metric Mapping (LDDMM). One specific registration task is pairwise registration where a registration between exactly two objects is calculated [26, 27, 28, 29, 30, 31, 1]. To register multiple objects simultaneously, groupwise registration algorithms [32, 33, 34, 30] register all the objects to the same template. In this process, the template can also be estimated to perform atlas building. The estimated atlas is used for proper assessment of the differences between objects. Finally, besides plain diffeomorphic registration, LDDMM is also used for modeling the variability of the data. Specifically, using the LDDMM Riemannian distance, one can use the manifold variant of principal component analysis (PCA), called principal geodesic analysis (PGA), to estimate the factors of variation given the data [35, 36].

Recently, these tasks have been solved via neural network models. For instance, neural network models exist for speeding up the computation of pairwise registrations [2, 3, 4, 5, 11, 12], groupwise registrations [5, 7, 14, 13, 17], and joint groupwise registration and atlas building [7, 14, 17]. Furthermore, attempts have been made to improve LDDMM PGA by adding neural networks [16, 15]. Finally, all the approaches described above require a specific discretization of the objects and the underlying registration domain. Often one does not immediately know the optimal discretization, one might get memory issues when one refines the discretization, and neural network models might not generalize well to other resolutions. Recently, resolution-independent methods avoiding this discretization showed improved performance for diffeomorphic registration [21, 10, 18, 8, 9] and non-diffeomorphic registration [37].

Table 1 compares a comprehensive selection of relevant and representative neural network methods based on their main features. In particular, we focus on atlas building, statistical latent modeling, physical consistency using LDDMM Riemannian geometry, and resolution invariance/independence. In this work, we only classify models as physically consistent if they use LDDMM, as it enables time-dependent non-autonomous behavior. In contrast, SVF, for example, only allows autonomous stationary velocity vector fields. From the table, we notice that many methods do not have all

Table 1: Table comparing neural network-based diffeomorphic registration methods. The columns showcase the features that a paper treats or does not treat. Our novel RDA-INR framework is the only work that addresses all the features.

<i>Model</i>	<i>Features</i>			
	Atlas-building	Stat. latent modelling	Physical consistency	Res. indep.
ResNet-LDDMM [1]	x	x	✓	x
R2Net [2]	x	x	x	x
Krebs. et al. [3]	x	✓	x	x
Shen et al. [4]	x	x	x	x
Quicksilver [5]	x	x	✓	x
Niethammer et al. [6]	x	x	x	x
Diff. Voxelmorph [7]	✓	x	x	x
DNVF [8]	x	x	x	✓
NePhi [9]	x	x	x	✓
Zou et al. [10]	x	x	x	✓
Fourier-Net+ [11]	x	x	x	x
DeepFLASH [12]	x	x	✓	x
LapIRN [13]	x	x	x	x
Aladdin [14]	✓	x	x	x
LDDMM-AE [15]	✓	✓	✓	x
DAE [16]	✓	✓	✓	x
Geo-Sic [17]	✓	x	✓	x
NeurEPDiff [18]	x	x	✓	✓
FlowSSM [19]	x	✓	x	✓
NMF [20]	x	✓	x	✓
NDF [21]	✓	✓	x	✓
RDA-INR (Ours)	✓	✓	✓	✓

the properties. For instance, to the best of our knowledge, no method exists for performing atlas building and statistical latent modeling that is physically consistent and resolution independent.

There are two types of algorithms for joint atlas building and statistical latent modeling. On the one hand, we have models that consider resolution-independent methods for shape data [38, 39, 21] but do not consider LDDMM physical consistency. This physical consistency is required for properly dealing with longitudinal data and to properly calculate the Fréchet mean and variance of the dataset:

$$\mu = \arg \min_O \frac{1}{N} \sum_{i=1}^N d(O, O_i)^2, \quad \sigma^2 = \frac{1}{N} \sum_{i=1}^N d(\mu, O_i)^2$$

where  $d$  is the Riemannian distance induced by LDDMM and  $\{O_i\}_{i=1}^N$  are  $N$  images or shapes. On the other hand, we have the LDDMM literature that considers physically consistent methods [16, 15] that are not resolution-independent.

## 1.1 Contribution

Our work aims to illustrate the general benefit of the LDDMM physical consistency and the benefit of resolution-independence properties. To achieve this goal, we introduce a novel model called Riemannian Diffeomorphic Autoencoding via Implicit Neural Representations (RDA-INR). Our model is designed to deal with the previously mentioned issues of joint atlas building and statistical latent modeling. Although our model is also applicable to images, we mainly focus on the special case of shapes.

Figure 1 shows how we arrive at our model. More precisely, our RDA-INR adds resolution-independent implicit neural representations (INRs) to LDDMM PGA and adds Riemannian geometry to deep learning models for shape variability modeling. Our method shares similarities with the deformable template model developed by Sun et al. [21]. However, our model differs from theirs as we merge it with the LDDMM framework. Specifically, we use the Riemannian deformation cost of LDDMM as a deformation regularizer. Moreover, instead of the signed distance representation of shapes used by Sun et al. [21], we consider point cloud and mesh data.

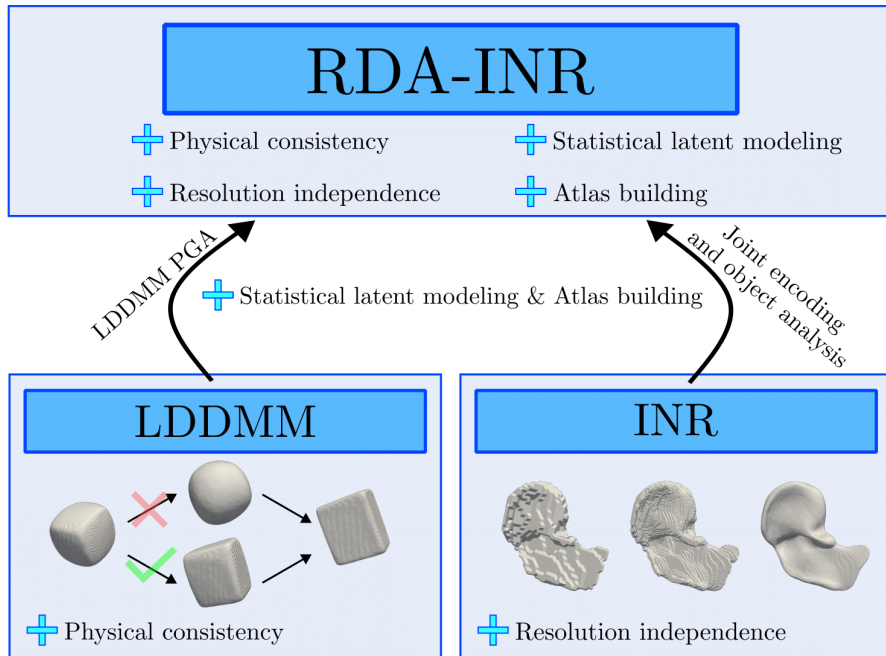


Figure 1: Overview of our RDA-INR framework. We combine LDDMM PGA and resolution-independent implicit neural representation (INR) methods used for joint encoding and registration. This combination yields a method for statistical latent modeling and atlas building that is (i) physically consistent via LDDMM Riemannian geometry and (ii) resolution independent.

First, we investigate the benefit of physical consistency. We treat the situation depicted in Figure 2 and show that the Riemannian geometry is required for a proper mean-variance analysis. More precisely, we show that the Riemannian geometry is needed to simultaneously achieve two objectives: obtaining a template as the Fréchet mean of the data and

obtaining physically plausible deformations that align the template to the data. Finally, we show that the Riemannian geometry improves the latent modeling as it leads to improved reconstruction generalization and more robustness to noise.

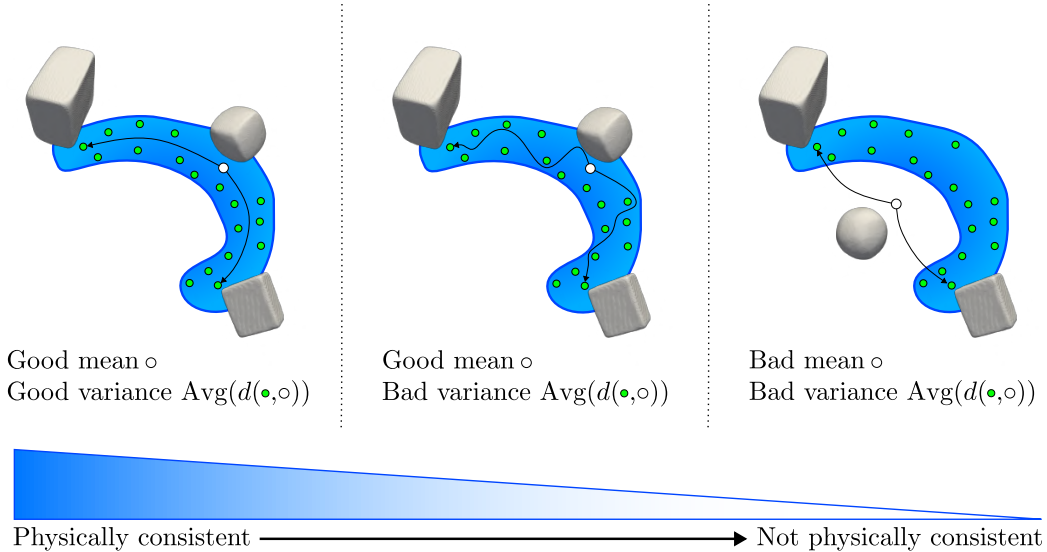


Figure 2: Three latent model classes defined by mean-variance analysis capabilities. The calculation of the Fréchet mean and the variance of a dataset depends on the length of the shortest path between objects. However, not all latent models take this into account, which results in three categories of methods: methods considering the geodesic distance, methods that do not consider the geodesic distance but still obtain a good Fréchet mean, and methods that do not consider the geodesic distance and do not obtain a good Fréchet mean.

While the previous results illustrate the importance of physical consistency, we also treat the importance of resolution independence by comparing our method to DAE [16]. We show that our model obtains higher-quality templates and higher-quality reconstructions, demonstrating the benefit of resolution independence.

## 1.2 Outline

We organize the rest of the paper as follows. In Section 2, we discuss the related works. Subsequently, in Section 3, we treat the background theory needed for our method. Specifically, we treat INRs and the implicit representation of shapes (Section 3.1), the LDDMM framework (Section 3.2), and LDDMM PGA (Section 3.3). Section 4 introduces our neural network model. In Section 5 we compare this model to the same model with a non-Riemannian regularization. We also compare our method to DAE [16]. Finally, we summarize the paper, discuss some limitations, and discuss future work in Section 6. In the appendices, we provide additional experiments and details, and some basics of Riemannian geometry.

## 2 Related work

In this section, we contextualize our contributions by discussing related literature on principal geodesic analysis (PGA), Riemannian geometry for latent space models, and neural ordinary differential equations (NODE).

### 2.1 Principal geodesic analysis

Principal component analysis (PCA) is a well-known technique that has applications in, among others, dimensionality reduction, data visualization, and feature extraction. In addition, the probabilistic extension of PCA, called Probabilistic PCA, can also be used to generate new samples from the data distribution [40]. A limitation of PCA and probabilistic PCA is that they can only be applied to Euclidean data and not Riemannian manifold-valued data. To solve this limitation, principal geodesic analysis (PGA) [41] and probabilistic principal geodesic analysis (PPGA) [42] are introduced as extensions of PCA and probabilistic PCA, respectively.

PGA is applied in various domains. For instance, optimal transport (OT) uses PGA to define Fréchet means and interpolations of data distributions called barycenters [43, 44]. Shape analysis uses PGA as well [41]. Different versions

of shape PGA are obtained by choosing different Riemannian distances on the shape space. For instance, while in Zhang and Fletcher [35] and Zhang et al. [36] an LDDMM-based distance is used for PGA, Heeren et al. [45] use a thin shell energy as the Riemannian distance. Furthermore, besides shape analysis based on PGA, shape analysis tools exist that use tangent PCA [46, 30] or that use a combination of tangent PCA and geodesic PCA [47].

Similar to how an autoencoder [48, 49] is a nonlinear version of PCA, our encoding framework can be viewed as a nonlinear version of PGA and PPGA. Another neural network model that can be regarded as a nonlinear version of PPGA is the Riemannian variational autoencoder (Riemannian VAE) [50]. The Riemannian VAE only considers learning a latent space for data on finite-dimensional Riemannian manifolds for which the exponential map and the Riemannian distance have an explicit formula. Hence, it can not be used for the infinite-dimensional Riemannian manifold of shapes and images for which the LDDMM Riemannian distance does not have an explicit formula. Our deformable template model extends the Riemannian VAE to this setting.

Bone et al. [16] and Hinkle et al. [15] construct an autoencoder that resembles LDDMM PGA [35]. While Hinkle et al. [15] is only applicable to images, Bone et al. [16] is also applicable to meshes. Similar to our work, both use an ordinary differential equation (ODE) to deform a learned template into a reconstruction. Moreover, they introduce a regularizer on the velocity vector fields to let the ODE flow be a diffeomorphism. In contrast, while they use a specific resolution for the images, shapes, and diffeomorphisms, we construct a resolution-independent method. For instance, while Bone et al. [16] consider the mesh representation of shapes, we consider the state-of-the-art implicit shape representation. Implicit shape representations allow us to acquire infinite-resolution shape reconstructions because the shapes are represented by the zero level set of a function.

## 2.2 Riemannian geometry for latent space models

Latent space models based on neural networks are used for, among others, dimensionality reduction, clustering, and data augmentation. Most of these models have the drawback that distances between latent codes do not always correspond to a ‘semantic’ distance between the objects they represent. One mathematical tool to solve this issue is Riemannian geometry. Arvanitidis et al. [51] design a Riemannian distance between latent codes by defining the corresponding Riemannian metric. This metric is defined as the standard Euclidean metric pulled back under the action of the decoder. The shortest paths under this Riemannian metric are found by solving the geodesic equation via a numerical solver. However, one can also find the shortest path by solving the geodesic equation via Gaussian processes and fixed point iterations [52]. Another approach is to solve the geodesic minimization problem [53, 54]. In Arvanitidis et al. [55] a surrogate Riemannian metric is proposed that approximates the Euclidean pullback metric and yields more robust calculations of shortest paths. In Chadebec et al. [54], the Riemannian metric is incorporated into a Hamiltonian Markov Chain to sample from the posterior distribution of a VAE. Geng et al. [56] enforce a Riemannian distance structure by encouraging Euclidean distances in latent space to equal the Riemannian distance in the original space. Consequently, linear interpolations correspond to points on the learned manifold.

Besides using the Riemannian metrics and the shortest paths for defining good distances, creating good interpolations, and defining probability distributions on the latent space, they are also used for improved shape generation [54], for improved clustering [57, 54], and for data augmentation [58].

Our novel latent model that uses implicit neural representations (INRs) is related to the Riemannian geometry for latent space models literature. The reason is that we embed a Riemannian distance between a learned template and the other reconstructed data points by using LDDMM as the Riemannian distance. The most relevant work that also attempts to incorporate shape Riemannian geometry into an INR latent space model is Atzmon et al. [59]. However, their model can not perform shape encoding and shape registration jointly. Moreover, it does not enforce a diffeomorphic structure.

## 2.3 Neural ODEs

The neural ODE (NODE) [60] is a first-order differential equation where the time derivative is parameterized by a neural network. It has applications in learning dynamics [61, 62, 63, 64], control [65, 66], generative modeling [60, 67], and joint shape encoding, reconstruction, and registration [68, 20, 21]. Furthermore, there is a bidirectional connection between the NODE and optimal transport (OT). First, due to the fluid dynamics formulation of OT [69], NODEs can be used to solve high-dimensional OT problems [70]. In addition, the learned dynamics can be complex and require many time steps to be solved accurately. As a consequence, training NODEs can be challenging and time-consuming. OT-inspired regularization functionals solve this issue as they simplify the dynamics and reduce the time needed to solve the NODE [71, 72].

Our framework fits the bidirectional connection of NODEs and OT. First, we use a neural ODE to solve the LDDMM problem, which is a problem that in formulation bears similarities to the fluid dynamics formulation of OT. Hence, our

approach shares similarities with the works using NODEs to solve OT problems. In addition, we use cost functionals inspired by LDDMM to regularize the NODE used for reconstruction and registration. This regularization fits the works that use OT for regularizing the dynamics of NODEs.

### 3 Preliminaries

#### 3.1 Implicit neural representations and implicit shape representations

In many variational problems, we need to optimize for an unknown image, a solution to a partial differential equation, a velocity vector field for registration, or a shape. The idea behind INRs is to represent such objects by a neural network  $f_\theta : \Omega \rightarrow \mathbb{R}^n$ , where  $\Omega$  can be a spatial or a spatiotemporal domain. This representation avoids discretization of the objects and by optimizing over the neural network weights, we obtain a solution in an infinite dimensional space. For example, methods that use  $f_\theta$  as parameterization of a velocity vector field or a deformation field achieve good performance in diffeomorphic [10, 18, 8, 9] and non-diffeomorphic registration [37].

When representing shapes with INRs, the implicit shape representation is of interest. A shape  $\mathcal{S} := \{x \mid f(x) = c\}$  is described by the  $c$  level set of a function  $f : \Omega \rightarrow \mathbb{R}$  on the image domain  $\Omega \subset \mathbb{R}^d$ . Given a shape  $\mathcal{S} = \partial\mathcal{S}_{\text{int}}$  with interior  $\mathcal{S}_{\text{int}} \subset \Omega$ , two examples of common implicit representations are the signed distance function (SDF) with  $c = 0$  and the occupancy function (OCC) with  $c = 0.5$ , respectively:

$$\text{SDF}_{\mathcal{S}}(x) = \begin{cases} \inf_{y \in \mathcal{S}} \|x - y\|_2 & \text{if } x \in \mathcal{S}_{\text{int}}, \\ -\inf_{y \in \mathcal{S}} \|x - y\|_2 & \text{if } x \notin \mathcal{S}_{\text{int}}, \end{cases} \quad (1)$$

$$\text{OCC}_{\mathcal{S}}(x) = \begin{cases} 0 & \text{if } x \notin \mathcal{S}_{\text{int}}, \\ 0.5 & \text{if } x \in \mathcal{S}, \\ 1 & \text{if } x \in \mathcal{S}_{\text{int}}. \end{cases} \quad (2)$$

Unlike the traditional discrete representations such as point clouds, meshes, or voxel grids, the implicit shape representation is continuous as it represents the shape using a function. Consequently, we can obtain a mesh or point cloud at an arbitrary resolution by using marching cubes [73]. INR methods that use this continuous representation, such as DeepSDF [74], SIREN [75], and Occupancy Network [76], yield state-of-the-art shape encodings and reconstructions. However, unlike the discrete representations, the implicit representation has the disadvantage of not providing point correspondences between shape deformations. To remedy this issue, recent works [38, 39, 21] create template-based INR models that allow for joint registration and encoding. These models parameterize a template shape with an INR and transform it into other shapes to allow for shape matching.

#### 3.2 Diffeomorphic registration and the LDDMM Riemannian distance

In this work, we focus on the LDDMM diffeomorphic registration framework. The objective of LDDMM is to find a diffeomorphism  $\phi : \Omega \rightarrow \Omega$  that matches two objects  $O_1$  and  $O_2$  (e.g., images or shapes defined on the spatial domain  $\Omega \subset \mathbb{R}^d$ ) by deforming  $O_1$  into  $O_2$ . More precisely, LDDMM wants  $\phi \cdot O_1 = O_2$  with  $\phi \cdot O$  a left group action of the diffeomorphism group on the set of objects. For instance, an image  $I : \Omega \rightarrow \mathbb{R}^n$  can be deformed by a left group action  $\phi \cdot I := I \circ \phi^{-1}$  and a shape  $\mathcal{S} := \{x \mid x \text{ on the shape}\}$  can be deformed by a left group action  $\phi \cdot \mathcal{S} := \phi(\mathcal{S})$ .

To construct a diffeomorphism that achieves  $\phi \cdot O_1 = O_2$ , LDDMM considers a subgroup of the diffeomorphism group. This subgroup consists of diffeomorphisms emerging as flows of ordinary differential equations (ODEs). Define a time-dependent vector field  $v : \Omega \times [0, 1] \rightarrow \mathbb{R}^d$  and assume  $v(\cdot, t) \in V$  for some Banach space  $V$ . The corresponding ODE is  $\frac{dx}{dt} = v(x, t)$  and it gives us a flow of diffeomorphisms:

$$\begin{aligned} \frac{\partial \phi_t}{\partial t}(x) &= v(\phi_t(x), t), \\ \phi_0(x) &= x. \end{aligned} \quad (3)$$

To make these ODE flows diffeomorphisms, one has to enforce smoothness on the velocity vector fields  $v(\cdot, t) \in V$ . Generally, this is accomplished by choosing an appropriate Banach space  $V$  and assuming  $v \in L^1([0, 1], V)$ . One class of such spaces are admissible Banach spaces:

**Definition 1** (Admissible Banach spaces [30]). Let  $C_0^1(\Omega, \mathbb{R}^d)$  be the Banach space of continuously differentiable vector fields  $\nu$  on the open and bounded domain  $\Omega \subset \mathbb{R}^d$  such that both  $\nu$  and its Jacobian  $J\nu$  vanish on  $\partial\Omega$  and at infinity. Furthermore, define a Banach space  $V \subset C_0^1(\Omega, \mathbb{R}^d)$  and let  $\|\nu\|_{1,\infty} := \|\nu\|_\infty + \|J\nu\|_\infty$  for  $\nu \in V$ . The Banach space  $V$  is admissible if it is (canonically) embedded in  $(C_0^1(\Omega, \mathbb{R}^d), \|\cdot\|_{1,\infty})$ . In other words, there exist a constant  $C$  such that  $\forall \nu \in V, \|\nu\|_V \geq C\|\nu\|_{1,\infty}$ .

Using these considerations, the subgroup that LDDMM considers is defined as

$$G := \{\phi_1 \mid \phi_t \text{ satisfies Equation (3) for some } v \in L^1([0, 1], V)\}.$$

There might exist multiple  $\phi \in G$  such that  $\phi \cdot O_1 = O_2$  for objects  $O_1$  and  $O_2$ . The LDDMM approach selects the 'smallest'  $\phi \in G$  such that  $\phi \cdot O_1 = O_2$ . To define 'smallest', LDDMM uses the following Riemannian distance  $d_G$  on  $G$ :

$$\begin{aligned} d_G(\phi, \psi) &:= \inf_{v \in L^1([0,1],V)} \left( \int_0^1 \|v(\cdot, t)\|_V dt \right) \\ \text{s.t.} \quad &\frac{\partial \phi_t}{\partial t}(x) = v(\phi_t(x), t), \quad \phi_0(x) = x, \\ &\psi = \phi_1 \circ \phi. \end{aligned}$$

where  $\|\cdot\|_V$  is a norm on  $V$ . The LDDMM approach selects the  $\phi \in G$  such that  $\phi \cdot O_1 = O_2$  and  $d_G(\text{id}, \phi)$  is small. Alternatively, in case the objects  $O_1$  and  $O_2$  belong to a set  $\mathcal{O}$  of diffeomorphic objects, we can reformulate this problem as finding a distance between the two objects  $O_1$  and  $O_2$ :

**Theorem 1** ([30]). *Assume that  $V$  is an admissible Banach space. Then  $(G, d_G)$  is a complete metric space. Furthermore, assume we consider a set  $\mathcal{O} := \{\phi \cdot O_{\text{temp}} \mid \phi \in G\}$  for some template object  $O_{\text{temp}}$ . Then we can define a pseudo-distance  $d_{\mathcal{O}}(O_1, O_2)$  on  $\mathcal{O}$  as*

$$d_{\mathcal{O}}(O_1, O_2) := \inf_{\phi} (d_G(\text{id}, \phi) \mid O_2 = \phi \cdot O_1, \phi \in G) \quad (4)$$

or alternatively:

$$\begin{aligned} d_{\mathcal{O}}(O_1, O_2) &= \inf_{v(\cdot, t)} \left( \int_0^1 \|v(\cdot, t)\|_V dt \right) \\ \text{s.t.} \quad &\frac{\partial \phi_t}{\partial t}(x) = v(\phi_t(x), t), \quad \phi_0(x) = x, \\ &\phi_1 \cdot O_1 = O_2. \end{aligned} \quad (5)$$

In addition,  $d_{\mathcal{O}}$  is a distance if the action  $\phi \rightarrow \phi \cdot O_{\text{temp}}$  is continuous from  $G$  to  $\mathcal{O}$  given (i) the topology induced by  $d_G$  on  $G$  and (ii) some topology on  $\mathcal{O}$ .

One particular example when  $\phi \rightarrow \phi \cdot O_{\text{temp}}$  is continuous as a mapping from  $G$  to  $\mathcal{O}$  is when considering the shape  $O_{\text{temp}} = \{x \mid f(x) = 0\} = \partial A$  for some continuous function  $f$  and  $A \subset \mathbb{R}^d$  some non-empty compact set. In this case  $\phi \cdot O_{\text{temp}} := \phi(O_{\text{temp}})$  and we use the topology induced by the Hausdorff distance on  $\mathcal{O}$ . For more in-depth information on the case of shapes seen as submanifolds of  $\mathbb{R}^d$ , we refer to Bauer et al. [77]. Additionally, when considering images as functions  $I : \Omega \rightarrow \mathbb{R}^n$  and using the group action  $\phi \cdot I := I \circ \phi^{-1}$ ,  $d_{\mathcal{O}}$  is a distance for a specific class of  $\|\cdot\|_V$  [26, 78].

As mentioned above, the goal of LDDMM is to approximately solve Equations (4) and (5). However, it is known that the solution  $\phi_t$  to Equation (5) is reparameterization invariant. Hence, there exists an infinite number of  $t \rightarrow \phi_t$  that trace out the same curve in the diffeomorphism group and that all have the same integral cost. To resolve this issue, LDDMM minimizes an energy that is not reparameterization invariant:

**Theorem 2** ([30]). *Assume that  $V$  is an admissible Banach space and let  $\mathcal{O} := \{\phi \cdot O_{\text{temp}} \mid \phi \in G\}$  for some template object  $O_{\text{temp}}$ . Then we can define the energy  $E_G$  between two diffeomorphisms  $\phi, \psi \in G$  as*

$$\begin{aligned} E_G(\phi, \psi) &:= \inf_{v \in L^1([0,1],V)} \left( \int_0^1 \|v(\cdot, t)\|_V^2 dt \right) \\ \text{s.t.} \quad &\frac{\partial \phi_t}{\partial t}(x) = v(\phi_t(x), t), \quad \phi_0(x) = x, \\ &\psi = \phi_1 \circ \phi. \end{aligned} \quad (6)$$

Then  $E_G(\phi, \psi) = d_G(\phi, \psi)^2$ . Moreover, we define the induced energy  $E_{\mathcal{O}}$  on  $\mathcal{O}$  as

$$E_{\mathcal{O}}(O_1, O_2) := \inf_{\phi} (E_G(\text{id}, \phi) \mid O_2 = \phi \cdot O_1, \phi \in G)$$

or alternatively:

$$\begin{aligned} E_{\mathcal{O}}(O_1, O_2) &= \inf_{v(\cdot, t)} \left( \int_0^1 \|v(\cdot, t)\|_V^2 dt \right) \\ \text{s.t.} \quad &\frac{\partial \phi_t}{\partial t}(x) = v(\phi_t(x), t), \quad \phi_0(x) = x, \\ &\phi_1 \cdot O_1 = O_2. \end{aligned} \quad (7)$$

As  $E_G(\phi, \psi) = d_G(\phi, \psi)^2$ ,  $E_{\mathcal{O}}(O_1, O_2) = d_{\mathcal{O}}(O_1, O_2)^2$ . More precisely, it can be shown that any  $\phi_t$  solving the energy minimization problem (7) solves the distance problem in Equation (5). Conversely, any constant speed curve  $t \rightarrow \phi_t$  (i.e.,  $\|v(\cdot, t)\|_V$  stays constant over time) that solves the distance problem in Equation (5) solves the energy minimization problem in Equation (7).

Using Theorem 2, LDDMM solves optimization problem (7) instead of optimization problem (5). To solve the energy minimization problem (7), LDDMM solves the relaxed problem:

$$\begin{aligned} \inf_{v(\cdot, t)} \quad & \mathcal{D}(\phi_1 \cdot O_1, O_2) + \sigma^2 \int_0^1 \|v(\cdot, t)\|_V^2 dt \\ \text{s.t.} \quad & \frac{\partial \phi_t}{\partial t}(x) = v(\phi_t(x), t), \quad \phi_0(x) = x, \end{aligned}$$

where  $\mathcal{D}$  is some data fidelity term and  $\sigma \in \mathbb{R}$ . For instance, when dealing with images, one can use  $\mathcal{D}(I_1, I_2) = \|I_1 - I_2\|_{L_2(\Omega)}^2$  and  $\phi \cdot I = I \circ \phi^{-1}$ .

### 3.3 Diffeomorphic latent modeling via LDDMM PGA

Besides registering two objects  $O_1$  and  $O_2$ , the Riemannian distance in Theorem 1 can be used for statistical latent modeling. Specifically, LDDMM is combined with principal geodesic analysis (PGA) for diffeomorphic latent modeling. In the original papers, a Bayesian approach introduces LDDMM PGA. However, we introduce LDDMM PGA from a slightly different perspective, namely closer to the original PGA formulation in [41].

The basic idea behind PGA is to first estimate a Fréchet mean and subsequently estimate a geodesic submanifold that best explains the data. For finding the Fréchet mean within the LDDMM PGA framework, we consider the set  $\mathcal{O} := \{\phi \cdot O_{\text{temp}} \mid \phi \in G\}$  for some reference template  $O_{\text{temp}}$ . This reference template is arbitrary as for every  $O \in \mathcal{O}$ , we have  $O = \{\phi \cdot O \mid \phi \in G\}$ , which means that any  $O$  can be seen as the template. In LDDMM PGA, the Fréchet mean is a template  $\mathcal{T} \in \mathcal{O}$  that minimizes the variance of the data. More precisely, the Fréchet mean should solve:

$$\min_{\mathcal{T}} \quad \mathbb{E}_{O \sim \rho} [d_{\mathcal{O}}(\mathcal{T}, O)^2],$$

with  $\rho$  the distribution of objects  $O$  and  $d_{\mathcal{O}}$  given as the distance in Equations (4) and (5). Using Theorem 2, we can also minimize:

$$\min_{\mathcal{T}} \quad \mathbb{E}_{O \sim \rho} [E_{\mathcal{O}}(\mathcal{T}, O)].$$

Writing out  $E_{\mathcal{O}}$  using Equation (7) and approximating the expectation using  $N$  available samples  $\{O_i\}_{i=1}^N$ , we get:

$$\begin{aligned} \min_{\{v_i(\cdot, t)\}_{i=1}^N, \mathcal{T}} \quad & \frac{1}{N} \sum_{i=1}^N \left[ \int_0^1 \|v_i(\cdot, t)\|_V^2 dt \right] \\ \text{s.t.} \quad & \frac{\partial \phi_t^i}{\partial t}(x) = v_i(\phi_t^i(x), t), \quad \phi_0^i(x) = x, \\ & \phi_1^i \cdot \mathcal{T} = O_i. \end{aligned} \tag{8}$$

An optimization over a time-dependent velocity field is difficult. To circumvent this issue, the diffeomorphism can be parameterized via an initial velocity vector field  $v_0$ . More precisely, if  $V$  is a Hilbert space, the extrema of Equations (6) and (7) satisfy the so-called EPDiff equation. This equation maps an initial velocity vector field  $v_0 \in V$  to a time-dependent velocity vector field  $v(\cdot, t)$  that satisfies  $\int_0^1 \|v(\cdot, t)\|_V^2 dt = \|v_0(\cdot)\|_V^2$ . Hence, defining  $\phi_t$  as the flow of diffeomorphisms corresponding to  $v(\cdot, t)$ , we can construct a map  $\exp : V \rightarrow G$  as  $\exp(v_0) := \phi_1$ . The exp map allows optimization over initial velocity vector fields and it has been used for, among others, images [79, 35, 36, 31, 80] and landmarks [28, 34]. While providing an in-depth treatment of the EPDiff equation is outside the scope of this work, for our discussion here, it is sufficient to know that the initial velocities constitute the tangent space at the identity diffeomorphism and that there exists an exponential map  $\exp(v_0)$  that maps an initial velocity vector field to a diffeomorphism. This situation is similar to PGA where an initial tangent vector is mapped to an element on the manifold via the exponential map. For more information on the EPDiff equation, we refer to Younes [30, 81].

Using the exp map, we replace the ODE in Equation (8) with  $\phi^i := \exp(v_{i,0})$  for some initial velocity vector field  $v_{i,0} \in V$ . In addition, using the properties of the exp map, we replace  $\int_0^1 \|v_i(\cdot, t)\|_V^2 dt$  by  $\|v_{i,0}\|_V^2$ . Finally, we replace



the hard constraint  $\phi_1^i \cdot \mathcal{T} = O_i$  by a soft penalty  $\mathcal{D}(\phi^i \cdot \mathcal{T}, O_i)$ . Combining these adjustments yields:

$$\begin{aligned} \min_{\{v_{i,0}\}_{i=1}^N, \mathcal{T}} \quad & \frac{1}{N} \sum_{i=1}^N [\mathcal{D}(\phi^i \cdot \mathcal{T}, O_i) + \sigma^2 \|v_{i,0}\|_V^2] \\ \text{s.t.} \quad & \phi^i = \exp(v_{i,0}), \end{aligned} \quad (9)$$

where  $\sigma \in \mathbb{R}$ . To optimize the initial velocity vector fields and the template, one creates discrete initial velocities  $v_{i,0}$  and a discrete version of the template. For instance, the template can be represented by a finite-resolution image or as a mesh.

Now that we know how to solve for the Fréchet mean in LDDMM PGA, we only need to find a geodesic submanifold that best explains the data. LDDMM PGA restricts the discretized initial velocities to a linear subspace via a, possibly orthonormal, matrix  $W$  and a diagonal matrix  $\Lambda$ . Putting this into Equation (9) yields the LDDMM PGA problem:

$$\begin{aligned} \min_{W, \Lambda, \{z_i\}_{i=1}^N, \mathcal{T}} \quad & \frac{1}{N} \sum_{i=1}^N [\mathcal{D}(\phi^i \cdot \mathcal{T}, O_i) + \sigma^2 \|v_{i,0}\|_V^2] \\ \text{s.t.} \quad & \phi^i = \exp(v_{i,0}), v_{i,0} = W\Lambda z_i. \end{aligned} \quad (10)$$

Additional loss components related to priors on  $W$ ,  $\Lambda$ , and the latent vectors can be added to the above optimization problem. These priors relate to the standard Bayesian derivation to obtain the LDDMM PGA model [35, 36]. However, these considerations are outside the scope of this work.

## 4 Riemannian Diffeomorphic Autoencoding via Implicit Neural Representations

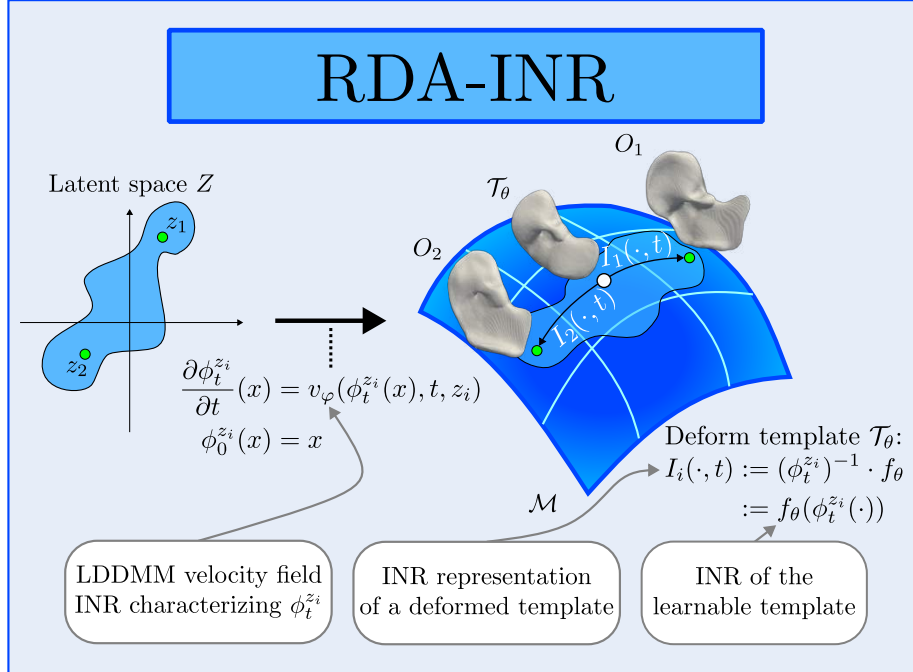


Figure 3: Overview of our RDA-INR framework for resolution independent and physically-consistent statistical latent modeling and atlas building. Our method maps a latent vector  $z_i$  to an INR representing a time-dependent velocity vector field  $v_\varphi(\cdot, t, z_i)$ . This velocity vector field defines a flow of diffeomorphisms  $\phi_t^{z_i}$  via an ODE. The diffeomorphisms  $\phi_1^{z_i}$  create (reconstructed) objects  $O_i$  on a subset of the Riemannian LDDMM manifold  $\mathcal{M}$  by deforming a learned template  $\mathcal{T}_\theta$ . We obtain this deformation by parameterizing  $\mathcal{T}_\theta$  with an INR  $f_\theta$  and deforming  $f_\theta$  via a group action:  $I_i(\cdot, t) := (\phi_t^{z_i})^{-1} \cdot f_\theta$ . As the template and the velocity vector fields are parameterized by INRs,  $I_i$  is a 4D INR deforming the template at  $t = 0$  to a reconstruction at  $t = 1$ . The goal is to learn the template  $\mathcal{T}_\theta$  as Fréchet mean of the data and to learn the  $I_i(\cdot, t)$  paths as geodesics on  $\mathcal{M}$  between the template  $\mathcal{T}_\theta$  and the data.

Our model depicted in Figure 3 can be obtained similarly to how we derived LDDMM PGA in Section 3.3. Specifically, we derive our model by starting at the LDDMM Fréchet mean problem [32, 82, 33, 34].

As shown in Section 3.3, if we want to find the Fréchet mean based on LDDMM, we should solve the problem in Equation (8). In LDDMM PGA, one first parameterizes the time-dependent diffeomorphisms via an initial velocity field, replaces the hard constraint with a soft penalty, and subsequently discretizes the template and the initial velocity vector fields. For our model, we skip the initial velocity field parameterization and immediately replace the hard constraint with a soft constraint using a general data-fitting term  $\mathcal{D}$  and a  $\sigma \in \mathbb{R}$ :

$$\begin{aligned} \min_{\{v_i(\cdot, t)\}_{i=1}^N, \mathcal{T}} \quad & \frac{1}{N} \sum_{i=1}^N \left[ \mathcal{D}(\phi_1^i \cdot \mathcal{T}, O_i) + \sigma^2 \int_0^1 \|v_i(\cdot, t)\|_V^2 dt \right] \\ \text{s.t.} \quad & \frac{\partial \phi_t^i}{\partial t}(x) = v_i(\phi_t^i(x), t), \quad \phi_0^i(x) = x. \end{aligned} \quad (11)$$

Here different data fidelity terms  $\mathcal{D}$  can be chosen depending on the data.

For LDDMM PGA, we required discretization of the template and initial velocity vector fields to solve the optimization problem in Equation (11). Furthermore, the discretization allows the restriction of the initial velocity vector fields to a linear subspace. Our approach circumvents the need for discretization by parameterizing the template and time-dependent velocity vector fields via INRs. Specifically, we parameterize  $\phi_1^i$  by specifying the corresponding flow in reverse time. Defining  $z_i \in \mathbb{R}^{d_z}$  as the latent code belonging to the object  $O_i$  and  $v_\varphi : \Omega \times [0, 1] \times \mathbb{R}^{d_z} \rightarrow \mathbb{R}^d$  as an implicit neural representation, the reverse time flow is given by  $\frac{\partial \phi_t^{z_i}}{\partial t}(x) = v_\varphi(\phi_t^{z_i}(x), t, z_i)$ . Jointly learning the weights  $\varphi$  and the latent codes  $z_i$  gives:

$$\begin{aligned} \min_{\varphi, \{z_i\}_{i=1}^N, \mathcal{T}} \quad & \frac{1}{N} \sum_{i=1}^N \left[ \mathcal{D}((\phi_1^{z_i})^{-1} \cdot \mathcal{T}, O_i) + \sigma^2 \int_0^1 \|v_\varphi(\cdot, t, z_i)\|_V^2 dt \right] \\ \text{s.t.} \quad & \frac{\partial \phi_t^{z_i}}{\partial t}(x) = v_\varphi(\phi_t^{z_i}(x), t, z_i), \quad \phi_0^{z_i}(x) = x. \end{aligned} \quad (12)$$

Finally, we represent the template  $\mathcal{T}$  by an implicit neural representation to obtain infinite-resolution reconstructions. Specifically, when dealing with images we represent  $\mathcal{T}$  as a neural network  $f_\theta : \Omega \rightarrow \mathbb{R}^n$  and for shapes we represent it with the zero-level set of a neural network  $f_\theta : \Omega \rightarrow \mathbb{R}$ . In both cases, the group action on the implicit representation becomes:

$$I(x, z, t) := ((\phi_t^z)^{-1} \cdot f_\theta)(x) := f_\theta(\phi_t^z(x)), \quad (13)$$

where  $I(x, z, 0)$  equals the implicit representation of the template and  $I(x, z, 1)$  is the reconstructed implicit representation of an object  $O$ . When dealing with shapes, we can obtain a mesh and point cloud for the reconstructed shape by applying marching cubes on  $I(x, z, 1)$ . Another strategy is to apply marching cubes to the template INR  $f_\theta$  to obtain a template mesh  $\mathcal{M}_\mathcal{T}$  and then retrieve the reconstructed mesh  $\mathcal{M}_z$  by  $(\phi_1^z)^{-1}(\mathcal{M}_\mathcal{T})$ .

Note that the group action in Equation (13) is a group action on the implicit representation. When dealing with shapes, the group action in Equation (13) can be associated to a group action on the template shape  $\mathcal{T}_\theta := \{x \mid f_\theta(x) = 0\}$ , namely  $\phi \cdot \mathcal{T}_\theta := \phi(\mathcal{T}_\theta)$ . In the case of images,  $\mathcal{T}_\theta = f_\theta$  and we can use the group action in Equation (13). Incorporating the template  $\mathcal{T}_\theta$  and the associated group action into Equation (12) yields the final optimization problem:

$$\begin{aligned} \min_{\theta, \varphi, \{z_i\}_{i=1}^N} \quad & \frac{1}{N} \sum_{i=1}^N \left[ \mathcal{D}((\phi_1^{z_i})^{-1} \cdot \mathcal{T}_\theta, O_i) + \sigma^2 \int_0^1 \|v_\varphi(\cdot, t, z_i)\|_V^2 dt \right] \\ \text{s.t.} \quad & \frac{\partial \phi_t^{z_i}}{\partial t}(x) = v_\varphi(\phi_t^{z_i}(x), t, z_i), \quad \phi_0^{z_i}(x) = x. \end{aligned} \quad (14)$$

Comparing Equation (14) with the LDDMM PGA optimization problem in Equation (10), we see several similarities and differences. In both problems, we optimize over latent vectors  $z_i$  and weights to define diffeomorphisms. However, in LDDMM PGA, the weights construct a linear relationship between the latent vectors and discrete initial velocity vector fields, while the weights in our model parameterize neural networks. This parameterization yields a nonlinear relationship between latent vectors and the time-dependent velocity vector fields. Moreover, the velocity vector fields and the template are not discretized in our model. Hence, our model still views the velocity vector fields and the template as elements of infinite-dimensional spaces, making the model resolution-independent. Finally, due to the nonlinear relationship between latent vectors and velocity vector fields, and the infinite-dimensional nature of the vector

fields, it is not possible to enforce the orthogonality properties of LDDMM PGA using an orthogonal weight matrix  $W$  as shown in Equation (10).

Up until now, we kept the data fidelity term  $\mathcal{D}$  general such that the optimization problem in Equation (14) applies to both images and shapes. In the case of images  $I_1$  and  $I_2$ ,  $\mathcal{D}(I_1, I_2) = \|I_1 - I_2\|_2^2$  is an appropriate data fidelity. When considering point cloud or mesh data without point correspondences, it is not immediately obvious what data fidelity  $\mathcal{D}$  to use in combination with our INR representation of the template. As LDDMM PGA is focused on image data, our numerical experiments focus on the novel point cloud and mesh data. We introduce the used data fidelity term for this data in Section 4.1. After introducing the data fidelity term, we introduce the chosen  $\|\cdot\|_V$  in Section 4.2. Subsequently, in Section 4.3, we introduce the neural network parameterization of  $v_\varphi$  and we treat how we numerically deform the template to obtain a reconstruction. Finally, we treat how we can obtain latent codes for unseen instances in Section 4.4.

#### 4.1 Data fidelity term for shape data

When dealing with shapes in the form of point cloud or mesh data, recent resolution-independent methods employ a deformable template for joint shape encoding and registration [39, 21] and use  $\mathcal{D}(I_1, I_2) = \|I_1 - I_2\|_{L_1(\Omega)}$  with  $I_i$  a ground truth signed distance representation of the point cloud or mesh. However, there are two issues with such a  $\mathcal{D}$ : (i) signed distance functions of diffeomorphic shapes are not always diffeomorphic, and (ii) if they are diffeomorphic, they influence the diffeomorphism that we find.

To illustrate the first issue, we take two circles of radii 0.75 and 0.1, respectively. Moreover, assume they are represented by their SDF, as presented in Equation (1). These SDFs are shown in Figure 4. We note that  $\min_{x \in \Omega} \text{SDF}_{\mathcal{S}_0}(x) < \min_{x \in \Omega} \text{SDF}_{\mathcal{S}_1}(x)$  as dark blue is present in the figure of the circle with radius 0.75 but not in the other figure. Consequently, there does not exist a diffeomorphism that matches the two implicit representations (i.e., SDF images). However, in Equation (14) we want to diffeomorphically match both to the same template implicit function. This matching is not possible as the two SDFs can not be diffeomorphically registered.

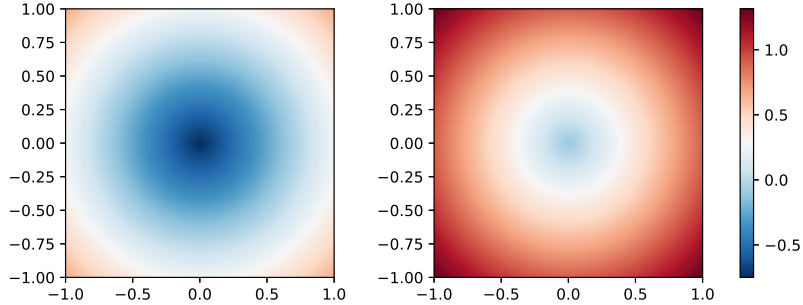


Figure 4: SDF values in  $[-1, 1]^2$  of a circle with radius 0.75 (left) and of a circle with radius 0.1 (right).

For the second issue, assume shapes  $\mathcal{S}_0$  and  $\mathcal{S}_1$  are represented by implicit functions that can be exactly matched. If a point  $x_0$  in shape  $\mathcal{S}_0$  has implicit function value  $s$ , it must be matched with a point  $x_1$  in shape  $\mathcal{S}_1$  that also has implicit function value  $s$ . By also matching points outside and inside the shape based on information like the SDF value, we influence how the points on the shape are matched.

To overcome these issues, we use the approach used by Sitzmann et al. [75], Deng et al. [38], and Gropp et al. [83] for learning an SDF from point cloud and mesh data. This approach uses the point cloud representation for training shapes  $\{\mathcal{S}_i\}_{i=1}^N$  and uses the following data fidelity term  $\mathcal{D}$ :

$$\mathcal{D}((\phi_1^z)^{-1} \cdot \mathcal{T}_\theta, \mathcal{S}_i) := \mathbb{E}_{x \in \mathcal{S}_i} [|I(x, z, 1)| + \tau (1 - F_{\cos}(\nabla_x I(x, z, 1), n_i(x)))] ,$$

with  $I(x, z, t)$  given in Equation (13),  $\tau \in \mathbb{R}$ ,  $n_i(x)$  the unit normal of the shape  $\mathcal{S}_i$  at  $x \in \mathcal{S}_i$ , and  $F_{\cos}$  the cosine similarity:

$$F_{\cos}(a, b) := \frac{\langle a, b \rangle_2}{\max(\|a\|_2, \|b\|_2, 10^{-8})}, \quad a, b \in \mathbb{R}^d .$$

This data-fitting term solves the previously mentioned issues since it only focuses on matching points on the training shapes with points on the template shape. In other words, all points on the training shapes are matched to the zero level set of the INR  $f_\theta$ . However, using only this loss function encourages  $f_\theta = 0$  as this ensures that all points are matched

to the zero level set of  $f_\theta$ . To avoid this trivial template shape, we constrain the learned template in optimization problem (14) via additional loss terms:

$$\begin{aligned}
 \min_{\theta, \varphi, \{z_i\}_{i=1}^N} \quad & \frac{1}{N} \sum_{i=1}^N \left[ \mathbb{E}_{x \in \mathcal{S}_i} [|I(x, z_i, 1)| + \tau (1 - F_{\cos}(\nabla_x I(x, z_i, 1), n_i(x)))] \right. \\
 & \left. + \beta \mathbb{E}_{x \in \Omega \setminus \mathcal{S}_i} [\exp(-\alpha |I(x, z_i, 1)|)] + \sigma^2 \int_0^1 \|v_\varphi(\cdot, t, z_i)\|_V^2 dt \right] \\
 & + \lambda \mathbb{E}_{x \in \Omega} [\|\nabla_x f_\theta(x)\|_2 - 1] \\
 \text{s.t.} \quad & \frac{\partial \phi_t^{z_i}}{\partial t}(x) = v_\varphi(\phi_t^{z_i}(x), t, z_i), \quad \phi_0^{z_i}(x) = x, \\
 & I(x, z, t) := ((\phi_t^z)^{-1} \cdot f_\theta)(x) := f_\theta(\phi_t^z(x)).
 \end{aligned} \tag{15}$$

Here the penalty corresponding to  $\lambda \in \mathbb{R}$  is an eikonal penalty, regularizing the template shape to be a signed distance function as in Equation (1) and ensuring that  $f_\theta \neq 0$ . Furthermore, the loss function involving the  $\beta$  parameter ensures that only points on  $\mathcal{S}_i$  are matched to the zero level set of  $f_\theta$ . Both terms are used by Sitzmann et al. [75], Deng et al. [38], and Gropp et al. [83] for learning an SDF from point cloud and mesh data.

*Remark 1.* Another approach to solve the issues is using the occupancy function in Equation (2) as the implicit representation for the data and the template shape. Earlier works using occupancy functions as implicit representations are Mescheder et al. [76] and Niemeyer et al. [68]. The reason occupancy functions solve the issues is that we only match points inside (outside) shape  $\mathcal{S}_0$  to points inside (outside) shape  $\mathcal{S}_1$  and do not take into account information like the SDF value. As the occupancy function and the occupancy values resemble probabilities, we can use the binary cross entropy loss as data fidelity term  $\mathcal{D}$  in optimization problem (14) [76, 68]. However, as shown in Appendix B, our strategy that uses point clouds as data representation allows for higher-quality reconstructions. Consequently, we use the point cloud data representation instead of the occupancy value data representation. Another reason for using point cloud data is related to the shape representation used by diffeomorphic registration methods. Specifically, these methods use either (i) meshes and point clouds or (ii) occupancy functions/segmentation masks. Our approach can deal with both data types as (i) our method deals with point cloud data via an image-based technique, and (ii) we can deal with segmentation masks via occupancy functions. We are showcasing the point cloud approach as, to the best of our knowledge, it is the first approach for dealing with point clouds using image-based representations.

## 4.2 Choice of velocity field regularization term

For the norm  $\|\cdot\|_V$  in optimization problems (14) and (15) we choose an isometric (rigid) deformation prior on the velocity vector fields by combining the Killing energy [84, 85] with an  $L_2(\Omega)$  penalty:

$$\|\nu\|_V^2 := \int_\Omega \|(J\nu) + (J\nu)^T\|_F^2 + \eta \|\nu\|_2^2 dx, \tag{16}$$

where  $\nu$  is a velocity vector field,  $J\nu$  the Jacobian of  $\nu$  with respect to  $x$ , and  $\eta \in \mathbb{R}$ . If  $\eta$  and the functional in Equation (16) are small, then we expect the template to deform almost isometrically to the reconstructed objects via  $I(x, z, t)$  in Equation (13). This property is interesting as it allows us to jointly perform affine registration and LDDMM registration. Moreover, as shown in Appendix C, under certain conditions the Killing energy can be viewed as an extension of a standard norm used in LDDMM.

## 4.3 Solving the ordinary differential equation

To solve optimization problems (14) and (15), we need to parameterize the velocity vector field  $v_\varphi$  and solve the resulting ordinary differential equation. Similarly to Gupta et al. [20] and Sun et al. [21], our model parameterizes  $v_\varphi$  as a quasi-time-varying velocity vector field. Concretely, using  $\chi_A$  as the indicator function of  $A$ , we define  $K$  neural networks  $v_{\varphi_k} : \Omega \times \mathbb{R}^{d_z} \rightarrow \mathbb{R}^d$  representing stationary velocity fields and define  $v_\varphi$  as

$$v_\varphi(x, t, z) = \sum_{k=1}^K \chi_{[\frac{k-1}{K}, \frac{k}{K})}(t) \cdot v_{\varphi_k}(x, z). \tag{17}$$

The main reason for this parameterization is that training time-varying velocity vector fields can be difficult as the model does not have training data for  $0 < t < 1$ . During training, we solve the ODE using an Euler discretization with  $K$  time steps. Consequently, similar to ResNet-LDDMM [1], we approximate the ODE with a ResNet architecture.

#### 4.4 Encoding objects

We use the strategy from DeepSDF [74] to encode (new) objects. More precisely, we solve the following optimization problem for encoding the object  $\mathcal{S}$ :

$$\min_z \mathcal{D}_{rec}((\phi_1^z)^{-1} \cdot \mathcal{T}_\theta, \mathcal{S}) + \gamma \|z\|_2^2, \quad (18)$$

where  $\gamma \in \mathbb{R}$ ,  $\phi_t^z$  is defined via the ODE in optimization problems (14) and (15),  $\mathcal{T}_\theta$  is the learned template, and  $\mathcal{D}_{rec}$  is some reconstruction data-fitting term. For instance, in case  $\mathcal{S}$  is represented by a mesh or a point cloud, we use  $\mathbb{E}_{x \in \mathcal{S}} [|I(x, z, 1)|]$  as  $\mathcal{D}_{rec}$  with  $I(x, z, t)$  given by Equation (13).

*Remark 2.* Our encoding procedure presented in Equation (18) is similar to the encoding strategy in PGA [41]. In PGA, the first step is to find the Fréchet mean  $\mu \in M$  of the data lying on some manifold  $M$ . Subsequently, PGA identifies a subspace  $W$  of the tangent space  $T_\mu M$  such that most variability in the data is described by  $H = \exp_\mu(W)$ , where  $\exp_\mu$  is the exponential map at  $\mu$ . Finally, to project a data point  $p \in M$  onto  $H$ , one calculates:

$$\pi_H(p) = \arg \min_{q \in H} d(p, q)^2,$$

where  $d$  is the Riemannian distance on the Riemannian manifold  $M$ . Alternatively, this can be reformulated into a minimization over the subspace  $W$ :

$$\arg \min_{w \in W} d(p, \exp_\mu(w))^2.$$

Equation (18) resembles this optimization problem. First, we have  $p = \mathcal{S}$ . Furthermore,  $(\phi_1^z)^{-1} \cdot \mathcal{T}_\theta$  should approximate  $\exp_\mu$  in case we consider the LDDMM manifold with  $\mu = \mathcal{T}_\theta$ . Although  $\mathcal{D}_{rec}$  is not a Riemannian distance, it replaces the Riemannian distance  $d$ , as also done in, e.g., Charlier et al. [47]. Finally, instead of searching over the vector space  $W$ , we search over a latent space that defines a point  $(\phi_1^z)^{-1} \cdot \mathcal{T}_\theta$  on the manifold. Hence, instead of finding an initial velocity  $w \in W$  to define a point  $q = \exp_\mu(w)$  on the manifold, we find a latent code  $z$  that determines a point on the manifold.

## 5 Numerical results

In this section, we demonstrate the benefit of the physical consistency and resolution independence of our model. To evaluate the benefit of physical consistency, we compare the Riemannian LDDMM regularizer to a non-Riemannian regularizer. We utilize two shape datasets for this comparison: a synthetic rectangles dataset and a liver dataset [21, 86]. Subsequently, we compare our method to DAE [16] on the liver dataset to illustrate the benefit of resolution independence for LDDMM statistical latent modeling.

### 5.1 Physical consistency and statistical shape modeling via latent spaces

To assess whether physical consistency is desirable for statistical shape modeling via latent spaces, we compare our model that regularizes the flow via the Riemannian LDDMM regularization (see Equation (15)) to our model utilizing the non-Riemannian pointwise regularization, which is presented in Appendix D.2. The latter model is the state-of-the-art baseline model by Sun et al. [21] with a different data-fitting term and a different approach to solving the ordinary differential equation.

We assess the mean-variance analysis capabilities of the models by visualizing the learned templates and deformations from the template to the training shapes. We relate our findings to Figure 2. Subsequently, we discuss reconstruction generalization and robustness of the reconstruction procedure to noisy data. This discussion shows the effect of Riemannian regularization on the quality and stability of the reconstruction procedure. We also relate this discussion to Figure 2. For details about the data source, the data preprocessing, and the training, we refer to Appendix D.

#### 5.1.1 Learning shape Fréchet means

For training the models with Riemannian LDDMM regularization, we need to pick a value for  $\eta$ . We choose  $\eta = 0.05$  for the rectangles dataset as we expect rigid body motions from the template to the training shapes. For the liver data, the only prior that we have is that the liver should not change too quickly. Hence, we pick  $\eta = 50$  such that  $\|\cdot\|_V \approx \eta \|\cdot\|_{L_2(\Omega)}$ .

After training, we obtain the template shapes in Figure 5. The templates obtained via the Riemannian regularization and the non-Riemannian regularization look almost identical for the liver dataset. The reason is that there is no clear deformation prior for the liver dataset besides that the liver should not change too quickly. The non-Riemannian

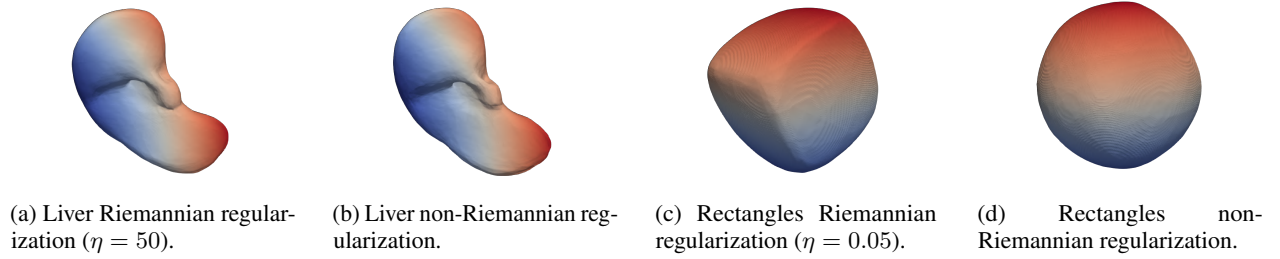


Figure 5: The learned templates of two different models trained on the rectangles dataset and the liver dataset. We use the model learned using the Riemannian LDDMM regularization ((a) and (c)) and the model learned using the non-Riemannian pointwise loss ((b) and (d)). We use  $\eta = 0.05$  and  $\eta = 50$  in Equation (16) for the rectangles and liver dataset, respectively.

regularization induces this prior by construction while the Riemannian regularization enforces this by our choice of  $\eta = 50$ . More precisely, as  $\|\cdot\|_V \approx \eta \|\cdot\|_{L_2(\Omega)}$  and as the pointwise loss can be interpreted as a non-Riemannian version of the LDDMM regularization with  $\|\cdot\|_V = \|\cdot\|_{L_2(\Omega)}$  (see Appendix D.2), both regularizations induce the same prior on the deformation. Hence, we expect similar templates in Figures 5a and 5b.

While the templates of the liver dataset look the same, the templates for the rectangles dataset differ. This difference between the rectangles and the liver dataset stems from the difference in the prior. While the liver shapes should only change gradually, the rectangles should also move rigidly. Consequently, we let the Killing energy play a more prominent role in  $\|\cdot\|_V$  in the Riemannian LDDMM regularization. This choice yields almost rigid deformations from the template to the training shapes, which results in the square template in Figure 5c. As the pointwise loss does not induce an isometry prior and is a non-Riemannian version of the LDDMM regularization with  $\|\cdot\|_V = \|\cdot\|_{L_2(\Omega)}$ , we obtain the spherical template in Figure 5d. This template is not desired as it lies outside the data distribution, as depicted in the third category in Figure 2.

In summary, while on the liver dataset, the templates look similar, the Riemannian regularization is required for the rectangles dataset to learn a template that fits the data distribution. Hence, the Riemannian LDDMM regularization allows for more flexibility in learning a template shape that resembles the data.

### 5.1.2 Learning geodesic shape deformations

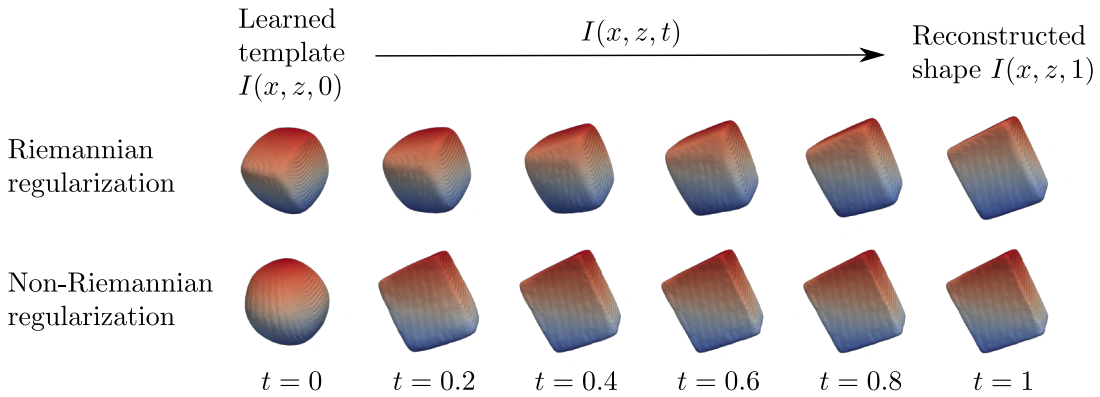


Figure 6: The deformation of the template into a reconstructed rectangle using the model learned using the Riemannian LDDMM regularization (top) and using the model learned using the non-Riemannian pointwise loss (bottom). The colors represent the matching of the points to their template.

In the previous section, we showed the effect of a regularizer on the learned Fréchet mean. To finalize the discussion on Figure 2, we assess the variance calculation by qualitatively assessing whether the template deformations are geodesics.

In Figure 6 the template is transformed into a reconstructed training shape for each model trained on the rectangles dataset. We notice that the non-Riemannian model quickly transitions from the spherical template to the target shape and then has a tendency to stay in a nearly identical configuration. In contrast, the model with the Riemannian LDDMM regularization smoothly rotates and scales the rectangular template shape to the final reconstructed shape. Hence, it fits

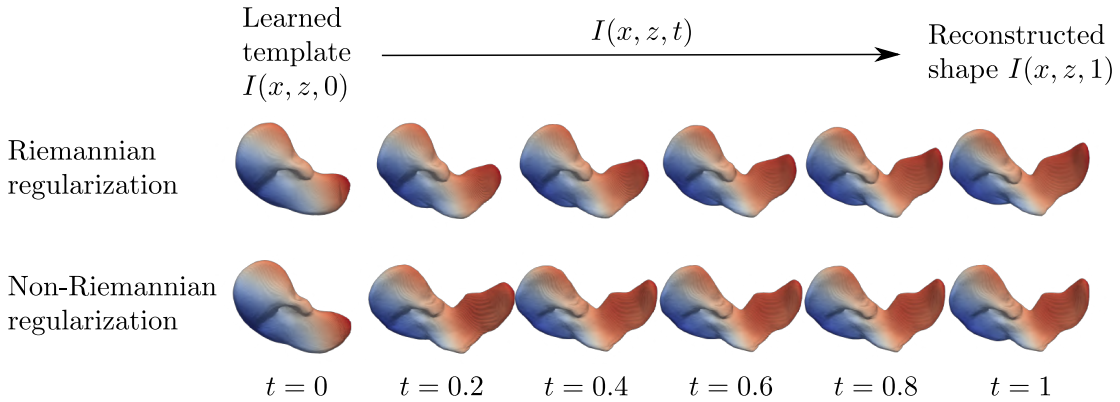


Figure 7: The deformation of the template into a reconstructed liver using the model learned using the Riemannian LDDMM regularization (top) and using the model learned using the non-Riemannian pointwise loss (bottom). The colors represent the matching of the points to their template.

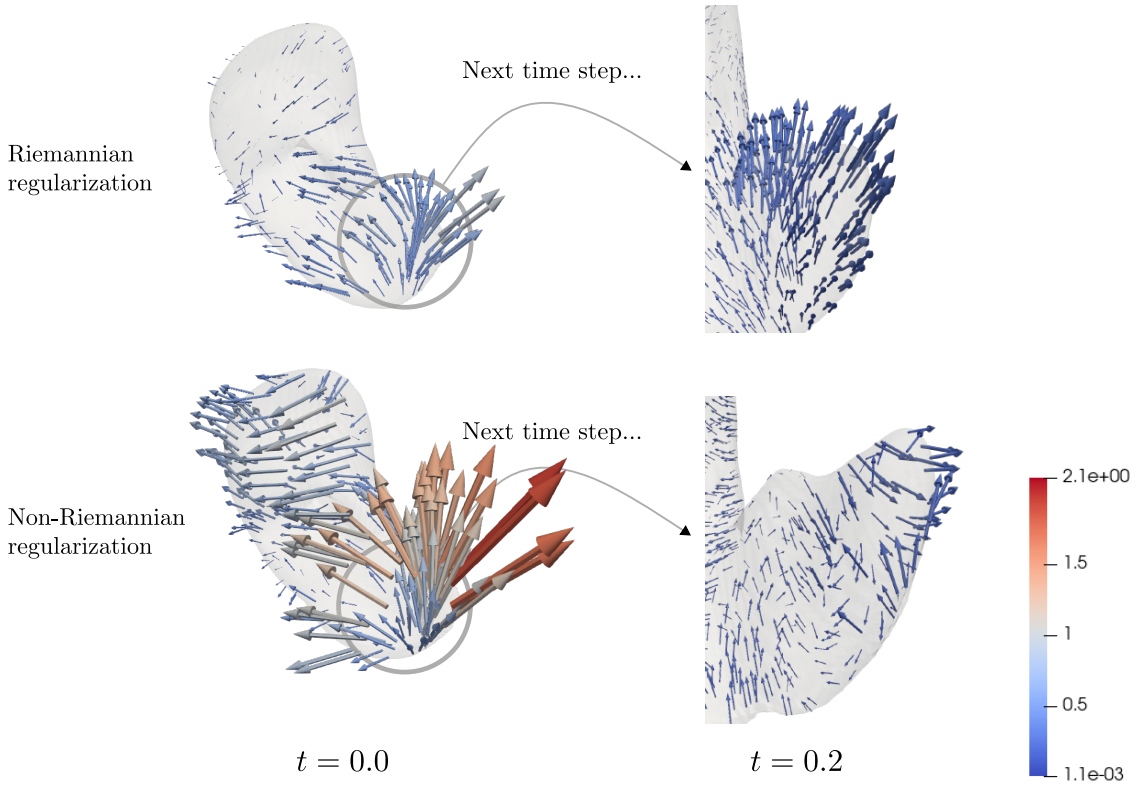


Figure 8: The velocity vector fields at  $t = 0$  and  $t = 0.2$  of the deformations in Figure 7. The colors represent the magnitude of the velocity vector field. At  $t = 0.2$ , only the vector field at the tip of the shape is shown.

the rigid body motion prior induced by the Killing energy. As the non-Riemannian pointwise loss does not penalize big abrupt deformations and does not use a rigid body motion prior, we obtain non-smooth and non-rigid deformations in this case.

Regarding the liver dataset, Figure 7 shows that the deformation with the Riemannian LDDMM regularization is smoother in time than the deformation with the non-Riemannian pointwise loss. Figure 8 reinforces this as it shows that the vector field of the non-Riemannian model at time  $t = 0.0$  is everywhere much larger than the vector field of

the Riemannian model. Moreover, at  $t = 0.2$ , the vector field of the Riemannian model points in approximately the same direction as at  $t = 0.0$ , while this does not apply to the non-Riemannian model. Hence, the vector field of the Riemannian model is smoother in time.

To explain these observations, we note that the non-Riemannian pointwise loss encourages a rapid transition to the target shape followed by a tendency to stay in a nearly identical configuration. Consequently, the velocity vector fields at later times  $t$  fine-tune the obtained reconstructions. Hence, the velocity vector field at  $t = 0.2$  does not necessarily point in approximately the same direction as the velocity vector field at  $t = 0.0$ , as shown in Figure 8. In contrast, with the Riemannian LDDMM regularization, rapid transitions are penalized, and it is preferred to gradually get closer to the target shape. Consequently, the Riemannian regularization allows for a smooth, physically plausible template deformation into the reconstructed shapes.

In Table 2, we summarize the discussion regarding Fréchet means and variance. The table shows that the Riemannian LDDMM regularization is more flexible in influencing the template than the non-Riemannian pointwise loss. Moreover, to obtain good variance estimates, the LDDMM regularization is needed to obtain geodesic deformations of the template to the reconstructions. Hence, the model with the Riemannian LDDMM regularization is the only model in category 1 of Figure 2 and, therefore, is the model with the best mean-variance analysis. The Riemannian model allows us to calculate Fréchet means of the data, to calculate physically-plausible geodesic deformations between the template shape and another shape, and to approximate the Riemannian distance between the template and a target shape. Hence, we have added Riemannian geometry to the latent space model.

Table 2: PGA performance. A summary of the results of Sections 5.1.1 and 5.1.2 regarding the models with Riemannian LDDMM regularization and non-Riemannian pointwise regularization. The results relate to Figure 2.

<i>Model</i>	<i>Dataset</i>	<i>Properties</i>	
		<i>Good Fréchet mean</i>	<i>Good variance</i>
<i>Non-Riemannian</i>	<i>Rectangles</i>	✗	✗
<i>Non-Riemannian</i>	<i>Liver</i>	✓	✗
<i>Riemannian</i>	<i>Rectangles</i>	✓	✓
<i>Riemannian</i>	<i>Liver</i>	✓	✓

### 5.1.3 Generalizability of shape encoding

This section evaluates the reconstruction quality of the two different models by reconstructing the test sets of the rectangles and liver dataset. We use the Chamfer Distance (CD) and the Earth Mover Distance (EM) as evaluation metrics. Table 3 shows the reconstruction metrics of the training data. We see that both models perform approximately equally on the training data.

Table 3: Training set evaluation. The model with the Riemannian LDDMM regularization and the model with the non-Riemannian pointwise regularization are evaluated by reconstructing the training sets of the rectangles (Rect.) and liver dataset. We calculate the average value and median (between brackets) of the Chamfer Distance (CD) and Earth Mover distance (EM). The Chamfer Distance values are of the order  $10^{-4}$ . The best (smallest) values are in bold.

<i>Model</i>	<i>Dataset (metric)</i>							
	<i>Rect. (CD)</i>		<i>Rect. (EM)</i>		<i>Liver (CD)</i>		<i>Liver (EM)</i>	
<i>Non-Riemannian</i>	<b>0.27</b>	<b>(0.26)</b>	<b>0.0187</b>	(0.0186)	<b>1.108</b>	<b>(0.92)</b>	0.0274	(0.0269)
<i>Riemannian</i>	0.33	(0.32)	0.0188	<b>(0.0185)</b>	1.12	(0.95)	<b>0.0273</b>	<b>(0.0265)</b>

Table 4 shows the reconstruction metrics on the test sets. We can immediately see that on the rectangle dataset, the model with Riemannian LDDMM regularization performs the best. On the liver dataset, the method with Riemannian regularization and the method with non-Riemannian pointwise regularization perform approximately equally.

The previous observations are related to the learned Fréchet mean. In the case of the rectangles dataset, we saw that the model with the Non-Riemannian pointwise loss did not learn a Fréchet mean resembling the data. As a consequence, the deformations of the spherical template do not end in proper rectangles. Hence, the fact that the model falls into category 3 in Figure 2 might cause generalization problems for the pointwise loss model on the rectangles dataset. This



Table 4: Test set evaluation. The model with the Riemannian LDDMM regularization and the model with the non-Riemannian pointwise regularization are evaluated by reconstructing the test sets of the rectangles (Rect.) and liver dataset. We calculate the average value and median (between brackets) of the Chamfer Distance (CD) and Earth Mover distance (EM). The Chamfer Distance values are of the order  $10^{-4}$ . The best (smallest) values are in bold.

<i>Model</i>	<i>Dataset (metric)</i>							
	<i>Rect. (CD)</i>		<i>Rect. (EM)</i>		<i>Liver (CD)</i>		<i>Liver (EM)</i>	
<i>Non-Riemannian</i>	8.20	(3.74)	0.0326	(0.0306)	<b>5.23</b>	(4.25)	<b>0.0369</b>	(0.0329)
<i>Riemannian</i>	<b>1.83</b>	( <b>1.43</b> )	<b>0.0245</b>	( <b>0.0245</b> )	5.31	( <b>3.68</b> )	0.0385	(0.0340)

view is strengthened by the results on the liver dataset where the pointwise loss model performs the best. In that case, a physically plausible Fréchet mean is learned and the deformations end at a liver shape again. To summarize, a good Fréchet mean improves generalization performance. Since we have already shown that physical consistency improves the Fréchet mean, physical consistency can also enhance generalization performance.

#### 5.1.4 Robustness to noise

In this section, we investigate how noise affects the reconstruction performance of the two models. We add random Gaussian noise with mean zero and a standard deviation of 0.01 or 0.02 to the vertices of the meshes in the test set. Subsequently, we reconstruct these noisy meshes and compare the reconstructions to the noiseless ground truth meshes. Figure 9 provides some of these reconstructions. In general, both models are effective in denoising the illustrated input meshes. On the rectangles, the Riemannian regularization produces good, noise-resistant reconstructions. The model with the non-Riemannian regularization generates worse reconstructions and is affected by the noise, as illustrated by the reconstructions, which gradually lose their rectangular shape when the noise increases. On the liver dataset, both models produce satisfactory denoised reconstructions. However, while the model with Riemannian regularization yields reconstructions that are approximately independent of the noise level, the model with non-Riemannian regularization results in visibly different reconstructions when the noise level varies.

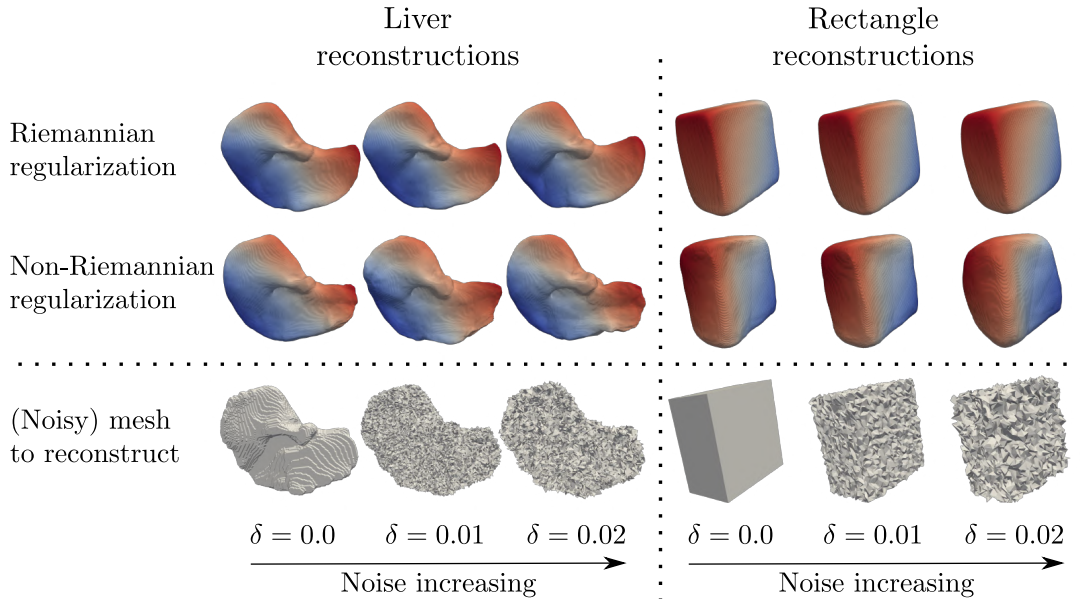


Figure 9: The reconstruction of two test shapes with added zero-mean Gaussian noise for different standard deviations  $\delta$ . The red and blue colors represent the matching of the points to the learned template.

To assess whether the observations concerning Figure 9 extend to other liver and rectangle shapes, Table 5 presents the average and median reconstruction errors between the reconstructed shapes and the ground truth noiseless shapes. Comparing the values to Table 4, we notice that on the rectangles dataset, the model with Riemannian LDDMM regularization is much less sensitive to the noise than the model with the non-Riemannian regularization. On the liver

Table 5: Noisy test set evaluation. The model with the Riemannian LDDMM regularization and the model with the non-Riemannian pointwise regularization are evaluated by reconstructing noisy test sets ( $\delta = 0.01$  and  $\delta = 0.02$ ) of the rectangles (Rect.) and liver dataset. We calculate the average value and median (between brackets) of the Chamfer Distance (CD) and Earth Mover distance (EM). The values are obtained by comparing the noiseless shapes to the reconstructions of their noisy versions. The Chamfer Distance values are of the order  $10^{-4}$ . The two best (smallest) values are in bold.

<i>Model</i>	$\delta$	<i>Dataset (metric)</i>							
		<i>Rect. (CD)</i>		<i>Rect. (EM)</i>		<i>Liver (CD)</i>		<i>Liver (EM)</i>	
<i>Non-Riemannian</i>	0.01	39.37	(4.48)	0.0409	(0.0310)	6.57	<b>(3.97)</b>	<b>0.0405</b>	(0.0369)
<i>Non-Riemannian</i>	0.02	54.82	(8.17)	0.0502	(0.0367)	10.49	(5.48)	0.0451	(0.0387)
<i>Riemannian</i>	0.01	<b>2.01</b>	<b>(1.72)</b>	<b>0.0242</b>	<b>(0.0252)</b>	<b>5.95</b>	<b>(3.82)</b>	<b>0.0390</b>	<b>(0.0322)</b>
<i>Riemannian</i>	0.02	<b>4.06</b>	<b>(3.43)</b>	<b>0.0275</b>	<b>(0.0276)</b>	<b>6.53</b>	(4.89)	0.0405	<b>(0.0365)</b>

dataset, the median values are relatively stable for both models. Moreover, the average reconstruction errors of both models are not influenced much by a noise level of 0.01. However, when increasing the standard deviation to 0.02, the non-Riemannian model attains a much bigger average reconstruction error. In contrast, the Riemannian model attains a similar average error to the scenario with a 0.01 noise level.

The latter observation can be related to Figure 9 where the liver reconstructions obtained via the non-Riemannian model are dependent on the noise level. As discussed in Section 5.1.2, the non-Riemannian regularization encourages a rapid transition to the target shape followed by smaller deformations for fine-tuning. Such large displacements can cause sensitivity to small changes and might yield worse or different reconstructions. Disallowing such large displacements by applying the Riemannian regularization to the velocity vector fields stabilizes the problem.

Finally, we again notice a difference between the rectangle and liver dataset, which again stems from the learned Fréchet mean. In the case of the rectangles dataset, the pointwise loss model does not learn a physically plausible Fréchet mean of the data and the end of the template deformations do not correspond to test rectangles. In contrast, on the liver dataset a physically plausible template is learned and the end of the template deformations do correspond to livers. These observations strengthen our finding, as discussed in Section 5.1.3, that a physically plausible Fréchet mean, and hence LDDMM physical consistency, enhances generalization performance.

## 5.2 INRs for LDDMM statistical latent modeling

To illustrate the benefit of INRs for topology-preserving latent modeling, we use the liver dataset and compare our method to DAE [16]. We only compare to this method as this is the only neural network-based LDDMM latent model that is designed to work for meshes.

First, we train on our original mesh data containing many vertices but without a smooth surface. Subsequently, we do the same experiment for downsampled meshes with a much smaller amount of vertices: one dataset with non-smooth meshes and one dataset with smoother meshes. The three mentioned datasets are depicted in Figure 10. To obtain the downsampled non-smooth meshes, we take the original mesh data, calculate a  $64 \times 64 \times 64$  occupancy grid for the meshes, and subsequently perform marching cubes [73] on the occupancy grid. To obtain a smooth version of the meshes, we also calculate a signed distance grid and apply marching cubes to it.

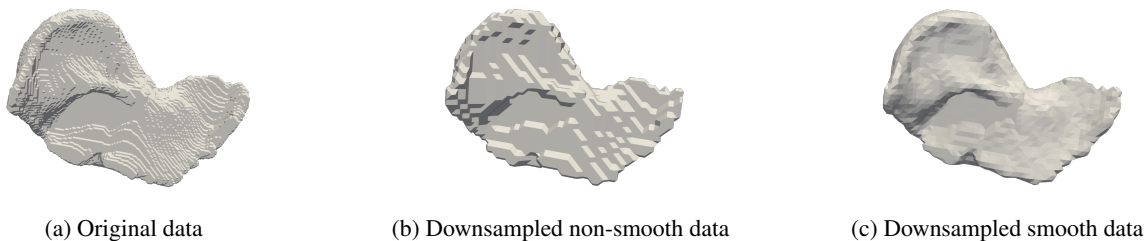


Figure 10: From left to right: the original mesh data, the downsampled version, and the downsampled smooth version.

We train the DAE model only on the downsampled data as the model failed to converge to a proper solution on the original data. On the other hand, we only train our model on the original data and the downsampled non-smooth data. The reason is that this is noisier data and we shall see that our algorithm is agnostic to such noise.

First, we evaluate the learned templates trained on the downsampled data. These learned templates can be found in Figure 11. The template learned by our model resembles the template that is learned using the original data, which is depicted in Figure 5a. Hence, even with less detailed meshes we can learn a high-resolution template. Furthermore, while the DAE template found using the non-smooth data looks non-smooth, the template found with the smooth data looks quite smooth. The latter looks similar to the liver template found with the RDA-INR model. However, the template found by DAE is less smooth than the template found by our model. The main reason for this is the smooth structure of the INR parameterizing the Fréchet mean. In DAE the Fréchet mean is calculated by Deformetrica [87]. For Deformetrica we take a single sample from the dataset as initialization of the Fréchet mean. Hence, we inherit the structure of this initialization, possibly causing a problem with the template.

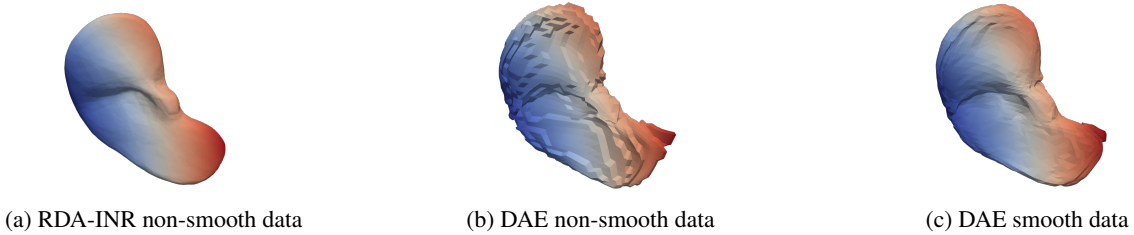


Figure 11: The learned templates of RDA-INR and DAE on the downsampled datasets in Figure 10.

The effect of the templates on the reconstructions of test samples is shown in Figure 12. We see that the reconstructions are heavily affected by the quality of the templates. On the other hand, our model does not seem affected by the resolution of the dataset as the reconstructions of the model trained on the original data and the downsampled non-smooth data are very similar. To strengthen these observations, we quantitatively assess reconstruction performance. We compare the reconstructed meshes to the ground-truth mesh with many vertices via the Chamfer Distance and the Earth Mover Distance. These metrics are calculated between the reconstructed mesh and many points sampled from the original mesh. The results are presented in Table 6. We can see that our model outperforms the DAE method and that our model does not deteriorate much when using the downsampled data.

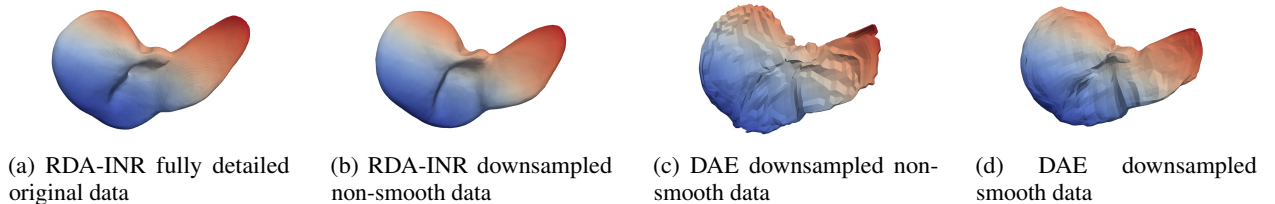


Figure 12: Reconstructions of a specific test sample using models trained on different variants of the original liver data.

Table 6: Train and test evaluation of RDA-INR and DAE. The table presents the average values and medians (between brackets) of the Chamfer Distance (CD) and Earth Mover distance (EM). The values are obtained by comparing the original data to the reconstructions obtained using the dataset the model is trained on. These values are calculated for the shapes in the train and test splits of the liver dataset. The Chamfer Distance values are of the order  $10^{-4}$ . The best (smallest) values for the downsampled data are in bold.

<i>Model (Dataset)</i>		<i>Data split (metric)</i>							
		<i>Train (CD)</i>		<i>Train (EM)</i>		<i>Test (CD)</i>		<i>Test (EM)</i>	
<i>RDA-INR</i>	<i>(Original data)</i>	1.12	(0.95)	0.0273	(0.0265)	5.31	(3.68)	0.0385	(0.0340)
<i>DAE</i>	<i>(Non-smooth)</i>	4.47	(4.01)	0.0350	(0.0339)	13.52	(10.22)	0.0479	(0.0443)
<i>DAE</i>	<i>(Smooth)</i>	4.12	(3.54)	0.0344	(0.0330)	14.46	(11.10)	0.0492	(0.0464)
<i>RDA-INR</i>	<i>(Non-smooth)</i>	<b>1.61</b>	<b>(1.45)</b>	<b>0.0288</b>	<b>(0.0282)</b>	<b>6.34</b>	<b>(4.17)</b>	<b>0.0402</b>	<b>(0.0351)</b>

## 6 Conclusion and future work

Several works recently developed neural network models regarding diffeomorphic registration, such as models for pairwise and groupwise registration, atlas building, and data variability modeling. We presented an overview of the current literature on neural networks for diffeomorphic registration and categorized the works depending on several features. This categorization highlights several research gaps. For instance, the LDDMM-based statistical latent modeling approaches are not resolution independent. Moreover, resolution-independent neural network algorithms for joint shape encoding and groupwise registration do not use the Riemannian geometry of shape space, which is a crucial component of LDDMM PGA. Consequently, these latent space models do not provide insights about shape Fréchet means, geodesics, Riemannian distances between shapes, and data variance.

We have addressed the aforementioned two limitations to highlight the importance of resolution independence and LDDMM physical consistency. Specifically, we presented an INR-based latent model inspired by LDDMM PGA. This resolution-independent model is a deformable template model that solves the Fréchet mean finding problem in LDDMM-based PGA. The main ingredient in this model is the Riemannian regularization on the neural network that deforms the template. Although our model also applies to images, we mainly focus on the case of point cloud and mesh data.

First, we discussed the importance of LDDMM physical consistency by comparing our model to the model with a non-Riemannian pointwise regularization on the deformation. We show that the Riemannian regularization is necessary for the model to perform a proper mean-variance analysis. In other words, our model allows calculating a Fréchet mean of the data, obtaining geodesics and approximating the distance between the template and another object, and estimating the data variance. Furthermore, we demonstrate that the Riemannian regularization can improve the reconstruction of an object and the stability of the reconstruction procedure. In other words, the Riemannian regularization induces a prior that enables us to find more stable factors of variation. Finally, we assess the importance of resolution-independence by comparing our model to another neural network model inspired by LDDMM PGA. We demonstrated improved template learning and improved reconstructions.

In summary, we show how shape and image analysis, Riemannian geometry, and deep learning can be connected. This connection paves the way to more research into how these different disciplines can reinforce each other.

### 6.1 Future work

An example of such research for our deformable template model is related to our model’s template deformations. Our learned template deformations constitute geodesics between the template and the reconstructed objects. This makes it possible to obtain template deformations that fit a prior and to reconstruct objects using the end point of such a deformation. However, for some objects in the geodesic, there might not exist a latent code and corresponding reconstruction. As these objects fit a modeling prior, we would like them to correspond to a latent code. To achieve this objective, we might employ the velocity vector field parameterization from Lüdke et al. [19]. In addition, even though our model can estimate the distance between the Fréchet mean and the training data via the aforementioned geodesics, we do not use this for generative modeling. In other words, we do not use the distance calculation for defining a Riemannian distribution on the latent space.

Moreover, we focus on the latent modeling capabilities of our model and not the registration capabilities. Hence, it might be interesting to compare our model to neural network models for pairwise and groupwise registration. As these methods often use the more expressive resolution-dependent U-NET architectures, we hypothesize that these methods would still outperform our latent model. We believe it would be interesting to research resolution-independent neural operator variants of our model that more resemble the U-NET architectures. Finally, our model is not only applicable to shapes but also to images. In future work, we would like to assess our model on image data as well.

### **Conflict of interest statement**

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

### **Data availability statement**

The data that support the findings of this study are currently only available upon reasonable request from the authors. In the appendices, we discuss the datasets used in the manuscript. Instructions on how to obtain the datasets will be provided in a GitHub repository that will be made publicly available in the future. This GitHub repository also contains the code used to perform the numerical experiments.

# Appendices

## Appendix A Riemannian geometry

In this section, we briefly present a high-level overview of some key concepts of differential geometry and Riemannian geometry. For more in-depth information, we refer to [88].

Manifolds are the main object in Riemannian geometry and differential geometry. Intuitively, a  $d$ -dimensional manifold  $M$  is a set that locally looks like  $\mathbb{R}^d$ . An example of a manifold is the sphere. Moreover, a submanifold  $N$  of  $M$  is a subset of  $M$  that is also a manifold.

To define differentiability on manifolds and submanifolds, we need a differentiable manifold. An important notion for differentiable manifolds is the tangent space  $T_p M$  at a point  $p \in M$ , which can be thought of as the tangent plane to the manifold at  $p$ . More formally, the tangent space can be defined as:

**Definition 2** (Tangent space). Assume we have a smooth curve  $\gamma : \mathbb{R} \rightarrow M$  with  $\gamma(0) = p$ . Define the directional derivative operator at  $p$  along  $\gamma$  as

$$\begin{aligned} X_{\gamma,p} : C^\infty(M) &\rightarrow \mathbb{R} \\ f &\rightarrow (f \circ \gamma)'(0), \end{aligned}$$

where  $C^\infty(M)$  denotes the set of smooth scalar functions on  $M$ . Then the tangent space  $T_p M$  is defined as  $T_p M := \{X_{\gamma,p} \mid \gamma(0) = p, \gamma \text{ smooth}\}$ .

These tangent spaces can be used to define shortest paths on manifolds. For defining shortest paths, we need a Riemannian manifold:

**Definition 3** (Riemannian manifold). Let  $M$  be a differentiable manifold. Define a Riemannian metric  $g$  as a smoothly varying metric tensor field. In other words, for each  $p \in M$ , we have an inner product  $g_p : T_p M \times T_p M \rightarrow \mathbb{R}$  on the tangent space  $T_p M$ . The pair  $(M, g)$  is called a Riemannian manifold.

We note that submanifolds  $N$  inherit the differential structure and the Riemannian metric structure of  $M$ . Using the metric structure, we define shortest paths between  $p$  and  $q$  as minimizers of:

$$\begin{aligned} d_M(p, q) &= \min_{\gamma} \int_0^1 \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))} dt \\ \text{s.t. } &\gamma(0) = p, \gamma(1) = q, \end{aligned} \tag{A.1}$$

where  $\dot{\gamma}(t)$  is the tangent vector at  $\gamma(t)$  generated by  $\gamma$ . In this case, Equation (A.1) defines a Riemannian distance on  $M$  and the minimizer  $\gamma^*$  is called a (Riemannian) geodesic.

*Remark 3.* In case  $M = \mathbb{R}^d$  and  $g_p(a, b) = \langle a, b \rangle_2$ , we have  $d_M(p, q) = \|p - q\|_2$  and  $\gamma^*(t) = p + t(q - p)$ .

Given a geodesically complete manifold, for any point  $p \in M$  and tangent vector  $\dot{p} \in T_p M$ , there exists a unique geodesic  $\gamma$  with  $\gamma(0) = p$  and  $\dot{\gamma}(0) = \dot{p}$ . The unique solution at  $t = 1$  is given by the exponential map  $\exp_p(\dot{p}) := \gamma(1)$ . The exponential map allows mapping tangent vectors in  $T_p M$  to points on the manifold. Consequently, we can execute several manifold operations on a tangent space instead of on the manifold. For instance, the exponential map is used in PGA to obtain a manifold version of PCA.

Finally, the Riemannian distance allows us to define manifold extensions of means in vector spaces and allows us to define a specific type of submanifold:

**Definition 4** (Fréchet mean). Let  $\rho$  be a probability distribution on  $M$ . The Fréchet mean  $\mu$  is defined as

$$\mu = \arg \min_{q \in M} \int_M d_M^2(p, q) d\rho(p).$$

If we only have a finite sample  $\{p_i\}_{i=1}^N$  with  $p_i \in M$ , the Fréchet mean  $\mu$  is defined as

$$\mu = \arg \min_{q \in M} \frac{1}{N} \sum_{i=1}^N d_M^2(p_i, q).$$

**Definition 5** (Geodesic submanifold). A geodesic submanifold of a Riemannian manifold  $M$  is a submanifold  $N$  such that  $\forall p \in N$ , all geodesics of  $N$  passing through  $p$  are also geodesics of  $M$ .

## Appendix B Occupancy data versus point cloud data

In the main text, we discuss two data fidelity terms that can be used in our implicit encoding model. One approach uses occupancy functions with a binary cross entropy data fidelity term, while the other approach uses point cloud data. To showcase the difference between the two approaches, we perform an experiment on the rectangle data as used in the numerical results section. The exact same training parameters are used.

In Figures B.1 and B.2, we see the learned templates and an example of a reconstructed training shape, respectively.



Figure B.1: The learned template of the model learned using occupancy functions (left) and the model learned using the point cloud data (right).



Figure B.2: The reconstructions of a particular training shape when using the model learned using occupancy functions (left) and the model learned using the point cloud data (right).

Figure B.1 shows that both templates resemble a square. Figure B.2 demonstrates that the reconstructions with the occupancy data are worse than the reconstructions with the point cloud data. One possible explanation is that the point cloud loss function encourages the shape's points to lie on the zero level set of the implicit representation. When using occupancy values, we focus more on the domain around the shape and train on uniformly sampled occupancy values to regress the occupancy function. Hence, when working with point clouds, the emphasis is on the shape itself, rather than the surrounding domain, which is the case with occupancy functions. The focus on the shape itself makes it possible to better reconstruct its details. As the point cloud method yields better reconstructions, we use this method for the numerical experiments.

## Appendix C Killing energy as standard LDDMM norm

In the LDDMM literature, a commonly used  $\|\cdot\|_V$  is the norm induced by  $\langle \nu, \omega \rangle_V := \langle L\nu, \omega \rangle$  with  $L : V \rightarrow V^*$ ,  $\langle \cdot, \cdot \rangle$  the canonical duality pairing, and  $\nu, \omega \in V$ . In many papers,  $L = (\text{id} - \alpha\Delta)^c$  for some  $\alpha \in \mathbb{R}$  and  $c \in \mathbb{N}$ , and  $\langle L\nu, \omega \rangle := \langle L\nu, \omega \rangle_{L^2(\Omega)}$ . In this section, we show that under certain conditions the Killing energy can be interpreted as an extension of  $\|\nu\|_V^2 = \langle L\nu, \nu \rangle_{L^2(\Omega)}$  with  $L = (\text{id} - \alpha\Delta)$ :

**Theorem 3.** *Assume we use the following norm:*

$$\|\nu\|_V^2 = \int_{\Omega} \frac{1}{2} \|J\nu + (J\nu)^T\|_F^2 + \eta \|\nu\|_2^2 dx, \quad (\text{C.1})$$

where  $\Omega \subset \mathbb{R}^d$  is a bounded domain. Moreover, we assume that  $\nu \in V$  is sufficiently many times differentiable and that:

$$\int_{\partial\Omega} \langle \nu, (J\nu + (J\nu)^T)n \rangle_{l_2} dx = 0, \quad (\text{C.2})$$

where  $n$  is the normal to the boundary. Both assumptions are, for instance, satisfied when  $V \subset C_0^2(\Omega, \mathbb{R}^d)$  where  $C_0^2(\Omega, \mathbb{R}^d)$  is the space of twice continuously differentiable vector fields  $\nu$  on the open and bounded domain  $\Omega \subset \mathbb{R}^d$  such that both  $\nu$  and its Jacobian  $J\nu$  vanish on  $\partial\Omega$  and at infinity.

Given the assumptions, we have:

$$\|\nu\|_V^2 = \int_{\Omega} \langle (\eta \text{id} - \Delta - \nabla \cdot \nabla^T)\nu, \nu \rangle_{l_2} dx = \int_{\Omega} \langle \tilde{L}\nu, \nu \rangle_{l_2} dx,$$

where  $\tilde{L} := \eta \text{id} - \Delta - \nabla \cdot \nabla^T$  and  $(\nabla \cdot \nabla^T)\nu = \nabla \cdot (J\nu)^T$  with the divergence taken row-wise.

*Proof.* First, define  $\nu_i$  as the  $i$ -th component of the vector field  $\nu$ . As the Frobenius norm comes from an inner product and  $\|A\|_F^2 = \|A^T\|_F^2$ , we obtain

$$\begin{aligned} \frac{1}{2} \|J\nu + (J\nu)^T\|_F^2 &= \|J\nu\|_F^2 + \langle J\nu, (J\nu)^T \rangle_F \\ &= \sum_{i=1}^d \langle \nabla \nu_i, \nabla \nu_i \rangle_{l_2} + \left\langle \nabla \nu_i, \frac{\partial}{\partial x_i} \nu \right\rangle_{l_2} \\ &= \sum_{i=1}^d \left\langle \nabla \nu_i, \nabla \nu_i + \frac{\partial}{\partial x_i} \nu \right\rangle_{l_2}. \end{aligned}$$

Using this identity, we obtain:

$$\int_{\Omega} \frac{1}{2} \|J\nu + (J\nu)^T\|_F^2 dx = \int_{\Omega} \sum_{i=1}^d \left\langle \nabla \nu_i, \nabla \nu_i + \frac{\partial}{\partial x_i} \nu \right\rangle_{l_2} dx = \sum_{i=1}^d \int_{\Omega} \left\langle \nabla \nu_i, \nabla \nu_i + \frac{\partial}{\partial x_i} \nu \right\rangle_{l_2} dx.$$

Subsequently, using  $\nabla \cdot (\nu_i \nabla \nu_i) = \langle \nabla \nu_i, \nabla \nu_i \rangle_{l_2} + \nu_i \Delta \nu_i$  and  $\nabla \cdot (\nu_i \frac{\partial}{\partial x_i} \nu) = \left\langle \nabla \nu_i, \frac{\partial}{\partial x_i} \nu \right\rangle_{l_2} + \nu_i \nabla \cdot \left( \frac{\partial}{\partial x_i} \nu \right)$ , we get:

$$\int_{\Omega} \frac{1}{2} \|J\nu + (J\nu)^T\|_F^2 dx = \sum_{i=1}^d \int_{\Omega} \nabla \cdot \left( \nu_i \left( \nabla \nu_i + \frac{\partial}{\partial x_i} \nu \right) \right) - \nu_i \left( \Delta \nu_i + \nabla \cdot \left( \frac{\partial}{\partial x_i} \nu \right) \right) dx.$$

Applying the divergence theorem yields:

$$\int_{\Omega} \frac{1}{2} \|J\nu + (J\nu)^T\|_F^2 dx = \sum_{i=1}^d \left( \int_{\partial\Omega} \left\langle \nu_i \left( \nabla \nu_i + \frac{\partial}{\partial x_i} \nu \right), n \right\rangle_{l_2} dx - \int_{\Omega} \nu_i \left( \Delta \nu_i + \nabla \cdot \left( \frac{\partial}{\partial x_i} \nu \right) \right) dx \right).$$

Doing some rewriting yields:

$$\int_{\Omega} \frac{1}{2} \|J\nu + (J\nu)^T\|_F^2 dx = \int_{\partial\Omega} \langle \nu, (J\nu + (J\nu)^T)n \rangle_{l_2} dx - \int_{\Omega} \langle \nu, (\Delta + \nabla \cdot \nabla^T)\nu \rangle_{l_2} dx.$$

Finally, using our assumption in Equation (C.2) gives:

$$\int_{\Omega} \frac{1}{2} \|J\nu + (J\nu)^T\|_F^2 dx = - \int_{\Omega} \langle (\Delta + \nabla \cdot \nabla^T)\nu, \nu \rangle_{l_2} dx.$$



Using the above in combination with Equation (C.1), we get:

$$\begin{aligned}
 \|\nu\|_{\tilde{V}}^2 &= \int_{\Omega} \frac{1}{2} \|J\nu + (J\nu)^T\|_F^2 + \eta \|\nu\|_2^2 dx \\
 &= \int_{\Omega} \langle (\eta \text{id} - \Delta - \nabla \cdot \nabla^T) \nu, \nu \rangle_{l^2} dx \\
 &= \int_{\Omega} \langle \tilde{L}\nu, \nu \rangle_{l^2} dx.
 \end{aligned}$$

□

## Appendix D Implementation details

### D.1 Neural network architectures

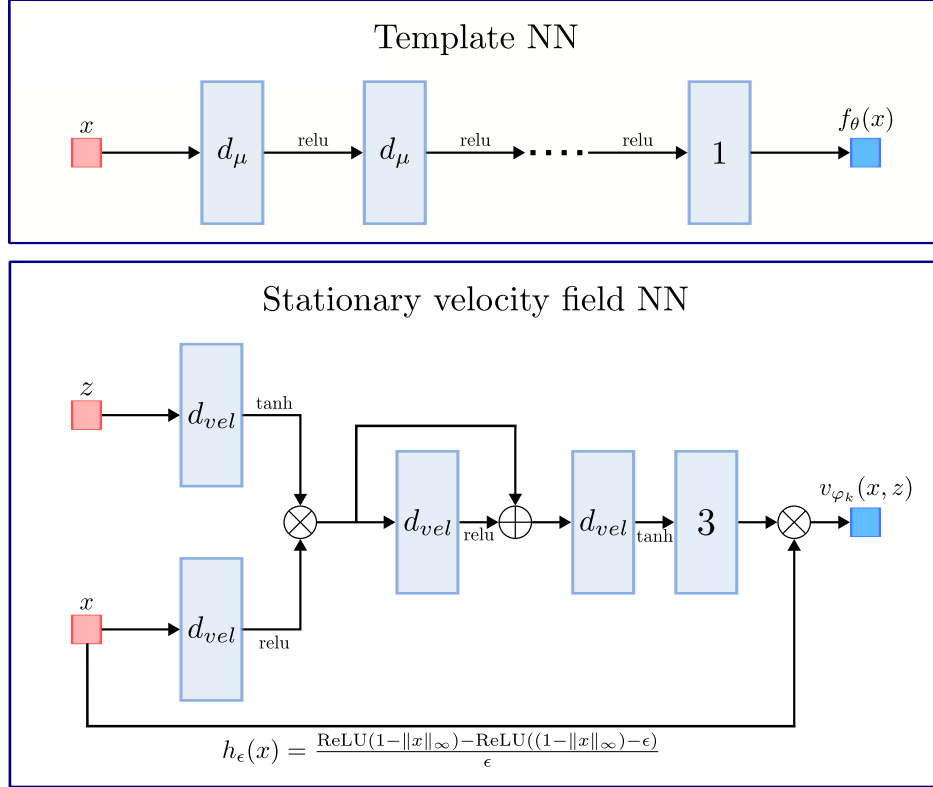


Figure D.1: The architectures for the template neural network  $f_\theta$  and the stationary velocity vector fields  $v_{\varphi_k}$  in Equation (17). The small red boxes correspond to the latent code input  $z$  and the spatial input  $x$ . Furthermore, the small blue box corresponds to an output, while the rectangles are linear layers with  $d_{vel}$  or  $d_\mu$  output dimensions. Finally, the  $\oplus$  and  $\otimes$  stand for elementwise addition and scalar/elementwise multiplication, respectively.

Figure D.1 shows the used neural network architectures. For the template neural network, we use nearly the same architecture as the template neural network in [38]. The only difference is that we use ReLU activation functions instead of sine activation functions. In our experiments, we clamp the output of the template neural network to the interval  $[-0.5, 0.5]$ . The stationary velocity vector field neural networks  $v_{\varphi_k}$  in Equation (17) use the architecture in [21]. However, there are some differences. First, we add an extra linear layer at the output. We also add a scalar multiplication with the function  $h_\epsilon$ . This component ensures the velocity vector field is zero outside  $\Omega$ , which is needed to let the ODE be a diffeomorphism on  $\Omega$ .

### D.2 Pointwise loss

The models by [39] and [21] are similar to our model. However, in contrast to our work, they do not use a Riemannian distance to regularize the time-dependent deformation of the template. Hence, we can not expect a physically plausible deformation that constitutes a geodesic. They use the pointwise regularization given by:

$$L_{pw} = \sum_{t \in T} \sum_{i=1}^N \sum_{j=1}^M \mathcal{L}_{0.25} (\|\phi_i(p_{ij}, t) - p_j\|_2),$$

where  $\mathcal{L}_{0.25}$  is the Huber loss with parameter equal to 0.25,  $T$  is a set of predefined time instances at which to evaluate the pointwise loss,  $\{p_{ij}\}_{j=1}^M$  are  $M$  points from the domain, and  $\phi_i(\cdot, t) := \phi_i^{z_i}(\cdot)$  with  $\frac{\partial}{\partial t} \phi_t^z(x) = v_\varphi(\phi_t^z(x), t, z)$  and  $\phi_0^z(x) = x$ . For our purposes, we follow [39] and [21] and choose  $T = \{\lfloor K/4 \rfloor \cdot i \mid 1 \leq i \leq K/\lfloor 0.25K \rfloor, i \in \mathbb{N}\}$  with  $K$  the number of stationary velocity vector fields in  $v_\varphi(x, t, z) = \sum_{k=1}^K \chi_{[(k-1)/K, k/K)}(t) \cdot v_{\varphi_k}(x, z)$ .

In [21], the model uses the above loss to learn a template shape with similar features to the training shapes. Hence, similar to our Riemannian regularization  $\int_0^1 \|v(\cdot, t)\|_V^2 dt$ , the pointwise loss is aimed at learning a proper template. Furthermore, note that when choosing  $\|\cdot\|_V = \|\cdot\|_{L_2(\Omega)}$  and defining  $\frac{d}{dt}x(t) = v_\varphi(x(t), t, z)$ , our Riemannian regularization penalizes large differences between  $x(1)$  and  $x(0)$ . As the pointwise loss penalizes a large  $\|x(1) - x(0)\|_2$  as well, the pointwise loss can be seen as a non-Riemannian version of our Riemannian LDDMM regularization.

### D.3 Datasets

In our work, we use two datasets: a synthetic rectangles dataset and a shape liver dataset [21, 86].

The synthetic rectangles dataset is created by generating random boxes with edge lengths uniformly distributed in  $[0.15, 0.85]$ . Subsequently, these boxes are rotated using a random rotation matrix. For training the point cloud-based model, we sample 100 000 uniform points and corresponding normals from the meshes. For training using the occupancy data (see Appendix B), we uniformly sample 100 000 points from  $\Omega = [-1, 1]^3$  and calculate the signed distance to the boxes. Subsequently, we calculate the occupancy values from these signed distance values. The training dataset consists of 100 randomly generated parallelograms, while the test dataset consists of 20 parallelograms.

For training the models on the liver dataset, we use the preprocessed data of [21]. The only additional preprocessing step is a scaling of their point cloud data and their mesh data. We multiply the points and the mesh vertices with a scaling factor of 0.75 to make sure that all the livers are present in the unit cube. As training data, we sample 100 000 uniform points and corresponding normals from the meshes. Finally, we use the same train-test split as [21], where the training dataset uses 145 samples and the test dataset uses 45 samples.

The liver data has downsampled versions, which are used in Section 5.2. These versions are obtained as outlined in the same section. For training, we use the same train-test split as for the original data. However, we exclude the samples with names `contrast_117`, `contrast_136`, and `noncontrast_034`. The reason for this is that the original meshes of these samples were not watertight, which caused issues while generating downsampled meshes. However, since these problematic meshes were only in the training dataset, testing on the test set is not affected.

### D.4 Training details

We jointly learn the latent codes  $z_i$ , the template implicit neural representation  $f_\theta$ , and the stationary velocity vector fields  $v_{\varphi_k}$  ( $k \in \{1, \dots, K\}$ ). As the template neural network architecture  $f_\theta$  comes from [38] and the stationary velocity vector fields  $v_{\varphi_k}$  from [21], we inherit the weight initialization schemes from these works. The latent codes  $z_i \in \mathbb{R}^{d_z}$  are initialized by sampling from  $\mathcal{N}(0, 1/d_z)$ , as in [21].

We use a batch size of 10 for each model and dataset pair. We approximate the expectations and spatial integrals in the point cloud data loss via Monte Carlo. For each expectation and integral, we sample 5000 points to estimate it. In particular, for the eikonal loss, we calculate the loss on 5000 random samples in  $\Omega = [-1, 1]^3$  and on 5000 warped surface samples. Subsequently, we average all the resulting values to obtain the final eikonal loss. Furthermore, for this eikonal loss, we only backpropagate the gradients concerning the parameters of the template INR  $f_\theta$ . Finally, for the experiments with the pointwise loss, the  $p_{ij}$  values of the pointwise loss are taken as the 5000 surface samples as well as the 5000 random samples used to calculate the regularization corresponding to  $\beta$  in Equation (15).

For the experiment done in Appendix B with the occupancy data, we first sample 5000 points in  $\Omega$  with their accompanying occupancy value. Half of the points lie inside, while the other half lie outside the shape. We calculate the binary cross entropy loss by averaging the binary cross entropies between the ground truth occupancy value and the estimated occupancy probability.

We update the neural network parameters and the latent codes  $z_i$  via separate Adam optimizers for the latent codes, the template NN  $f_\theta$ , and velocity field NN  $v_\varphi$ . The initial learning rate for the latent codes is  $10^{-3}$ , for the  $f_\theta$  parameters  $5 \cdot 10^{-4}$ , and for the  $v_\varphi$  parameters  $5 \cdot 10^{-4}$ . Moreover, for each Adam optimizer, we use a learning rate scheduler that multiplies the learning rate with 0.7 every 250 epochs. Finally, we bound the latent codes  $z_i$  to be within the unit sphere. For the remainder of the hyperparameters, see Table D.1.

Using the above considerations, we train our models on a high-performance computing cluster and use two NVIDIA A40 GPUs or two NVIDIA L40 GPUs. All of the GPUs have 48 GB of memory. The training took approximately 17 hours when using the A40’s and approximately 13.5 hours when using the L40’s.

Table D.1: The hyperparameter values every trained model uses on a specific dataset (rectangles or liver). Here  $N_h$  is the number of linear layers between the first and last linear layer in the template neural network (as presented in Figure D.1). Furthermore,  $c_{PW}$  denotes the regularization constant for the pointwise loss when training the model that uses this loss as the deformation regularizer.

<i>Hyperparameter</i>	<i>Value (Rect.)</i>	<i>Value (Liver)</i>
Epochs	4000	3000
Latent dimension	32	32
$d_{vel}$	512	512
$d_\mu$	256	256
$\epsilon$	0.05	0.05
$K$	10	10
$d_\mu$	256	256
$N_h$	5	5
$\sigma^2$	0.025	0.002
$\tau$	0.01	0.01
$\lambda$	0.005	0.005
$\beta$	1.5	1.5
$\alpha$	100	100
$\eta$	0.05	50
$c_{PW}$	0.1	0.1

## D.5 Inference details

For reconstructing shapes, we solve the following problem:

$$\min_z \mathcal{D}_{rec}((\phi_1^z)^{-1} \cdot \mathcal{T}_\theta, \mathcal{S}) + \gamma \|z\|_2^2,$$

where

- $\gamma \in \mathbb{R}$ ,
- $\frac{\partial}{\partial t} \phi_t^z(x) = v_\varphi(\phi_t^z(x), t, z)$  and  $\phi_0^z(x) = x$ ,
- $\mathcal{D}_{rec}((\phi_1^z)^{-1} \cdot \mathcal{T}_\theta, \mathcal{S}) = \mathbb{E}_{x \in \mathcal{S}} [ |I(x, z, 1)| ]$ ,
- $I(x, z, t) := ((\phi_t^z)^{-1} \cdot f_\theta)(x) := f_\theta(\phi_t^z(x))$ .

We solve this optimization problem by running an Adam optimizer with an initial learning rate of  $5 \cdot 10^{-2}$  for 800 iterations. At iteration 400 the learning rate is decreased by a factor of 10. Furthermore, we put  $\gamma = 10^{-4}$  and we initialize the latent code from  $\mathcal{N}(0, 0.01 \cdot I)$ .

## D.6 DAE training and inference details

To train and evaluate the DAE model [16], we utilized the original code provided by the authors. For training DAE we tried to tune the hyperparameters to the best of our capabilities.

First, we used Deformetrica’s atlas estimation [87] to compute an initial template. We set the noise standard deviation of the current similarity metric to 0.01 and the kernel width to 0.1. For the deformation kernel width, we used a value of 0.15. Furthermore, we used ScipyLBFGS as the optimizer with an initial step size of  $3 \cdot 10^{-5}$  and a very low convergence tolerance. Once we obtained the template, we employed Deformetrica’s PGA function to obtain a 32-dimensional latent space. We initialized the initial template with the estimated template and ran PGA with the same parameters as for the atlas estimation.

We utilized the calculated PGA to initialize the DAE model. We use 15000 epochs for this initialization. After that, we trained the DAE model for 15000 epochs with a learning rate of  $5 \cdot 10^{-4}$  and a batch size of 32. We kept the template fixed as learning it resulted in a lot of instability. We chose a splatting grid size of 16, a deformation grid size of 32, and a kernel width of 0.15 for the splatting layer and the current metric. We multiplied the data-fitting term by  $1e4$  and the velocity field regularization term by 0.1.

To evaluate the model, we used the encoder to estimate a latent vector. For the test dataset, we also performed an additional optimization step for 1500 iterations on the latent code using an Adam optimizer with a learning rate of  $5 \cdot 10^{-3}$ . This additional optimization step corresponds to the DAE+ version of the model, which was used to improve reconstruction performance on the test set.

## References

- [1] Boulbaba Ben Amor, Sylvain Arguillere, and Ling Shao. Resnet-lddmm: Advancing the lddmm framework using deep residual networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3707–3720, 2022.
- [2] Ankita Joshi and Yi Hong. Diffeomorphic image registration using lipschitz continuous residual networks. In *Proceedings of The 5th International Conference on Medical Imaging with Deep Learning*, volume 172 of *Proceedings of Machine Learning Research*, pages 605–617. PMLR, 2022.
- [3] Julian Krebs, Hervé Delingette, Boris Mailhé, Nicholas Ayache, and Tommaso Mansi. Learning a probabilistic model for diffeomorphic registration. *IEEE transactions on medical imaging*, 38(9):2165–2176, 2019.
- [4] Zhengyang Shen, Xu Han, Zhenlin Xu, and Marc Niethammer. Networks for joint affine and non-parametric image registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4224–4233, 2019.
- [5] Xiao Yang, Roland Kwitt, Martin Styner, and Marc Niethammer. Quicksilver: Fast predictive image registration—a deep learning approach. *NeuroImage*, 158:378–396, 2017.
- [6] Marc Niethammer, Roland Kwitt, and Francois-Xavier Vialard. Metric learning for image registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8463–8472, 2019.
- [7] Adrian Dalca, Marianne Rakic, John Guttag, and Mert Sabuncu. Learning conditional deformable templates with convolutional networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [8] Kun Han, Shanlin Sun, Xiangyi Yan, Chenyu You, Hao Tang, Junayed Naushad, Haoyu Ma, Deying Kong, and Xiaohui Xie. Diffeomorphic image registration with neural velocity field. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1869–1879, 2023.
- [9] Lin Tian, Soumyadip Sengupta, Hastings Greer, Raúl San José Estépar, and Marc Niethammer. Nephi: Neural deformation fields for approximately diffeomorphic medical image registration. *arXiv preprint arXiv:2309.07322*, 2023.
- [10] Jing Zou, Noémie Debroux, Lihao Liu, Jing Qin, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. Homeomorphic image registration via conformal-invariant hyperelastic regularisation. *arXiv preprint arXiv:2303.08113*, 2023.
- [11] Xi Jia, Alexander Thorley, Alberto Gomez, Wenqi Lu, Dipak Kotecha, and Jinming Duan. Fourier-net+: Leveraging band-limited representation for efficient 3d medical image registration. *arXiv preprint arXiv:2307.02997*, 2023.
- [12] Jian Wang and Miaomiao Zhang. Deepflash: An efficient network for learning-based medical image registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4444–4452, 2020.
- [13] Tony C. W. Mok and Albert C. S. Chung. Large deformation diffeomorphic image registration with laplacian pyramid networks. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 211–221. Springer, 2020.
- [14] Zhipeng Ding and Marc Niethammer. Aladdin: Joint atlas building and diffeomorphic registration learning with pairwise alignment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20784–20793, 2022.
- [15] Jacob Hinkle, David Womble, and Hong-Jun Yoon. Diffeomorphic autoencoders for lddmm atlas building. 2018.
- [16] Alexandre Bône, Maxime Louis, Olivier Colliot, Stanley Durrleman, Alzheimer’s Disease Neuroimaging Initiative, et al. Learning low-dimensional representations of shape data sets with diffeomorphic autoencoders. In *Information Processing in Medical Imaging*, pages 195–207. Springer, 2019.
- [17] Jian Wang and Miaomiao Zhang. Geo-sic: Learning deformable geometric shapes in deep image classifiers. In *Advances in Neural Information Processing Systems*, volume 35, pages 27994–28007. Curran Associates, Inc., 2022.
- [18] Nian Wu and Miaomiao Zhang. Neurepdiff: Neural operators to predict geodesics in deformation spaces. In *Information Processing in Medical Imaging*, pages 588–600. Springer, 2023.

- [19] David Lüdke, Tamaz Amiranashvili, Felix Ambellan, Ivan Ezhov, Bjoern H Menze, and Stefan Zachow. Landmark-free statistical shape modeling via neural flow deformations. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 453–463. Springer, 2022.
- [20] Kunal Gupta and Manmohan Chandraker. Neural mesh flow: 3d manifold mesh generation via diffeomorphic flows. In *Advances in Neural Information Processing Systems*, volume 33, pages 1747–1758. Curran Associates, Inc., 2020.
- [21] Shanlin Sun, Kun Han, Deying Kong, Hao Tang, Xiangyi Yan, and Xiaohui Xie. Topology-preserving shape reconstruction and registration via neural diffeomorphic flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20845–20855, 2022.
- [22] Vincent Arsigny, Olivier Commowick, Xavier Pennec, and Nicholas Ayache. A log-euclidean framework for statistics on diffeomorphisms. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 924–931. Springer, 2006.
- [23] John Ashburner. A fast diffeomorphic image registration algorithm. *Neuroimage*, 38(1):95–113, 2007.
- [24] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Diffeomorphic demons: Efficient non-parametric image registration. *NeuroImage*, 45(1):S61–S72, 2009.
- [25] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Symmetric log-domain diffeomorphic registration: A demons-based approach. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 754–761. Springer, 2008.
- [26] 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(2):139–157, 2005.
- [27] Marc Vaillant and Joan Glaunès. Surface matching via currents. In *International Conference on Information Processing in Medical Imaging*, pages 381–392. Springer, 2005.
- [28] Michael I Miller, Alain Trouvé, and Laurent Younes. Geodesic shooting for computational anatomy. *Journal of Mathematical Imaging and Vision*, 24:209–228, 2006.
- [29] François-Xavier Vialard, Laurent Risser, Daniel Rueckert, and Colin J Cotter. Diffeomorphic 3d image registration via geodesic shooting using an efficient adjoint calculation. *International Journal of Computer Vision*, 97:229–241, 2012.
- [30] Laurent Younes. *Shapes and Diffeomorphisms*. Springer, 2 edition, 2019.
- [31] Miaomiao Zhang and P Thomas Fletcher. Fast diffeomorphic image registration via fourier-approximated lie algebras. *International Journal of Computer Vision*, 127:61–73, 2019.
- [32] Sarang Joshi, Brad Davis, Matthieu Jomier, and Guido Gerig. Unbiased diffeomorphic atlas construction for computational anatomy. *NeuroImage*, 23:S151–S160, 2004.
- [33] Jun Ma, Michael I Miller, Alain Trouvé, and Laurent Younes. Bayesian template estimation in computational anatomy. *NeuroImage*, 42(1):252–261, 2008.
- [34] Jun Ma, Michael I Miller, and Laurent Younes. A bayesian generative model for surface template estimation. *International Journal of Biomedical Imaging*, 2010:1–14, 2010.
- [35] Miaomiao Zhang and P Thomas Fletcher. Bayesian principal geodesic analysis for estimating intrinsic diffeomorphic image variability. *Medical Image Analysis*, 25(1):37–44, 2015.
- [36] Miaomiao Zhang, William M Wells III, and Polina Golland. Probabilistic modeling of anatomical variability using a low dimensional parameterization of diffeomorphisms. *Medical Image Analysis*, 41:55–62, 2017.
- [37] Jelmer M Wolterink, Jesse C Zwienenberg, and Christoph Brune. Implicit neural representations for deformable image registration. In *International Conference on Medical Imaging with Deep Learning*, pages 1349–1359. PMLR, 2022.
- [38] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10286–10296, 2021.
- [39] Zerong Zheng, Tao Yu, Qionghai Dai, and Yebin Liu. Deep implicit templates for 3d shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1429–1439, 2021.
- [40] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Springer, 2006.
- [41] P Thomas Fletcher, Conglin Lu, Stephen M Pizer, and Sarang Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, 23(8):995–1005, 2004.

- [42] Miaomiao Zhang and Tom Fletcher. Probabilistic principal geodesic analysis. In *Advances in Neural Information Processing Systems*, volume 26, page 1178–1186. Curran Associates, Inc., 2013.
- [43] Martial Agueh and Guillaume Carlier. Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011.
- [44] Vivien Seguy and Marco Cuturi. Principal geodesic analysis for probability measures under the optimal transport metric. In *Advances in Neural Information Processing Systems*, volume 28, pages 3312–3320. Curran Associates, Inc., 2015.
- [45] Behrend Heeren, Chao Zhang, Martin Rumpf, and William Smith. Principal geodesic analysis in the space of discrete shells. *Computer Graphics Forum*, 37(5):173–184, 2018.
- [46] Marc Vaillant, Michael I Miller, Laurent Younes, and Alain Trouvé. Statistics on diffeomorphisms via tangent space representations. *NeuroImage*, 23:S161–S169, 2004.
- [47] Benjamin Charlier, Jean Feydy, David W Jacobs, and Alain Trouvé. Distortion minimizing geodesic subspaces in shape spaces and computational anatomy. In *VipIMAGE 2017*, pages 1135–1144. Springer, 2018.
- [48] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AICHE journal*, 37(2):233–243, 1991.
- [49] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [50] Nina Miolane and Susan Holmes. Learning weighted submanifolds with variational autoencoders and riemannian variational autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14503–14511, 2020.
- [51] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations*, 2018.
- [52] Georgios Arvanitidis, Soren Hauberg, Philipp Hennig, and Michael Schober. Fast and robust shortest paths on manifolds learned from data. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1506–1515. PMLR, 2019.
- [53] Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1540–1550. PMLR, 2018.
- [54] Clément Chadebec, Clément Mantoux, and Stéphanie Allasonnière. Geometry-aware hamiltonian variational auto-encoder. *arXiv preprint arXiv:2010.11518*, 2020.
- [55] Georgios Arvanitidis, Bogdan M. Georgiev, and Bernhard Schölkopf. A prior-based approximate latent riemannian metric. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 4634–4658. PMLR, 2022.
- [56] Cong Geng, Jia Wang, Li Chen, Wenbo Bao, Chu Chu, and Zhiyong Gao. Uniform interpolation constrained geodesic learning on data manifold. *arXiv preprint arXiv:2002.04829*, 2020.
- [57] Tao Yang, Georgios Arvanitidis, Dongmei Fu, Xiaogang Li, and Søren Hauberg. Geodesic clustering in deep generative models. *arXiv preprint arXiv:1809.04747*, 2018.
- [58] Clément Chadebec and Stéphanie Allasonnière. Data augmentation with variational autoencoders and manifold sampling. In *Deep Generative Models, and Data Augmentation, Labelling, and Imperfections*, pages 184–192. Springer, 2021.
- [59] Matan Atzmon, David Novotny, Andrea Vedaldi, and Yaron Lipman. Augmenting implicit neural shape representations with explicit deformation fields. *arXiv preprint arXiv:2108.08931*, 2021.
- [60] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, page 6572–6583. Curran Associates, Inc., 2018.
- [61] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 15379–15389. Curran Associates, Inc., 2019.
- [62] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, 2019.
- [63] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.



- [64] Alexander Tong, Jessie Huang, Guy Wolf, David Van Dijk, and Smita Krishnaswamy. TrajectoryNet: A dynamic optimal transport network for modeling cellular dynamics. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9526–9536. PMLR, 2020.
- [65] Derek Onken, Levon Nurbekyan, Xingjian Li, Samy Wu Fung, Stanley Osher, and Lars Ruthotto. A neural network approach for high-dimensional optimal control applied to multiagent path finding. *IEEE Transactions on Control Systems Technology*, 31(1):235–251, 2023.
- [66] Yannik P. Wotte, Sven Dummer, Nicolò Botteghi, Christoph Brune, Stefano Stramigioli, and Federico Califano. Discovering efficient periodic behaviors in mechanical systems via neural approximators. *Optimal Control Applications and Methods*, 44(6):3052–3079, 2023.
- [67] Liu Yang and George Em Karniadakis. Potential flow generator with l2 optimal transport regularity for generative models. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):528–538, 2022.
- [68] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5379–5389, 2019.
- [69] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.
- [70] Lars Ruthotto, Stanley J Osher, Wuchen Li, Levon Nurbekyan, and Samy Wu Fung. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences*, 117(17):9183–9193, 2020.
- [71] Chris Finlay, Joern-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ODE: the world of Jacobian and kinetic regularization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3154–3164. PMLR, 2020.
- [72] Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9223–9232, 2021.
- [73] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987.
- [74] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019.
- [75] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020.
- [76] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019.
- [77] Martin Bauer, Martins Bruberis, and Peter W Michor. Overview of the geometries of shape spaces and diffeomorphism groups. *Journal of Mathematical Imaging and Vision*, 50:60–97, 2014.
- [78] Michael I Miller, Alain Trouvé, and Laurent Younes. On the metrics and euler-lagrange equations of computational anatomy. *Annual review of biomedical engineering*, 4(1):375–405, 2002.
- [79] Nikhil Singh, Jacob Hinkle, Sarang Joshi, and P Thomas Fletcher. A vector momenta formulation of diffeomorphisms for improved geodesic regression and atlas construction. In *2013 IEEE 10th International Symposium on Biomedical Imaging*, pages 1219–1222. IEEE, 2013.
- [80] Jian Wang and Miaomiao Zhang. Bayesian atlas building with hierarchical priors for subject-specific regularization. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 76–86. Springer, 2021.
- [81] Laurent Younes. Jacobi fields in groups of diffeomorphisms and applications. *Quarterly of Applied Mathematics*, 65(1):113–134, 2007.
- [82] Mirza Faisal Beg and Ali Khan. Computing an average anatomical atlas using lddmm and geodesic shooting. In *3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro*, pages 1116–1119. IEEE, 2006.
- [83] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020.
- [84] Justin Solomon, Mirela Ben-Chen, Adrian Butscher, and Leonidas Guibas. As-killing-as-possible vector fields for planar deformation. *Computer Graphics Forum*, 30(5):1543–1552, 2011.

- [85] Michael Tao, Justin Solomon, and Adrian Butscher. Near-isometric level set tracking. *Computer Graphics Forum*, 35(5):65–77, 2016.
- [86] Xuming Chen, Shanlin Sun, Narisu Bai, Kun Han, Qianqian Liu, Shengyu Yao, Hao Tang, Chupeng Zhang, Zhipeng Lu, Qian Huang, et al. A deep learning-based auto-segmentation system for organs-at-risk on whole-body computed tomography images for radiation therapy. *Radiotherapy and Oncology*, 160:175–184, 2021.
- [87] Alexandre Bône, Maxime Louis, Benoît Martin, and Stanley Durrleman. Deformetrica 4: An open-source software for statistical shape analysis. In *Shape in Medical Imaging*, pages 3–13. Springer, 2018.
- [88] Jürgen Jost. *Riemannian geometry and geometric analysis*. Springer, 7 edition, 2017.