



Disentangling Geometric Deformation Spaces in Generative Latent Shape Models

Tristan Aumentado-Armstrong^{1,2,3} · Stavros Tsogkas^{1,2} · Sven Dickinson^{1,2,3} · Allan Jepson^{1,2}

Received: 17 December 2020 / Accepted: 2 January 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

A complete representation of 3D objects requires characterizing the space of deformations in an interpretable manner, from articulations of a single instance to changes in shape across categories. In this work, we improve on a prior generative model of geometric disentanglement for 3D shapes, wherein the space of object geometry is factorized into rigid orientation, non-rigid pose, and intrinsic shape. The resulting model can be trained from raw 3D shapes, without correspondences, labels, or even rigid alignment, using a combination of classical spectral geometry and probabilistic disentanglement of a structured latent representation space. Our improvements include more sophisticated handling of rotational invariance and the use of a diffeomorphic flow network to bridge latent and spectral space. The geometric structuring of the latent space imparts an interpretable characterization of the deformation space of an object. Furthermore, it enables tasks like pose transfer and pose-aware retrieval without requiring supervision. We evaluate our model on its generative modelling, representation learning, and disentanglement performance, showing improved rotation invariance and intrinsic-extrinsic factorization quality over the prior model.

Keywords 3D Shape · Generative models · Disentanglement · Articulation · Deformation · Representation learning

Communicated by Edmond Boyer.

Disclaimer: Tristan Aumentado-Armstrong and Stavros Tsogkas contributed to this article in their personal capacity as PhD student and Adjunct Professor at the University of Toronto, respectively. Sven Dickinson and Allan Jepson contributed to this article in their personal capacity as Professors at the University of Toronto. The views expressed (or the conclusions reached) by the authors are their own and do not necessarily represent the views of Samsung Research America, Inc.

✉ Tristan Aumentado-Armstrong
tristan.aumentado@mail.utoronto.ca

Stavros Tsogkas
tsogkas@cs.toronto.edu

Sven Dickinson
sven@cs.toronto.edu

Allan Jepson
jepson@cs.toronto.edu

¹ University of Toronto, Toronto, ON, Canada

² Samsung Toronto AI Research Center, Toronto, ON, Canada

³ Vector Institute for Artificial Intelligence, Toronto, ON, Canada

1 Introduction

A major goal of representation learning is to discover and separate the underlying explanatory factors that give rise to some set of data (Bengio et al., 2013). For many objects, such as 3D shapes of biological entities, structuring their representation within a learned model means understanding the different modes of their deformation spaces. For instance, rotating a chair does not affect its category, nor does articulated deformation of a cat alter its identity. In general, different geometric deformations may be semantically distinct, e.g., shape style (Marin et al., 2020), intrinsic versus extrinsic alterations (Corman et al., 2017), or geometric texture details (Berkiten et al., 2017). In other words, for many objects, we can naturally factorize the associated deformation space, based on geometric characteristics (Fig. 1).

Such a disentanglement can provide a useful structuring of the 3D shape representation. For example, in a vision context, one could constrain inference of a 3D model from a motion sequence to change in pose, but not intrinsic shape. Or, in the context of graphics, separating shape and pose allows for tasks such as deformation transfer or shape interpolation.

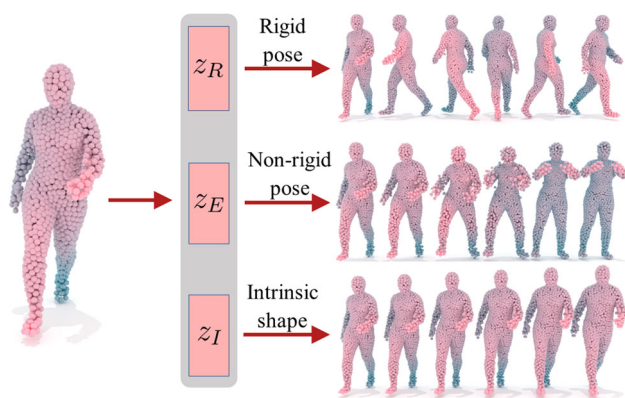


Fig. 1 Depiction of overall framework goal. We factorize the latent deformation space of a given 3D object into rigid pose z_R , extrinsic non-rigid pose z_E , and intrinsic shape z_I , without supervision

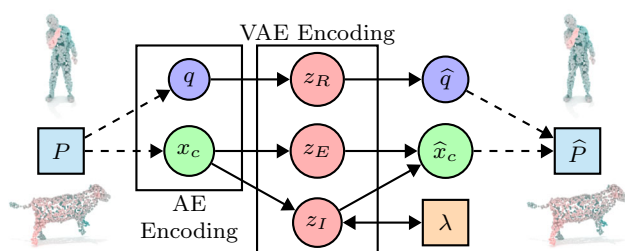


Fig. 2 A schematic overview of the combined two-level architecture used as the generative model. A point cloud P is first encoded into (q, x_c) by a *deterministic* AE, where q is the quaternion representing the rotation (rigid pose) of the shape, and x_c the compressed representation of the input P , in its *canonical* orientation. (q, x_c) is then further compressed into a latent representation $z = (z_R, z_E, z_I)$ of a VAE. The hierarchical latent variable z has disentangled subgroups in red (representing rotation, extrinsics, and intrinsics, respectively). The intrinsic latent subgroup z_I is used to contain the information in the LBO spectrum λ using an invertible mapping. Both the extrinsic z_E and intrinsic z_I are utilized to compute the shape \hat{x}_c in the AE's latent space. The latent rotation z_R is used to predict the quaternion \hat{q} . Finally, the decoded representation (\hat{q}, \hat{x}_c) is used to reconstruct the original point cloud \hat{P} . The deterministic AE mappings are shown as dashed lines; VAE mappings are represented by solid lines

In this work, we consider a purely geometric decomposition of object deformations, separating the space into rigid orientation, non-rigid pose, and shape. Our method is based on methods from spectral geometry, utilizing the isometry invariance of the Laplace–Beltrami operator spectrum (LBOS). The LBOS characterizes the intrinsic geometry of the shape; in contrast, we refer to the space of non-rigid isometric deformations of the shape as its extrinsic geometry, in a manner similar to Corman et al. (2017). This decomposition is performed in the latent space of a generative model, using information-theoretic methods for disentangling random variables, resulting in three latent vectors for rigid orientation, pose, and shape. We apply our model to several tasks requiring this factorized structure, including pose-aware retrieval and pose-versus-shape interpolation (for

which pose transfer is a special case). See Fig. 2 for an overview of our approach.

We focus on minimizing the supervision required for our model, eschewing requirements for identical meshing, correspondence, or labels. Thus, our method is orthogonal to advances in neural architectures, as it can be applied to any encoder or decoder model. For the same reason, it is also agnostic to the 3D modality (e.g., meshes, voxels, or implicit fields). We include experiments on meshes and point clouds, to showcase the versatility of our method with respect to shape modality, but we choose to focus on the latter, as they are a common data type in computer vision.¹

Our method builds on a prior model (Aumentado-Armstrong et al., 2019), the geometrically disentangled VAE (GDVAE), with two major algorithmic improvements: (1) we enhance the ability of the network to factorize rotation, and (2) we replace a simple spectrum regressor with a diffeomorphic flow network. For the first point, we investigate two representation learning approaches that allow the model to discern a canonical rigid orientation, with or without assuming aligned training data. The latter change not only guarantees that spectral information is preserved by the mapping (due to the invertibility requirement), but it can be readily applied to generative modelling (due to the tractability of the likelihood calculation) and it permits shape-from-spectrum computations that prevent contaminating learned latent intrinsics with extrinsic information. This allows us to define a better training procedure, in which we use a shape-from-spectrum starting point, instead of the initial input shape, thus ensuring that the latent intrinsics cannot access extrinsics. These two improvements result in superior disentanglement quality, compared to the prior GDVAE model.

2 Background

2.1 Rotation Invariant Shape Representation

Invariance to rotation is generally a desirable property of shape representations, since many tasks (such as categorization or retrieval) tend to consider orientation a nuisance variable. Hence, there is a significant body of work on how to learn such rigid invariance.

Classical research includes many types of geometric features, directly computed from input shapes, that are rotation invariant (e.g., Guo et al., 2014), such as structural indexing (Stein et al., 1992), signature of histogram orientations (Tombari et al., 2010), spin images (Johnson & Hebert, 1999), and point signatures (Chua & Jarvis, 1997). More recently, SRINet (Sun et al., 2019), ClusterNet (Chen et al.,

¹ However, we note that, by default, we use spectra derived from meshes, unless otherwise specified (but see Sect. 5.3.3).

2019a), and RConv (Zhang et al., 2019) design rotation invariant hand-crafted features that can be extracted from point clouds (PCs), for use in learning algorithms.

Separately, rotation *equivariance* has been achieved in voxel shapes using group convolutions (Worrall & Brostow, 2018) and spherical correlations (Cohen et al., 2018), which can be utilized to obtain invariance. PRIN (You et al., 2018) computes rotation invariant features for point clouds, but requires the application of convolutions on spherical voxel grids. SPHNet (Poulenard et al., 2019) attains rotation invariance without voxelization, by extending feature signals defined on a shape into \mathbb{R}^3 , and then using a specific non-linear transform of the signal, convolved with a spherical harmonic kernel. Additional network architectures have been applied to modelling equivariations, including tensor field networks (Thomas et al., 2018; Fuchs et al., 2020), graph-theoretic methods (Kondor et al., 2018) and quaternion-based approaches (Zhao et al., 2020; Zhang et al., 2020). See also Dym and Maron (2020) for additional discussions and theoretical analysis.

Other works focus on changing the input and/or utilizing other representation learning techniques, which are more closely related to our work. The PCA-RI model (Xiao et al., 2020) achieves rotation invariance by transforming each shape into an intrinsic reference frame, defined by its principal components, handling frame ambiguity (due to eigenvector signs) by duplicating the input. Info3D (Sanghi, 2020) uses techniques from unsupervised contrastive learning to encourage rotation invariance in the representation, including the ability to handle unaligned data. Li et al. (2019) attain equivariance by rotating each input point cloud by a discrete rotation group. Similar to this, an approximately rotation invariant encoder can be defined by feeding in randomly rotated copies of the input (Sanghi & Danielyan, 2019). We build on this latter approach to define one version of our 3D autoencoder (AE). For our other approach, we utilize Feature Transform Layers (FTLs) (Worrall et al., 2017), which allow us to make latent space rotations equivalent to 3D data space rotations. In both cases, rather than removing rigid transforms from the embedding, we attempt to factorize such transforms out, as part of the deformation space of the object.

More specifically, we consider two general approaches to learning rotation invariant representations, building on related work as noted above. Both methods are modality agnostic (e.g., not requiring spherical voxelization), architecture independent (e.g., not necessitating particular types of convolution), able to avoid information loss in feature extraction, and do not increase the cost of a forward pass (e.g., no duplication of inputs). In this sense, our method is largely orthogonal to architectural improvements for PC processing, as well as the aforementioned approaches to rotation invariance. Indeed, they can be readily applied to other 3D shape

modalities. This is because our approaches modify only the latent representation and loss calculation procedure, allowing the use of arbitrary features as input, including rotation invariant ones. Nevertheless, we show that, despite obtaining features from a simple PointNet (Qi et al., 2017), we can still approximately attain rotation invariance without architectural alterations. Finally, the utility of much of the related work above for generative modelling and/or autoencoding is unclear; hence, we choose to use simpler architectures already known to work for these purposes (Achlioptas et al., 2017; Aumentado-Armstrong et al., 2019).

2.2 Shape Analysis via Spectral Geometry

Any 3D surface can be viewed as a 2D Riemannian manifold (\mathcal{M}, g) , with metric tensor g , which allows the application of differential geometry to shape analysis in computer graphics and vision. One major technique in this area is the use of *spectral geometry*, which is mainly concerned with the Laplace–Beltrami Operator (LBO), Δ_g , and its associated spectrum (i.e., the eigenvalues λ of $-\Delta_g \phi_i = \lambda_i \phi_i$) for shape processing (Patané, 2016). Use of the spectrum generalizes classical Fourier analysis on Euclidean domains to manifolds, transferring concepts from signal processing to transforms of non-Euclidean geometry itself (Taubin, 1995). The LBO spectrum (LBOS) characterizes the intrinsic properties of a manifold (Lévy, 2006; Rustamov, 2007; Vallet & Lévy, 2008), sufficiently matching human intuition on the meaning of “shape”, to the extent it is considered as its “DNA” (Reuter et al., 2006). Mathematically, *intrinsic* properties of a shape are those that depend only on its metric tensor, independent of its embedding (Corman et al., 2017); this includes, for example, geodesic distances and the LBOS. Among the most useful advantages of intrinsic shape properties is *isometry invariance*, meaning intrinsics do not change in response to alterations that do not affect the metric. This includes rigid transforms, as well as certain non-rigid deformations, such as biological articulations (approximately). Algorithms relying on shape intrinsics are therefore able to ignore such deformations (e.g., recognize a person regardless of articulated pose). We show some examples of the intrinsic-extrinsic geometric decomposition provided by the LBOS in Figs. 3 and 4. We remark that we also refer to extrinsic shape as non-rigid pose, since this is the most intuitive interpretation for the case of approximately isometrically articulating objects, like animals.

Intrinsic spectral geometry processing has thus yielded numerous useful techniques for vision and graphics, often due to its isometry invariance. This includes semi-localized, articulation invariant feature extraction, such as the heat (Gebal et al., 2009; Sun et al., 2009) and wave (Aubry et al., 2011) kernel signatures, later extended to learned generalizations (Boscaini et al., 2015b). Similar techniques can

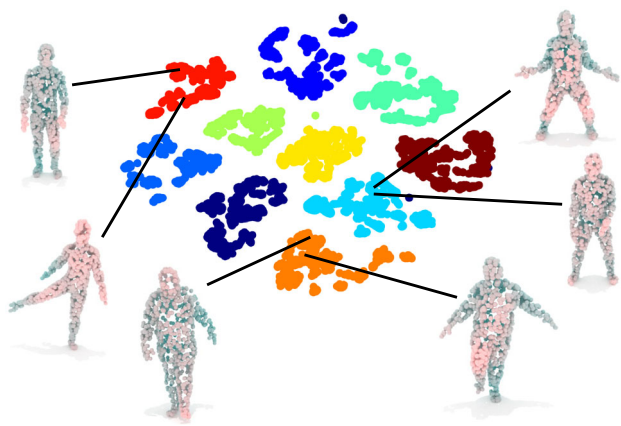


Fig. 3 Pose versus shape factorization via LBOS. A t-sne (van der Maaten & Hinton, 2008) plot of LBOSs from the Dyna dataset (Pons-Moll et al., 2015), with illustrative accompanying point cloud representations for several spectra. Different body shapes are mapped close together, regardless of articulated pose

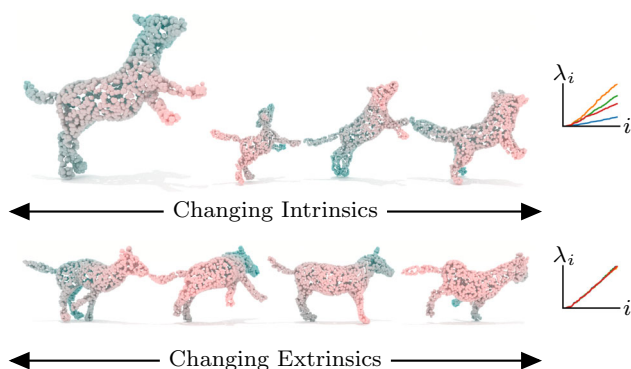


Fig. 4 Visual explanation of the use of spectral geometry in characterizing intrinsic versus extrinsic shape. We display two rows of animals, showing different *intrinsic* (first row) and *extrinsic* (second row), across the columns per inset. The plots show the LBOS λ across shapes (indices i range from 1 to 50); notice the lack of variability as extrinsics change

be applied to a variety of downstream tasks for 3D shapes as well, including correspondence (Ovsjanikov et al., 2012; Rodolà et al., 2017), retrieval (Bronstein et al., 2011), segmentation (Reuter, 2010), analogies (Boscaini et al., 2015a), classification (Masoumi & Hamza, 2017), and manipulation (Vallet & Lévy, 2008). Beyond the standard LBOS, more recent research has also explored localized manifold harmonics (Melzi et al., 2018; Neumann et al., 2014), modifications of the LBO (Andreux et al., 2014; Choukroun et al., 2018), and extrinsic spectral geometry (Liu et al., 2017; Wang et al., 2017; Ye et al., 2018).

While the above applications rely on the spectral intrinsics of existing shapes, the inverse problem seeks to reconstruct a shape from an intrinsic operator (or function thereof), such as the LBO (Boscaini et al., 2015a; Chern et al., 2018; Huang et al., 2019). In particular, the *shape-from-spectrum* (SfS)

task seeks to recover a shape from its LBOS, an instance of an “inverse eigenvalue problem” investigated in other fields (e.g., Chu & Golub, 2005; Panine & Kempf, 2016). This enables useful spectral-space tasks, such as shape style transfer and correspondence matching (Cosmo et al., 2019; Marin et al., 2021). Fortunately, despite theoretical results suggesting such recovery is not always possible, due to the existence of non-isometric isospectral shapes (i.e., “one cannot hear the shape of a drum”) (Gordon et al., 1992; Kac, 1966), it appears *practically* possible in many circumstances (Cosmo et al., 2019; Panine & Kempf, 2016). Indeed, Cosmo et al. (2019) show several applications of their approach to SfS recovery, though it is computationally costly and difficult to constrain. More recently, Rampini et al. (2021) utilize spectral perturbations to define universal geometric deformations, while Moschella et al. (2022) apply a learning framework to process unions of partial shapes in the spectral domain. Closest to our work, Marin et al. (2020, 2021) apply a data-driven approach to the SfS problem, among other tasks.

In this work, we focus on utilizing the classical LBOS as a purely intrinsic characterization of the shape. By exploiting the approximate articulation invariance conferred by its isometry invariance, we gain access to a signal that can separate intrinsic shape from articulated pose, without supervision beyond the geometry itself. While the LBO has been used to perform disentangled shape manipulations in the context of computer graphics and vision, such as isometric shape interpolation (Baek et al., 2015), spectral pose transfer (Yin et al., 2015), and shape-from-spectrum recovery (Cosmo et al., 2019), we show how to do such manipulations within a generative model, as a byproduct of the learned representation.

2.3 Learning Shape-Pose Disentanglement

A common task that has been tackled in the context of computer graphics is *pose transfer*. Utilizing a small set of correspondences, an optimization-based approach can be applied to perform deformation transfer (Sumner & Popović, 2004). Later work utilized the LBO eigenbases to perform pose transfer (Kovnatsky et al., 2013; Yin et al., 2015), via exchanging low-frequency coefficients of the manifold harmonics. In our work, we use the LBOS instead, which avoids issues of basis computation and spectral compatibility (Kovnatsky et al., 2013). Basset et al. (2020) consider transferring shape instead of pose; due to our symmetric formulation, our approach is also capable of this. We refer to Roberts et al. (2020) for a survey of related work.

Recently, several works have attacked pose transfer from a machine learning point of view. Gao et al. (2018) present a method for mesh deformation transfer using a cycle consistent GAN and a visual similarity metric, but require retraining new models for each source and target set. Levinson et al.

(2019) utilize a mesh VAE, which relies on data having identical meshing, to separate pose and shape via batching with identical pose and shape labels. LIMP (Cosmo et al., 2020) disentangles intrinsic and extrinsic deformations in a generative model, utilizing a differentiable geodesic distance regularizer; identical meshing or labels are not required, though correspondence is. Zhou et al. (2020) devise a method for separating intrinsics and extrinsics using corresponding meshes known only to have the same shape but different pose, and applying a powerful as-rigid-as-possible geometric prior. Similarly, Fumero et al. (2021) make use of data pairs with shared transforms to obtain a general disentanglement mechanism. Su et al. (2021) also use identity-based semantic supervision, but with an adversarial mechanism on point clouds. Finally, Marin et al. (2020) consider learning a bijective mapping of the LBOS as well, examining its use in the context of neural networks for several tasks, including spectrum estimation from point clouds and shape style transfer; however, they do not focus on deformation space factorization or generative representation learning. Followup work Marin et al. (2021) investigates shape-from-spectrum tasks, as well as shape-pose disentanglement via optimization.

In our work, we focus on learning a generative representation that factorizes the latent deformation space into intrinsic shape and extrinsic pose, without supervision. We do not require labels (e.g., identity, pose, or shape), identical meshing, correspondence, or even rigid alignment—only the raw geometry, which we use to compute the LBOS. Rather than targeting pose transfer specifically, in our model, the ability to transfer articulation arises naturally from the learned representation. In particular, we build on the GDVAE model (Aumentado-Armstrong et al., 2019), which disentangles shape and pose into two continuous and independent latent factors. Our method, which we refer to as the GDVAE++ model, includes adding a bijective mapping from an LBO spectrum to the space of latent intrinsics, and defining a new training scheme based on this function. We show that the resulting model is significantly improved in terms of disentanglement.

3 Autoencoder Model

Our model consists of two components: an autoencoder (AE) on the 3D shape data and a variational autoencoder (VAE) defined on the latent space of the AE. We show an overview of the complete framework in Fig. 2.

In this work, the AE is used to map a 3D point cloud (PC) to a latent vector, and then decode it back to a reconstruction of the original input. In contrast to the AE used in the prior GDVAE model (Aumentado-Armstrong et al., 2019), we specifically consider the rotational invariance properties of the AE architecture.

Notation We assume our input is a PC $P \in \mathbb{R}^{N_p \times 3}$, which we want to reconstruct as $\hat{P} \in \mathbb{R}^{N_p \times 3}$. To do so, we encode P into a rigid rotation, represented as a quaternion $q \in \mathbb{R}^4$, and canonically oriented non-rigid latent shape, $x_c \in \mathbb{R}^n$, using learned mappings E_r and E_x . We can also obtain a canonical PC, $P_c = D(x_c) \in \mathbb{R}^{N_p \times 3}$, via a decoder D . The details for obtaining this rigid versus non-rigid factorization are given below.

3.1 Autoencoder Architecture

We consider two possible AE architectures on PCs. Both models attempt to regress a rotation matrix and a rotation invariant latent shape representation from an input. The first type, which we denote “standard” (STD), uses a straightforward reconstruction loss, but also includes a random rotation before attempting to encode the shape, inspired by prior work (Li et al., 2019; Sanghi & Danielyan, 2019). The second type relies on feature transform layers (FTLs) (Worrall et al., 2017) to learn a latent vector space that transforms covariantly with the 3D data space under rotation, thus allowing the model to learn how to “derotate” to a canonical representation (denoted “FTL-based”).

Implementation-wise, we use PointNet (Qi et al., 2017) to encode an input point cloud, P (which allows us to handle dynamic PC sizes), and fully connected layers (with batch normalization and ReLU) for all other learned mappings, unless otherwise specified. See Appendix F.1 for details.

3.1.1 Standard Architecture

Let P be an input PC, that has potentially undergone an arbitrary rotation. We learn two mappings as our encoder, E_r and E_x , which map P to a quaternion $q = E_r(P)$ and a latent shape embedding $x_c = E_x(P)$. Our decoder D generates a canonically oriented PC $P_c = D(x_c)$, which can be rotated to match the input via $\hat{P} = P_c R(q)$, where $R(q)$ is the parameter-less conversion from quaternion to rotation matrix. Inspired by Li et al. (2019) and Sanghi and Danielyan (2019), we insert an additional layer before E_x that randomly rotates P (i.e., $x_c = E_x(P\tilde{R})$, \tilde{R} being a random sample), to further encourage learning rotation invariant features. We only do this for the standard architecture, shown in Fig. 5.

3.1.2 FTL-Based Architecture

We also consider a slightly more complex architecture with a latent space designed for interpretability under rotation transformations, using a Feature Transform Layer (FTL) (Worrall et al., 2017). Several methods have utilized latent-space rigid transforms for mapping 3D data between views (Chen et al., 2019d,c; Rhodin et al., 2018, 2019). Our design is in particular inspired by prior work that extracts *canonical rep-*

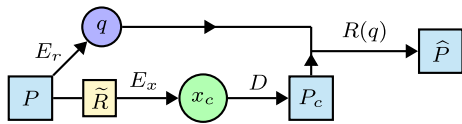


Fig. 5 STD AE architecture. Our standard AE encodes an input PC P into a rigid pose component (quaternion q) and canonically oriented shape embedding x_c . Before passing P to the shape encoder E_x , a random rotation \tilde{R} is sampled and applied. The decoder D then generates the canonical PC P_c , which is rotated by $R(q)$ into the final reconstruction \hat{P}

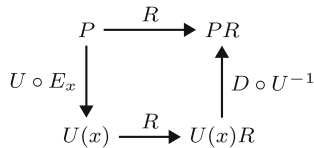


Fig. 6 Desired commutativity structure of FTL-based architecture. Ideally, latent rotations should have the same effect as in the data space

representations in the context of 3D human pose using FTLs (Remelli et al., 2020). Nevertheless, the architecture components of the FTL-AE are nearly the same as those of the STD-AE.

Rotational Feature Transform Layers The main idea behind FTLs is to view a latent vector $x \in \mathbb{R}^n$ as an ordered set of subvectors $U(x) = (u(x)_1, \dots, u(x)_{N_s}) \in \mathbb{R}^{N_s \times 3}$, where $N_s = n/3$ and $u(x)_i \in \mathbb{R}^3$, by simply folding it into a matrix. Consider rotating a point cloud $P \in \mathbb{R}^{N_p \times 3}$ by a 3D rotation operation $R \in \mathbb{R}^{3 \times 3}$, to get a new shape PR . By folding, one can analogously perform this rigid transformation on a “latent point cloud”, as $U(x)R$. Ideally, applying R to P or $U(x)$ has the *same effect* (i.e., rotates the underlying shape in the same way), resulting in an interpretable latent space, with respect to rotation. We define the rotational feature transform layer $F(R, x) = U^{-1}(U(x)R)$ as a latent rotation R of the subvectors of x , where the inverse U^{-1} “unfolds” the ordered set of subvectors into a single vector-valued latent variable again (as opposed to the “folding” operator U). We will use the FTL mapping F to enforce a rotation equivariant structure onto the latent space, thus allowing us to “derotate” the shape embedding to some canonical rigid pose. We depict the desired duality over rotations in Fig. 6.

Architectural Details Utilizing similar notation to Sect. 3.1.1, we first encode $q = E_r(P)$, as before, and convert it to a predicted rotation $\hat{R} = R(q)$. We then compute a *non-canonical* latent shape $\tilde{x} = \tilde{E}_x(P)$, which encodes the *rotated* shape P . We then use the FTL to obtain the canonical latent shape via $x_c = F(\hat{R}, \tilde{x})$, which can be decoded via $P_c = D(x_c)$ with shared parameters. As before, we obtain the final reconstruction via $\hat{P} = P_c \hat{R}$. For notational consistency, we write $E_x(P) = x_c = F(\hat{R}, \tilde{E}_x(P))$. See Fig. 7 for a visual depiction.

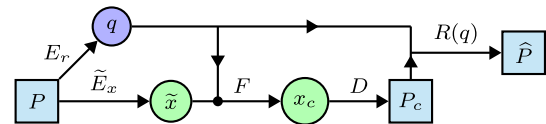


Fig. 7 FTL-based AE architecture. An input PC, P , is encoded into a quaternion q and a *pose-aware* embedding \tilde{x} , representing the rotated (rather than the canonical) shape. The rotational FTL F is then used to de-rotate \tilde{x} to obtain the *canonically* oriented shape $x_c = E_x(P) = F(R(q), \tilde{x})$. Finally, a reconstruction of the input, \hat{P} , is produced by rotating the decoded canonical PC $P_c = D(x_c)$ using the predicted rigid pose $R(q)$

This FTL-based architecture provides greater interpretability in terms of the effect of a rigid transform on the representation; rather than trying to remove the dependence on rotation, we attempt to explicitly characterize it. Rotations in the 3D *data* space should thus have an identical effect on the resulting *latent* space representation (and vice versa).

3.2 Autoencoder Loss Objective

The overall loss function for the AE can be written

$$\mathcal{L}_{\text{AE}} = \mathcal{L}_c + \mathcal{L}_R + \mathcal{L}_P + \mathcal{L}_x, \quad (1)$$

where the terms control representational consistency \mathcal{L}_c , rotation prediction \mathcal{L}_R , reconstruction \mathcal{L}_P and regularization \mathcal{L}_x . These terms will be different depending on whether one uses the STD-AE (Sect. 3.1.1) or FTL-AE (Sect. 3.1.2).

3.2.1 Standard Loss Objective

Reconstruction Loss The reconstruction loss term for the STD architecture is $\mathcal{L}_P = \gamma_P D_P(P, P_c)$, where D_P includes a Chamfer distance and an approximate Hausdorff loss, similar to prior work (Aumentado-Armstrong et al., 2019; Chen et al., 2019b):

$$D_P(P_1, P_2) = \alpha_C d_C(P_1, P_2) + \alpha_H d_H(P_1, P_2), \quad (2)$$

in which the squared L_2 distance is used between matched points.

Cross-Rotational Consistency Loss is a simple loss designed to promote consistency of the latent representation across rotations of the input, i.e., encourage rotation invariance. First, we split each batch into N_R copies of the same PCs; we then apply a different random rotation to each copy. Letting $x_{c,i}$ be the embedding of P after having undergone the i th rotation, the loss is then

$$\mathcal{L}_c = \frac{\gamma_c}{M_c} \sum_{i=1}^{N_R} \sum_{j>i} \|x_{c,i} - x_{c,j}\|_2^2, \quad (3)$$

where M_c is the number of pairwise distances. Note that, unlike combining features across rotated copies (Xiao et al., 2020; Li et al., 2019), this approach does not increase the computational cost of a forward pass for a single input.

Rotation Loss depends on whether we assume the data is rigidly aligned or unaligned, i.e., whether we have rotational supervision or not. In the supervised case, where the canonical rigid pose is shared across data examples, we simply predict the real rotation for every example: $\mathcal{L}_R = \gamma_R d_R(R, \widehat{R})$, where

$$d_R(R_k, R_\ell) = \frac{1}{\pi} \arccos \left(\frac{\text{tr}(R_k R_\ell^T) - 1}{2} \right) \quad (4)$$

is the geodesic distance on $SO(3)$ (Huynh, 2009).

In the *unsupervised* case, we enforce a consistency loss across rotational predictions, which does not rely on a ground truth P being canonical across multiple shapes. Instead, it only asks that the predicted rotations of an object have the same *relative* difference as the original rotations of the input (which should be true regardless of whether P was originally canonically oriented). Consider rotations of a PC, $P_i = P_0 R_0 R_i$, where our data follows $P = P_0 R_0$, in which P_0 (the ground truth PC in canonical orientation) and R_0 (the rotation of the ground truth datum) are both unknown. Our predictions are $\widehat{P}_i = P_c \widehat{R}_i$, so for any two rotations of a single observed PC (e.g., P_i and P_j), we want $\widehat{R}_i \approx R_0 R_i$ and $\widehat{R}_j \approx R_0 R_j$, meaning we want to predict R_k *relative to* R_0 . Combining these equations means we want $R_i^T \widehat{R}_i \approx R_j^T \widehat{R}_j$ for each such pair. Formally, we write this constraint as

$$\mathcal{L}_R = \frac{\gamma_r}{M_c} \sum_{i=1}^{N_R} \sum_{j>i} d_R(R_i^T \widehat{R}_i, R_j^T \widehat{R}_j), \quad (5)$$

where R_k and \widehat{R}_k are the true and predicted rotations for the k th copy in the duplicated batch, respectively. As noted above, in the unsupervised case, we do not necessarily wish to regress R_i as \widehat{R}_i , because the initial (derotated) input P is not assumed to be in the canonical orientation of P_c .

Regularization Loss The primary purpose of the AE is to provide a space with reduced complexity and dimensionality, for training the generative VAE model. Following work on learning probabilistic samplers with latent-space generative autoencoders (Ghosh et al., 2019), we apply a small weight decay and latent radius loss: $\mathcal{L}_x = \gamma_w L_2(\Theta) + \gamma_d \|x_c\|_2^2$, where $L_2(\Theta)$ is an L_2 weight decay on the network parameters Θ .

3.2.2 FTL-based Loss Objective

Similar to the network functions, the FTL-AE objective terms, as well as the training regime, are largely reused from

the STD-AE. The only major difference is that we compute reconstruction losses for both the instance representation, \tilde{x} , and the canonical representation, x_c :

$$\mathcal{L}_P = \gamma_{\tilde{P}} D_P(P_r, \tilde{P}) + \gamma_P D_P(P, P_c). \quad (6)$$

Here, the decoder output $\tilde{P} = D(\tilde{x})$ is encouraged to be similar to the rotated input P_r .

We note that the penalty \mathcal{L}_c , enforcing consistency of the canonical latent shape vectors $x_{c,j}$ in the FTL architecture, ties the non-canonical embeddings, \tilde{x}_j , through an FTL operation (across rotated inputs), as follows:

$$\|x_{c,i} - x_{c,j}\|_2^2 = \|(U(\tilde{x}_i)R_i - U(\tilde{x}_j)R_j)R_i^T\|_F^2 \quad (7)$$

$$= \|U(\tilde{x}_i) - U(\tilde{x}_j)R_j R_i^T\|_F^2, \quad (8)$$

where we have used $U(x_{c,k}) = U(\tilde{x}_k)R_k$, and the orthogonality of R_k implies $\|R_k v\|_2^2 = \|v\|_2^2$ for any $v \in \mathbb{R}^3$.

4 Latent Variational Autoencoder Model

4.1 Overview

Our goal is to define a disentangled generative model of 3D shapes, using a VAE. The model should be capable of encoding for representation inference, decoding random noise for novel sample generation, and allowing factorized latent control of intrinsic shape and extrinsic non-rigid pose. The latter decomposition is made possible by use of the LBO spectrum, which allows us to separate non-rigid deformations into intrinsic shape and extrinsic (articulated or non-rigid) pose (see Sect. 2.2).

Following Ghosh et al. (2019) and Achlioptas et al. (2017), we use the AE latent space to define our generative model and disentangled representation learning. This allows us to train with much larger batch sizes (useful for information-theoretic objectives based on estimating marginal distribution properties from samples), and generally obtain better computational efficiency. See Fig. 2 for a pictographic overview.

Compared to our prior GDVAE model (Aumentado-Armstrong et al., 2019), we replace a simple predictor of the LBOS from the latent intrinsic shape with a diffeomorphic mapping between the two quantities. This allows us to use the spectrum directly in training (see Sect. 4.4) and increase the dimensionality of the latent intrinsics, improving representation performance.

4.2 Model Architecture

4.2.1 Hierarchically Factorized VAE

The core of our VAE model is the Hierarchically factorized VAE (HFVAE) model (Esmaili et al., 2018), which permits penalization of mutual information between sets of vector-valued random variables. This allows us to enforce the latent intrinsics to be separate from the latent extrinsics, specifically.

Let (q, x_c) be an encoded input from the AE. We define $z_R \sim \mathcal{N}(\mu_R(q), \Sigma_R(q))$, $z_E \sim \mathcal{N}(\mu_E(x_c), \Sigma_E(x_c))$, and $z_I \sim \mathcal{N}(\mu_I(x_c), \Sigma_I(x_c))$ to be the latent encodings of the rotation, extrinsic shape, and intrinsic shape, respectively, sampled from their variational latent posteriors. Our decoder is deterministic: $\hat{q} = D_q(z_R)$ and $\hat{x}_c = D_x(z_E, z_I)$. All three variables use isotropic Gaussians as latent priors. See Appendix F.2 for further details.

4.2.2 Normalizing Flow for Spectrum Encoding

In order to encourage z_I to hold only shape intrinsics, we utilize the LBOS. In particular, we define an *invertible* mapping between λ and μ_I . Let $\tilde{\mu}_I = f_\lambda(\lambda)$ be the latent encoding of a *real* spectrum (i.e., computed from a shape), λ , and $\hat{\lambda} = g_\lambda(\mu_I)$ be the predicted spectrum, with $g_\lambda = f_\lambda^{-1}$. We implement f_λ as a *normalizing flow* network (Papamakarios et al., 2019; Kobayev et al., 2020), defining a bijective mapping between z_I -space and the space of spectra. For VAE calculations, we use $\tilde{z}_I \sim \mathcal{N}(\tilde{\mu}_I, \tilde{\Sigma}_I(\lambda))$.

Briefly, flow networks are specialized neural modules with two general properties: (1) being a diffeomorphic mapping, and (2) having a simple analytic Jacobian determinant. These properties allow tractable exact likelihood computations through the network, via the probability chain rule through each layer (Papamakarios et al., 2019). Many architectures have been proposed with these functional properties (e.g., Kingma et al., 2016; Kingma & Dhariwal, 2018; Dinh et al., 2014; 2016) and they have been applied to generative modelling tasks in both 2D and 3D (Kingma & Dhariwal, 2018; Yang et al., 2019), as the tractable exact likelihood allows for stable training of the distribution matching loss to the prior, at the cost of requiring the dimensions of the input and output space to match and restricting the class of allowed neural architectures.

Using a flow mapping ensures that $f_\lambda(\lambda)$ can hold complete information about λ , since the learned network is guaranteed to be diffeomorphic (i.e., it is invertible and differentiable in either direction). Unlike Aumentado-Armstrong et al. (2019), this approach also allows various “shape-from-spectrum” applications (Marin et al., 2020), which we explore in Sect. 5.4.2. Thus, the flow network confers an additional benefit, which is the presence of a mapping from

λ -space to z_I -space, which allows us to define a novel training regime that prevents encouraging the network to store extrinsic information in the z_I -space for reconstruction, by instead using $\tilde{\mu}_I$ for reconstruction and pushing μ_I to match it (see Sect. 4.4). Finally, it has the benefit of being specifically designed for likelihood-based generative modelling, hence its training procedure synergizes well with the HFVAE. In particular, since we want the latent intrinsic space z_I to conform to a Gaussian prior (which we enforce with the HFVAE prior-matching losses), we also wish to ensure anything mapped from λ -space to there does as well. Fortunately, the tractable likelihood of flow networks allows us to directly optimize a prior-matching likelihood, which is not an upper-bound (unlike for VAEs). See Sect. 4.3.2 for details.

4.3 VAE Loss Function

The VAE model is trained with the following objective:

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{HF}} + \mathcal{L}_\lambda + \mathcal{L}_F + \mathcal{L}_D, \quad (9)$$

where \mathcal{L}_{HF} is the hierarchically factorized VAE loss (Esmaili et al., 2018), \mathcal{L}_λ measures the likelihood defined by the spectral flow network between spectra λ and latent intrinsics z_I , \mathcal{L}_F is a consistency loss between the VAE (mapping between x_c and z space) and the flow network, and \mathcal{L}_D is an additional disentanglement penalty. We next define the component loss functions used in this complete objective in detail. Note that we assemble two versions of this loss, expounded in Sects. 4.4.1 and 4.4.2, which differ in whether to use the latent intrinsics derived from x_c or λ .

4.3.1 HFVAE Loss \mathcal{L}_{HF}

Recall that our latent space $z = (z_R, z_E, z_I)$ is *structured*, in that we can partition it into three sub-vectors. Our goals are to (1) push z to follow an isotropic Gaussian latent prior and (2) force each component group z_g , with $g \in \{R, E, I\}$, to be independent from the other two groups, in an information-theoretic sense. Specifically, we use *total correlation* (TC), a measure of multivariate mutual information, between latent groups to optimize disentanglement (Watanabe, 1960).

Prior work on structured disentanglement (Esmaili et al., 2018) has shown that the VAE objective can be decomposed in a hierarchical fashion via

$$\begin{aligned} \mathcal{L}_{\text{HF}}[z] &= \omega_R L_R + \beta_1 \sum_g \mathcal{I}_{\text{TC}}[z_g] \\ &+ \beta_2 \sum_{d,g} \mathcal{D}_{\text{KL}}[q_\phi(z_{g,d}) \parallel P(z_{g,d})] \\ &+ \beta_3 \mathcal{I}[x_c, z] + \beta_4 \mathcal{I}_{\text{TC}}[z] \end{aligned} \quad (10)$$

where L_R denotes the reconstruction loss, the β_1 term controls the *intra*-group total correlation, the β_2 term penalizes the dimension-wise KL-divergence from the latent prior, the β_3 term controls the mutual information between x_c and z , and the β_4 term controls the *inter*-group total correlation. The latter term, $\mathcal{I}_{TC}(z)$, is the most important for our application, as it encourages statistical independence between latent intrinsics and extrinsics—this is our *disentanglement* objective.

Recall that the VAE input is a quaternion q and canonical shape vector x_c , while the output are the regressions \hat{q} and \hat{x}_c . The reconstruction loss, L_R , is written as follows:

$$L_R((q, x_c), (\hat{q}, \hat{x}_c)) = \frac{\|x_c - \hat{x}_c\|_2^2}{n \mathbb{E}[\|x_c\|_2^2]} + \omega_q d_q(q, \hat{q}), \quad (11)$$

where $n = \dim(x_c)$, the expected norm $\mathbb{E}[\|x_c\|_2^2]$ normalizes for differing AEs (making hyper-parameter setting across models easier), and $d_q(q_1, q_2) = 1 - |q_1 \cdot q_2|$ is a distance metric on rotations, through unit quaternions q_1, q_2 (Huynh, 2009).

4.3.2 Flow Likelihood Loss \mathcal{L}_λ

Since f_λ is a normalizing flow network and we want to enforce \tilde{z}_I to follow the Gaussian latent prior, we can simply use the standard likelihood objective (Kobyzev et al., 2020; Papamakarios et al., 2019):

$$P_\lambda(\lambda) = P_{z_I}(f_\lambda(\lambda)) |\det \mathcal{J}[f_\lambda](\lambda)|, \quad (12)$$

where P_{z_I} represents the density of an isotropic Gaussian (latent prior of z_I) and $\mathcal{J}[f]$ is the Jacobian of f . We use a weighted log-likelihood as the final loss: $\mathcal{L}_\lambda = -\omega_p \log P_\lambda(\lambda)$. This loss enforces \tilde{z}_I to follow the latent prior, as in most flow-based generative models. While it is similar to the HFVAE loss on z_I , it is an exact likelihood (Kobyzev et al., 2020; Papamakarios et al., 2019), rather than a lower bound. As discussed in Sect. 4.2.2, this is intuitively possible due to the use of a diffeomorphic transform, constrained to have a computationally tractable Jacobian determinant.

4.3.3 Spectral Intrinsics Consistency Loss \mathcal{L}_F

We also want the VAE encoder to be consistent with the spectral flow network, so we apply a loss between the spectral and latent intrinsic space outputs:

$$\mathcal{L}_F = \omega_I \|\mu_I - \tilde{\mu}_I\|_2^2 + \omega_\lambda d_\lambda(\lambda, \hat{\lambda}). \quad (13)$$

$\tilde{\mu}_I = f_\lambda(\lambda)$, $\hat{\lambda} = g_\lambda(\mu_I)$, and d_λ is a weighted distance between spectra (Aumentado-Armstrong et al., 2019),

$$d_\lambda(\lambda, \hat{\lambda}) = \frac{1}{N_\lambda} \sum_{n=1}^{N_\lambda} \frac{|\lambda_n - \hat{\lambda}_n|}{n}, \quad (14)$$

where N_λ is the number of elements used in the spectrum. This formulation is inspired by Weyl’s estimate (Reuter et al., 2006; Weyl, 1911), which posits approximately linear eigenvalue growth asymptotically. The motivation is to avoid overweighing the higher elements of the spectrum (corresponding to higher geometric frequencies and thus noisier, small-scale shape details). See also Cosmo et al. (2019). Note that this does not assume a particular structure for the LBO, nor for the growth of its eigenvalues; rather, it is a heuristic for reducing the effect of the monotonic growth of λ (i.e., non-linear growth will simply change the relative importance of the frequencies in the loss).

4.3.4 Additional Disentanglement Losses \mathcal{L}_D

Following Aumentado-Armstrong et al. (2019), we utilize two additional losses to promote disentanglement. The first is motivated by Kumar et al. (2017), penalizing the covariance between latent groups:

$$\mathcal{L}_\Sigma = \sum_{g \neq \tilde{g}} \sum_{i,j} |\hat{\Sigma}[\mu_g, \mu_{\tilde{g}}]_{i,j}|, \quad (15)$$

where $\hat{\Sigma}$ is the empirical covariance matrix between latent vectors, computed per batch, and $g, \tilde{g} \in \{R, E, I\}$. The second takes advantage of the differentiable nature of the networks involved, directly penalizing the rate of change in the intrinsics as the extrinsics are varied (and vice versa). This is implemented as a penalty on the Jacobian between latent groups

$$\mathcal{L}_\mathcal{J} = \left\| \frac{\partial \hat{\mu}_E}{\partial \mu_I} \right\|_F^2 + \left\| \frac{\partial \hat{\mu}_I}{\partial \mu_E} \right\|_F^2, \quad (16)$$

where $\hat{\mu}_g = \mu_g(\hat{x}_c)$ is the re-encoding of the reconstructed shape from the AE, $\hat{x}_c = D_x(z_E, z_I)$, such that $\frac{\partial \hat{\mu}_g}{\partial \mu_{\tilde{g}}} = \frac{\partial \hat{\mu}_g}{\partial \hat{x}_c} \frac{\partial \hat{x}_c}{\partial \mu_{\tilde{g}}}$ for $g \neq \tilde{g}$ and μ_g is the approximate posterior mean from which z_g is sampled. Hence, the final loss term is given by $\mathcal{L}_D = \omega_\Sigma \mathcal{L}_\Sigma + \omega_\mathcal{J} \mathcal{L}_\mathcal{J}$.

4.4 Training Regimes

We consider two methods of training, which differ in the manner in which the latent variables are obtained at training time. The first is similar to the original Geometrically Disentangled

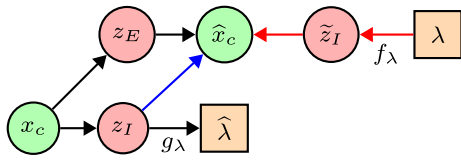


Fig. 8 Diagram of VAE mappings, depicting the ability to use latent intrinsics derived from x_c versus λ . Black lines indicate mappings always run in training. The blue arrow is used when predicting the latent reconstruction from x_c rather than λ , which is useful at inference time (when λ may not be known) and for the GDVAE-FO training scheme (see Sect. 4.4). The red arrows depict using latent intrinsics directly procured from the LBOS λ , as in the GDVAE++ training scheme (Color figure online)

VAE (GDVAE) model, where z_I is used for reconstruction and predicting the spectrum. This is the “flow-only” (FO) model. The second takes advantage of the shape-from-spectrum capabilities of the bijective flow mapping, using $\tilde{\mu}_I = f_\lambda(\lambda)$ for reconstruction (which does not depend on x_c), and encouraging $\mu_I(x_c)$ to be close to $\tilde{\mu}_I$. We refer to these models as **GDVAE-FO** and **GDVAE++**, respectively. Notice that the latter approach more stringently separates extrinsics and intrinsics, as the decoder has more limited access to extrinsics from $f_\lambda(\lambda)$, as opposed to using x_c . We visualize the two pathways in Fig. 8. Notice that the two training regimes do not differ in their architecture, hyperparameters, and structure of the forward pass at inference, but only in the structure of the forward pass at training time.

4.4.1 GDVAE-FO Loss

The “flow-only” model is most similar to the prior GDVAE model (Aumentado-Armstrong et al., 2019). We want the encoded intrinsic shape vector $\mu_I(x_c)$ to hold as much information as possible about the spectrum. This is accomplished through the diffeomorphic mapping to λ and the spectral losses in \mathcal{L}_F . In other words, we reconstruct via $\hat{x}_c = D_x(z_E, z_I)$ and $\hat{\lambda} = g_\lambda(\mu_I)$. The disentanglement losses \mathcal{L}_{HF} and \mathcal{L}_D are computed with μ_I .

4.4.2 GDVAE ++ Loss

For the GDVAE++ loss, we use the known spectrum to compute the output latent shape. The idea during training is to enforce the latent intrinsics used for reconstruction (in this case, \tilde{z}_I) to *only* hold intrinsic geometry (using $f_\lambda(\lambda)$), and push z_I (inferred from x_c) to be close to it. Thus, $\hat{x}_c = D_x(z_E, \tilde{z}_I)$ is used for reconstruction, where $\tilde{z}_I \sim \mathcal{N}(\tilde{\mu}_I = f_\lambda(\lambda), \tilde{\Sigma}_I(\lambda))$. In addition, the disentanglement losses \mathcal{L}_{HF} and \mathcal{L}_D are computed with $\tilde{\mu}_I$. Note that this training strategy does not preclude us from processing shapes *without* spectra at test time, which we do for our evaluations.

5 Experimental Results

5.1 Datasets

We use the same datasets as in Aumentado-Armstrong et al. (2019). Specifically, we consider MNIST (LeCun et al., 1998), SMAL (Zuffi et al., 2017), and SMPL (Loper et al., 2015). We also assemble a Human-Animal (HA) mixed dataset by combining data from SMAL and SMPL. Note that, in all cases, we perform a scalar rescaling of the dataset such that the largest bounding box length is scaled down to unit length. This scale is the same across PCs (otherwise the change in scale would affect the spectrum for each shape differently). We apply random rotations about the gravity axis (SMAL and SMPL) or the out-of-image axis (MNIST). For rotation supervision, the orientation of the raw data is treated as canonical. We also remark that we use LBOSs derived from the mesh shape, rather than PCs, unless otherwise specified. See Appendix D for additional dataset details.

5.2 Autoencoder Results

Our AE is designed to factorize out rigid pose, as well as encode a complete representation of a canonical shape. In Fig. 9, we show example reconstructions, as well as the canonicalization capability of the model. In Fig. 11, we show latent embeddings of the shape representations x_c across different rotations of input shapes. The results show that the AE is not only able to accurately reconstruct the inputs, but also correctly derotate the canonical PCs in 3D, and that the encodings are close to being orientation invariant in the latent space.

We consider two AE types, the STD and FTL models with their differing rotation handling techniques. We also examine two ablations: the unsupervised (U) scenario, which removes the assumption of aligned data, and the HA-trained model, which eliminates the use of specialized single models for SMAL and SMPL.

Quantitatively, we evaluate our autoencoders on (1) reconstruction capability and (2) rotation invariance in their representation. Reconstruction quality is computed with the standard Chamfer distance between the output PC and a uniform random sampling from the raw shape mesh. We average over five randomly rotated copies of the test set.

Rotation invariance is assessed with two measures. The first is in 3D space, and checks that canonicalizations of the same PC under different rotations are close (according to the Chamfer distance between PCs):

$$\mathcal{C}_{3D} = \frac{1}{M_R} \sum_{i=1}^{m_R} \sum_{j \neq i} d_C(D(E(P_i)), P_j), \quad (17)$$

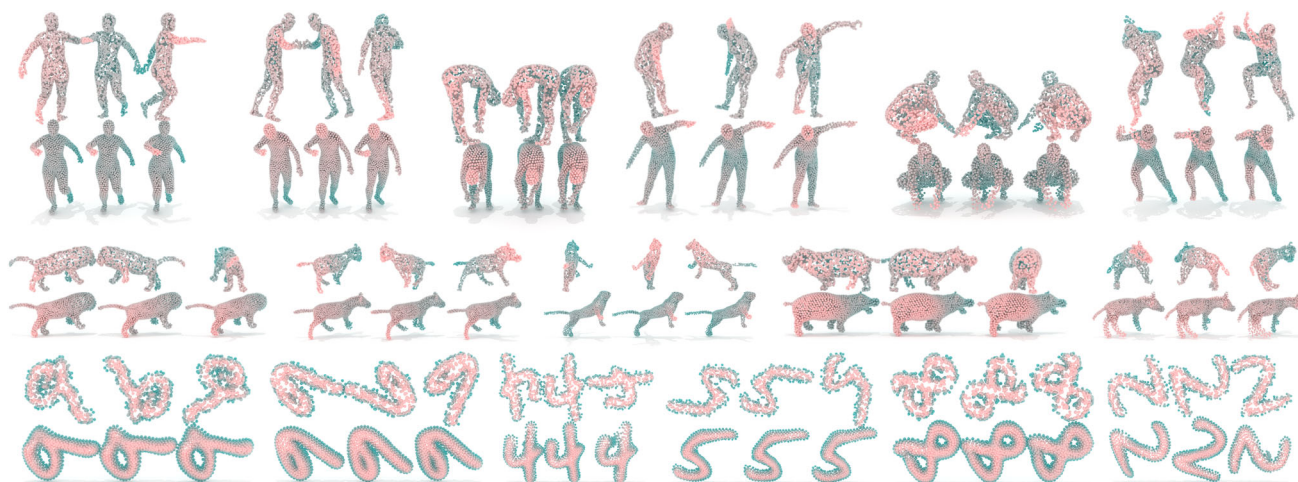


Fig. 9 Example AE reconstructions and rigid canonicalizations of input point clouds. Per inset, the top row shows input point clouds P under random rotations, while the bottom row displays the resulting canonical decodings P_c . Colors indicate depth. Architectures used are SMPL-FTL-S, SMAL-FTL-S, and MNIST-FTL-U, respectively (see Table 1). Since MNIST is unsupervised, we rotate the canonical output by a constant rotation for visualization purposes. In particular, notice that the first two MNIST insets (“9” and “6”) are rotated to match (which would be

incorrect in the supervised case). We show some failure cases in the last column of each row: for SMPL, the pose (e.g., arms) is incorrectly reconstructed; for SMAL, the hind-legs of the third canonical PC does not match its counterparts, though the overall pose does; for MNIST, the canonical decoding simply fails to match across rotations. Note that we show additional reconstructions (through the VAE) in Fig. 10 (Color figure online)

Table 1 AE evaluation on held-out test data

Dataset	Model	$d_C(P, \hat{P}) \downarrow$	$C_{3D} \downarrow$	$C_X \uparrow$
MNIST	STD-U	1.19	1.57	0.92
	FTL-U	0.94	2.73	0.65
SMAL	STD-S	0.35	0.03	0.97
	FTL-S	0.10	0.14	0.93
	STD-U	0.29	0.01	0.97
SMPL	FTL-U	0.10	0.21	0.88
	STD-S	0.34	0.03	0.97
	FTL-S	0.19	0.30	0.71
HA	STD-U	0.23	0.05	0.97
	FTL-U	0.18	0.45	0.70
	STD-S	0.36/0.44	0.03/0.05	0.97/0.97
	FTL-S	0.11/0.19	0.24/0.22	0.66/0.62
	STD-U	0.33/0.34	0.02/0.06	0.97/0.97
	FTL-U	0.11/0.19	0.20/0.20	0.72/0.66

Metrics (left to right) refer to the Chamfer distance in reconstructions and the rotational consistency measures (in 3D and x_c -space, respectively). HA is the humans and animals dataset (SMPL+SMAL). For each model, STD and FTL refer to the type of AE architecture, and U and S denote the use (S) or lack of (U) rotational supervision. For the HA dataset, a/b denote the values on the SMAL and SMPL test sets, respectively. Also, note that, for HA, SMPL shapes are scaled with the SMAL maximum bounding box length; therefore, we scale the Chamfer distances in the evaluations to match the other SMPL models, to make them comparable. \uparrow (\downarrow) means the higher (lower) the better

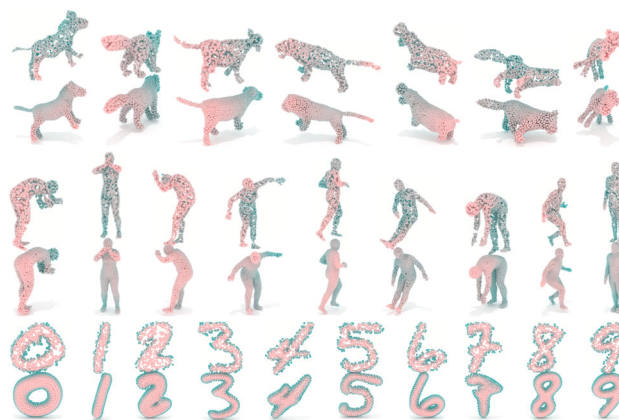


Fig. 10 Reconstructions through the VAE. Odd rows are inputs; even rows are reconstructions. Since the model is learned in x_c -space, reconstruction error manifests as shapes being valid, but slightly off in shape and/or pose (e.g., left-most and right-most insets of top row). All models use the FTL-based AE. See Fig. 9 for qualitative example reconstructions through only the AE. Note that inputs and outputs have the same number of points

where m_R is the number of random copies we use for evaluation and M_R is the number of pairs tested.

The second measure is in the latent canonical shape space (i.e., x_c). Since latent distances are less meaningful (e.g., dimensions may have very different scales) and will differ across AEs, we choose to measure performance by *clustering quality*. Ideally, a representation that canonicalizes an input shape should map rotated copies of a given PC to the

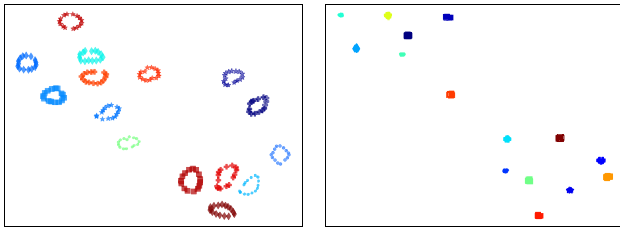


Fig. 11 Latent rotational invariance via t-sne plots of x_c vector embeddings. Plots are done with the HA-FTL-U (left) and HA-STD-U (right) models, respectively (see Table 1). Colors denote a single shape; markers with the same color are rotations of that single shape. Marker types are only meant to help tell apart shapes with similar colors. Notice the non-FTL (STD) architecture gives a tighter latent invariance (Color figure online)

same latent encoding—exactly fulfilling this would make it *rotation invariant*. Hence, we create rotated copies of many input shapes, encode them, and then cluster in the AE embedding space. We expect that rotated copies of the same instance should cluster together; hence, we treat instance identity as a ground truth cluster label and use Adjusted Mutual Information (AMI) to measure quality (Vinh et al., 2010). An AMI of 1 indicates perfect matching of the predicted and real partitions, while an AMI of 0 is the expected value of a random clustering. We average AMIs over clusterings obtained from different random sample sizes (i.e., the number of unique shapes duplicated and clustered). The resulting “area-under-the-curve”-like latent space clustering metric for rotational invariance is denoted C_X . See Appendix C.1 for additional details.

The original GDVAE model (Aumentado-Armstrong et al., 2019) was trained on limited angles of rotation about the canonical one, since otherwise reconstruction quality was degraded but in this work we always consider full rotation about a single axis. Despite the fact that two models use essentially the same architectural components, our AE is better able to obtain canonical orientations, while maintaining reconstruction quality.

The results in Table 1 show a few patterns between the AE types.² First, we find that the FTL-based AE has superior reconstruction quality, while the STD AE has much better rotation invariance. Second, the difference between the unsupervised and supervised scenarios is relatively smaller, with the unsupervised reconstruction quality being slightly better than the supervised, whereas the supervised case has superior rotation invariance. Finally, performance on the HA dataset (which is a union of the SMAL and SMPL data) is only slightly degraded compared to the per-category models (more so for FTL than STD).

² We remark that these results utilize single-axis (planar) rotations; we refer the reader to Appendix G for tests with full rotations, which results in reduced rotational robustness.



Fig. 12 Random sample generations from the VAE. All models use the FTL-based AE. Note that the MNIST model was trained on digits at all orientations and thus should output samples at any orientation (as for SMAL and SMPL). Rows: SMAL, SMPL, MNIST, and HA

5.2.1 Results Summary

Since the FTL-based AE maintains strong rotation invariance, with superior latent interpretability and reconstruction error, we suggest using it as a starting point. We also find that rotation factorization can be done without aligned data supervision, at little cost to reconstruction or rotational invariance quality.

5.3 Latent Variational Autoencoder Results

We evaluate our VAE model on three main criteria: (1) representational fidelity, (2) generative modeling, and (3) intrinsic-extrinsic disentanglement. Representational fidelity is captured simply as the reconstruction error, measured via the Chamfer distance between input and output (see Fig. 10 for qualitative examples). To assess generative modeling capability, we utilize the coverage and fidelity metrics (Achlioptas et al., 2017), which examine how well samples from our VAE represent a held-out test set. In addition, utilizing our flow network, we can measure the quality of spectrum generation using the standard log-likelihood. Finally, using the known ground truth intrinsics and extrinsics of our synthetic SMAL and SMPL data, we can measure disentanglement quality via a pose-aware retrieval task. We discuss our results and the details of these metrics in the following sections. Figures 15, 16, and 17 (as well as Appendix Table 5) show our quantitative results on metrics for all of these criteria.

We explore two variants of our model, using the STD and FTL AEs, as well as several ablations. Two ablations involve the AE: removing rotational supervision (the “S” vs. “U” models) and using only *one* model for both SMAL and SMPL (via the HA dataset), as opposed to having specialized models for each. Note that the latter scenario not only increases data complexity without altering model capacity, but it also removes some regularities that are present in the independent datasets due to their restricted categories. The

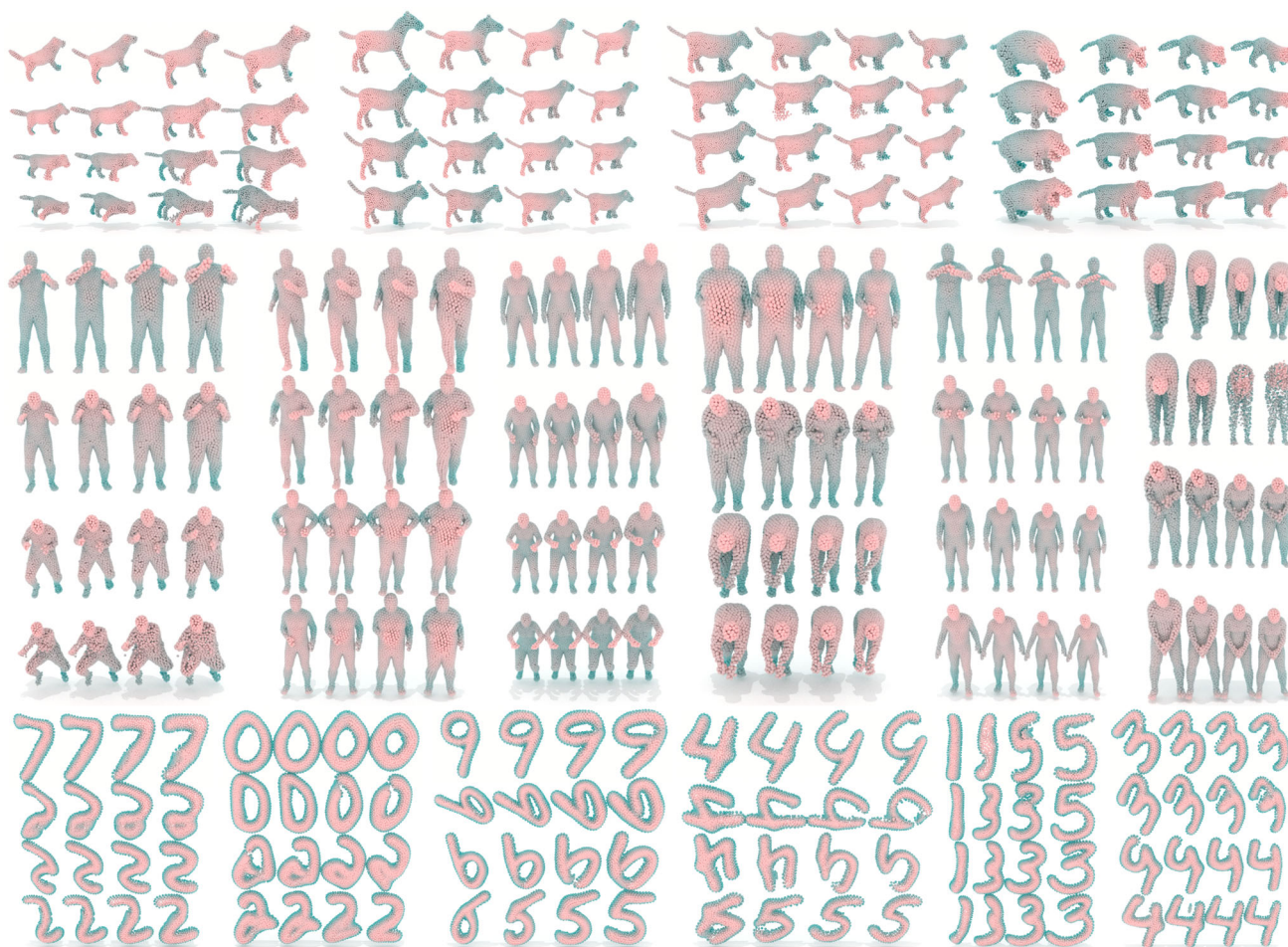


Fig. 13 Intrinsic-extrinsic latent space interpolations using the disentangled VAE representation. In each 4×4 panel, top-left and bottom-right shapes are reconstructions of real inputs. Along the horizontal axis, we interpolate along the intrinsics (z_I), whereas we do so for the extrinsics (z_E) along the vertical axis. Notice that the bottom-left and top-right shapes are *pose transfers*, for which one of the latent

factors is exchanged, while the other remains unchanged. SMAL and SMPL shapes are shown in the learned canonical orientation (using FTL-S); for MNIST only (using FTL-U), we interpolate the estimated AE rotation encodings q as well, via sleeping the quaternions (Shoemaker, 1985) between the two inputs along with the extrinsics (i.e., along the vertical axis)

remaining ablations affect only the VAE: using a PC-derived LBO (rather than the mesh-derived one we use by default), altering our algorithm to not use the spectrum-derived latent intrinsics in training (GDVAE-FO), and removing the additional disentanglement loss \mathcal{L}_D (see Sect. 4.3.4).

5.3.1 Generative Modeling

We measure generative modeling quality using the metrics introduced by Achlioptas et al. (2017). Consider two sets of PC shapes: ζ_G , a random set of generated samples, and ζ_R , a set of real PCs. Note that generations are computed via $P = D(D_x(z_E, z_I))R(D_q(z_R)) \in \zeta_G$, where $z_R, z_E, z_I \sim \mathcal{N}(0, I)$ (see Sects. 3.1 and 4.2.1). Briefly, we consider two measures: *coverage*, which checks how well ζ_G covers the modes of ζ_R (a proxy for set diversity), and

fidelity, which considers how faithful each element in ζ_G is to its closest counterpart in ζ_R (a proxy for per-element realism). Coverage is computed by matching each $Q \in \zeta_G$ to its closest PC in ζ_R , and counting the percent of PCs chosen (matched) in ζ_R (high coverage meaning most of the PCs in ζ_R are represented in ζ_G). Fidelity (also called minimum matching distance) is computed by matching each $P \in \zeta_R$ to its closest pair in ζ_G , taking the Chamfer distance between them, and averaging these distances over the dataset. Fidelity is needed because coverage does not measure the quality of the matchings (e.g., low quality PCs could be used to cover a given real PC). Matching is always computed as the minimum Chamfer distance. Similar to Achlioptas et al. (2017), we generate a synthetic set five times larger than the held-out test set, and report the average of running the same evaluation twice. See Fig. 15 for a plot of generative metrics



Fig. 14 Example retrievals using the disentangled factors of the GDVAE++ latent representation. Shapes are in order of latent similarity from left to right). SMAL, SMPL, and MNIST use FTL-S, FTL-S, and FTL-U AE models, respectively. For SMAL and SMPL, notice that z_E retrieves a variety of animals/body types in the same articulated pose, while z_I retrieves the same animal/body type in an array of different non-rigid poses. Though MNIST does not have a natural sense of articulation, notice that the extrinsic retrievals tend to have the

same digit identity, but vary most noticeably in thickness (which is a non-isometric alteration). In contrast, digits retrieved via z_I appear to be bent largely isometrically; that is, “wiggled” around in a way that preserves the metric tensor (and the distribution of geodesic distances among points)—see the “8” digit. For the “6”, notice that it retrieves several “9” digits, showing its blindness to orientation, as well as two “5” shapes that were “thickened” sufficiently in a manner similarly to the query

and Appendix Table 5 for quantitative scores. For qualitative visualizations, random sample generations are shown in Fig. 12.

Separately, our flow model f_λ provides a generative model on LBOSSs. Using its bijectivity, we can directly compute the log-likelihood (shown in Fig. 18 and Appendix Table 5). This measures how well our spectral encoder maps real spectra into the Gaussian latent space of the intrinsics.

Looking at Figs. 15 and 18 (as well as Appendix Table 5), we can see that the GDVAE++ and GDVAE-FO score similarly for generative fidelity and coverage, and obtain mixed results on $\log P_\lambda(\lambda)$ (the FO method performs better or similar with the FTL AE, but worse with the STD AE), but GDVAE-FO always has better reconstruction results. In terms of AE type, results are mixed, though the FTL approach does tend to have slightly better coverage and worse fidelity.

We discuss results related to disentanglement quality in the next subsection.

5.3.2 Shape-Pose Disentanglement

To measure disentanglement quantitatively, we rely on a *pose-aware* retrieval task in which ground truth continuous values for intrinsics and extrinsics are known.

We start with a set of shapes (SMAL or SMPL) for which parameters for intrinsic shape β and extrinsic pose θ are known. These shapes *were not* used in training. Let point cloud P_i have parameters (β_i, θ_i) . Using our model, we encode P_i into a latent representation $\rho_i \in \{x_c(P_i), z(P_i), z_E(P_i), z_I(P_i)\}$. We then measure distances between representations as $d_\rho(P_i, P_j) = \|\rho_i - \rho_j\|_2^2$, and rank the retrieved shapes based on ρ_i . We measure the disentangled retrieval quality for a retrieved PC, P_j , using query

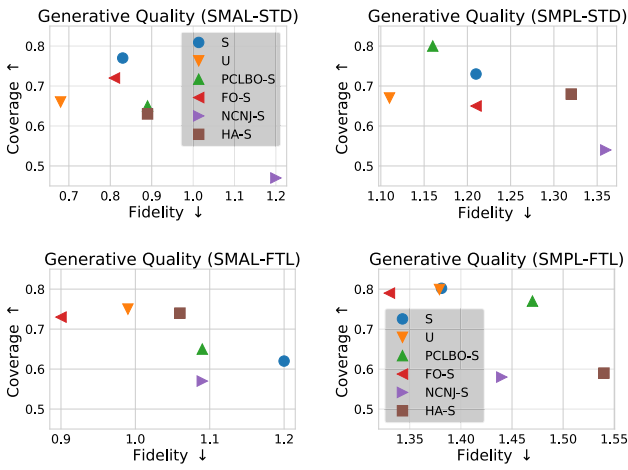


Fig. 15 VAE generative modelling evaluation scores on SMAL (left) and SMPL (right) with the FTL AE architecture. Notationally, “S”/“U” refers to the supervised/unsupervised AE settings (see Sect. 3.2.1), PCLBO refers to the LBOS being procured from a point cloud (see Sect. 5.3.3), NCNJ refers to our disentanglement ablation (see Sect. 5.3.4), and HA refers to a model trained on both SMAL and SMPL at the same time (see Sect. 5.3). All methods use the GDVAE++ setting (see Sect. 4.4), except for the FO case. The upper-left corner of the plots is preferred. See Appendix Table 5 for detailed values. Notice that NCNJ tends to have poor generative quality, while the GDVAE++ (S or U) and GDVAE-FO generally perform well. In most cases, HA underperforms the GDVAE++ (except for the SMAL-FTL case), likely due to the additional complexity of the dataset straining model capacity

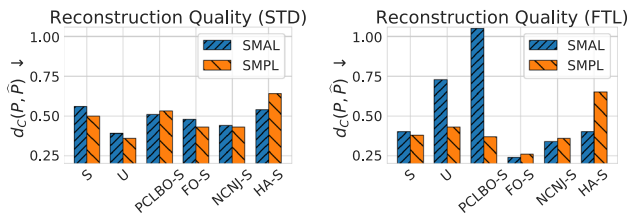


Fig. 16 VAE reconstruction quality evaluation, measured in Chamfer distance (lower is better), using the STD (left) and FTL (right) architectures. See Fig. 15 for explanation of model types and Appendix Table 5 for detailed values. We find that the PCLBO and HA models are similar or worse, while NCNJ and FO are similar or slightly better, compared to GDVAE++ (S). Note that NCNJ can take advantage of the weaker disentanglement requirements, while the FO case simply fails to disentangle (see Fig. 17)

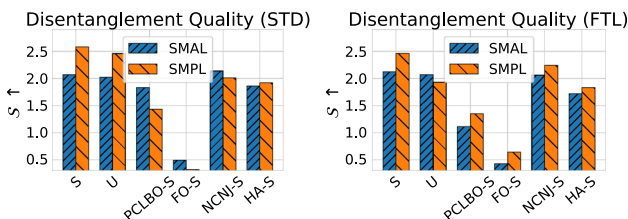


Fig. 17 VAE disentanglement quality evaluation (see Sect. 5.3.2 and Eq. 18 for discussion of the disentanglement score \mathcal{S}). See Fig. 15 for explanation of model types and Appendix Table 5 for detailed values. Compared to the regular GDVAE++, the flow-only (FO) case is severely degraded, whereas PCLBO, HA, and NCNJ experience moderate deterioration (for the latter case, moreso on SMPL than SMAL)

P_i , by separately checking how well the intrinsic shape and non-rigid pose match. This is done by comparing the query ground truth parameters, (β_i, θ_i) , to (β_j, θ_j) , from the retrieved shape.

We compute the distance between these parameters, as the mean squared error between β values and the average rotational distance d_q across corresponding joint rotations, denoted E_β and E_θ , respectively. Note that we normalize E_β and E_θ by the mean pairwise error across the dataset for each measure, so that it is relative to the expected error of a *uniformly random* retrieval algorithm (1 corresponds to random retrieval, while 0 implies obtaining the same parameter set). More specifically, we use the encoding(s) ρ of a PC P , to retrieve the three closest shapes (in terms of d_ρ), and compute the errors E_β and E_θ averaged over these three retrievals, to obtain two errors per shape. For a fixed encoding type $\rho \in \{x_c(P), z(P), z_E(P), z_I(P)\}$, we get a final error by averaging over an entire held-out test set. Hence, we obtain two scalars E_β and E_θ for each choice of ρ .

We then convert these errors into scores, $s_\psi(\rho) = 1 - E_\psi(\rho)$, where $\psi \in \{\theta, \beta\}$. We expect using z_E for retrieval (i.e., as ρ) to result in a *high* intrinsic error $E_\beta(z_E)$ (*low* score $s_\beta(z_E)$), but a *low* extrinsic error $E_\theta(z_E)$ (*high* score $s_\theta(z_E)$). Using z_I should result in the converse: a *high* intrinsic score $s_\beta(z_I)$ and a *low* extrinsic score $s_\theta(z_I)$. We expect retrieval with x_c or z to obtain high scores for both parameters.

Lastly, we wish to have a final scalar score that expresses the quality of disentanglement obtained by the model. Notice that s_β and s_θ are normalized with respect to a random retriever, but are still not comparable (as the errors are originally different units and at different scales). Hence we compute $\widehat{s}_\psi(z_g) = s_\psi(z_g)/s_\psi(x_c)$, with $z_g \in \{z, z_E, z_I\}$, normalizing beta and theta retrievals to be in approximately the “same” units (both are errors relative to the AE).

With these normalizations, we make the following interpretations: $\widehat{s}_\psi(z_g) = 0$ means that using z_g to retrieve shapes is no better (with respect to ψ) than random retrieval. In contrast, $\widehat{s}_\psi(z_g) = 1$ implies that z_g performs just as well as using x_c ; this comparison is relevant, because the AE limits the amount of information available to the VAE. Higher scores (e.g., $\widehat{s}_\psi(z_g) = 2$) imply that z_g performs $\widehat{s}_\psi \times$ better than x_c (specifically for retrieving pose alone, when $\psi = \theta$, or intrinsic shape alone, when $\psi = \beta$).

Our normalized retrieval scores $\widehat{s}_\psi(z_g)$ are then used to compute a final disentanglement scalar

$$\mathcal{S} = \widehat{s}_\beta(z_I) + \widehat{s}_\theta(z_E) - \widehat{s}_\beta(z_E) - \widehat{s}_\theta(z_I). \tag{18}$$

Higher \mathcal{S} requires higher accurate extrinsics-based retrieval in terms of pose (high $\widehat{s}_\theta(z_E)$), but poor retrieval (when using z_E) with respect to intrinsic β (low $\widehat{s}_\beta(z_E)$); at the same time, it requires the opposite performance for the latent intrinsic z_I . Note that random retrieval performance results in all

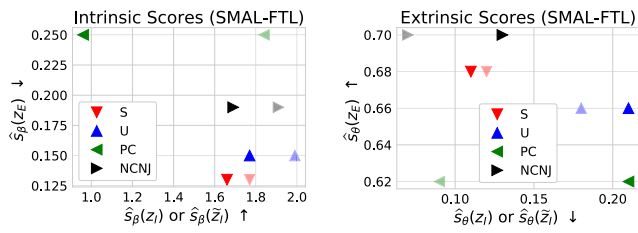


Fig. 18 Pose-aware retrieval scores with the FTL AE model. Model notation refers to the GDVAE++ model with (S) or without (U) rotation supervision, use of the PC-derived LBOS (PC; see Sect. 5.3.3), and the partial disentanglement loss ablation (NCNJ; see Sect. 5.3.4). The lighter (partially transparent) counterparts of each point corresponds to using $\tilde{z}_I = f_\lambda(\lambda)$ instead of z_I for retrieval. The leftmost two insets show results with SMAL, while the rightmost two do so for SMPL; for each set of two, left and right correspond to scores in intrinsic (\hat{s}_β) and extrinsic (\hat{s}_θ) retrieval, respectively. Preferred values lie in the bottom-right for intrinsic scores and top-left for extrinsic scores. See Appendix

terms being zero (hence $\mathcal{S} = 0$); however, one *also* obtains $\mathcal{S} = 0$ if performance for each term is the same as the AE (since all four terms would be one). In other words, good performance retrieving intrinsics (extrinsics) with z_I (z_E) will be cancelled out by good performance retrieving extrinsics (intrinsics) with z_I (z_E). This shows that a high \mathcal{S} requires disentanglement between z_E and z_I .

Disentanglement scores are shown in Fig. 17 (as well as Appendix Table 5). Note that retrieval scores \mathcal{S} are 1.08 and 1.04, for SMAL and SMPL respectively, in the original GDVAE work. As such, the GDVAE++ model obtains significantly superior disentanglement scores across both datasets (including from the HA model)—around double the score of the original model.

From Fig. 18 (as well as Appendix Table 6 and Fig. 22), we also observe the superiority of $\tilde{z}_I = f_\lambda(\lambda)$ over z_I in retrieving intrinsics, suggesting one should use the spectrum directly when it is available for such a task, though the raw spectrum λ cannot be used for other tasks (e.g., smooth interpolation, generation, or same-pose-different-shape retrieval).

Qualitatively, we can assess disentanglement by looking at interpolations within the factorized latent space (shown in Fig. 13). The interpolation plots also show examples of *pose transfers* (upper-right and lower left corners per inset). For SMAL and SMPL, one can see that the network correctly disentangles articulated pose and shape. For MNIST, where an obvious notion of articulation is not present, moving in z_I tends to change digit thickness or allow large-scale shape alterations, while changing z_E approximately leaves geodesic distance distributions unchanged (though it can change major factors, like topology).

We can also consider the retrievals qualitatively based on the disentangled latent vectors. Figure 14 shows what shapes the networks think are most similar to each query, in terms of intrinsics versus extrinsics. We observe that z_E is able to

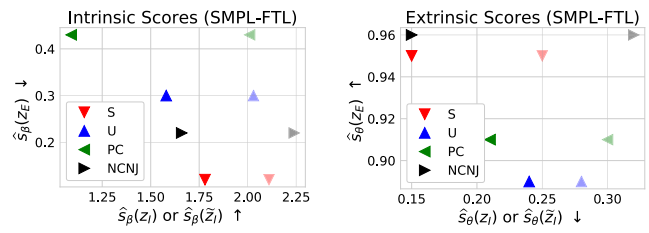


Fig. 22 for plots with the STD AE and Appendix Table 6 for detailed values. We see that (i) using λ -derived (rather than mesh-derived) latents is consistently better for intrinsic retrieval, but more mixed for extrinsic retrieval (particularly for SMAL); (ii) the PCLBO struggles more on intrinsic than extrinsic retrieval scores (see also Sect. 5.3.3); and (iii) removing disentanglement losses (NCNJ) increases the intrinsic score on the *extrinsics* $\hat{s}_\beta(z_E)$ (meaning more intrinsic information is erroneously in z_E), but allows for a modest improvement in extrinsic score $s_\theta(z_E)$, potentially due to the weaker constraints on the latent representation

retrieve very similar articulations across many animals and/or human body types, while z_I correctly retrieves similar shapes without regard for non-rigid pose. For MNIST, retrieval with z_E tends to mostly return the same digit with differing thicknesses, while retrieval with z_I also largely results in the same digit, but under isometric (non-geodesic-altering) deformations. There are some exceptions to these, such as the nines retrieved by the z_I from the six (as the spectrum is unaffected by rotation) or the fives there (potentially due to the closeness of the end of the last stroke in the five to the upper portion of the digit, as well as its thickness, leading to greater intrinsic similarity). The ones retrieved for the eight by z_E are less obvious to interpret; they may be due to the low dimensionality of z_E or the similarity of ones to thin eights.

We conclude by noting that the GDVAE++ (S or U) generally has the best disentanglement scores (see Fig. 17), while NCNJ has the second-best, but suffers from worse generative quality (Figs. 15 and 19). In comparison, the HA and PCLBO models are generally slightly worse across all metrics (generation, reconstruction, and disentanglement). The FO scenario has by far the worst disentanglement score among all models, underscoring the importance of our altered training regime. While there is some noise (e.g., higher reconstruction error for SMAL-FTL-U in Fig. 16 or superior generative quality for HA on SMAL-FTL in Fig. 15), these trends broadly hold across datasets and AE types (STD and FTL), suggesting our new approach is generally better.

5.3.3 Spectral Robustness

Although we use PCs as our shape representation for these experiments, our spectra are computed on the mesh forms of the shapes, via the cotan weight formulation (Meyer et al., 2003). This provides a useful measure of performance for our model (effectively bounding the performance we can expect

with lower quality LBOs), as well as allowing comparison to the original GDVAE model, upon which we are trying to improve. Further, we expect methods for LBOS extraction from PCs to improve over time (e.g., via advances in machine learning (Marin et al., 2021) and geometry processing (Sharp et al., 2021)), making the use of higher quality operators more feasible.

However, for completeness, we also investigated the effect of computing the spectra directly on our subsampled point clouds. This mesh-to-point-cloud conversion process introduces several additional sources of noise: for instance, parts far in geodesic distance may be close in Euclidean space (altering the LBO), and the subsampling of the surface (our PCs being smaller than the number of vertices in SMPL and SMAL) also introduces noise. Hence, we expect results to be degraded, compared to the prior section. For computing the point cloud LBO (PCLBO), we use the robust “tufted” Laplacian operator (Sharp & Crane, 2020).

The scalar disentanglement results are shown in Fig. 17 and Appendix Table 5. While the scores do decrease overall, they are still superior to the scores from the original GDVAE (which used mesh-derived LBOSs to obtain 1.08 and 1.04, for SMAL and SMPL respectively) and the GDVAE-FO models. From Fig. 18 (as well as Appendix Table 6 and Fig. 22), we see that two major terms are negatively affected in the PCLBO case, likely due to noise in the estimated LBOSs: (1) the ability of z_I to capture intrinsics degrades, indicated by the decline in $\hat{\sigma}_\beta(z_I)$, scores; and (2) intrinsic information is not removed as effectively from z_E , indicated by high values of $\hat{\sigma}_\beta(z_E)$ (especially for SMPL).

5.3.4 Ablations

Lastly, we consider the effect of ablating two aspects of the model: the additional disentanglement loss \mathcal{L}_D and the shape-from-spectrum reconstruction used in the GDVAE++ training.

First, we investigate the utility of the additional disentanglement penalties. By removing these losses, we have no covariance and no Jacobian terms; we denote this scenario NCNJ. For SMAL, the disentanglement scores seem unaffected by this ablation; however, it seems to have introduced a trade-off between reconstruction and generative modelling errors, with d_C improving (see Fig. 16), but coverage and $\log P_\lambda(\lambda)$ degrading (see Figs. 15 and 19). For SMPL, NCNJ results in degradations in the disentanglement and generative coverage scores (see Figs. 15 and 17). Note that since the VAE prior is Gaussian, it presupposes latent independence (Higgins et al., 2017); hence, disentanglement is likely to affect the prior fitting (and hence generative quality and $\log P_\lambda(\lambda)$ as well).

Second, we look at the effectiveness of the “flow-only” training approach, where we do not perform latent shape-

from-spectrum during training to perform reconstruction, and instead only use the direct encoding of the AE output. We find that this incurs the most significant degradation in terms of disentanglement score across both datasets (see Fig. 17), showing the importance of using the uncontaminated spectrum for training, rather than relying on the \mathcal{L}_F to force z_I to carry only intrinsic information. One may notice that, even though GDVAE-FO is similar to the GDVAE model (Aumentado-Armstrong et al., 2019),³ it has a much lower disentanglement score. This can be partly explained by the increase in dimensionality of the latent intrinsics, as the newer model has a 4–5 times larger $\dim(z_I)$ than the original GDVAE, making disentanglement more difficult.

5.3.5 Results Summary

The GDVAE++ shows substantial improvements over the original GDVAE model in terms of disentanglement. Using the PCLBO or the combined model (HA dataset) ablations decrease performance, but still maintain this advantage. This improvement also holds regardless of AE type or whether rotational supervision is ablated, showcasing the robustness of our model to AE settings. Much of this gain stems from our shape-from-spectrum training regime: when ablated (the GDVAE-FO model), disentanglement capabilities are crippled.

5.4 Mesh Experiments

The previous results demonstrated the improvements of our approach over the prior GDVAE model. To illustrate applicability to a different 3D shape modality, as well as facilitate comparison to other works, we also tested our method on mesh data.

5.4.1 Human Bodies (AMASS)

First, we utilize the AMASS dataset (Mahmood et al., 2019), which combines a number of human motion datasets and provides parametric fitting via SMPL, in order to compare with “Unsupervised Shape and Pose Disentanglement for 3D Meshes” (USPD) (Zhou et al., 2020) on disentangled retrieval and pose transfer tasks.

We alter the AE to (1) process a mesh input, instead of a PC, and (2) output ordered vertex coordinates instead of arbitrary PC sample points. Following other work (Marin et al., 2021; Tan et al., 2018), we use a fully connected encoder. Each output position of the decoder is now semantically associated to a fixed vertex. We alter the loss function to use vertex-to-vertex mean squared error for reconstruction, rather than Eq. (2) (with other terms remaining the

³ Except for the flow network and altered AEs.

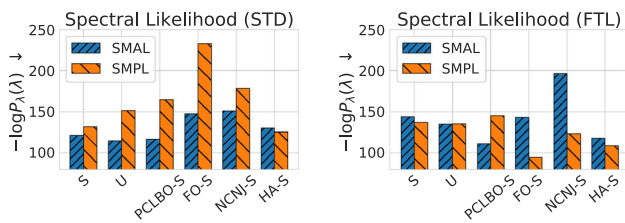


Fig. 19 VAE spectral negative log-likelihood (NLL) evaluation, measuring generative quality of the diffeomorphic flow network on the LBOS (see Sect. 4.3.2). See Fig. 15 for explanation of model types and Appendix Table 5 for detailed values. In most scenarios, NCNJ experiences some degradation, while NLL in the FO case increases only for the STD case

same). Notice that we use the vertex correspondence to compute reconstruction loss during the AE training, but this information is *not* utilized for disentanglement by the VAE, which only has access to latent encodings x in our two-stage training regime. See Appendix H.1 for details, including hyper-parameter settings.

We test on two tasks, pose transfer and pose-aware retrieval, on held-out subsets of AMASS. We use the same evaluation methodology and splits as USPD for consistency, which induces minor differences with the evaluations on PCs from previous sections. We first measure pose transfer quality: given two meshes, we can obtain a ground truth transfer

by exchanging the SMPL parameters for articulation θ , while fixing those for body shape β , and obtain our prediction by doing so for z_I and z_E . After decoding, we can measure the average *vertex-to-vertex* Euclidean distance between the predicted and true transfers. These values are shown in Table 2. While we greatly outperform the original GDVAE, we still underperform USPD for this task. Nevertheless, beyond the additional requirements of USPD (subject labels and vertex correspondence), we note that our VAE is trained to reconstruct AE latent vectors (i.e., it is not trained end-to-end to reduce real-space vertex-to-vertex error), which also potentially contributes to worse performance on this task. In

Table 2 Pose transfer scores (lower is better) on mesh data from AMASS, measured in vertex-to-vertex distance in millimeters (with comparative numbers from Zhou et al., 2020)

	GDVAE	GDVAE++	USPD
Error ↓	54.44*	31.54	19.43

Note that the GDVAE model (which outputs a PC) measures error via the more forgiving one-sided Chamfer distance instead (as in Zhou et al. 2020). See Sect. 5.4.1 for additional details. We find that our GDVAE++ outperforms the GDVAE, but underperforms the more strongly supervised USPD, which uses subject labels and mesh correspondence for disentanglement. We mark the GDVAE score with a *, as it is computed with a different (more lenient) metric

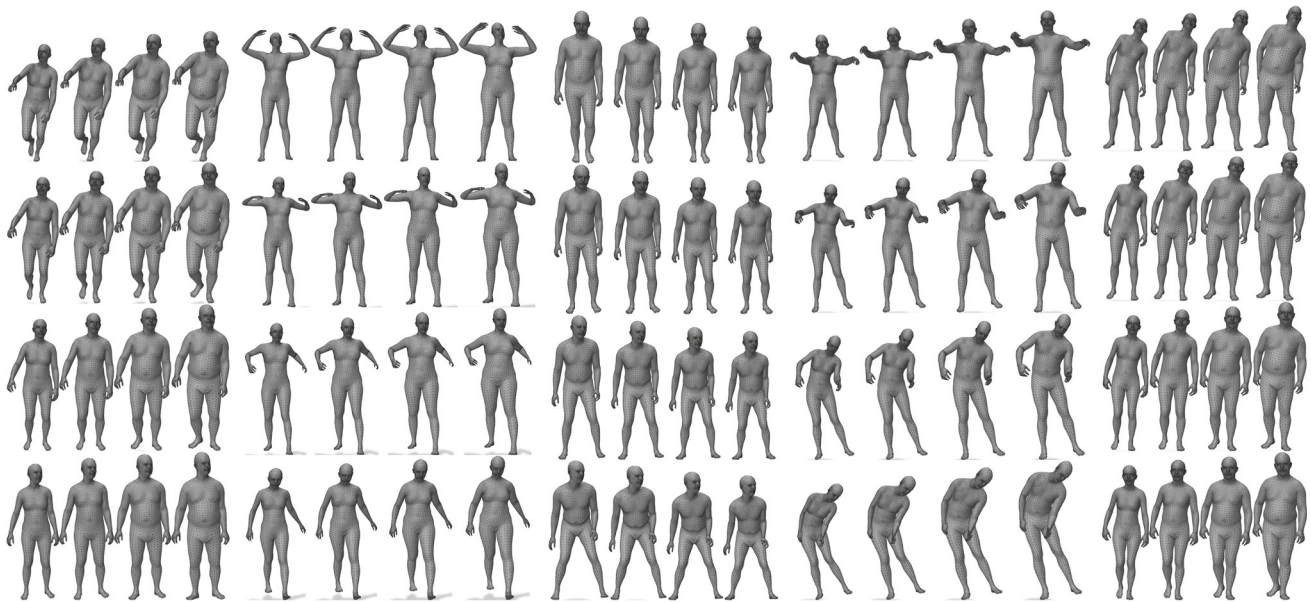


Fig. 20 Latent interpolations of AMASS validation mesh shapes. As in Fig. 13, per inset, horizontal or vertical movement traverses latent intrinsics z_I or extrinsics z_E , respectively, via linear interpolation. See

Sect. 5.4.1 for model details. Upper-left and lower-right shapes correspond to real input meshes; upper-right and lower-left shapes therefore correspond to *pose transfers*

Fig. 20, we show example latent interpolations in the disentangled space, including pose transfers.

We then examine pose-aware retrieval quality. For ease of comparison, we use the error measures on SMPL parameters from USPD: $\tilde{E}_\beta(z_\psi) = \mathbb{E}_{M_Q} \|\beta(M_Q) - \beta(M(z_\psi(M_Q)))\|_2$ and $\tilde{E}_\theta(z_\psi) = \mathbb{E}_{M_Q} \|q(\theta(M_Q)) - q(\theta(M(z_\psi(M_Q))))\|_2$, where $\psi \in \{E, I\}$, β, θ refer to shape and pose SMPL parameters, M_Q is a query mesh from a held-out test set, $M(z_\psi(M_Q))$ is the nearest neighbour mesh to M_Q as measured by MSE in z_ψ space, and q converts pose angles to unit quaternions. We also examine the differences $\Delta(\tilde{E}_\beta) = \tilde{E}_\beta(z_E) - \tilde{E}_\beta(z_I)$ and $\Delta(\tilde{E}_\theta) = \tilde{E}_\theta(z_I) - \tilde{E}_\theta(z_E)$, which should ideally be high. Quantitative results are compiled in Table 3. Compared to USPD, our method has higher $\tilde{E}_\beta(z_I)$ and $\tilde{E}_\theta(z_E)$, but outperforms in terms of both differences $\Delta(\tilde{E}_\beta)$ and $\Delta(\tilde{E}_\theta)$. Intuitively, when querying with latent intrinsics/extrinsics, USPD obtains shapes with very close intrinsics/extrinsics, but those shapes also have similar extrinsics/intrinsics; in other words, some shape-pose entanglement remains. By comparison, the GDVAE++

Table 3 Pose-aware disentangled retrieval scores on mesh data from AMASS

Retrieval with latent:		Intrinsics	Extrinsics	Δ
GDVAE	\tilde{E}_β	2.80 ↓	4.71 ↑	1.91 ↑
	\tilde{E}_θ	1.47 ↑	1.44 ↓	0.03 ↑
GDVAE++	\tilde{E}_β	0.41 ↓	1.36 ↑	0.94 ↑
	\tilde{E}_θ	1.15 ↑	0.80 ↓	0.35 ↑
USPD	\tilde{E}_β	0.14 ↓	0.92 ↑	0.78 ↑
	\tilde{E}_θ	0.94 ↑	0.76 ↓	0.18 ↑
GDVAE++ (PCA)	\tilde{E}_β	0.50 ↓	1.49 ↑	0.98 ↑
	\tilde{E}_θ	1.21 ↑	0.82 ↓	0.40 ↑
USPD (PCA)	\tilde{E}_β	0.34 ↓	2.14 ↑	1.80 ↑
	\tilde{E}_θ	1.23 ↑	0.87 ↓	0.36 ↑

Note that our latent intrinsics and extrinsics nomenclature refers to the latent “shape” and “pose” (or articulation) vectors in other works. Comparative numbers from Zhou et al. (2020). See Sect. 5.4.1 for additional details and Appendix H.1.3 for empirical standard deviations. We show the difference Δ between retrieval scores as well ($\tilde{E}_\beta(z_E) - \tilde{E}_\beta(z_I)$ and $\tilde{E}_\theta(z_I) - \tilde{E}_\theta(z_E)$), such that higher is better. The GDVAE appears to perform well on $\Delta(\tilde{E}_\beta)$; however, this is due to the high overall error magnitude in intrinsics retrieval, \tilde{E}_β . The authors of USPD previously observed a reduction in entanglement when using PCA (measured by Δ); we therefore compare against this dimensionally reduced version as well (using $\dim(z_I) = 5$ and $\dim(z_E) = 15$). We show the best score across categories between the PCA and non-PCA models in **bold**. In particular, notice that our method underperforms the non-PCA USPD in terms of $\tilde{E}_\beta(z_I)$ and $\tilde{E}_\theta(z_E)$, but outperforms it in terms of the Δ differences; in other words, while USPD retrieves shapes with close intrinsics/extrinsics (when querying with latent intrinsics/extrinsics), those shapes also have similar extrinsics/intrinsics, suggesting a level of shape-pose entanglement remains. In contrast, the PCA-reduced version of USPD has better Δ values; however, in this case, our PCA-based method has better $\tilde{E}_\theta(z_E)$, as well as better $\Delta(\tilde{E}_\theta)$ overall

has less entanglement (higher error when retrieving intrinsics/extrinsics with latent extrinsics/intrinsics), but also higher error in terms of retrieving intrinsics/extrinsics via latent intrinsics/extrinsics.

The authors of USPD also considered a version of their model with reduced dimensionality via PCA, which controlled for the difference in dimensionality between USPD and the GDVAE. They found it had better disentanglement properties, as evidenced by the higher differences Δ , but worse $\tilde{E}_\theta(z_E)$ and $\tilde{E}_\beta(z_I)$ values. We observe a similar effect occurs with our model when using PCA to transform z_E and z_I to that same dimensionality as well (from 9 to 5 for $\dim(z_I)$ and 18 to 15 for $\dim(z_E)$). Comparing the PCA-reduced case, USPD has superior retrieval results in terms of intrinsics, but ours has better values in terms of extrinsics ($\tilde{E}_\theta(z_E)$ and $\Delta(\tilde{E}_\theta)$).

We note that these Δ measures effectively weight the two terms equally, which may not be ideal. However, we find that a uniformly random retrieval algorithm incurs average errors of 6.5 for \tilde{E}_β and 1.76 for \tilde{E}_θ (as well as Δ values close to zero), suggesting none of these models are actually selecting random intrinsics/extrinsics for given query extrinsics/intrinsics, as one would expect from perfectly disentangled retrieval.

Overall, our model underperforms USPD on pose transfer, but is more competitive on retrieval. However, we remark that USPD relies on known subject identities to obtain sets of people with identical intrinsics, but different extrinsic pose, providing the network with explicit information about the articulated pose space for a given shape. It also utilizes vertex correspondence, which our method does not use for disentanglement. Together, these provide powerful learning signals to the network. This is different than our use of the LBOS, which is specific to a geometric entity, extractable from raw geometry, and not based on semantic knowledge about identity. In other words, USPD performs better for these tasks, but is more specialized, whereas our approach defines a generic structural prior on the deformation space of objects, which happens to disentangle articulation and intrinsic shape as a natural geometric consequence. Other factors, such as our need for low latent dimensionality and inability to do end-to-end training (necessitated by our information-theoretic disentanglement) also contribute to reduced performance.

5.4.2 Human Faces (CoMA)

We also investigated our approach on human face meshes, derived from the CoMA dataset (Ranjan et al., 2018). In particular, we consider the utility of our approach on a shape-from-spectrum task, under identical experimental conditions to recent work by Marin et al. (2021). Given an LBOS λ , our goal is to reconstruct the original shape S . Due to our use of a flow network, we can easily encode λ , to obtain the latent

intrinsic $\tilde{z}_I(\lambda)$. However, we also require latent extrinsics, which we must obtain without access to S . Fortunately, our VAE-based formulation permits a straightforward, principled solution: simply use the mode of the Gaussian prior over the latent extrinsics, meaning we set $z_E \equiv 0$. We can then decode $z = (\tilde{z}_I(S), \bar{0})$ to obtain the reconstructed shape \hat{S} with “mean” extrinsic pose, according to the prior. In practice, if we use more eigenvalues, more of the shape will be represented in \tilde{z}_I ; for fair comparison, we use the same number as Marin et al. (2021) (i.e., $\dim(\lambda) = 30$). Error is simply the vertex-to-vertex Euclidean distance between the meshes S and \hat{S} . Appendix H.2 contains additional details.

Our results are displayed in Table 4. We consider two nearest neighbour baselines (λ -NN- L_2 and λ -NN- d_λ), which simply retrieve the closest shape in the training set to the given spectrum, using the Euclidean distance or our weighted d_λ (Eq. 14), respectively. We remark that using d_λ provides superior retrievals than the L_2 metric, as it corrects for the growth of the monotonic LBOS, which overweights high frequency geometric details. The method by Marin et al. (2021) outperforms these baselines, but our method (using the mode of the VAE prior for z_E) performs the best overall. We observe that there is still a performance gap compared to using $z_E(x)$ (bottom row of the table); however, this is to be expected, since using the truncated spectrum alone will lose some information.

We also provide example latent interpolations on the CoMA dataset in Fig. 21. Notice that our latent intrinsics capture overall head shape, while the latent extrinsics contain deformations of the mouth and other facial expressions, despite only using raw meshes as input to the algorithm. Compared to Marin et al. (2021), which must perform a regularized optimization to obtain such disentanglement, our method simply linearly interpolates z_I and z_E .

6 Discussion

In this work, we have devised a method for separating the deformation space of an object into rigid orientation, non-rigid extrinsic pose, and intrinsic shape. We require no information other than the geometry of the shapes themselves (i.e., no labels or correspondences). Our method relies on the isometry invariance of the LBOS, which can be estimated from the geometry directly, and uses disentanglement techniques to partition the latent space of a generative model into these independent components.

In particular, we have built upon the GDVAE model (Aumentado-Armstrong et al., 2019) with two primary technical improvements. First, we investigated two approaches to improving rotation factorization: STD, which utilizes randomly rotated inputs to enforce rotation invariance (Li et al., 2019; Sanghi & Danielyan, 2019), and FTL, which pro-

Table 4 Empirical shape-from-spectrum results on CoMA, following the experimental settings of Marin et al. (2021)

Method	Error ↓	Spectrum Only
λ -NN- L_2	4.47	Yes
λ -NN- d_λ	2.63	Yes
Marin et al. (2021)	1.61	Yes
$\tilde{z}_I(\lambda)$ & $z_E \equiv 0$ (Ours)	1.52	Yes
Full z (Ours)	1.24	No

Columns: reconstruction methods, test set error (in terms of vertex-to-vertex L_2 distance), and whether or not some form of information about the shape extrinsics is used. Rows refer to different approaches: λ -NN- L_2 simply retrieves the closest shape in the training set, based on the L_2 distance between LBOSs; λ -NN- d_λ is the same nearest neighbour approach, but using our d_λ metric (Eq. 14) instead (which avoids over-emphasizing high frequency geometric details); “ $\tilde{z}_I(\lambda)$ & $z_E \equiv 0$ ” denotes simply setting z_E to be zero; and “Full z ” means $z = (z_I(x), z_E(x))$ is used, which forms a lower bound on the error we can expect, as it uses both intrinsic and extrinsic information from the full shape S . Our approach with $z_E \equiv 0$ uses only the spectrum λ of the shape (and no other information from S or x); it is equivalent to simply choosing z_E as the mode of the VAE prior, over the space of latent extrinsics. Overall, our method, which separates latent intrinsics and extrinsics, as well as guarantees invertibility, performs best. We observe the standard error of the mean for our two VAE-based approaches as 0.02 (using only λ) and 0.009 (using full z). All error values are $\times 10^{-5}$.

vides an interpretable latent space in which 3D rotations in a “folded” latent space mirror the effects of those rotations in real-space (Remelli et al., 2020; Worrall et al., 2017). Compared to the GDVAE, which was only able to maintain robustness to small rotations, both new AEs can handle arbitrary rotations about a single axis; the FTL method has the additional benefit of latent interpretability. Second, we utilized a diffeomorphic normalizing flow network to map between LBOSs and latent intrinsic space. Unlike the GDVAE, which did *not* have a mapping from LBOS space to latent intrinsic space (and thus could not *architecturally* stop latent intrinsics z_I computed from an encoded shape x from being affected by extrinsic pose information), utilizing this mapping in our GDVAE++ training procedure (see Sect. 4.4) allows us to compute reconstructions through \tilde{z}_I instead, guaranteeing this separation. Further, the bijectivity of the flow ensures that (i) spectral information is not lost and (ii) generative likelihood is tractably computable. Altogether, these changes result in greatly improved unsupervised disentanglement, without sacrificing other representational aspects.

Our results show that we have significantly improved on the GDVAE. Firstly, we are able to handle larger orientation changes with far better robustness in both the 3D data space and latent space (see Sect. 5.2), utilizing rotational invariance techniques that do not rely on a specific feature extraction or neural architecture. Secondly, we obtain nearly double

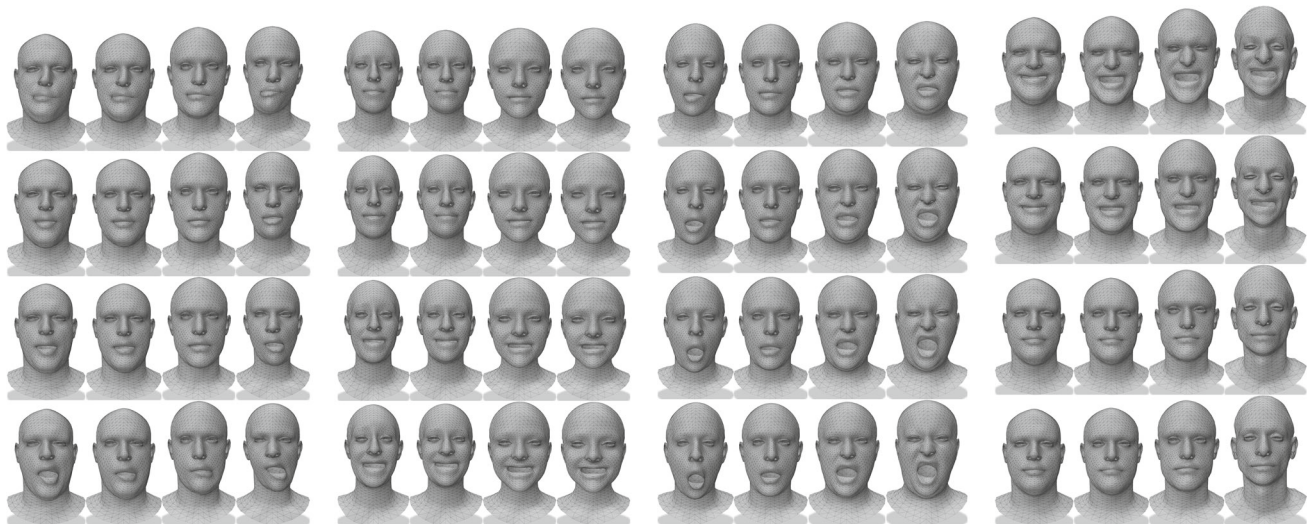


Fig. 21 Example latent interpolations on the CoMA dataset. Per inset, horizontal movement corresponds to linear changes in z_I , while vertical movement corresponds to linear changes in z_E . Upper left and lower right images are starting and ending points; upper right and lower left are thus pose (or intrinsics) transfers. The model is capable of correctly preserving identity when the intrinsics, z_I , are fixed (i.e., within columns), as well as representing subtle expressions (e.g., the orientation of the

mouth in the leftmost inset). We remark that there are a few failure cases (e.g., bottom row of the third inset, top row of the fourth inset) where the non-rigid pose (facial expression) is not exactly preserved as the latent intrinsics change. However note that our formulation demands only that latent shape and extrinsics are disentangled; it does not directly enforce which non-rigid pose should semantically correspond across different intrinsics

the quantitative disentanglement score, for data from both SMAL and SMPL, using our GDVAE++ training scheme (see Sect. 5.3.2). We also examined the ability of the model to generate novel shape samples, its capacity to smoothly and independently control latent shape and non-rigid pose (see Figs. 13 and 20), and the effect of several ablations and modifications of the model. Finally, we compare the GDVAE++ to existing techniques for disentanglement and shape-from-spectrum recovery.

For future work, we expect research on *localized* spectral geometry (Melzi et al., 2018; Neumann et al., 2014), LBO modifications (Andreux et al., 2014; Choukroun et al., 2018), and *extrinsic* spectral shape (Liu et al., 2017; Wang et al., 2017; Ye et al., 2018) to be potentially useful. Furthermore, our formulation is readily applicable to other 3D shape modalities (e.g., tetrahedral meshes or implicit fields), as the only elements of our architecture that would require alteration are the AE encoders (E_r and E_x) and decoder (D), provided one has a way to estimate the LBOS. Our VAE model is also agnostic to the neural architecture of the AE. Hence, our approach could be used in conjunction with other methods for factorizing deformations. Lastly, our method can also be utilized for applications in computer vision. For instance, it can be used for controllable shape generation or manipulation, for regularizing visual inference (e.g., by acting as a prior on expected deformation types), or for pose-aware shape retrieval. In general, we hope that our model can

serve as an interpretable unsupervised prior for understanding shape deformations.

Acknowledgements We are grateful for support from NSERC (CGSD3-534955-2019) and Samsung Research.

Data Availability All datasets are from publicly available sources, detailed below.

A Glossary of Notation

Symbol	Sec/Eq	Definition
P	Section 3	Shape
x_c	Section 3	Canonical AE encoding
\tilde{x}	Section 3.1.2	Non-canon FTL-AE encoding
q	Section 3	Quaternion
D_P	Equation 2	Distance between PCs
z_I	Section 4.2.1	Latent intrinsics
z_E	Section 4.2.1	Latent extrinsics
z_R	Section 4.2.1	Latent rigid pose
λ	Section 2.2	LBO Spectrum
\tilde{z}_I	Section 4.2.2	Latent intrinsics (from λ)
f_λ	Section 4.2.2	Spectral flow network
g_λ	Section 4.2.2	Inverse of f_λ
S	Equation 18	Disentanglement score
E_β	Section 5.3.2	Retrieval error wrt SMPL shape
E_θ	Section 5.3.2	Retrieval error wrt SMPL pose

Symbol	Sec/Eq	Definition
$P_\lambda(\lambda)$	Equation 12	Spectral likelihood
d_λ	Equation 14	Distance between spectra
d_R	Equation 4	Distance between rotations
\mathcal{L}_{AE}	Section 3.2	AE total loss
\mathcal{L}_c	Section 3.2.1	AE x -consistency
\mathcal{L}_R	Section 3.2.1	AE rotation prediction
\mathcal{L}_P	Section 3.2.1/3.2.2	AE shape prediction
\mathcal{L}_{VAE}	Section 4.3	GDVAE total loss
\mathcal{L}_{HF}	Section 4.3.1	HFVAE loss
L_R	Section 4.3.1	VAE reconstruction loss
\mathcal{L}_λ	Section 4.3.2	Spectral log-likelihood loss
\mathcal{L}_D	Section 4.3.4	Additional disentanglement loss
\mathcal{L}_F	Section 4.3.4	Intrinsics-Spectrum consistency

B Invariant FTL-Based Mapping

As an aside, in an FTL-based model, we remark that it is possible to transform $x \in \mathcal{X}$, in a way that is invariant to latent-space rotation operators. Let $\mathcal{I}[x] = (U(x)_i^T U(x)_j)_{i,j \in [1, N_s]; i \leq j}$ be the collection of inner products of the subvectors of x . Then $\mathcal{I}[x]$ is rotation invariant; i.e., $\mathcal{I}[x] = \mathcal{I}[F(R, x)]$, for any $R \in SO(3)$. This idea is noted by Worrall et al. (2017).

However, we found that using $\mathcal{I}[x]$ only slightly improved rotation invariance, yet slightly decreased reconstruction performance, and further was computationally expensive, due to the quadratic dependence of $\dim(\mathcal{I}[x])$ on N_s . Nevertheless, this may be specific to our particular architectural setup, and could still be an interesting direction for future work.

C Evaluation Metric Details

C.1 Latent Rotation Invariance Measure

We provide a more detailed description of the latent clustering metric \mathcal{C}_X here. Recall that our goal is to take a set of random shapes (potentially differing in intrinsics, non-rigid extrinsics, and orientation) and duplicate each shape, before randomly rotating the copies. We then encode each shape into x_c -space and cluster them. We use K-means clustering (MacQueen et al., 1967) to obtain the labeling. We expect our canonicalization method to bring the latent representations close in latent space, such that rotated copies should cluster together. We can therefore measure rotational invariance by supervised clustering quality metrics, in which the instance identity (i.e., which shape a vector originated from) is a ground truth cluster label. We use Adjusted Mutual Information (AMI) for this (Vinh et al., 2010), which returns 1 for a perfect partitioning (as compared to the ground truth) and a 0 for a random clustering.

While this captures the representational invariance to rotation in the embedding space, the number of disparate sample shapes to use is unclear. We therefore average over a sequentially larger set of samples, thus giving an “area-under-the-curve”-like measure of quality across sample sizes.

More formally, let $\Gamma = \{P_1, \dots, P_{N_I}\}$ be a set of $|\Gamma| = N_I$ PC shape instances. Now consider a set that includes N_{rc} rotated copies of each PC: $\tilde{\Gamma} = \bigcup_{k=1}^{N_I} \{\tilde{P}_{k,1}, \dots, \tilde{P}_{k,N_{rc}}\}$, where $\tilde{P}_{k,j} = P_k \tilde{R}_j$ for a randomly sampled \tilde{R}_j and $|\tilde{\Gamma}| = N_I N_{rc}$. We then encode the set into canonical representations $\tilde{\Gamma}_E = \{E_x(\tilde{P}_{k,j}) \mid \tilde{P}_{k,j} \in \tilde{\Gamma}\}$ and run our clustering algorithm on $\tilde{\Gamma}_E$ to get $\text{AMI}(N_I)$ for a given instance set size N_I . Let \mathcal{N}_S be a set of sample sizes (we chose eight sizes, linearly spaced from 20 to 10^3). Finally, the x_c -space rotational consistency metric is given by

$$\mathcal{C}_X = \frac{1}{|\mathcal{N}_S|} \sum_{N_I \in \mathcal{N}_S} \text{AMI}(N_I). \quad (19)$$

Note that for each size we always run two clusterings with different randomly chosen sample shapes, and use their average AMI in the above equation. For implementations, we use scikit-learn (Pedregosa et al., 2012).

D Dataset Details

Except for HA, all our datasets are identical to those in (Aumentado-Armstrong et al., 2019). We denote N_p as the size of the input point cloud (PC) and N_λ as the dimensionality of the spectrum used. In all cases, we output the same number of points as we input.

We also perform a scalar rescaling of the dataset such that the largest bounding box length is scaled down to unit length. This scale is the same across PCs in a given dataset (otherwise the change in scale would affect the spectrum for each shape differently). For augmentation and rigid orientation learning, we apply random rotations about the gravity axis (SMAL and SMPL) or the out-of-image axis (MNIST). For rotation supervision, the orientation of the raw data is treated as canonical.

D.1 MNIST

Meshes are extracted from the greyscale MNIST images, followed by area-weighted point cloud (PC) sampling. See Aumentado-Armstrong et al. (2019) for extraction details. We set $N_p = 512$ and $N_\lambda = 20$. The dataset has 59483 training examples and 9914 testing examples.

D.2 SMAL

Using the SMAL model (Zuffi et al., 2017), we generate set of 3D animal shapes with varying shape and pose. Using densities provided by authors, we generate 3200 shapes per animal category. Following 3D-CODED (Groueix et al., 2018), we sample poses by taking a Gaussian about the joint angles with a standard deviation of 0.2. We use 15,000 shapes for training and 1000 for testing, and set $N_p = 1600$ and $N_\lambda = 24$.

D.3 SMPL

Based on the SMPL model (Loper et al., 2015), we again follow the procedure in 3D-CODED (Groueix et al., 2018) to assemble a dataset of human models. This results in 20500 meshes per gender, using random samples from the SUR-REAL dataset (Varol et al., 2017), plus an additional 3100 meshes of “bent” people per gender, following Groueix et al. (2018). Ultimately, we get 45,992 training and 1199 testing meshes, equally divided by gender, after spectral calculations. We used $N_p = 1600$ and $N_\lambda = 20$.

D.4 Human-Animal (HA)

Since our model uses only geometry, we are able to simply mix SMAL and SMPL data together. The testing sets are left alone, and used separately during evaluation (for comparison to the unmixed models). For training, we use the entire SMAL training set, plus 9000 unbent and 1500 bent samples from the SMPL training set, per gender. We set $N_p = 1600$ and $N_\lambda = 22$.

Note that the use of a single scalar scaling factor (setting the maximum bounding box length to 1) means that SMPL models are smaller in the HA data than in the isolated SMPL dataset. We correct for this in the evaluation tables so that they are comparable (e.g., for Chamfer distances).

E Results Tables

In this section, we provide the detailed results tables for the experiments discussed in Sect. 5.3. See Table 5 for measurements of VAE quality, including reconstruction, generative modelling, and disentanglement metrics. See Table 6 for pose-aware retrieval scores, with various choices of latent vector, and Fig. 22 for plots of those scores for the STD AE (as well as Fig. 19 for the FTL AE case).

F Implementation Details

All models were implemented in Pytorch (Paszke et al., 2019). Notationally, let $n = \dim(x_c)$, $n_E = \dim(z_E)$,

$n_I = \dim(z_I)$, and $n_R = \dim(z_R)$. For this section, we assume that the number of points N_p in a point cloud (PC) is the same for inputs and outputs (though the architectures themselves do not require this). Validation sets of size 40 (or 250 for MNIST) were set aside from the training set to observe generalization error estimates. Hyper-parameters were largely set based on qualitative examination of training outputs.

F.1 Autoencoder Details

F.1.1 AE Network Architectures

Both the STD and FTL architectures used the same network components, with slight hyper-parameter alterations. Our encoders $E_r : \mathbb{R}^{3N_p} \rightarrow \mathbb{R}^4$ and $E_x : \mathbb{R}^{3N_p} \rightarrow \mathbb{R}^n$ were implemented as PointNets (Qi et al., 2017) without spatial transformers, with hidden channel sizes (64, 128, 256, 512, 128) and (128, 256, 512, 836, 1024). The inputs are only the point coordinates (i.e., three channels) and the output is a four-dimensional quaternion for E_r and an n -dimensional vector for E_x . The decoder $D : \mathbb{R}^n \rightarrow \mathbb{R}^{3N_p}$ is implemented as a fully connected network, with hidden layer sizes (K , $2K$, $4K$), where $K = 1200$ for STD and $K = 1250$ for FTL. Within D , each layer consisted of a linear layer, layer normalization (Ba et al., 2016), and ReLU (except for the last, which had only a linear layer). For the MNIST dataset only, we changed the hidden layer sizes of the decoder D to be (512, 1024, 1536).

F.1.2 AE Hyper-Parameters and Loss Weights

For architectural parameters, in the FTL case, we set $N_s = 333$, and hence $n = 999$. For STD, we let $n = 600$ and did not notice improvements when increasing it. For MNIST, we set $N_s = 32$ and, for the STD case, $n = 150$. Regarding loss parameters, we set the reconstruction loss weights to $\alpha_C = 200$, $\alpha_H = 1$, $\tilde{\gamma}_P = 100$, and $\gamma_P = 20$, in the FTL case, altering only $\gamma_P = 250$ in the STD case. Rotational consistency and prediction loss weights were set to $\gamma_c = 1$ and $\gamma_r = 10$. Regularization loss weights were $\gamma_w = \gamma_d = 2 \times 10^{-5}$. For MNIST, we altered $\gamma_c = 50$ for STD, while we let $\gamma_d = 10^{-6}$, $\gamma_w = 5 \times 10^{-5}$, and $\gamma_c = 100$ for FTL.

F.1.3 AE Training Details

We train all AEs with Adam (Kingma & Ba, 2014), using an initial learning rate of 0.0005. For supervised AEs, we pretrain the rotation predictor for 2000 iterations before the rest of the network. We use a scheduler that decreases the learning rate by 5% upon hitting a loss plateau, until it reaches 0.0001. We trained MNIST, SMAL, SMPL, and HA for 200,

Table 5 VAE Evaluation on held-out test data

Dataset	VAE Model	AE Model	$d_C(P, \hat{P}) \downarrow$	Fidelity \downarrow	Coverage \uparrow	$\log P_\lambda(\lambda) \uparrow$	$S \uparrow$
SMAL	GDVAE++	STD-S	0.56	0.83	0.77	-121.60	2.07
		FTL-S	0.40	1.20	0.62	-144.17	2.12
		STD-U	0.39	0.68	0.66	-114.73	2.02
		FTL-U	0.73	0.99	0.75	-135.21	2.07
	GDVAE-FO	STD-S	0.48	0.81	0.72	-147.60	0.49
		FTL-S	0.24	0.90	0.73	-143.39	0.43
	GDVAE++ (PCLBO)	STD-S	0.51	0.89	0.65	-116.62	1.83
		FTL-S	1.05	1.09	0.65	-111.12	1.11
	GDVAE++ (NCNJ)	STD-S	0.44	1.20	0.47	-151.32	2.14
		FTL-S	0.34	1.09	0.57	-196.15	2.06
SMPL	GDVAE++	STD-S	0.50	1.21	0.73	-131.82	2.58
		FTL-S	0.38	1.38	0.80	-137.06	2.46
		STD-U	0.36	1.11	0.67	-151.74	2.46
		FTL-U	0.43	1.38	0.80	-135.45	1.93
	GDVAE-FO	STD-S	0.43	1.21	0.65	-232.73	0.32
		FTL-S	0.26	1.33	0.79	-94.65	0.64
	GDVAE++ (PCLBO)	STD-S	0.53	1.16	0.80	-164.64	1.43
		FTL-S	0.37	1.47	0.77	-145.43	1.35
	GDVAE++ (NCNJ)	STD-S	0.43	1.36	0.54	-178.26	2.01
		FTL-S	0.36	1.44	0.58	-123.40	2.24
HA	GDVAE++	STD-S	0.54/0.64	0.89/1.32	0.63/0.68	-130.13/-125.44	1.86/1.92
		FTL-S	0.40/0.65	1.06/1.54	0.74/0.59	-117.82/-108.69	1.72/1.83

The first three columns denote the dataset, the VAE type, and the AE type (S and U mean supervised and unsupervised, respectively). GDVAE-FO means using the x_c -derived latent intrinsics (as opposed to the λ -derived one) in training; PCLBO and NCNJ denote using point cloud-based LBOs and ablating the \mathcal{L}_D loss, respectively. The five right-most columns are evaluation metrics, considering reconstruction (d_C), generative modelling (Fidelity, Coverage, and $\log P_\lambda(\lambda)$), and unsupervised disentanglement (S). See Sects. 5.3.4 and 5.3.3 for additional details on the alterations (PCLBO and NCNJ), Sect. 4.4 for the difference between GDVAE++ and GDVAE-FO, Sect. 5.3.1 for the generative modelling metrics, and Sect. 5.3.2 for the scalar disentanglement measure. For the HA dataset, we show A/B as the scores on the SMAL and SMPL test sets, respectively, and we also scale Chamfer distances for the HA dataset to make them comparable across datasets, as in Table 1. We see that the GDVAE++ model obtains much better disentanglement scores than the GDVAE model (across SMAL, SMPL, and HA), while GDVAE-FO does significantly worse. In terms of retrieval quality, using the PCLBO degrades performance, but it stays above the GDVAE as well, while ablating \mathcal{L}_D (the NCNJ scenario) worsens performance on SMPL but has little effect for SMAL.

1250, 350, and 400 epochs, respectively, and batch sizes of 64/100 (FTL/STD) and 36/40 for MNIST and non-MNIST datasets. We set the number of rotated copies (which expands the batch sizes above) to $N_R = 3$, except in the case of MNIST (for which we used $N_R = 6$ in the FTL case and $N_R = 4$ in the STD case). Finally, note that, during training, for the supervised case only, we replace the predicted rotation \hat{R} with the real one R in all operations.

F.2 Variational Autoencoder Details

For implementation of the HFVAE, we use ProbTorch (Narayanaswamy et al., 2017). Our normalizing flow sub-network used nflows (Durkan et al., 2020).

F.2.1 VAE Network Architectures

For the VAE, all networks except for the flow mapping f_λ are implemented as fully connected networks (linear-layer+norm+ReLU, as above). Approximate variational posteriors have diagonal covariances. Thus, we have the following mappings with their hidden sizes:

- The rotation distribution parameter encoders, $\mu_R : \mathbb{R}^4 \rightarrow \mathbb{R}^{n_R}$ and $\Sigma_R : \mathbb{R}^4 \rightarrow \mathbb{R}^{n_R}$, are implemented with an initial shared network, with hidden sizes (256, 128) into an intermediate dimensionality of 64, followed by single linear layer each.
- The quaternion decoder $D_q : \mathbb{R}^{n_R} \rightarrow \mathbb{R}^4$ is structured as (64, 128, 256).
- The intrinsic and extrinsic parameter encoders, $\mu_\xi : \mathbb{R}^n \rightarrow \mathbb{R}^{n_\xi}$ and $\Sigma_\xi : \mathbb{R}^n \rightarrow \mathbb{R}^{n_\xi}$, for $\xi \in \{E, I\}$,

Table 6 Retrieval scores

Data	Model	\widehat{s}_ψ	$z \uparrow$	z_E	z_I	\widetilde{z}_I	
SMAL	STD-S	\widehat{s}_β	1.46	0.23 ↓	1.63 ↑	2.04 ↑	
		\widehat{s}_θ	0.69	0.83 ↑	0.16 ↓	0.08 ↓	
	FTL-S	\widehat{s}_β	1.52	0.13 ↓	1.66 ↑	1.77 ↑	
		\widehat{s}_θ	0.61	0.68 ↑	0.11 ↓	0.12 ↓	
	STD-U	\widehat{s}_β	1.35	0.19 ↓	1.56 ↑	1.90 ↑	
		\widehat{s}_θ	0.73	0.79 ↑	0.15 ↓	0.06 ↓	
	FTL-U	\widehat{s}_β	1.54	0.15 ↓	1.77 ↑	1.99 ↑	
		\widehat{s}_θ	0.63	0.66 ↑	0.21 ↓	0.18 ↓	
	STD-S	\widehat{s}_β	1.15	0.16 ↓	1.30 ↑	2.00 ↑	
	(PCLBO)	\widehat{s}_θ	0.74	0.89 ↑	0.20 ↓	0.04 ↓	
	FTL-S	\widehat{s}_β	0.74	0.25 ↓	0.96 ↑	1.84 ↑	
	(PCLBO)	\widehat{s}_θ	0.56	0.62 ↑	0.21 ↓	0.09 ↓	
	STD-S	\widehat{s}_β	1.38	0.20 ↓	1.60 ↑	1.93 ↑	
	(NCNJ)	\widehat{s}_θ	0.73	0.91 ↑	0.17 ↓	0.07 ↓	
	FTL-S	\widehat{s}_β	1.55	0.19 ↓	1.69 ↑	1.91 ↑	
	(NCNJ)	\widehat{s}_θ	0.73	0.70 ↑	0.13 ↓	0.07 ↓	
	SMPL	STD-S	\widehat{s}_β	0.72	0.12 ↓	1.97 ↑	1.73 ↑
			\widehat{s}_θ	0.94	0.93 ↑	0.20 ↓	0.34 ↓
FTL-S		\widehat{s}_β	0.85	0.12 ↓	1.78 ↑	2.11 ↑	
		\widehat{s}_θ	0.90	0.95 ↑	0.15 ↓	0.25 ↓	
STD-U		\widehat{s}_β	0.65	0.06 ↓	1.95 ↑	1.80 ↑	
		\widehat{s}_θ	0.90	0.87 ↑	0.30 ↓	0.29 ↓	
FTL-U		\widehat{s}_β	0.97	0.30 ↓	1.58 ↑	2.03 ↑	
		\widehat{s}_θ	0.87	0.89 ↑	0.24 ↓	0.28 ↓	
STD-S		\widehat{s}_β	0.42	0.23 ↓	0.90 ↑	1.74 ↑	
(PCLBO)		\widehat{s}_θ	0.93	0.91 ↑	0.14 ↓	0.31 ↓	
FTL-S		\widehat{s}_β	0.71	0.43 ↓	1.09 ↑	2.01 ↑	
(PCLBO)		\widehat{s}_θ	0.91	0.91 ↑	0.21 ↓	0.30 ↓	
STD-S		\widehat{s}_β	1.26	0.24 ↓	1.63 ↑	2.23 ↑	
(NCNJ)		\widehat{s}_θ	0.74	0.96 ↑	0.34 ↓	0.38 ↓	
FTL-S		\widehat{s}_β	1.35	0.22 ↓	1.66 ↑	2.24 ↑	
(NCNJ)		\widehat{s}_θ	0.77	0.96 ↑	0.15 ↓	0.32 ↓	

All models use the GDVAE++ training regime. \uparrow (\downarrow) means the higher (lower) the better. See Sect. 5.3.2 for additional details

have identical network architectures across latent group types: (2000, 1600, 1200, 400) and (2000, 1200, 400), for μ_ξ and Σ_ξ , respectively.

- The only mapping that is not a fully connected network is the bijective flow f_λ (and its inverse, g_λ). Recall that we use f_λ as $\widetilde{\mu}$. Hence, $f_\lambda : \mathbb{R}^{N_\lambda} \rightarrow \mathbb{R}^{n_I}$ and $N_\lambda = n_I$. This is implemented as a normalizing flow with nine layers, where each layer consists of an affine coupling transform (Dinh et al., 2016), an activation normalization (actnorm) (Kingma & Dhariwal, 2018), and a random feature ordering permutation. The last layer does not have normalization or permutation. Each affine coupling uses an internal FC network with one hidden layer of size 400.

- For the GDVAE++ training regime, we require a covariance parameter estimator for inference during training: $\Sigma_\xi : \mathbb{R}^{N_\lambda} \rightarrow \mathbb{R}^{n_I}$. This is implemented via hidden layers ($2n_I, 2n_I$).
- The shape decoder $D_x : \mathbb{R}^{n_I+n_E} \rightarrow \mathbb{R}^n$ is an FC network with hidden layers (600, 1200, 1600, 2000).

F.2.2 VAE Hyper-Parameters and Loss Weights

Recall that the loss hyper-parameters control the following terms: the intra-group total correlation (TC) β_1 , dimension-wise KL divergence β_2 , mutual information β_3 , inter-group TC β_4 , log-likelihood reconstruction ω_R , relative quaternion reconstruction ω_q , flow likelihood ω_p , intrinsics consistency in z_I -space ω_I , intrinsics consistency in λ -space ω_λ , covariance disentanglement ω_Σ , and Jacobian disentanglement $\omega_{\mathcal{J}}$.

In all cases, we set $n_R = 3$, $\beta_1 = 1$, $\beta_3 = 1$, $\omega_p = 1$, $\omega_{\mathcal{J}} = 200$, and $\omega_q = 10$. See Table 7 for dataset-dependent parameters. An additional L_2 weight decay was applied to all networks, with a strength of 10^{-4} . For the flow-only (GDVAE-FO) approach, the parameters are the same per dataset, except for ω_I and ω_λ (which were tuned more in line with original GDVAE model, in an effort to improve disentanglement). For the FO case, we set $\omega_\lambda = 800$ and ω_I to 0, 250, and 0, for SMAL, SMPL and HA, respectively.

F.2.3 VAE Training Details

As in the AE case, optimization is done with Adam, using a reduce-on-plateau scheduler. The initial learning rate was set to 0.0001, with a minimum of 0.00001. A batch size of 264 was used, except for MNIST, for which we used 512. The networks were trained for 25000 iterations for MNIST and 40000 iterations for all other datasets. We note that for the GDVAE++ mode only, we also cut the gradient of the ω_I loss term from flowing through $\widetilde{\mu}_I$ (preventing extrinsic information in μ_I from contaminating $\widetilde{\mu}_I$).

G Full Rotation-Space Experiments

We also provide some limited tests our method on full 3D rotations, rather than single-axis rotations. We find that the invariance properties are severely reduced in this more difficult scenario. In particular, we train two AEs on SMAL, both using the FTL architecture and allowing arbitrary rotations. We try both the supervised (S) and unsupervised (U) cases.

Results are shown in Table 8. We see that both reconstruction and rotation invariance are worsened; however, note that (1) \mathcal{C}_{3D} is of a smaller magnitude than $d_C(P, \widehat{P})$, suggesting the presence of some rotation invariance, and (2) \mathcal{C}_X are larger than zero (the expected value if there were no latent structure

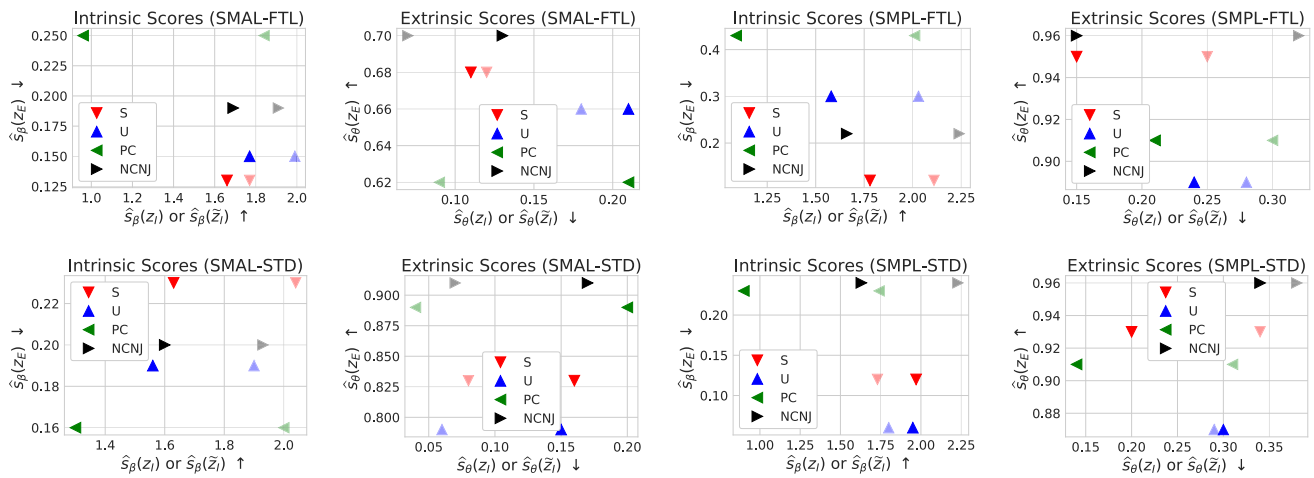


Fig. 22 Pose-aware retrieval scores with the STD AE model. Model notation refers to the GDVAE++ model with (S) or without (U) rotation supervision, use of the PC-derived LBOS (PC; see Sect. 5.3.3), and the partial disentanglement loss ablation (NCNJ; see Sect. 5.3.4). The lighter (partially transparent) counterparts of each point corresponds to using \tilde{z}_I instead of z_I for retrieval. We reproduce the FTL AE plots (from Fig. 19) to aid in comparison. See also Appendix Table 6 for detailed values. Compared to the FTL case, for SMPL, U performs rel-

atively better on intrinsic scores, while S and U are relatively similar for extrinsic scores. For SMAL, we see that the extrinsic scores are generally better with the STD AE, compared to the FTL one. We also see that, in the STD case, the PCLBO scenario performs relatively better on SMAL than its S/U counterparts. Finally, we note that using the spectrum-derived latents \tilde{z}_I are generally better, but not always (e.g., on SMPL-STD)

in the space). Corroborating this latter point, in Fig. 23, we can qualitatively see that the tight latent clustering of rotated objects (as in Fig. 11) is no longer present, but that there is still some structure in the space, by which same-identity objects stay nearby under rotation.

We utilized slightly different hyper-parameters compared to the standard AE case. In particular, for S, we set the batch size to $B = 3$ and $N_R = 48$ trained for 24 epochs; for U, we set $B = 2$ and $N_R = 60$. Decoder layers were set via $K = 1400$ (see Sect. F.1). Loss weights were modified to $\gamma_c = 8000$, $\tilde{\gamma}_P = 50/7$, $\gamma_P = 10/7$ (S case) and $\gamma_c = 500$, $\tilde{\gamma}_P = 400/7$, $\gamma_P = 80/7$ (U case), to compensate for the larger rotational space.

H Mesh Experiments

H.1 AMASS Experiments

H.1.1 AE Settings

Following other works (e.g., Marin et al., 2021; Tan et al., 2018), we use a fully connected AE for the AMASS meshes. In particular, the encoder and decoder have hidden layers (1024, 512) and (512, 1024). A latent dimension of $\dim(x) = 128$ was used. Both networks use the SELU activation (Klambauer et al., 2017) and no normalization. Note that AMASS shapes use the SMPL mesh, with $N_{SMPL} = 6890$ nodes.

Table 7 VAE hyper-parameters across datasets

	β_2	β_4	ω_R	ω_Σ	ω_I	ω_λ	n_E	n_I
MNIST	50	100	350	80	600	0	4	24
SMAL	10	50	400	40	300	200	8	24
SMPL	20	100	350	80	600	0	12	20
HA	20	80	360	60	450	100	10	22

These values are for the GDVAE++ training method. See text in Appendix F.2.2 for details

Table 8 AE evaluation on held-out test data with full 3D rotations

Dataset	Model	$d_C(P, \hat{P}) \downarrow$	$C_{3D} \downarrow$	$C_X \uparrow$
SMAL	FTL-S	0.46 [0.10]	0.41 [0.14]	0.11 [0.93]
	FTL-U	0.29 [0.10]	1.32 [0.21]	0.17 [0.88]

Metrics (left to right) refer to the Chamfer distance in reconstructions and the rotational consistency measures (in 3D and x_c -space, respectively); see Sect. 5.2. We place the scores obtained by the corresponding single-axis models in square brackets beside each value (from Table 1), for ease of comparison. Notice the deterioration in both reconstruction and rotation invariance, compared to the single axis case. Nevertheless, note that (1) C_{3D} is of a smaller magnitude than $d_C(P, \hat{P})$, suggesting the presence of some rotation invariance, and (2) C_X are larger than zero (the expected value if there were no latent structure in the space)

The loss utilized for training modifies only \mathcal{L}_P , which (in the FTL case) is given by

$$\mathcal{L}_P = \gamma_P D_V(V, \hat{V}), \quad (20)$$

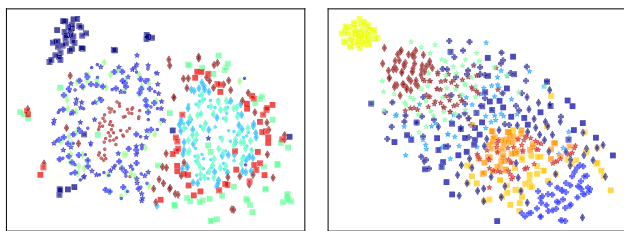


Fig. 23 Qualitative visualization of latent AE space with respect to full 3D rotation. Similar to Fig. 11, we show a t-SNE of the latent embeddings of random shapes under random rotations (we show more rotations as the set of rotations is now much larger). Heuristically, we can see there is some clustering structure in the space, but it does not have the tight invariance of the single-axis case. We show the supervised (S) case on the left and unsupervised (U) case on the right. See Sect. G for additional details

where $D_V(V_1, V_2)$ is the vertex-to-vertex mean squared error between the input nodal coordinate sets. Other loss terms remain the same as in the PC case. We set $\gamma_P = 5$ and use weight decay with $\gamma_w = 10^{-3}$. For simplicity, following Zhou et al. (2020), we include the global rotation in z_E rather than using a separate latent variable. Note that the input and output size is much larger for AMASS than for the PC case (6890 vs. 1600 points). The same learning setup was used as in the PC case, except we apply AdamW (Loshchilov & Hutter, 2017) with a learning rate of 10^{-4} and batch size of 100. We run for 250 epochs, using the same train, validation, and test splits as USPD. Notice that, while the AE uses the identical meshing of the input for the reconstruction loss, it does not perform any disentanglement. The VAE, which does perform disentanglement, uses only the raw x values (and does not update the AE), without correspondence or label information.

H.1.2 VAE Settings

We slightly modify the architecture of the VAE, removing batch normalization and replacing ReLU with SELU (as in the AE). We also increase the layer sizes: the flow network is given 10 layers, the encoders that predict μ_E and μ_I use hidden layers (2400, 2000, 1600, 800), and the decoder uses (2000, 1800, 1600, 800) for hidden layers; other networks are unchanged. We then use following hyper-parameters, with the SMPL (PC) settings as the default unless otherwise mentioned $\omega_R = 750$, $\beta_2 = 5$, $\beta_4 = 500$, $\omega_\Sigma = 5$, $\omega_\mathcal{J} = 50$, $\omega_I = \omega_\lambda = 1000$, $n_E = 18$, and $n_I = 9$. No weight decay was used. We trained with a batch size of 2200 for 40K iterations, starting from initial learning rate 5×10^{-5} .

Table 9 Pose-aware retrieval *standard errors of the mean* on mesh data from AMASS

Retrieval with latent:		Intrinsics	Extrinsics
GDVAE++	$\tilde{E}_\beta \pm \sigma(\tilde{E}_\beta)$	0.41 ± 0.0162	1.36 ± 0.0272
	$\tilde{E}_\theta \pm \sigma(\tilde{E}_\theta)$	1.15 ± 0.0065	0.80 ± 0.0066
GDVAE++	$\tilde{E}_\beta \pm \sigma(\tilde{E}_\beta)$	0.50 ± 0.0179	1.49 ± 0.0282
(PCA)	$\tilde{E}_\theta \pm \sigma(\tilde{E}_\theta)$	1.21 ± 0.0065	0.82 ± 0.0067

Data is shown as mean plus/minus standard error (with means from Table 3). Note that our latent intrinsics and extrinsics nomenclature refers to the latent “shape” and “pose” (or articulation) vectors in other works. See Sect. H.1.3 for additional details

H.1.3 Empirical Variation

We also compute variabilities on our mesh experiments (see Sect. 5.4.1), to give an indication of the variability in the results for our method. For pose transfers, we obtain a standard deviation of 8.08. Table 9 shows the standard errors of the mean for the pose-aware retrieval task. In general, the standard deviations are fairly high. However, following USPD, the held-out data sets from AMASS are of size 10,733 for pose transfer and 11,738 for retrieval, meaning the standard *error* of the mean is relatively small.

H.2 CoMA Experiments

H.2.1 Dataset

We use the same data as in Marin et al. (2021), namely 1853 training meshes with 100 faces from an unseen subject for the shape-from-spectrum recovery test set. We also use their data and dimensionality for the LBOS eigenvalues, so we set $\dim(\lambda) = 30$, and treat the meshes at full resolution (3931 vertices and 7800 faces).

H.2.2 AE Settings

Following Marin et al. (2021), we use the same fully connected AE to derive the initial latent representation x : tanh was used as the non-linearity, no normalization was applied, and the hidden layers were given by (300, 200, 30, 200) (with input and output in $\mathbb{R}^{3|V|}$), with $\dim(x) = 30$. The reconstruction loss was the vertex-to-vertex MSE, with weight $\gamma_P = 5$. We set the weight decay to $\gamma_w = 0.01$, the radial regularization to $\gamma_d = 0$, and the batch size to 16. Since this dataset has no orientation changes, we fix our rotation prediction to be identity.

H.2.3 VAE Settings

We use the same VAE architecture as the PC experiments. Only the hyper-parameters and training settings are altered,

which we leave at the SMPL settings by default, except for the following changes (see also Sect. F.2.2 for details): $\omega_R = 250$, $\beta_2 = 5$, $\beta_4 = 250$, $\omega_\Sigma = 100$, $\omega_{\mathcal{J}} = 250$, $\omega_I = \omega_\lambda = 1000$, $n_E = 1$, and $n_I = 30$. A lighter weight decay of 10^{-6} was used. We trained with a batch size of 720 for 30K iterations, starting from initial learning rate 5×10^{-5} . While this setup works well for the shape-from-spectrum task, and it mimics the $\dim(\lambda) = 30$ setting from Marin et al. (2021), we found qualitatively that disentangled interpolations could be improved by altering these settings to $n_E = 4$, $n_I = 12$, $\omega_R = 50$, and $\beta_4 = \omega_I = 500$, which we use for Fig. 21. This is likely due to facial deformations not being exactly isometric; hence, using too high LBOS dimensionality (and too low $\dim(z_E)$) leads to z_I capturing information we might not expect to be intrinsic (but improving shape-from-spectrum performance).

References

- Achlioptas, P., Diamanti, O., Mitliagkas, I., & Guibas, L. (2017). Learning representations and generative models for 3d point clouds. arXiv preprint [arXiv:1707.02392](https://arxiv.org/abs/1707.02392)
- Andreux, M., Rodola, E., Aubry, M., & Cremers, D. (2014). Anisotropic Laplace–Beltrami operators for shape analysis. In *European conference on computer vision* (pp. 299–312). Springer.
- Aubry, M., Schlickewei, U., & Cremers, D. (2011). The wave kernel signature: A quantum mechanical approach to shape analysis. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)* (pp. 1626–1633). IEEE.
- Aumentado-Armstrong, T., Tsogkas, S., Jepson, A., & Dickinson, S. (2019). Geometric disentanglement for generative latent shape models. In *Proceedings of the IEEE international conference on computer vision* (pp. 8181–8190)
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint [arXiv:1607.06450](https://arxiv.org/abs/1607.06450)
- Baek, S. Y., Lim, J., & Lee, K. (2015). Isometric shape interpolation. *Computers & Graphics*, *46*, 257–263.
- Basset, J., Wuhrer, S., Boyer, E., & Multon, F. (2020). Contact preserving shape transfer: Retargeting motion from one shape to another. *Computers & Graphics*.
- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.
- Berkiten, S., Halber, M., Solomon, J., Ma, C., Li, H., & Rusinkiewicz, S. (2017). Learning detail transfer based on geometric features. *Computer Graphics Forum, Wiley Online Library*, *36*, 361–373.
- Boscaini, D., Eynard, D., Kourounis, D., & Bronstein, M. M. (2015). Shape-from-operator: Recovering shapes from intrinsic operators. *Computer Graphics Forum, Wiley Online Library*, *34*, 265–274.
- Boscaini, D., Masci, J., Melzi, S., Bronstein, M. M., Castellani, U., & Vandergheynst, P. (2015). Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Computer Graphics Forum, Wiley Online Library*, *34*, 13–23.
- Bronstein, A. M., Bronstein, M. M., Guibas, L. J., & Ovsjanikov, M. (2011). Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, *30*(1), 1.
- Chen, C., Li, G., Xu, R., Chen, T., Wang, M., & Lin, L. (2019a). Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4994–5002).
- Chen, X., Chen, B., & Mitra, N. J. (2019b). Unpaired point cloud completion on real scans using adversarial training. arXiv preprint [arXiv:1904.00069](https://arxiv.org/abs/1904.00069)
- Chen, X., Lin, K. Y., Liu, W., Qian, C., & Lin, L. (2019c). Weakly-supervised discovery of geometry-aware representation for 3D human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10895–10904).
- Chen, X., Song, J., & Hilliges, O. (2019d). Monocular neural image based rendering with continuous view control. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4090–4100).
- Chern, A., Knöppel, F., Pinkall, U., & Schröder, P. (2018). Shape from metric. *ACM Transactions on Graphics (TOG)*, *37*(4), 1–17.
- Choukroun, Y., Shtern, A., Bronstein, A. M., & Kimmel, R. (2018). Hamiltonian operator for spectral shape analysis. *IEEE Transactions on Visualization and Computer Graphics*.
- Chu, M., & Golub, G. (2005). *Inverse eigenvalue problems: Theory, algorithms, and applications*. OUP Oxford
- Chua, C. S., & Jarvis, R. (1997). Point signatures: A new representation for 3D object recognition. *International Journal of Computer Vision*, *25*(1), 63–85.
- Cohen, T. S., Geiger, M., Köhler, J., & Welling, M. (2018). Spherical CNNs. arXiv preprint [arXiv:1801.10130](https://arxiv.org/abs/1801.10130)
- Corman, E., Solomon, J., Ben-Chen, M., Guibas, L., & Ovsjanikov, M. (2017). Functional characterization of intrinsic and extrinsic geometry. *ACM Transactions on Graphics (TOG)*, *36*(2), 14.
- Cosmo, L., Norelli, A., Halimi, O., Kimmel, R., & Rodolà, E. (2020). Limp: Learning latent shape representations with metric preservation priors. arXiv preprint [arXiv:2003.12283](https://arxiv.org/abs/2003.12283)
- Cosmo, L., Panine, M., Rampini, A., Ovsjanikov, M., Bronstein, M. M., & Rodolà, E. (2019). Isospectralization, or how to hear shape, style, and correspondence. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7529–7538).
- Dinh, L., Krueger, D., & Bengio, Y. (2014). Nice: Non-linear independent components estimation. arXiv preprint [arXiv:1410.8516](https://arxiv.org/abs/1410.8516)
- Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real NVP. arXiv preprint [arXiv:1605.08803](https://arxiv.org/abs/1605.08803)
- Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2020). nflows: Normalizing flows in PyTorch. <https://doi.org/10.5281/zenodo.4296287>
- Dym, N., & Maron, H. (2020). On the universality of rotation equivariant point cloud networks. arXiv preprint [arXiv:2010.02449](https://arxiv.org/abs/2010.02449)
- Esmaili, B., Wu, H., Jain, S., Bozkurt, A., Siddharth, N., Paige, B., Brooks, D. H., Dy, J., & van de Meent, J. W. (2018). Structured disentangled representations. arXiv preprint [arXiv:1804.02086](https://arxiv.org/abs/1804.02086)
- Fuchs, F. B., Worrall, D. E., Fischer, V., & Welling, M. (2020). Se (3)-transformers: 3D roto-translation equivariant attention networks. arXiv preprint [arXiv:2006.10503](https://arxiv.org/abs/2006.10503)
- Fumero, M., Cosmo, L., Melzi, S., & Rodolà, E. (2021). Learning disentangled representations via product manifold projection. In *International conference on machine learning* (pp. 3530–3540). PMLR.
- Gao, L., Yang, J., Qiao, Y. L., Lai, Y. K., Rosin, P. L., Xu, W., & Xia, S. (2018). Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (TOG)*, *37*(6), 1–15.
- Gebal, K., Bærentzen, J. A., Aanæs, H., & Larsen, R. (2009). Shape analysis using the auto diffusion function. *Computer Graphics Forum, Wiley Online Library*, *28*, 1405–1413.
- Ghosh, P., Sajjadi, M. S., Vergari, A., Black, M., & Schölkopf, B. (2019). From variational to deterministic autoencoders. arXiv preprint [arXiv:1903.12436](https://arxiv.org/abs/1903.12436)

- Gordon, C., Webb, D. L., & Wolpert, S. (1992). One cannot hear the shape of a drum. *Bulletin of the American Mathematical Society*, 27(1), 134–138.
- Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., & Aubry, M. (2018). 3D-coded: 3D correspondences by deep deformation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 230–246).
- Guo, Y., Bennamoun, M., Sohel, F., Lu, M., & Wan, J. (2014). 3D object recognition in cluttered scenes with local surface features: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11), 2270–2287.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). β -vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*.
- Huang, R., Rakotosaona, M. J., Achlioptas, P., Guibas, L. J., & Ovsjanikov, M. (2019). Operatornet: Recovering 3D shapes from difference operators. In *International conference on computer vision (ICCV)*.
- Huynh, D. Q. (2009). Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2), 155–164.
- Johnson, A. E., & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5), 433–449.
- Kac, M. (1966). Can one hear the shape of a drum? *The American Mathematical Monthly*, 73(4P2), 1–23.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
- Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1×1 convolutions. In *Advances in neural information processing systems* (pp. 10215–10224).
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, 29, 4743–4751.
- Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in Neural Information Processing Systems*, 30.
- Kobyzev, I., Prince, S., & Brubaker, M. (2020). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Kondor, R., Son, H. T., Pan, H., Anderson, B., & Trivedi, S. (2018). Covariant compositional networks for learning graphs. arXiv preprint [arXiv:1801.02144](https://arxiv.org/abs/1801.02144)
- Kovnatsky, A., Bronstein, M. M., Bronstein, A. M., Glashoff, K., & Kimmel, R. (2013). Coupled quasi-harmonic bases. *Computer Graphics Forum, Wiley Online Library*, 32, 439–448.
- Kumar, A., Sattigeri, P., & Balakrishnan, A. (2017). Variational inference of disentangled latent concepts from unlabeled observations. arXiv preprint [arXiv:1711.00848](https://arxiv.org/abs/1711.00848)
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
- Levinson, J., Sud, A., & Makadia, A. (2019). Latent feature disentanglement for 3D meshes. arXiv preprint [arXiv:1906.03281](https://arxiv.org/abs/1906.03281)
- Lévy, B. (2006). Laplace–Beltrami eigenfunctions towards an algorithm that understands geometry. In *IEEE international conference on shape modeling and applications, 2006. SMI 2006* (pp 13–13). IEEE.
- Li, J., Bi, Y., & Lee, G. H. (2019). Discrete rotation equivariance for point cloud recognition. In *2019 International conference on robotics and automation (ICRA)* (pp. 7269–7275). IEEE.
- Liu, H. T. D., Jacobson, A., & Crane, K. (2017). A Dirac operator for extrinsic shape analysis. *Computer Graphics Forum, Wiley Online Library*, 36, 139–149.
- Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., & Black, M. J. (2015). SMPL: A skinned multi-person linear model. *ACM Transactions (Proc SIGGRAPH Asia)*, 34(6), 248:1–248:16.
- Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint [arXiv:1711.05101](https://arxiv.org/abs/1711.05101)
- MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA* (Vol. 1, pp. 281–297).
- Mahmood, N., Ghorbani, N., Troje, N. F., Pons-Moll, G., & Black, M. J. (2019). AMASS: Archive of motion capture as surface shapes. In *International conference on computer vision* (pp. 5442–5451).
- Marin, R., Rampini, A., Castellani, U., Rodola, E., Ovsjanikov, M., & Melzi, S. (2020). Instant recovery of shape from spectrum via latent space connections. In *2020 International conference on 3D vision (3DV)* (pp. 120–129). IEEE.
- Marin, R., Rampini, A., Castellani, U., Rodola, E., Ovsjanikov, M., & Melzi, S. (2021). Spectral shape recovery and analysis via data-driven connections. *International Journal of Computer Vision*, 1–16.
- Masoumi, M., & Hamza, A. B. (2017). Spectral shape classification: A deep learning approach. *Journal of Visual Communication and Image Representation*, 43, 198–211.
- Melzi, S., Rodola, E., Castellani, U., & Bronstein, M. M. (2018). Localized manifold harmonics for spectral shape analysis. *Computer Graphics Forum, Wiley Online Library*, 37, 20–34.
- Meyer, M., Desbrun, M., Schröder, P., & Barr, A. H. (2003). Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III* (pp. 35–57). Springer.
- Moschella, L., Melzi, S., Cosmo, L., Maggioli, F., Litany, O., Ovsjanikov, M., et al. (2022). Learning spectral unions of partial deformable 3D shapes. *Computer Graphics Forum, Wiley Online Library*, 41, 407–417.
- Narayananwamy, S., Paige, B., Van de Meent, J. W., Desmaison, A., Goodman, N., Kohli, P., Wood, F., & Torr, P. (2017). Learning disentangled representations with semi-supervised deep generative models. In *Advances in neural information processing systems* (pp. 5925–5935).
- Neumann, T., Varanasi, K., Theobalt, C., Magnor, M., & Wacker, M. (2014). Compressed manifold modes for mesh processing. *Computer Graphics Forum, Wiley Online Library*, 33, 35–44.
- Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., & Guibas, L. (2012). Functional maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4), 1–11.
- Panine, M., & Kempf, A. (2016). Towards spectral geometric methods for Euclidean quantum gravity. *Physical Review D*, 93(8), 084033.
- Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. arXiv preprint [arXiv:1912.02762](https://arxiv.org/abs/1912.02762)
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, highperformance deep learning library. *Advances in Neural Information Processing Systems*, 32, 8024–8035.
- Patané, G. (2016). Star-Laplacian spectral kernels and distances for geometry processing and shape analysis. *Computer Graphics Forum, Wiley Online Library*, 35, 599–624.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2012). Scikit-learn: Machine learning in python. CoRR. [arxiv:1201.0490](https://arxiv.org/abs/1201.0490)

- Pons-Moll, G., Romero, J., Mahmood, N., & Black, M. J. (2015). Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics, (Proc SIGGRAPH)*, 34(4), 120:1–120:14.
- Poulenard, A., Rakotosaona, M. J., Ponty, Y., & Ovsjanikov, M. (2019). Effective rotation-invariant point CNN with spherical harmonics kernels. arXiv preprint [arXiv:1906.11555](https://arxiv.org/abs/1906.11555)
- Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proc computer vision and pattern recognition (CVPR)* (Vol. 1, issue 2, p. 4). IEEE.
- Rampini, A., Pestarini, F., Cosmo, L., Melzi, S., & Rodola, E. (2021). Universal spectral adversarial attacks for deformable shapes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3216–3226).
- Ranjan, A., Bolkart, T., Sanyal, S., & Black, M. J. (2018). Generating 3D faces using convolutional mesh autoencoders. In *European conference on computer vision (ECCV)* (pp. 725–741).
- Remelli, E., Han, S., Honari, S., Fua, P., & Wang, R. (2020). Lightweight multi-view 3D pose estimation through camera-disentangled representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6040–6049).
- Reuter, M. (2010). Hierarchical shape segmentation and registration via topological features of Laplace–Beltrami eigenfunctions. *International Journal of Computer Vision*, 89(2–3), 287–308.
- Reuter, M., Wolter, F. E., & Peinecke, N. (2006). Laplace–Beltrami spectra as shape-DNA of surfaces and solids. *Computer-Aided Design*, 38(4), 342–366.
- Rhodin, H., Constantin, V., Katircioglu, I., Salzmänn, M., & Fua, P. (2019). Neural scene decomposition for multi-person motion capture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7703–7713).
- Rhodin, H., Salzmänn, M., & Fua, P. (2018). Unsupervised geometry-aware representation for 3D human pose estimation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 750–767).
- Roberts, R. A., dos Anjos, R. K., Maejima, A., & Anjyo, K. (2020). Deformation transfer survey. *Computers & Graphics*.
- Rodolà, E., Cosmo, L., Bronstein, M. M., Torsello, A., & Cremers, D. (2017). Partial functional correspondence. *Computer Graphics Forum, Wiley Online Library*, 36, 222–236.
- Rustamov, R. M. (2007). Laplace–Beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing* (pp. 225–233). Eurographics Association.
- Sanghi, A. (2020). Info3D: Representation learning on 3D objects using mutual information maximization and contrastive learning. arXiv preprint [arXiv:2006.02598](https://arxiv.org/abs/2006.02598)
- Sanghi, A., & Danielyan, A. (2019). Towards 3D rotation invariant embeddings. In *CVPR 2019 workshop on 3D scene understanding for vision, graphics, and robotics*.
- Sharp, N., Gillespie, M., & Crane, K. (2021). Geometry processing with intrinsic triangulations. In *SIGGRAPH'21: ACM SIGGRAPH 2021 courses*.
- Sharp, N., & Crane, K. (2020). A Laplacian for nonmanifold triangle meshes. *Computer Graphics Forum, Wiley Online Library*, 39, 69–80.
- Shoemake, K. (1985). Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on computer graphics and interactive techniques* (pp. 245–254).
- Stein, F., & Medioni, G., et al. (1992). Structural indexing: Efficient 3-D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 125–145
- Su, F. G., Lin, C. S., & Wang, Y. C. F. (2021). Learning interpretable representation for 3d point clouds. In *2020 25th International conference on pattern recognition (ICPR)* (pp. 7470–7477). IEEE.
- Sumner, R. W., & Popović, J. (2004). Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG)*, 23(3), 399–405.
- Sun, X., Lian, Z., & Xiao, J. (2019). Srinet: Learning strictly rotation-invariant representations for point cloud classification and segmentation. In *Proceedings of the 27th ACM international conference on multimedia* (pp. 980–988).
- Sun, J., Ovsjanikov, M., & Guibas, L. (2009). A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum, Wiley Online Library*, 28, 1383–1392.
- Tan, Q., Gao, L., Lai, Y. K., & Xia, S. (2018). Variational autoencoders for deforming 3D mesh models. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5841–5850).
- Taubin, G. (1995). A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on computer graphics and interactive techniques* (pp. 351–358).
- Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., & Riley, P. (2018). Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds. arXiv preprint [arXiv:1802.08219](https://arxiv.org/abs/1802.08219)
- Tombari, F., Salti, S., & Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *European conference on computer vision* (pp. 356–369). Springer.
- Vallet, B., & Lévy, B. (2008). Spectral geometry processing with manifold harmonics. *Computer Graphics Forum, Wiley Online Library*, 27, 251–260.
- van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9, 2579–2605.
- Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., & Schmid, C. (2017). Learning from synthetic humans. In *CVPR*.
- Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11, 2837–2854.
- Wang, Y., Ben-Chen, M., Polterovich, I., & Solomon, J. (2017). Steklov spectral geometry for extrinsic shape analysis. arXiv preprint [arXiv:1707.07070](https://arxiv.org/abs/1707.07070)
- Watanabe, S. (1960). Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development*, 4(1), 66–82.
- Weyl, H. (1911). Über die asymptotische verteilung der eigenwerte. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse, 1911*, 110–117.
- Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. (2017). Interpretable transformations with encoder-decoder networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 5726–5735).
- Worrall, D., & Brostow, G. (2018). Cubenet: Equivariance to 3D rotation and translation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 567–584).
- Xiao, Z., Lin, H., Li, R., Geng, L., Chao, H., & Ding, S. (2020). Endowing deep 3D models with rotation invariance based on principal component analysis. In *2020 IEEE international conference on multimedia and expo (ICME)* (pp. 1–6). IEEE.
- Yang, G., Huang, X., Hao, Z., Liu, M. Y., Belongie, S., & Hariharan, B. (2019). Pointflow: 3D point cloud generation with continuous normalizing flows. arXiv preprint [arXiv:1906.12320](https://arxiv.org/abs/1906.12320)
- Ye, Z., Diamanti, O., Tang, C., Guibas, L., & Hoffmann, T. (2018). A unified discrete framework for intrinsic and extrinsic Dirac operators for geometry processing. *Computer Graphics Forum, Wiley Online Library*, 37, 93–106.
- Yin, M., Li, G., Lu, H., Ouyang, Y., Zhang, Z., & Xian, C. (2015). Spectral pose transfer. *Computer Aided Geometric Design*, 35, 82–94.

- You, Y., Lou, Y., Liu, Q., Tai, Y. W., Ma, L., Lu, C., & Wang, W. (2018). Pointwise rotation-invariant network with adaptive sampling and 3D spherical voxel convolution. arXiv preprint [arXiv:1811.09361](https://arxiv.org/abs/1811.09361)
- Zhang, Z., Hua, B. S., Rosen, D. W., & Yeung, S. K. (2019). Rotation invariant convolutions for 3D point clouds deep learning. In *International conference on 3D vision (3DV)*.
- Zhang, X., Qin, S., Xu, Y., & Xu, H. (2020). Quaternion product units for deep learning on 3D rotation groups. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7304–7313).
- Zhao, Y., Birdal, T., Lenssen, J. E., Menegatti, E., Guibas, L., & Tombari, F. (2020). Quaternion equivariant capsule networks for 3D point clouds. In *European conference on computer vision* (pp. 1–19). Springer.
- Zhou, K., Bhatnagar, B. L., & Pons-Moll, G. (2020). Unsupervised shape and pose disentanglement for 3D meshes. In *European conference on computer vision* (pp. 341–357). Springer.
- Zuffi, S., Kanazawa, A., Jacobs, D., & Black, M. J. (2017). 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on computer vision and pattern recognition (CVPR)*.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.