Frame Averaging for Equivariant Shape Space Learning

Matan Atzmon[1,2*] Koki Nagano[1] Sanja Fidler[1,3,4] Sameh Khamis[1] Yaron Lipman[2]
[1]NVIDIA [2]Weizmann Institute of Science [3]University of Toronto [4]Vector Institute

Abstract

The task of shape space learning involves mapping a train set of shapes to and from a latent representation space with good generalization properties. Often, real-world collections of shapes have symmetries, which can be defined as transformations that do not change the essence of the shape. A natural way to incorporate symmetries in shape space learning is to ask that the mapping to the shape space (encoder) and mapping from the shape space (decoder) are equivariant to the relevant symmetries. In this paper, we present a framework for incorporating equivariance in encoders and decoders by introducing two contributions: (i) adapting the recent Frame Averaging (FA) framework for building generic, efficient, and maximally expressive Equivariant autoencoders; and (ii) constructing autoencoders equivariant to piecewise Euclidean motions applied to different parts of the shape. To the best of our knowledge, this is the first fully piecewise Euclidean equivariant autoencoder construction. Training our framework is simple: it uses standard reconstruction losses, and does not require the introduction of new losses. Our architectures are built of standard (backbone) architectures with the appropriate frame averaging to make them equivariant. Testing our framework on both rigid shapes dataset using implicit neural representations, and articulated shape datasets using mesh-based neural networks show state of the art generalization to unseen test shapes, improving relevant baselines by a large margin. In particular, our method demonstrates significant improvement in generalizing to unseen articulated poses.

1. Introduction

Learning a shape space is the task of finding a latent representation to a collection of input training shapes that generalizes well to unseen, test shapes. This is often done within an autoencoder framework, namely an *encoder* $\Phi : X \to Z$, mapping an input shape in X (in some 3D representation) to the latent space Z, and a *decoder* $\Psi : Z \to Y$, mapping latent representations in Z back to shapes Y (possibly in other 3D representation than X).

*Work done during an internship at NVIDIA.

Many shape collections exhibit *symmetries*. That is, transformations that do not change the essence of the shape. For example, applying an Euclidean motion (rotation, reflection, and/or translation) to a rigid object such as a piece of furniture will produce an equivalent version of the object. Similarly, the same articulated body, such as an animal or a human, can assume different poses in space.

A natural way to incorporate symmetries in shape space learning is to require the mapping to the latent space, i.e., the encoder, and mapping from the latent space, i.e., the decoder, to be equivariant to the relevant symmetries. That is, applying the symmetry to an input shape and then encoding it would result in the same symmetry applied to the latent code of the original shape. Similarly, reconstructing a shape from a transformed latent code will result in a transformed shape.

The main benefit in imposing equivariance in shape space learning is achieving a very useful inductive bias: If the model have learned a single shape, it can already generalize perfectly to all its symmetric versions! Unfortunately, even in the presumably simpler setting of a global Euclidean motion, building an equivariant neural network that is both expressive and efficient remains a challenge. The only architectures that were known to be universal for Euclidean motion equivariant functions are Tensor Field Networks [17, 49] and group averaging [8, 57] both are computationally and memory intensive. Other architectures, *e.g.*, Vector Neurons [15] are efficient computationally but are not known to be universal.

In this paper, we present a novel framework for building equivariant encoders and decoders for shape space learning that are flexible, efficient and maximally expressive (*i.e.*, universal). In particular, we introduce two contributions: (i) we adapt the recent Frame Averaging (FA) framework [39] to shape space learning, showing how to efficiently build powerful shape autoencoders. The method is general, easily adapted to different architectures and tasks, and its training only uses standard autoencoder reconstruction losses without requiring the introduction of new losses. (ii) We construct what we believe is the first autoencoder architecture that is fully equivariant to piecewise Euclidean transformations of the shape's parts, *e.g.*, articulated human body.

We have tested our framework on two types of shape space learning tasks: learning implicit representations of shapes from real-life input point clouds extracted from sequences of images [42], and learning mesh deformations of human (body and hand) and animal shape spaces [1, 6, 31, 62]. In both tasks, our method produced state of the art results when compared to relevant baselines, often showing a big margin compared to the runner-up, justifying the efficacy of the inductive bias injected using the frame-averaging and equivariance.

2. Related work

Euclidean equivariant point networks. Original point cloud networks, such as PointNet [40, 41], PCNN [4], PointCNN [26], Spider-CNN [56], and DGCNN [52] are permutation equivariant but not Euclidean equivariant. Therefore, these architectures often struggle to generalize over translated and/or rotated inputs. Realizing that Euclidean equivariance is a useful inductive bias, much attention has been given to develop Euclidean equivariant point cloud networks. Euclidean *invariance* can be achieved by defining the network layers in terms of distances or angles between points [16, 60] or angles and distances measured from the input point cloud's normals [20]. Other works encode local neighborhoods using some local or global coordinate system to achieve invariance to rotations and translations. [16, 55, 58] use PCA to define rotation invariance. Equivariance is a desirable property for autoencoders. Some works use representation theory of the rotation group (*e.g.*, spherical harmonics) to build rotational equivariant networks [29, 53, 54]. Tensor Field Networks (TFN) [19, 43, 49] achieve equivariance to both translation and rotation. However, TFN architectures are tailored to rotations and require high order features for universality [17]. Recently [15] proposed a rotation equivariant network encoding features using the first two irreducible representations of the rotation group (tensor features) and constructed linear equivariant layers between features as well as equivariant non-linearities. This architecture is not proven universal. Another method achieving Euclidean equivariance is by group averaging or convolutions [57]. [12, 18] use spherical convolution to achieve rotation or Euclidean equivariance. [8] suggests to average of the 6D Euclidean group. Recently, [39] suggest Frame Averaging (FA) as a general purpose methodology for building equivariant architectures that are maximally expressive and often offer a much more efficient computation than group representation or averaging techniques.

Implicit shape space learning. Learning neural implicit representations from input point clouds is done by regressing signed distance function to the surface [35] or occupancy probabilities [10, 32]. The input point cloud is usually encoded in the latent space using a PointNet-like encoder [40, 59] or autodecoder [35]. [2, 3] regress the unsigned distance to the input point clouds avoiding the need of implicit function supervision for training. Normal data and gradient losses can be used to improve training and fidelity of learned implicits [3, 21, 27, 47]. Higher spatial resolution was achieved by using spatially varying latent codes [11, 37]. The above works did not incorporate Euclidean equivariance. As far as we are aware, [15] are the first to incorporate Euclidean equivariance in the implicit shape space learning framework.

Implicit representations are generalized to deformable and articulated shapes by composing the implicit representation with some backward parametric deformation such as Linear Blend Skinning (LBS) [23, 33, 44], displacement and/or rotation fields [36, 38] and flows [5, 34]. NASA [14] suggest to combine a collection of deformable components represented using individual occupancy networks sampled after reversing the Euclidean transformation of each component. SNARF [9] applies approximated inverse of LBS operator followed by an occupancy query. Both NASA and SNARF work on a single shape and do not learn the latent representation of pose.

Mesh shape space learning. Mesh shape spaces are often represented as coordinates assigned to a fixed template mesh and GNNs are used to learn their coordinates and latent representations [22, 24, 28, 50]. [25] adapt GNNs to surfaces advocating the dirac operator transferring information from nodes to faces and vice versa. [24, 28] use Variational AutoEncoders (VAEs) to improve generalization. The most recent and related work to ours in this domain is [22] that suggests to incorporate As-Rigid-As-Possible (ARAP) [48, 51] deformation loss to encourage Euclidean motions of local parts of the shape.

3. Method

3.1. Preliminaries: Group action

In this work, we consider vector spaces for representing shape and feature spaces. In the following, we define these vector spaces in general terms and specify how the different symmetry groups are acting on them. We use capital letters to represent vector spaces, *e.g.*, V, W, X, Y, Z. We use two types of vector spaces: i) $\mathbb{R}^{a+b\times 3}$, where $a, b \in \mathbb{N}_{\geq 0}$ are the invariant and equivariant dimensions, respectively, and ii) $C^1(\mathbb{R}^3)$, the space of continuously differentiable scalar volumetric functions. The symmetry considered in this paper is the group of Euclidean motions in $\mathbb{R}^3$, denoted $E(3) = O(3) \ltimes \mathbb{R}^3$, where $O(3)$ is the orthogonal matrix group in $\mathbb{R}^{3\times 3}$. We represent elements in this group as pairs $g = (\boldsymbol{R}, \boldsymbol{t})$, where $\boldsymbol{R} \in O(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$, where by default vectors are always column vectors.

The *action* of G on a vector space V, denoted ρ_V, is defined as follows. First, for $\boldsymbol{V} = (\boldsymbol{u}, \boldsymbol{U}) \in V = \mathbb{R}^{a+b\times 3}$, consisting of an invariant part $\boldsymbol{u} \in \mathbb{R}^a$, and equivariant part $\boldsymbol{U} \in \mathbb{R}^{b\times 3}$, we define the action simply by applying the transformation to the equivariant part:

$$\rho_V(g)\boldsymbol{V} = (\boldsymbol{u}, \boldsymbol{U}\boldsymbol{R}^T + \boldsymbol{1}\boldsymbol{t}^T) \tag{1}$$

where $g = (\boldsymbol{R}, \boldsymbol{t}) \in E(3)$ and $\boldsymbol{1} \in \mathbb{R}^b$ is the vector of all ones. Second, for $f \in V = C^1(\mathbb{R}^3)$ we define the action using change of variables:

$$(\rho_V(g)f)(\boldsymbol{x}) = f(\boldsymbol{R}^T(\boldsymbol{x} - \boldsymbol{t})) \tag{2}$$

for all $\boldsymbol{x} \in \mathbb{R}^3$ and $g = (\boldsymbol{R}, \boldsymbol{t}) \in G$.

3.2. Shape spaces and equivariance

We consider an input shape space X, a latent space Z, and output shape space Y, representing shapes in $\mathbb{R}^3$. All three spaces X, Z, Y are vector spaces as described above, each endowed with an action (using either equation 1 or 2) of the Euclidean group $G = E(3)$, denoted ρ_X, ρ_Z, ρ_Y, respectively.

Our goal is to learn an encoder $\Phi : X \to Z$, and decoder $\Psi : Z \to Y$ that are *equivariant*. Namely, given an $E(3)$-transformed input, $\rho_X(g)\boldsymbol{X}$ we would like its latent code to satisfy

$$\Phi(\rho_X(g)\boldsymbol{X}) = \rho_Z(g)\Phi(\boldsymbol{X}), \tag{3}$$

and its reconstruction to satisfy

$$\Psi(\rho_Z(g)\boldsymbol{Z}) = \rho_Y(g)\Psi(\boldsymbol{Z}). \tag{4}$$

Such X, Z, Y are called *steerable* spaces [13]. The following commutative diagram summarizes the interplay between the encoder, decoder, and the actions of the transformation group:

$$\begin{array}{ccccc} X & \xrightarrow{\Phi} & Z & \xrightarrow{\Psi} & Y \\ {\scriptstyle \rho_X(g)}\downarrow & & {\scriptstyle \rho_Z(g)}\downarrow & & {\scriptstyle \rho_Y(g)}\downarrow \\ X & \xrightarrow{\Phi} & Z & \xrightarrow{\Psi} & Y \end{array}$$

3.3. Frame averaging

We will use Frame Averaging (FA) [39] to build Φ, Ψ. FA allows to build both computationally efficient and maximally expressive equivariant networks. A *frame* is a map $\mathcal{F} : V \to 2^G \setminus \emptyset$. That is, for each element $\boldsymbol{V} \in V$ it provides a non-empty subset of the group $G = E(3)$, $\mathcal{F}(\boldsymbol{V}) \subset G$. The frame $\mathcal{F}$ is called *equivariant* if it satisfies

$$\mathcal{F}(\rho_V(g)\boldsymbol{V}) = g\mathcal{F}(\boldsymbol{V}) \tag{5}$$

for all $g \in G$, $\boldsymbol{V} \in V$, where for a set $A \subset G$ we define (as usual) $gA = \{ga \,|\, a \in A\}$, and the equality in equation 5 should be understood in the sense of sets. Then, as shown in [39], an arbitrary map $\phi : V \to W$ can be made equivariant by averaging over an equivariant frame:

$$\langle \phi \rangle_{\mathcal{F}}(\boldsymbol{V}) = \frac{1}{|\mathcal{F}(\boldsymbol{V})|} \sum_{g \in \mathcal{F}(\boldsymbol{V})} \rho_W(g)\phi\left(\rho_V(g)^{-1}\boldsymbol{V}\right). \tag{6}$$

The operator $\langle \cdot \rangle_{\mathcal{F}}$ is called Frame Averaging (FA). An alternative to FA is full group averaging [8, 57], that amounts to replacing the sum over $\mathcal{F}(\boldsymbol{V})$ in equation 6 with an integral over G. Full group averaging also provides equivariance and universality. The crucial benefit in FA, however, is that it only requires averaging over a small number of group elements without sacrificing expressive power. In contrast, averaging over the entire group $E(3)$ requires approximating a 6D integral (with an unbounded translation part). Therefore, it can only be approximated and is memory and computationally intensive [8].

Frame construction. All the frames we use in this paper are of the form $\mathcal{F} : V \to 2^G \setminus \emptyset$, for $V = \mathbb{R}^{d\times 3}$, $G = E(3)$, with the action defined as in equation 1. In some cases we further assume to have some non-negative weight vector $\boldsymbol{w} = (w_1, \ldots, w_d) \in \mathbb{R}^d_{\geq 0}$. Given $\boldsymbol{V} \in V = \mathbb{R}^{d\times 3}$ we define $\mathcal{F}(\boldsymbol{V}) \subset E(3)$ using weighted PCA, as follows. First,

$$\boldsymbol{t} = \frac{1}{\boldsymbol{1}^T\boldsymbol{w}} \boldsymbol{V}^T \boldsymbol{w} \tag{7}$$

is the weighted centroid. The covariance matrix is in $\mathbb{R}^{3\times 3}$

$$\boldsymbol{C} = (\boldsymbol{V} - \boldsymbol{1}\boldsymbol{t}^T)^T \mathrm{diag}(\boldsymbol{w})(\boldsymbol{V} - \boldsymbol{1}\boldsymbol{t}^T),$$

where $\mathrm{diag}(\boldsymbol{w}) \in \mathbb{R}^{d\times d}$ is a diagonal matrix with $\boldsymbol{w}$ along its main diagonal. In the generic case (which we assume in this paper) no eigenvalues of $\boldsymbol{C}$ are repeating, *i.e.*, $\lambda_1 < \lambda_2 < \lambda_3$ (for justification see *e.g.*, [7]). Let $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3$ be the corresponding eigenvectors. The frame is defined by $\mathcal{F}(\boldsymbol{V}) = \{(\boldsymbol{R}, \boldsymbol{t}) \mid \boldsymbol{R} = [\pm\boldsymbol{r}_1, \pm\boldsymbol{r}_2, \pm\boldsymbol{r}_3]\}$, which contains $2^3 = 8$ elements. Intuitively, $\boldsymbol{V}$ is a point cloud in $\mathbb{R}^3$ and its frame, $\mathcal{F}(\boldsymbol{V})$, contains all Euclidean motions that take the origin to the weighted centroid of $\boldsymbol{V}$ and the axes to the weighted principle directions. The proof of the following proposition is in the supplementary.

Proposition 1. *The frame $\mathcal{F}$ is equivariant.*

3.4. Shape space instances

Global Euclidean: Mesh → mesh. In this case, we would like to learn mesh encoder and decoder that are equivariant to global Euclidean motion. We consider the shape spaces $X = Y = \mathbb{R}^{n\times 3}$ that represent all possible coordinate assignments to vertices of some fixed n-vertex template mesh. The latent space is defined as $Z = \mathbb{R}^{m+d\times3}$ consisting of vectors of the form $\boldsymbol{Z} = (\boldsymbol{u}, \boldsymbol{U}) \in Z$, where the $\boldsymbol{u} \in \mathbb{R}^m$ part contains invariant features and the $\boldsymbol{U} \in \mathbb{R}^{d\times3}$ part contains equivariant features. The group actions ρ_X, ρ_Z, ρ_Y are as defined in equation 1. We define our encoder Φ and decoder Ψ by FA (equation 6), *i.e.*, $\Phi = \langle\phi\rangle_{\mathcal{F}}$, and $\Psi = \langle\psi\rangle_{\mathcal{F}}$ where the frames are defined as in Section 3.3 with constant weights $\boldsymbol{w} = \boldsymbol{1}$, $\phi : X \to Z$ and $\psi : Z \to Y$ are standard GNNs adapted to meshes (implementation details are provided in Section 4).

Global Euclidean: Point-cloud → implicit. Here we adopt the setting of [15] where $X = \mathbb{R}^{n\times3}$ represents all possible n-point clouds in $\mathbb{R}^3$, and $Y = C^1(\mathbb{R}^3)$ contains implicit representations of a shapes in $\mathbb{R}^3$. That is, for $f \in Y$ we consider its zero preimage,

$$f^{-1}(0) = \left\{\boldsymbol{x} \in \mathbb{R}^3 \mid f(\boldsymbol{x}) = 0\right\} \tag{8}$$

as our shape rerpesenation in $\mathbb{R}^3$. If 0 is a regular value of f then the Implicit Function Theorem implies that $f^{-1}(0)$ is a surface in $\mathbb{R}^3$. A regular value $r \in \mathbb{R}$ of f means that at every preimage $\boldsymbol{x} \in f^{-1}(r)$, the gradient does not vanish, $\nabla f(\boldsymbol{x}) \neq 0$. The latent space is again $Z = \mathbb{R}^{m+d\times3}$, consisting of vectors of the form $\boldsymbol{Z} = (\boldsymbol{u}, \boldsymbol{U}) \in Z$. The actions ρ_X, ρ_Z are defined as in equation 1, while the action ρ_Y is defined as in equation 2. The motivation behind the definition of ρ_Y is that $\rho_Y(g)f$ would transform the shape represented by f, that is $f^{-1}(0)$, by g:

$$\begin{aligned}(\rho_Y(g)f)^{-1}(0) &= \left\{\boldsymbol{x} \mid f(\boldsymbol{R}^T(\boldsymbol{x}-\boldsymbol{t})) = 0\right\} \\ &= \left\{\boldsymbol{R}\boldsymbol{x}+\boldsymbol{t} \mid f(\boldsymbol{x}) = 0\right\} \\ &= \boldsymbol{R}f^{-1}(0) + \boldsymbol{t}\end{aligned}$$

The encoder is defined as $\Phi = \langle\phi\rangle_{\mathcal{F}}$, where the frames are computed as described in Section 3.3 with constant weights $\boldsymbol{w} = \boldsymbol{1}$, and $\phi : X \to Z$ is a point cloud network (implementation details are provided in Section 4). Since the decoder needs to output an element in Y, which is a space of functions, we define the decoder by

$$\Psi(\boldsymbol{Z}) = \hat{\Psi}(\boldsymbol{Z}, \cdot), \tag{9}$$

where $\hat{\Psi} : Z \times \mathbb{R}^3 = \mathbb{R}^{m+3\times(d+1)} \to \mathbb{R}$. Following [15], to make the decoder Ψ equivariant as a map $Z \to Y$ it is enough to ask that $\hat{\Psi}$ is equivariant under appropriate actions. Namely, the action in equation 1 applied to $V = \mathbb{R}^{m+3\times(d+1)}$, and $W = \mathbb{R}$, where the latter is just the trivial action providing invariance, *i.e.*, $\rho_{\mathbb{R}}(g) \equiv 1$.

Proposition 2. *Ψ is equivariant iff $\hat{\Psi}$ is equivariant.*

The decoder is therefore defined as $\hat{\Psi} = \langle\psi\rangle_{\mathcal{F}}$, where $\psi : Z \times \mathbb{R}^3 \to \mathbb{R}$ is an MLP (implementation details are provided in Section 4), and the frame is defined as in Section 3.3 with constant weights $\boldsymbol{w} = \boldsymbol{1}$.

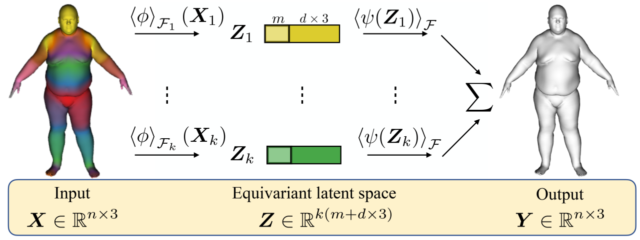

Figure 1. Piecewise Euclidean: Mesh → mesh. The same ϕ backbone is used for the equivariant encoding of each part. Similarly, the same ψ backbone is used for the equivariant decoding of each part's latent code. Lastly, the final prediction is a weighted sum of each part's equivariant output mesh.

Piecewise Euclidean: Mesh → mesh. In this scenario we generalize our framework to be equivariant to *different* Euclidean motions applied to different object parts (see Figure 1). We consider (as before) the shape spaces $X = Y = \mathbb{R}^{n\times3}$ that represent all possible coordinate assignments to vertices of some fixed n-vertex template mesh. The k parts are defined using a partitioning weight matrix (*e.g.*, as used in linear blend skinning) $\boldsymbol{W} \in \mathbb{R}^{n\times k}$, where $\boldsymbol{W}_{i,j} \in [0,1]$ indicates the probability that the i-th vertex belongs to the j-th part, and $\sum_{j=1}^{k} \boldsymbol{W}_{i,j} = 1$. The latent space has the form $Z = Z_1 \times \cdots \times Z_k$, where $Z_j \in \mathbb{R}^{m+d\times3}$. Note that $k = 1$ represents the case of a global Euclidean motion, as discussed above.

The actions $\rho_X, \rho_Y, \rho_{Z_j}$, $j \in [k] = \{1, \ldots, k\}$, are defined as in equation 1. Lastly, we define the encoder and decoder by

$$\Phi(\boldsymbol{X}) = \left(\langle\phi\rangle_{\mathcal{F}_j}(\boldsymbol{X}_j) \mid j \in [k]\right) \tag{10}$$

$$\Psi(\boldsymbol{Z}) = \sum_{j=1}^{k} \boldsymbol{w}_j \odot \langle\psi(\boldsymbol{Z}_j)\rangle_{\mathcal{F}} \tag{11}$$

where $\phi : X \to Z$, $\psi : Z \to Y$ are Graph Neural Networks (GNNs) as above; $\boldsymbol{X}_j \in X$ is the geometry of each part, where all other vertices are mapped to the part's centroid, *i.e.*,

$$\boldsymbol{X}_j = (\boldsymbol{1} - \boldsymbol{w}_j)\frac{\boldsymbol{w}^T\boldsymbol{X}}{\boldsymbol{w}^T\boldsymbol{1}} + \boldsymbol{w}_j \odot \boldsymbol{X},$$

$\boldsymbol{w}_j = \boldsymbol{W}_{:,j}$ the j-th column of the matrix $\boldsymbol{W}$, and each part's frame $\mathcal{F}_j$ is defined as in Section 3.3 with weights $\boldsymbol{w}_j$. The part's latent code is $\boldsymbol{Z}_j = \langle\phi\rangle_{\mathcal{F}_j}(\boldsymbol{X}_j) \in Z_j$. For vector $\boldsymbol{a} \in \mathbb{R}^n$ and matrix $\boldsymbol{B} \in \mathbb{R}^{n\times3}$ we define the multiplication $\boldsymbol{a} \odot \boldsymbol{B}$ by $(\boldsymbol{a} \odot \boldsymbol{B})_{i,j} = \boldsymbol{a}_i\boldsymbol{B}_{i,j}$.

If using hard weights, *i.e.*, $\boldsymbol{W} \in \{0,1\}^{n\times k}$, this construction guarantees *part-equivariance*. That is, if the j-th part of an input shape $\boldsymbol{X} \in X$ is transformed by $g_j \in G$, $j \in [k]$, that is,

$$\boldsymbol{X}' = \sum_{j=1}^{k} \boldsymbol{w}_j \odot (\rho_X(g_j)\boldsymbol{X})$$

then the corresponding latent codes, $\boldsymbol{Z}_j$, will be transformed by $\rho_{Z_j}(g_j)$, namely

$$\boldsymbol{Z}_j' = \rho_{Z_j}(g_j)\boldsymbol{Z}_j$$

and the decoded mesh will also transform accordingly,

$$\boldsymbol{Y}' = \sum_{j=1}^{k} \boldsymbol{w}_j \odot (\rho_Y(g_j)\boldsymbol{Y}).$$

Theorem 1. *The encoder and decoder in equations 10 and 11 are part-equivariant.*

In practice we work with a smoothed weighting matrix allowing values in $[0, 1]$, *i.e.*, $\boldsymbol{W} \in [0, 1]^{n\times k}$, losing some of this exact part equivariance for better treatment of transition areas between parts.

4. Implementation details

In this section we provide the main implementation details, further details can be found in the supplementary.

Mesh $\rightarrow$ mesh. The backbone architectures for ϕ, ψ is a 6 layer GNN, exactly as used in [22]; The specific dimensions of layers and hidden features for each experiment is detailed in the supplementary appendix. We denote the learnable parameters of both networks by θ. The training loss is the standard autoencoder reconstruction loss of the form

$$\mathcal{L}_{\text{rec}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| \Psi(\Phi(\boldsymbol{X}^{(i)})) - \boldsymbol{X}^{(i)} \right\|_F \tag{12}$$

where $\|\cdot\|_F$ is the Frobenious norm and $\left\{\boldsymbol{X}^{(i)}\right\}_{i=1}^{N} \subset \mathbb{R}^{n\times 3}$ is a batch drawn from the shape space's training set.

Point cloud $\rightarrow$ implicit. The backbone encoder architecture ϕ is exactly as in [32] constructed of PointNet [40] with 4 layers. The decoder is an MLP as in [3] with 8 layers with 512 features each. We trained a VAE where the latent space is $Z = \mathbb{R}^{d+m+d\times 3}$ containing codes of the form $(\boldsymbol{\mu}, \boldsymbol{\eta})$, where $\boldsymbol{\mu} \in \mathbb{R}^{m+d\times 3}$ is the latent mean, and $\boldsymbol{\eta} \in \mathbb{R}^d$ is the invariant latent log-standard-deviation. For training the VAE we use a combination of two losses

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{sald}}(\theta) + 0.001\mathcal{L}_{\text{vae}}(\theta), \tag{13}$$

where $\mathcal{L}_{\text{sald}}$ is the SALD loss [3],

$$\mathcal{L}_{\text{sald}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\int_{\Omega} \tau(\Psi(\mathcal{N}(\boldsymbol{\mu}^{(i)}, \boldsymbol{\eta}^{(i)})), h)(\boldsymbol{x})d\boldsymbol{x} \tag{14}$$

where $(\boldsymbol{\mu}^{(i)}, \boldsymbol{\eta}^{(i)}) = \Phi(\boldsymbol{X}^{(i)})$, $\mathcal{N}(\boldsymbol{a}, \boldsymbol{b})$ is an axis aligned Gaussian i.i.d. sample with mean $\boldsymbol{a}$ and standard deviation $\exp(\text{diag}(\boldsymbol{b}))$. $h(\cdot)$ is the unsigned distance function to $\boldsymbol{X}^{(i)}$, and $\tau(f, g)(\boldsymbol{x}) = ||f(\boldsymbol{x})| - g(\boldsymbol{x})| + \min\{\|\nabla f(\boldsymbol{x}) - \nabla g(\boldsymbol{x})\|_2, \|\nabla f(\boldsymbol{x}) + \nabla g(\boldsymbol{x})\|_2\}$. The domain of the above integral, $\Omega \subset \mathbb{R}^3$, is set according to the scene's bounding box. In practice, the integral is approximated using Monte-Carlo sampling. Note that this reconstruction loss is unsupervised (namely, using only the input raw point cloud). The VAE loss is defined also as in [3] by

$$\mathcal{L}_{\text{vae}}(\theta) = \sum_{i=1}^{N} \left\|\boldsymbol{\mu}^{(i)}\right\|_1 + \left\|\boldsymbol{\eta}^{(i)} + \boldsymbol{1}\right\|_1, \tag{15}$$

where $\|\cdot\|_1$ denotes the 1-norm.

Method	I	z	$SO(3)$
AE	5.16	9.96	15.41
AE-Aug	5.22	5.86	5.12
Ours	**4.39**	**4.35**	**4.66**

Table 1. Global Euclidean mesh$\rightarrow$mesh shape space experiment; MSE error (lower is better) in three test versions of the DFAUST [6] dataset, see text for details.

5. Experiments

We have tested our FA shape space learning framework under two kinds of symmetries G: global Euclidean transformations and piecewise Euclidean transformations.

5.1. Global Euclidean

In this case, we tested our method both in the mesh $\rightarrow$ mesh and the point-cloud $\rightarrow$ implicit settings.

Mesh $\rightarrow$ mesh. In this experiment we consider the DFaust dataset [6] of human meshes parameterized with SMPL [30]. The dataset consists of 41,461 human shapes where a random split is used to generate a training set of 37,197 models, and a test set of 4,264 models. We used the same generated data and split as in [22]. We generated two additional test sets of randomly oriented models: randomly rotated models about the up axis (uniformly), denoted by z, and randomly rotated models (uniformly), denoted by $SO3$. We denote the original, aligned test set by I. We compare our global Euclidean mesh$\rightarrow$mesh autoencoder versus the following baselines: Vanilla Graph autoencoder, denoted by AE; and the same AE trained with random rotations augmentations, denoted by AE-Aug. Note that the architecture used for AE and AE-Aug is the same backbone architecture used for our FA architecture. Table 1 reports the average per-vertex euclidean distance (MSE) on the various test sets: I, z and $SO3$. Note that FA compares favorably to the baselines in all tests.

Point cloud $\rightarrow$ implicit. In this experiment we consider the CommonObject3D dataset [42] that contains 19k objects from 50 different classes. We have used only the objects' point clouds extracted from videos using COLMAP [45]. The point clouds are very noisy and partial (see

Method	teddy bear $d_C^{\rightarrow}$	teddy bear d_C	bottle $d_C^{\rightarrow}$	bottle d_C	suitcase $d_C^{\rightarrow}$	suitcase d_C	banana $d_C^{\rightarrow}$	banana d_C
VAE	5.11	2.611	0.419	0.225	0.619	0.341	0.309	0.177
VN [15]	0.047	**0.421**	0.638	0.334	0.348	0.218	0.157	0.087
Ours	**0.046**	0.451	**0.226**	**0.129**	**0.079**	**0.086**	**0.118**	**0.074**

Table 2. Global Euclidean point cloud $\rightarrow$ implicit shape space experiment; CommonObject3D [42] dataset.

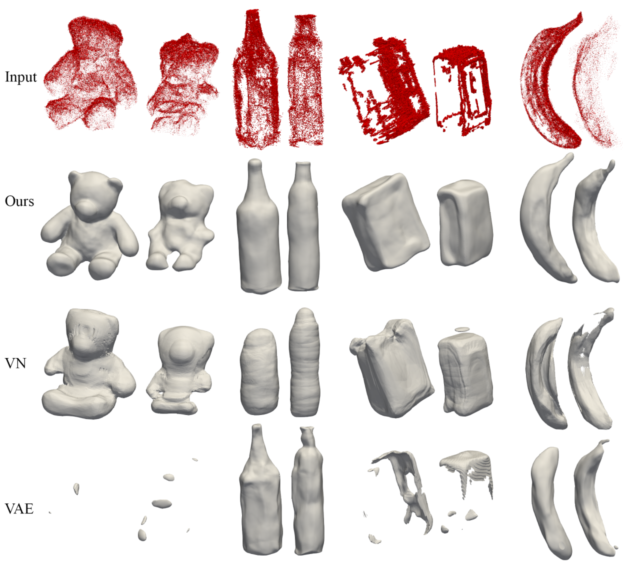

Figure 2. Global Euclidean point cloud $\rightarrow$ implicit, qualitative test results; CommonObject3D [42] dataset.

e.g., Figure 2, where input point clouds are shown in red), providing a "real-life" challenging dataset. Note that we have not used any other supervision for the implicit learning. We have used 4 object catagories: teddy bear (747 point clouds), bottle (296 point clouds), suitcase (480 point clouds), and banana (197 point clouds). We have divided each category to train and test sets randomly based on a 70%-30% split. We compare to the following baselines: Variational Autoencoder, denoted by VAE; Vector Neurons [15] version of this VAE, denoted VN. We used the official VN implementation. For our method we used an FA version of the same baseline VAE architecture. Table 2 reports two error metrics on the test set: $d_C^{\rightarrow}$ that denotes the one sided Chamfer distance from the input point cloud to the generated shape and d_C that denotes the symmetric Chamfer distance (see supplementary for exact definitions). Note that our method improves the symmetric Chamfer metric in almost all cases. Figure 2 shows some typical reconstructed implicits (after contouring) in each category, along with the input test point cloud (in red). Qualitatively our framework provides a more faithful reconstruction, even in this challenging noisy scenario without supervision. Note that for the "teddy bear" class we provide a visually improved reconstruction that is not apparent in the qualitative score due to high noise and outliers in the input point clouds.

5.2. Piecewise Euclidean

Mesh $\rightarrow$ mesh. In this experiment we consider three different datasets: DFaust [6], SMAL [62] and MANO [43]. For the DFaust dataset we used train-test splits in an increasing level of difficulty: random split of train-test taken from [22], as described above; unseen random pose - removing a random (different) pose sequence from each human and using it as test; and unseen pose - removing the *same* pose sequence from all humans and using it as test. The SMAL dataset contains a four-legged animal in different poses. We use the data generated in [22] of 400 shapes, randomly split into a 300 train, 100 test sets. The MANO dataset contains 3D models of realistic human hands in different poses. Using the MANO SMPL model we generated 150 shapes, randomly split into a 100 train and 50 test sets. We compared our method to the following baselines: Vanilla autoencoder, denoted by AE and ARAPReg [22] that reported state of the art results on this dataset. Note that both the AE and our method use the same backbone architecture. ARAPReg, report autodecoder to be superior in their experiments and therefore we compared to that version. Note that all compared methods have the same (backbone) decoder architecture. Figure 3 shows typical reconstruction results on the tests sets: green marks the random (easy) split; orange marks the random unseen pose split; and red marks the global unseen pose split. Note our method is able to produce very high-fidelity approximations of the ground truth, visually improving artifacts, noise and inaccuracies in the baselines (zoom in to see details). Lastly, we note that we use also partition skinning weight matrix (defined on the single rest-pose model) as an extra supervision not used by ARAPReg.

Method	random	unseen random pose	unseen pose	SMAL	MANO
AE	5.45	7.99	6.27	9.11	1.34
ARAPReg	4.52	7.77	3.38	6.68	1.15
Ours	**1.68**	**1.89**	**1.90**	**2.44**	**0.86**

Table 3. Piecewise Euclidean mesh $\rightarrow$ mesh experiment; MSE error (lower is better); DFaust [6], SMAL [62] and MANO [43] datasets.

Interpolation in shape space. In this experiment we show qualitative results for interpolating two latent codes $\boldsymbol{Z}^{(j)} = (\boldsymbol{q}^{(j)}, \boldsymbol{Q}^{(j)}) \in Z$, $j = 0, 1$, computed with our encoder for two input shapes $\boldsymbol{X}^{(j)}$, $j = 0, 1$. We use the encoder and decoder learned in the "unseen pose" split described above. Since Z is an equivariant feature space, if $\boldsymbol{X}^{(1)}$ is an Euclidean transformed version of $\boldsymbol{X}^{(0)}$, *i.e.*, $\boldsymbol{X}^{(1)} = \rho_X(g)\boldsymbol{X}^{(0)}$, then equivariance would mean that $\boldsymbol{Z}^{(1)} = \rho_Z(g)\boldsymbol{Z}^{(0)}$. Therefore interpolation in this case should be done by finding the optimal rotation and translation between the equivariant parts of $\boldsymbol{Z}^{(0)}$ and $\boldsymbol{Z}^{(1)}$ and continuously rotating and translating $\boldsymbol{Z}^{(0)}$ into $\boldsymbol{Z}^{(1)}$. This can be done using the closed form solution to the rotational

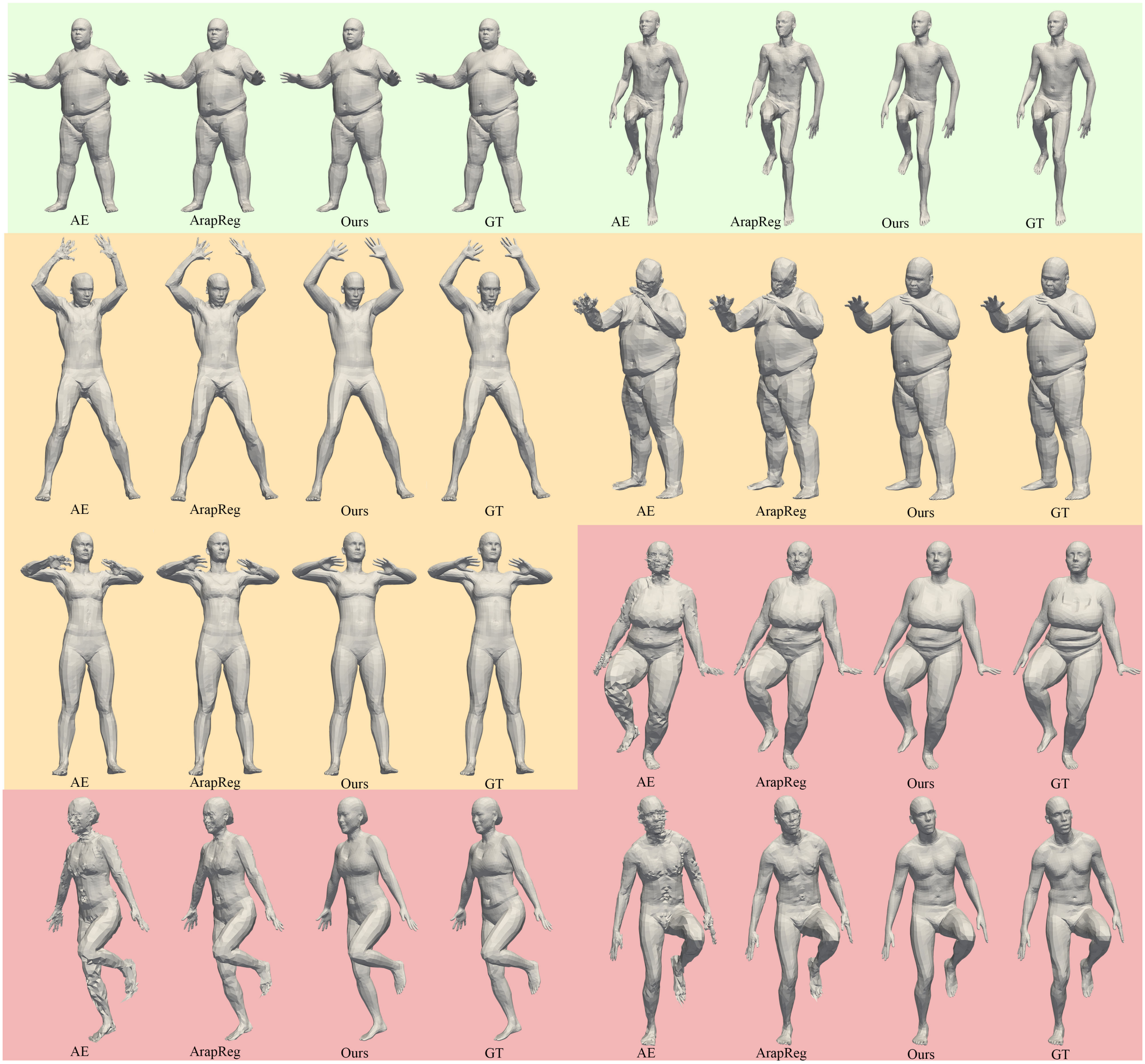

Figure 3. Piecewise Euclidean mesh → mesh, qualitative results; DFaust [6] dataset. Colors mark different splits: green is the random (easy) split; orange is the unseen random pose split; and red is the unseen pose split, see text for details. Our method demonstrates consistently high-quality results across splits of different difficulty levels.

Procrustes problem (see *e.g.*, [46, 61]). For two general codes $\boldsymbol{Z}_j$ we use this procedure while adding linearly the residual difference after cancelling the optimal rotation and translations between the codes. In the supplementary we provide the full derivation of this interpolation denoted $\boldsymbol{Z}^t$, $t \in [0, 1]$. Figure 4 shows the result of decoding the interpolated latent codes $\boldsymbol{Z}^t$, $t \in [0, 1]$ with the learned decoder. Note that both shape and pose gracefully and naturally change along the interpolation path.

Comparison with implicit pose-conditioned methods. Lastly, we have trained our piecewise Euclidean mesh → mesh framework on the DFaust subset of AMASS [31]. Following the protocol defined in [9], we trained our model on each of the 10 subjects and tested it on the "within distribution" SMPL [30] generated data from SNARF [9], as well as on their "out of distribution" test from PosePrior [1] dataset. We use both SNARF [9] and NASA [14] as baselines. Table 4 reports quantitative error metrics: The Intersection over Union (IoU) using sampling within the bound-

	Within Distribution						Out of Distribution					
	IoU bbox			IoU surface			IoU bbox			IoU surface		
	NASA	SNARF	Ours	NASA	SNARF	Ours	NASA	SNARF	Ours	NASA	SNARF	Ours
50002	96.56%	97.50%	**98.67%**	84.02%	89.57%	**93.28%**	87.71%	94.51%	**96.76%**	60.25%	79.75%	**85.06%**
50004	96.31%	97.84%	**98.64%**	85.45%	91.16%	**94.57%**	86.01%	95.61%	**96.19%**	62.53%	83.34%	**85.84%**
50007	96.72%	97.96%	**98.62%**	86.28%	91.02%	**94.11%**	80.22%	93.99%	**95.31%**	51.82%	77.08%	**81.91%**
50009	94.96%	96.68%	**97.75%**	84.52%	88.19%	**92.84%**	78.15%	91.22%	**94.75%**	55.86%	75.84%	**84.60%**
50020	95.75%	96.27%	**97.61%**	87.57%	88.81%	**92.60%**	83.06%	93.57%	**95.17%**	62.01%	81.37%	**85.66%**
50021	95.92%	96.86%	**98.55%**	87.01%	90.16%	**95.38%**	81.80%	93.76%	**96.35%**	65.49%	81.49%	**88.86%**
50022	97.94%	97.96%	**98.39%**	91.91%	92.06%	**93.68%**	87.54%	94.67%	**96.12%**	70.23%	83.37%	**85.80%**
50025	95.50%	97.54%	**98.48%**	86.19%	91.25%	**94.74%**	83.14%	94.48%	**95.99%**	60.88%	82.48%	**86.58%**
50026	96.65%	97.64%	**98.61%**	87.72%	91.09%	**94.64%**	84.58%	94.13%	**96.45%**	59.78%	80.01%	**87.10%**
50027	95.53%	96.80%	**97.95%**	86.13%	89.47%	**93.46%**	83.97%	93.76%	**95.61%**	61.82%	81.81%	**86.60%**

Table 4. Piecewise Euclidean mesh → mesh, comparison to implicit articulation methods. DFaust [6] and PosePrior [1] datasets.

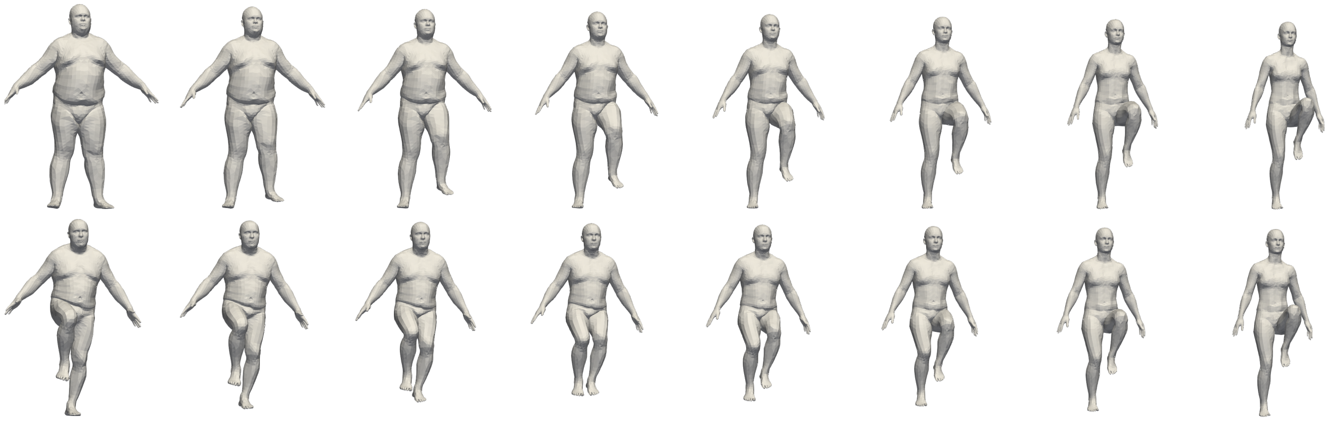

Figure 4. Interpolation in equivariant latent space between two test examples from the "unseen pose" split (leftmost and rightmost columns).

ing box (bbox) and near the surface, see supplementary for more details. Figure 5 shows comparison with SNARF of test reconstructions from the "out of distribution" set. We note that our setting is somewhat easier than that faced by SNARF and NASA (we perform mesh → mesh learning using a fixed mesh connectivity and skinning weights; the skinning weights is used by NASA and learned by SNARF).

Nevertheless, we do not assume or impose anything on the latent space besides Euclidean equivariance, not use an input pose explicitly, and train only with a simple reconstruction loss (see equation 12). Under this disclaimer we note we improve the reconstruction error both qualitatively and quantitatively in comparison to the baselines.

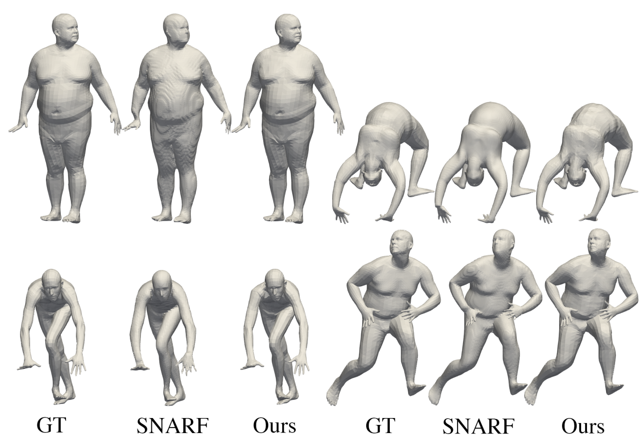

Figure 5. Comparison with SNARF on the "out of distribution" test set.

6. Limitations and future work

We have introduced a generic methodology for incorporating symmetries by construction into encoder and/or decoders in the context of shape space learning. Using Frame Averaging we showed how to construct expressive yet efficient equivariant autoencoders. We instantiated our framework for the case of global and piecewise Euclidean motions, as well as to mesh → mesh, and point cloud → implicit scenarios. We have achieved state of the art quantitative and qualitative results in all experiments.

Our method has several limitations: First in the mesh → mesh case we use fixed connectivity and skinning weights. Generalizing the piecewise Euclidean case to implicit representations, dealing with large scale scenes with multiple objects, or learning the skinning weights would be interesting future works. Trying to use linear blend skinning to define group action of $E(3)^k$ could be also interesting. Finally, using this framework to explore other symmetry types, combinations of geometry representations (including images, point clouds, implicits, and meshes), and different architectures could lead to exciting new methods to learn and use shape space in computer vision.

References

[1] Ijaz Akhter and Michael J Black. Pose-conditioned joint angle limits for 3d human pose reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1446–1455, 2015.

[2] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020.

[3] Matan Atzmon and Yaron Lipman. SALD: sign agnostic learning with derivatives. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[4] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[5] Matan Atzmon, David Novotny, Andrea Vedaldi, and Yaron Lipman. Augmenting implicit neural shape representations with explicit deformation fields. *arXiv preprint arXiv:2108.08931*, 2021.

[6] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[7] Paul Breiding, Khazhgali Kozhasov, and Antonio Lerario. On the geometry of the set of symmetric matrices with repeated eigenvalues. *Arnold Mathematical Journal*, 4(3):423–443, 2018.

[8] Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. Equivariant point network for 3d point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14514–14523, 2021.

[9] Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. *arXiv preprint arXiv:2104.03953*, 2021.

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[11] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020.

[12] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *International Conference on Learning Representations*, 2018.

[13] Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.

[14] Boyang Deng, John P Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa neural articulated shape approximation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 612–628. Springer, 2020.

[15] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021.

[16] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018.

[17] Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. In *International Conference on Learning Representations*, 2020.

[18] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[19] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[20] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5545–5554, 2019.

[21] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems 2020*, pages 3569–3579. 2020.

[22] Qixing Huang, Xiangru Huang, Bo Sun, Zaiwei Zhang, Junfeng Jiang, and Chandrajit Bajaj. Arapreg: An as-rigid-as possible regularization loss for learning deformable shape generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5815–5825, 2021.

[23] Timothy Jeruzalski, David IW Levin, Alec Jacobson, Paul Lalonde, Mohammad Norouzi, and Andrea Tagliasacchi. Nilbs: Neural inverse linear blend skinning. *arXiv preprint arXiv:2004.05980*, 2020.

[24] Boyi Jiang, Juyong Zhang, Jianfei Cai, and Jianmin Zheng. Disentangled human body embedding based on deep hierarchical neural network. *IEEE transactions on visualization and computer graphics*, 26(8):2560–2575, 2020.

[25] Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. Surface networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2540–2548, 2018.

[26] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[27] Yaron Lipman. Phase transitions, distance functions, and implicit neural representations. *arXiv preprint arXiv:2106.07689*, 2021.

[28] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE con-*

ference on computer vision and pattern recognition, pages 1886–1895, 2018.

[29] Min Liu, Fupin Yao, Chiho Choi, Ayan Sinha, and Karthik Ramani. Deep learning 3d shapes using alt-az anisotropic 2-sphere convolution. In *International Conference on Learning Representations*, 2018.

[30] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015.

[31] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5442–5451, 2019.

[32] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[33] Marko Mihajlovic, Yan Zhang, Michael J Black, and Siyu Tang. Leap: Learning articulated occupancy of people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10461–10471, 2021.

[34] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5379–5389, 2019.

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[36] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

[37] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020.

[38] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[39] Omri Puny, Matan Atzmon, Heli Ben-Hamu, Edward J Smith, Ishan Misra, Aditya Grover, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021.

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[41] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[42] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10901–10911, 2021.

[43] David W Romero and Jean-Baptiste Cordonnier. Group equivariant stand-alone self-attention for vision. In *ICLR*, 2021.

[44] Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J. Black. SCANimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[45] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[46] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[47] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[48] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[49] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[50] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2598–2606, 2018.

[51] Michael Wand, Philipp Jenke, Qixing Huang, Martin Bokeloh, Leonidas Guibas, and Andreas Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Symposium on Geometry processing*, pages 49–58, 2007.

[52] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[53] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

[54] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

[55] Zelin Xiao, Hongxin Lin, Renjie Li, Lishuai Geng, Hongyang Chao, and Shengyong Ding. Endowing deep 3d

models with rotation invariance based on principal component analysis. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

[56] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[57] Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, pages 1–68, 2021.

[58] Ruixuan Yu, Xin Wei, Federico Tombari, and Jian Sun. Deep positional and relational feature learning for rotation-invariant point cloud analysis. In *European Conference on Computer Vision*, pages 217–233. Springer, 2020.

[59] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

[60] Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*, pages 204–213. IEEE, 2019.

[61] Yuanchen Zhu and Steven J Gortler. 3d deformation using moving least squares. 2007.

[62] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6365–6373, 2017.