Shape-Pose Disentanglement using SE(3)-equivariant Vector Neurons

Oren Katzir[1], Dani Lischinski[2], and Daniel Cohen-Or[1]

[1] Tel-Aviv University
[2] Hebrew University of Jerusalem

Abstract. We introduce an unsupervised technique for encoding point clouds into a canonical shape representation, by disentangling shape and pose. Our encoder is stable and consistent, meaning that the shape encoding is purely pose-invariant, while the extracted rotation and translation are able to semantically align different input shapes of the same class to a common canonical pose. Specifically, we design an auto-encoder based on Vector Neuron Networks, a rotation-equivariant neural network, whose layers we extend to provide translation-equivariance in addition to rotation-equivariance only. The resulting encoder produces pose-invariant shape encoding by construction, enabling our approach to focus on learning a consistent canonical pose for a class of objects. Quantitative and qualitative experiments validate the superior stability and consistency of our approach.

Keywords: point clouds, canonical pose, equivariance, shape-pose disentanglement

1 Introduction

Point clouds reside at the very core of 3D geometry processing, as they are acquired at the beginning of the 3D processing pipeline and usually serve as the raw input for shape analysis or surface reconstruction. Thus, understanding the underlying geometry of a point cloud has a profound impact on the entire 3D processing chain. This task, however, is challenging since point clouds are unordered, and contain neither connectivity, nor any other global information.

In recent years, with the emergence of neural networks, various techniques have been developed to circumvent the challenges of analyzing and understanding point clouds [19,20,27,11,15,23,14,31]. However, most methods rely on *pre-aligned* datasets, where the point clouds are normalized, translated and oriented to have the same pose.

In this work, we present an unsupervised technique to learn a canonical shape representation by disentangling shape, translation, and rotation. Essentially, the canonical representation is required to meet two conditions: *stability* and *consistency*. The former means that the shape encoding should be invariant to any rigid transformation of the same input, while the latter means that different shapes of the same class should be semantically aligned, sharing the same canonical pose.

Canonical alignment is not a new concept. Recently, Canonical Capsules [25] and Compass [24] proposed self-supervised learning of canonical representations using augmentations with Siamese training. We discuss these methods in more detail in the next section. In contrast, our approach is to extract a *pose-invariant* shape encoding, which is explicitly disentangled from the separately extracted translation and rotation.

Specifically, we design an auto-encoder, trained on an unaligned dataset, that encodes the input point cloud into three disentangled components: (i) a pose-invariant shape encoding, (ii) a rotation matrix and (iii) a translation vector. We achieve pure $\mathbf{SE}(3)$-invariant shape encoding and $\mathbf{SE}(3)$-equivariant pose estimation (enabling reconstruction of the input shape), by leveraging a novel extension of the recently proposed Vector Neuron Networks (VNN) [6]. The latter is an $\mathbf{SO}(3)$-equivariant neural network for point cloud processing, and while translation invariance could theoretically be achieved by centering the input point clouds, such approach is sensitive to noise, missing data and partial shapes. Therefore we propose an extension to VNN achieving $\mathbf{SE}(3)$-equivariance.

It should be noted that the shape encodings produced by our network are stable (i.e., pose-invariant) *by construction*, due to the use of $\mathbf{SE}(3)$-invariant layers. At the same time, the extracted rigid transformation is equivariant to the pose of the input. This enables the learning process to focus on the consistency across different shapes. Consistency is achieved by altering the input point cloud with a variety of simple shape augmentations, while keeping the pose fixed, allowing us to constrain the learned transformation to be invariant to the identity, (i.e., the particular shape), of the input point cloud.

Moreover, our disentangled shape and pose representation is not limited to point cloud decoding, but can be combined with any 3D data decoder, as we demonstrate by learning a canonical implicit representation of our point cloud utilizing occupancy networks [16].

We show, both qualitatively and quantitatively, that our approach leads to a stable, consistent, and purely $\mathbf{SE}(3)$-invariant canonical representation compared to previous approaches.

2 Background and Related Work

2.1 Canonical representation

A number of works proposed techniques to achieve learnable canonical frames, typically requiring some sort of supervision [21,17,10]. Recently, two unsupervised methods were proposed: Canonical Capsules [25] and Compass [24]. Canonical Capsules [25] is an auto-encoder network that extracts positions and pose-invariant descriptors for k capsules, from which the input shape may be reconstructed. Pose invariance and equivariance are achieved only implicitly via Siamese training, by feeding the network with pairs of rotated and translated versions of the same input point cloud.

Compass [24] builds upon spherical CNN [4], a semi-equivariant $\mathbf{SO}(3)$ network, to estimate the pose with respect to the canonical representation. It should

be noted that Compass is inherently tied to spherical CNN, which is not purely equivariant [4]. Thus, similarly to Canonical Capsules, Compass augments the input point cloud with a rotated version to regularize an equivariant pose estimation. It should be noted that neither method guarantees pure equivariance.

Similarly to Canonical Capsules, we employ an auto-encoding scheme to disentangle pose from shape, i.e., the canonical representation, and similarly to Compass, we strive to employ an equivariant network, however, our network is $\mathbf{SE}(3)$-equivariant and not only $\mathbf{SO}(3)$-equivariant. More importantly, differently from these two approaches, the different branches of our network are $\mathbf{SE}(3)$-invariant or $\mathbf{SE}(3)$-equivariant *by construction*, and thus the learning process is free from the burden of enforcing these properties. Rather, the process focuses on learning a consistent shape representation in a canonical pose. Close to our work are Equi-pose [13] and a concurrent work ConDor [22]. Both methods use a relatively intricate equivariant backbone to estimate multiple proposal poses. Differently, we employ a simple equivariant network (VNN) and predict a single pose, easing the incorporation of our method with other SOTA methods for 3D shape reconstruction. We further discuss Equi-pose [13] in our suppl. material.

2.2 3D reconstruction

Our method reconstructs an input point cloud by disentangling the input 3D geometry into shape and pose. The encoder outputs a pose encoding and a shape encoding which is pose-invariant by construction, while the decoder reconstructs the 3D geometry from the shape encoding alone. Consequently, our architecture can be easily integrated into various 3D auto-encoding pipelines. In this work, we shall demonstrate our shape-pose disentanglement for point cloud encoding and implicit representation learning.

State-of-the-art point cloud auto-encoding methods rely on a folding operation of a template (optionally learned) hyperspace point cloud to the input 3D point cloud [30,9,7]. Following this approach, we employ AtlasNetV2 [7] which uses multiple folding operations from hyperspace patches to 3D coordinates, to reconstruct point clouds in a pose-invariant frame.

Implicit 3D representation networks [16,18,29] enable learning of the input geometry with high resolution and different mesh topology. We utilize occupancy networks [16] to learn an implicit pose-invariant shape representation.

2.3 Rotation-equivariance and Vector Neuron Network

The success of 2D convolutional neural networks (CNN) on images, which are equivariant to translation, drove a similar approach for 3D data with rotation as the symmetry group. Most works on 3D rotation-equivariance [8,4,26,28], focus on steerable CNNs [5], where each layer "steers" the output features according to the symmetry property (rotation and occasionally translation for 3D data). For example, Spherical CNNs [8,4] transform the input point cloud to a spherical signal, and use spherical harmonics filters, yielding features on $\mathbf{SO}(3)$-space.

Usually, these methods are tied with specific architecture design and data input which limit their applicability and adaptation to SOTA 3D processing.

Recently, Deng et al. [6] introduced Vector Neuron Networks (VNN), a rather light and elegant framework for $\mathbf{SO}(3)$-equivariance. Empirically, the VNN design performs on par with more complex and specific architectures. The key benefit of VNNs lies in their simplicity, accessibility and generalizability. Conceptually, any standard point cloud processing network can be elevated to $\mathbf{SO}(3)$-equivariance (and invariance) with minimal changes to its architecture.

Below we briefly describe VNNs and refer the reader to [6] for further details.

In VNNs the representation of a single neuron is lifted from a sequence of scalar values to a sequence of 3D vectors. A single vector neuron feature is thus a matrix $\mathbf{V} \in \mathbb{R}^{C\times 3}$, and we denote a collection of N such features by $\mathcal{V} \in \mathbb{R}^{N\times C\times 3}$. The layers of VNNs, which map between such collections, $f : \mathbb{R}^{N\times C\times 3} \to \mathbb{R}^{N\times C'\times 3}$, are equivariant to rotations $R \in \mathbb{R}^{3\times 3}$, that is:

$$f\left(\mathcal{V}R\right) = f\left(\mathcal{V}\right)R, \tag{1}$$

where $\mathcal{V}R = \{\mathbf{V}_n R\}_{n=1}^{N}$.

Ordinary linear layers fulfill this requirement, however, other non-linear layers, such as ReLU and max-pooling, do not. For ReLU activation, VNNs apply a truncation w.r.t to a learned half-space. Let $\mathbf{V}, \mathbf{V}' \in \mathbb{R}^{C\times 3}$ be the input and output vector neuron features of a single point, respectively. Each 3D vector $\mathbf{v}' \in \mathbf{V}'$ is obtained by first applying two learned matrices $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{1\times C}$ to project $\mathbf{V}$ to a feature $\mathbf{q} = \mathbf{QV} \in \mathbb{R}^{1\times 3}$ and a direction $\mathbf{k} = \mathbf{KV} \in \mathbb{R}^{1\times 3}$. To achieve equivariance, $\mathbf{v}' \in \mathbf{V}'$ is then defined by truncating the part of $\mathbf{q}$ that lies in the negative half-space of $\mathbf{k}$, as follows,

$$\mathbf{v}' = \begin{cases} \mathbf{q} & \text{if} \quad \langle \mathbf{q}, \mathbf{k} \rangle \geq 0, \\ \mathbf{q} - \left\langle \mathbf{q}, \frac{\mathbf{k}}{\|\mathbf{k}\|} \right\rangle \frac{\mathbf{k}}{\|\mathbf{k}\|} & \text{otherwise.} \end{cases} \tag{2}$$

In addition, VNNs employ rotation-equivariant pooling operations and normalization layers. We refer the reader to [6] for the complete definition.

Invariance layers can be achieved by inner product of two rotation-equivariant features. Let $\mathbf{V} \in \mathbb{R}^{C\times 3}$, and $\mathbf{V}' \in \mathbb{R}^{C'\times 3}$ be two equivariant features obtained from an input point cloud $\mathbf{X}$. Then rotating $\mathbf{X}$ by a matrix R, results in the features $\mathbf{V}R$ and $\mathbf{V}'R$, and

$$\langle \mathbf{V}R, \mathbf{V}'R \rangle = \mathbf{V}R(\mathbf{V}'R)^T = \mathbf{V}RR^T\mathbf{V}'^T = \mathbf{V}\mathbf{V}'^T = \langle \mathbf{V}, \mathbf{V}' \rangle. \tag{3}$$

Note that VNN is also reflection equivariant which may be beneficial for symmetrical objects, although we do not take advantage of this attribute directly.

In our work, we also utilize vector neurons, but we extend the different layers to be $\mathbf{SE}(3)$-equivariant, instead of $\mathbf{SO}(3)$-equivariant, as described in Section 3.1. This new design allow us to construct an $\mathbf{SE}(3)$-invariant encoder, which gradually disentangles the pose from the shape, first the translation and then the rotation, resulting in a pose-invariant shape encoding.

3 Method

We design an auto-encoder to disentangle shape, translation, and rotation. We wish the resulting representation to be stable, i.e., the shape encoding should be pose-invariant, and the pose $\mathbf{SE}(3)$-equivariant. At the same time, we wish multiple different shapes in the same class to have a consistent canonical pose. To achieve stability, we revisit VNNs and design new $\mathbf{SE}(3)$-equivariant and invariant layers, which we refer to as Vector Neurons with Translation (VNT). Consistency is then achieved by self-supervision, designed to preserve pose across shapes. In the following, we first describe the design of our new VNT layers. Next, we present our VNN and VNT-based auto-encoder architecture. Finally, we elaborate on our losses to encourage disentanglement of shape from pose in a consistent manner.

3.1 SE(3)-equivariant Vector Neuron Network

As explained earlier, Vector Neuron Networks (VNN) [6] provide a framework for $\mathbf{SO}(3)$-equivariant and invariant point cloud processing. Since a pose of an object consists of translation and rotation, $\mathbf{SE}(3)$-equivariance and invariance are needed for shape-pose disentanglement. While it might seem that centering the input point cloud should suffice, note that point clouds are often captured with noise and occlusions, leading to missing data and partial shapes, which may significantly affect the global center of the input. Specifically, for canonical representation learning, a key condition is consistency across different objects, thus, such an approach assumes that the center of the point cloud is consistently semantic between similar but different objects, which is hardly the case. Equivariance to translation, on the other-hand, allows identifying local features in different locations with the same filters, without requiring global parameters.

Therefore, we revisit the Vector Neuron layers and extend them to Vector Neurons with Translation (VNT), thereby achieving $\mathbf{SE}(3)$-equivariance.

Linear layers: While linear layers are by definition rotation-equivariant, they are not translation-equivariant. Following VNN, our linear module $f_{\text{lin}}(\cdot; \mathbf{W})$ is defined via a weight matrix $\mathbf{W} \in \mathbb{R}^{C' \times C}$, acting on a vector-list feature $\mathbf{V} \in \mathbb{R}^{C \times 3}$. Let $R \in \mathbb{R}^{3 \times 3}$ be a rotation matrix and $T \in \mathbb{R}^{1 \times 3}$ a translation vector. For $f_{\text{lin}}(\cdot; \mathbf{W})$ to be $\mathbf{SE}(3)$-equivariant, the following must hold:

$$f_{\text{lin}}(\mathbf{V}R + \mathbb{1}_C T) = \mathbf{W}\mathbf{V}R + \mathbb{1}_{C'} T, \tag{4}$$

where $\mathbb{1}_C = [1, 1, \ldots, 1]^T \in \mathbb{R}^{C \times 1}$ is a column vector of length C. A sufficient condition for (4) to hold is achieved by constraining each row of $\mathbf{W}$ to sum to one. Formally, $\mathbf{W} \in \mathcal{W}^{C' \times C}$, where

$$\mathcal{W}^{C' \times C} = \left\{ \mathbf{W} \in \mathbb{R}^{C' \times C} \mid \textstyle\sum_{j=1}^{C} w_{i,j} = 1 \quad \forall i = 1, \ldots, C' \right\}, \tag{5}$$

See our supplementary material for a complete proof.

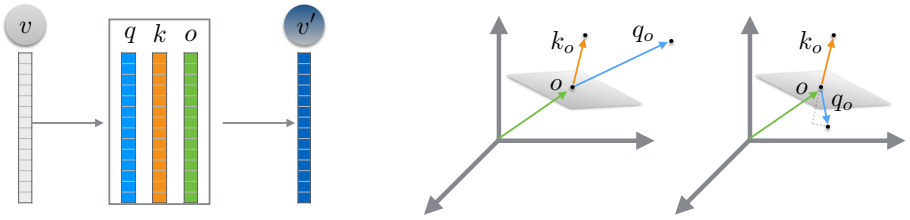

Fig. 1. Vector Neuron Translation equivariant non linear layer. We learn for each input point feature $\mathbf{v}$, three component $\mathbf{o}, \mathbf{q}, \mathbf{k}$, and interpret them as an origin $\mathbf{o}$, a feature $\mathbf{q_o} = \mathbf{q} - \mathbf{o}$ and a direction $\mathbf{k_o} = \mathbf{k} - \mathbf{o}$. Similarly to VNN operation, the feature component of $\mathbf{q_o}$ which is in the half-space defined by $-\mathbf{k_o}$ is clipped. In-addition, we translate the feature by the learned origin $\mathbf{o}$ outputting the $\mathbf{v}'$.

Non-linear layers: We extend each non-linear VNN layer to become $\mathbf{SE}(3)$-equivariant by adding a learnable origin. More formally, for the ReLU activation layer, given an input feature list $\mathbf{V} \in \mathbb{R}^{C\times 3}$, we learn three (rather than two) linear maps, $\mathbf{Q}, \mathbf{K}, \mathbf{O} \in \mathcal{W}^{1\times C}$ projecting the input to $\mathbf{q}, \mathbf{k}, \mathbf{o} \in \mathbb{R}^{1\times 3}$. The feature and direction are defined w.r.t the origin $\mathbf{o}$, i.e., the feature is given by $\mathbf{q_o} = \mathbf{q} - \mathbf{o}$, while the direction is given by $\mathbf{k_o} = \mathbf{k} - \mathbf{o}$, as illustrated in Fig. 1. The ReLU is applied by clipping the part of $\mathbf{q_o}$ that resides behind the plane defined by $\mathbf{k_o}$ and $\mathbf{o}$, i.e.,

$$\mathbf{v}' = \begin{cases} \mathbf{o} + \mathbf{q_o} & \text{if} \quad \langle \mathbf{q_o}, \mathbf{k_o} \rangle \geq 0, \\ \mathbf{o} + \mathbf{q_o} - \left\langle \mathbf{q_o}, \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|} \right\rangle \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|}, & \text{otherwise.} \end{cases}, \tag{6}$$

Note that $\mathbf{o} + \mathbf{q_o} = \mathbf{q}$, and that $\mathbf{K}, \mathbf{O}$ may be shared across the elements of $\mathbf{V}$.

It may be easily seen that we preserve the equivariance w.r.t $\mathbf{SO}(3)$ rotations, as well as translations. In the same manner, we extend the $\mathbf{SO}(3)$-equivariant VNN maxpool layer to become $\mathbf{SE}(3)$-equivariant. We refer the reader to the supplementary material for the exact adaptation and complete proof.

Translation-invariant layers: Invariance to translation can be achieved by subtracting two $\mathbf{SE}(3)$-equivariant features. Let $\mathbf{V}, \mathbf{V}' \in \mathbb{R}^{C\times 3}$ be two $\mathbf{SE}(3)$-equivariant features obtained from an input point cloud $\mathbf{X}$. Then, rotating $\mathbf{X}$ by a matrix R and translating by T, results in the features $\mathbf{V}R + \mathbb{1}_C T$ and $\mathbf{V}'R + \mathbb{1}_C T$, whose difference is translation-invariant:

$$(\mathbf{V}R + \mathbb{1}_C T) - (\mathbf{V}'R + \mathbb{1}_C T) = (\mathbf{V} - \mathbf{V}')\, R \tag{7}$$

Note that the resulting feature is still rotation-equivariant, which enables to process it with VNN layers, further preserving $\mathbf{SO}(3)$-equivariance.

3.2 SE(3)-equivariant Encoder-Decoder

We design an auto-encoder based on VNT and VNN layers to disentangle pose from shape. Thus, our shape representation is pose-invariant (i.e., stable), while

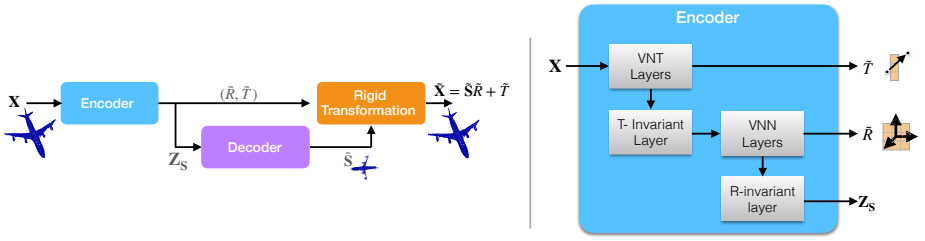

Fig. 2. The architecture of our auto-encoder for shape-pose disentanglement. The auto-encoder (left) disentangles the input point cloud $\mathbf{X}$ to rotation $\tilde{R}$, translation $\tilde{T}$ and a canonical representation $\tilde{\mathbf{S}}$. The shape encoding $\mathbf{Z_s}$ is invariant by construction to the pose, while the learned rotation and translation are equivariant to it. Our encoder (right) learns features that are initially equivariant to the pose, and gradually become invariant to it, first to translation and then to rotation, eventually yielding pose invariant features $\mathbf{Z_s}$.

our pose estimation is $\mathbf{SE}(3)$-pose-equivariant, by construction. The decoder, which can be an arbitrary 3D decoder network, reconstructs the 3D shape from the invariant features.

The overall architecture of our AE is depicted in Fig. 2. Given an input point cloud $\mathbf{X} \in \mathbb{R}^{N\times 3}$, we can represent it as a rigid transformation of an unknown canonical representation $\mathbf{S} \in \mathbb{R}^{N\times 3}$:

$$\mathbf{X} = \mathbf{S}R + \mathbb{1}_N T, \tag{8}$$

where $\mathbb{1}_N = [1, 1, \ldots, 1]^T \in \mathbb{R}^{N\times 1}$ is a column vector of length N, $R \in \mathbb{R}^{3\times 3}$ is a rotation matrix and $T \in \mathbb{R}^{1\times 3}$ is a translation vector.

Our goal is to find the shape $\mathbf{S}$, which is by definition pose-invariant and should be consistently aligned across different input shapes. To achieve this goal, we use an encoder that first estimates the translation $\tilde{T}$ using translation-equivariant VNT layers, then switches to a translation-invariant representation from which the rotation $\tilde{R}$ is estimated using rotation-equivariant VNN layers. Finally, the representation is made rotation-invariant and the shape encoding $\mathbf{Z_s}$ is generated. A reconstruction loss is computed by decoding $\mathbf{Z_s}$ into the canonically-positioned shape $\tilde{\mathbf{S}}$ and applying the extracted rigid transformation. In the following we further explain our encoder architecture and the type of decoders used.

SE(3)-equivariant Encoder Our encoder is composed of rotation and translation equivariant and invariant layers as shown in Fig. 3. We start by feeding $\mathbf{X}$ through linear and non-linear VNT layers yielding $\mathbf{X_{RT}} \in \mathbb{R}^{N\times C\times 3}$, where the RT subscript indicates $\mathbf{SE}(3)$-equivariant features, as described in Section 3.1.

$\mathbf{X_{RT}}$ is then fed-forward through additional VNT layers resulting in a single vector neuron per point $\bar{\mathbf{X}}_{\mathbf{RT}} \in \mathbb{R}^{N\times 1\times 3}$. We mean-pool the features to produce a 3D SE(3)-equivariant vector as our translation estimation, as shown in the upper branch of Fig. 3, yielding $\tilde{T} = f_T(\mathbf{X}) = f_T(\mathbf{S})R + T \in \mathbb{R}^{1\times 3}$, where we denote by $f_T : \mathbb{R}^{N\times 3} \to \mathbb{R}^{1\times 3}$ the aggregation of the VNT-layers from the input point

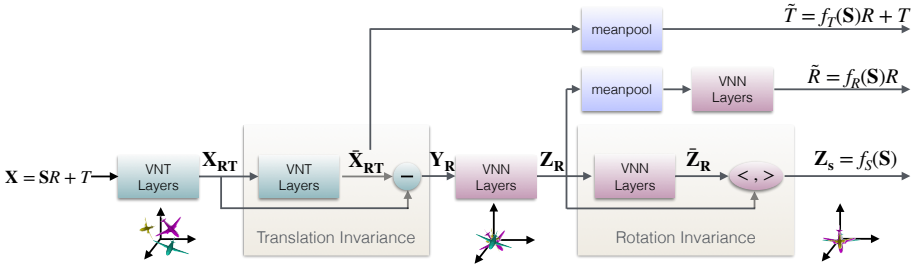

Fig. 3. The architecture of our encoder. The representation of the input point cloud $\mathbf{X}$ yields a pose-invariant (bottom branch) shape encoding $\mathbf{Z_s}$ in two steps: first, making it invariant to translation and then to rotation. At the same time, the learned rigid transformation $(\tilde{R}, \tilde{T})$ (top and middle branches) is equivariant to the input pose. The small rendered planes at the bottom, illustrate the alignment at each stage.

cloud $\mathbf{X}$ to the estimated $\tilde{T}$, thus, it is a translation and rotation equivariant network. In addition, as explained in Section 3.1, the following creates translation invariant features, $\mathbf{Y_R} = \mathbf{X_{RT}} - \bar{\mathbf{X}}_{\mathbf{RT}} \in \mathbb{R}^{N \times C \times 3}$.

While $\mathbf{Y_R}$ is translation invariant, it is still rotation equivariant, thus, we can proceed to further process $\mathbf{Y_R}$ with VNN layers, resulting in (deeper) rotation equivariant features $\mathbf{Z_R} \in \mathbb{R}^{N \times C' \times 3}$.

Finally, $\mathbf{Z_R}$ is fed forward through a VNN rotation-invariant layer as explained in Section 2.3, resulting in a shape encoding, $\mathbf{Z_s}$, which is by construction pose invariant. Similar to the translation reconstruction, the rotation is estimated by mean pooling $\mathbf{Z_R}$ and feeding it through a single VN linear layer yielding $\tilde{R} = f_R(\mathbf{X}) = f_R(\mathbf{S})R \in \mathbb{R}^{3 \times 3}$, where $f_R : \mathbb{R}^{N \times 3} \to \mathbb{R}^{3 \times 3}$ denotes the aggregation of the layers from the input point cloud $\mathbf{X}$ to the estimated rotation $\tilde{R}$ and, as such, it is a rotation-equivariant network. The entire encoder architecture is shown in Fig. 3 and we refer the reader to our supplementary for a detailed description of the layers.

Decoder The decoder is applied on the shape encoding $\mathbf{Z_s}$ to reconstruct the shape $\tilde{\mathbf{S}}$. We stress again that $\tilde{\mathbf{S}}$ is invariant to the input pose, regardless of the training process. Motivated by the success of folding networks [30,9,7] for point clouds auto-encoding, we opt to use AtlasNetV2 [7] as our decoder, specifically using the point translation learning module. For implicit function reconstruction, we follow Occupancy network decoder [16]. Please note, that our method is not coupled with any decoder structure.

3.3 Optimizing for shape-pose disentanglement

While our auto-encoder is pose-invariant by construction, the encoding has no explicit relation to the input geometry. In the following we detail our losses to encourage a rigid relation between $\tilde{\mathbf{S}}$ and X, and for making $\tilde{\mathbf{S}}$ consistent across different objects.

Rigidity To train the reconstructed shape $\tilde{\mathbf{S}}$ to be isometric to the input point cloud $\mathbf{X}$, we enforce a rigid transformation between the two, namely $\mathbf{X} = \tilde{\mathbf{S}}\tilde{R} + \mathbb{1}_N\tilde{T}$.

For point clouds auto-encoding we have used the Chamfer Distance (CD):

$$\mathcal{L}_{rec} = CD\left(\mathbf{X}, \tilde{\mathbf{S}}\tilde{R} + \mathbb{1}_N\tilde{T}\right), \tag{9}$$

Please note that other tasks such as implicit function reconstruction use equivalent terms, as we detail in our supplementary files.

In addition, while $\tilde{R} = f_R(\mathbf{X})$ is rotation-equivariant we need to constrain it to SO(3), and we do so by adding an orthonormal term:

$$\mathcal{L}_{ortho} = \|I - \tilde{R}\tilde{R}^T\|_2^2 \tag{10}$$

where $\|\cdot\|_2$ is mean square error (MSE) loss.

Consistency Now, our shape reconstruction $\tilde{\mathbf{S}}$ is isometric to $\mathbf{X}$ and it is invariant to T and R. However, there is no guarantee that the pose of $\tilde{\mathbf{S}}$ would be consistent across different instances.

Assume two different point clouds $\mathbf{X_1}, \mathbf{X_2}$ are aligned. If their canonical representations $\mathbf{S_1}, \mathbf{S_2}$ are also aligned, then they have the same rigid transformation w.r.t their canonical representation and vice versa, i.e., $\mathbf{X_i} = \mathbf{S_i}R + \mathbb{1}_N T$, $i = 1, 2$. To achieve such consistency, we require:

$$(f_T(\mathbf{X_1}), f_R(\mathbf{X_1})) = (f_T(\mathbf{X_2}), f_R(\mathbf{X_2}))\,. \tag{11}$$

We generate such pairs of aligned point clouds, by augmenting the input point cloud $\mathbf{X}$ with several simple augmentation processes, which do not change the pose of the object. In practice, we have used Gaussian noise addition, furthest point sampling (FPS), patch removal by k-nn (we select one point randomly and remove k of its nearest neighbors) and re-sampling of the input point cloud. We then require that the estimated rotation and translation is the same for the original and augmented versions,

$$\mathcal{L}^{aug}_{consist} = \sum_{A\in\mathcal{A}} \|f_R(\mathbf{X}) - f_R(A(\mathbf{X}))\|_2^2 + \|f_T(\mathbf{X}) - f_T(A(\mathbf{X}))\|_2^2, \tag{12}$$

where $\mathcal{A}$ is the group of pose preserving augmentations and $\|\cdot\|_2$ is MSE loss.

In addition, we can also generate a version of $\mathbf{X}$, with a known pose, by feeding again the learned canonical shape. Empirically, we found it beneficial to use the reconstructed shape, thus, we transform $\tilde{\mathbf{S}}$ by a random rotation matrix R^* and a random translation vector T^* and require the estimated pose to be consistent with this transformation .

$$\mathcal{L}^{can}_{consist} = \|f_R\left(\tilde{\mathbf{S}}R^* + T^*\right) - R^*\|_2^2 + \|f_T\left(\tilde{\mathbf{S}}R^* + T^*\right) - T^*\|_2^2, \tag{13}$$

Our overall loss is

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_1\mathcal{L}_{ortho} + \lambda_2\mathcal{L}^{aug}_{consist} + \lambda_3\mathcal{L}^{can}_{consist}, \tag{14}$$

where the λ_i are hyper parameters, whose values in all our experiments were set to $\lambda_1 = 0.5$, $\lambda_2 = \lambda_3 = 1$.

Fig. 4. Aligning planes and chairs. The input planes and chairs (first and third row, respectively) have different shapes and different poses, as can be seen separately and together (rightmost column). We apply the inverse learned pose, transforming the input to its canonical pose (second and fourth row).

3.4 Inference

At inference time, we feed forward point cloud $\mathbf{X} \in \mathbb{R}^{N\times 3}$ and retrieve its shape and pose. However, since our estimated rotation matrix $\tilde{R}$ is not guaranteed to be orthonormal, at inference time, we find the closest orthonormal matrix to $\tilde{R}$ (i.e., minimize the Forbenius norm), following [1], by solving:

$$\hat{R} = \tilde{R}\left(\tilde{R}^T\tilde{R}\right)^{-\frac{1}{2}}. \tag{15}$$

The inverse of the square root can be computed by singular value decomposition (SVD). While this operation is also differentiable we have found it harmful to incorporate this constraint during the training phase, thus it is only used during inference. We refer the reader to [1] for further details.

4 Results

We preform qualitative and quantitative comparison of our method for learning shape-invariant pose. Due to page limitations, more results can be found in our supplementary files.

4.1 Dataset and implementation details

We employ the ShapeNet dataset [2] for evaluation. For point cloud auto-encoding we follow the settings in [25] and [7], and use ShapeNet Core focusing on airplanes, chairs, tables and cars. While airplanes and cars are more semantically consistent and containing less variation, chairs exhibit less shape-consistency and

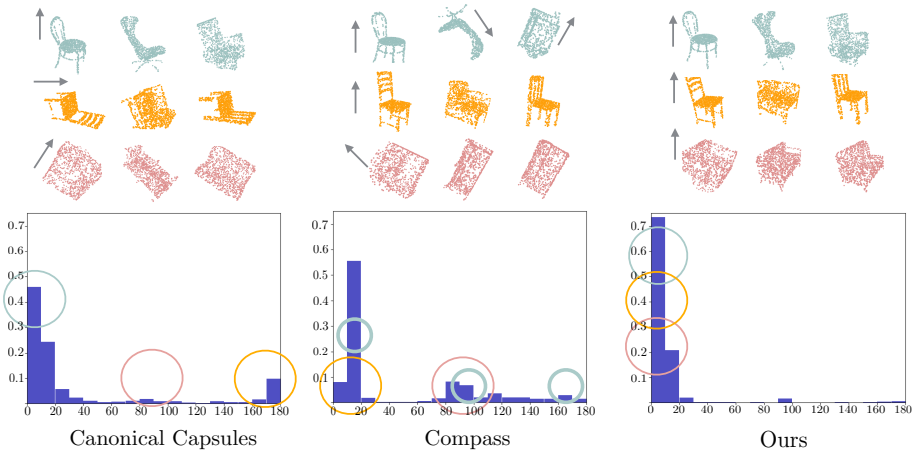

Fig. 5. Histogram of canonical pose deviation from the mean canonical pose. We estimate the canonical pose of aligned 3D point clouds from ShapeNet using our method, Canonical Capsules [25] and Compass [24]. In the bottom row, we show the normalized histogram of the deviation from the mean pose. It is clear that while our method is shape-consistent, both Compass and Canonical Capsules struggle to have a single canonical pose. In the top row, we focus on small, medium and large deviation cases of Canonical Capsules, marked by cyan, red, and orange circles on the histogram plot, respectively. The canonical pose of the same objects is shown for Compass and our method, as well as their location on the corresponding histogram plot. The arrow next to the objects is directed toward the local shape z+ direction.

may contain different semantic parts. All 3D models are randomly rotated and translated in the range of $[-0.1, 0.1]$ at train and test time.

For all experiments, unless stated otherwise, we sample random 1024 points for each point cloud. The auto-encoder is trained using Adam optimizer with learning rate of $1e^{-3}$ for 500 epochs, with drop to the learning rate at 250 and 350 by a factor of 10. We save the last iteration checkpoint and use it for our evaluation. The decoder is AtlasNetV2 [7] decoder with 10 learnable grids.

4.2 Pose consistency

We first qualitatively evaluate the consistency of our canonical representation as shown in Fig. 4. At test time, we feed different instances at different poses through our trained network, yielding estimated pose of the input object w.r.t the pose-invariant shape. We then apply the inverse transformation learned, to transform the input to its canonical pose. As can be seen, the different instances are roughly aligned, despite having different shapes. More examples can be found in our supplementary files.

We also compare our method, both qualitatively and quantitatively, to Canonical Capsules [25] and Compass [24] by using the alignment in ShapeNet (for Compass no translation is applied). First, we feed forward all of the aligned test point clouds $\{\mathbf{X}_i\}_{i=1}^{N_t}$ through all methods and estimate their canonical pose

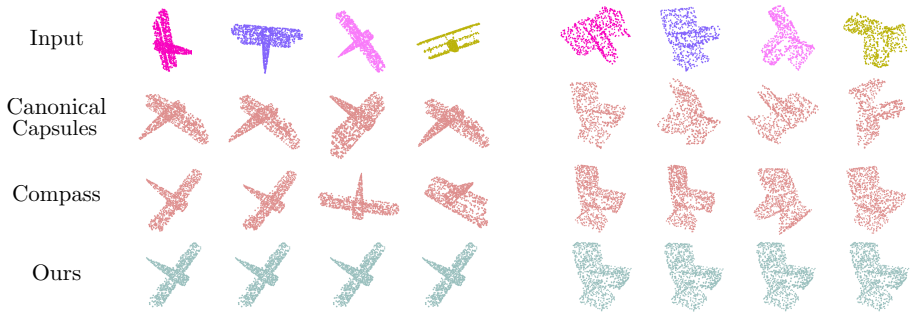

Fig. 6. Stability of the canonical representation to rigid transformation of the input. The location and orientation of the same point cloud affects its canonical representation in both Canonical Capsules [25] and Compass [24]. Our canonical representation (bottom row) is **SE**(3)-invariant to the rigid transformation of the input.

$\{\tilde{R}_i\}_{i=1}^{N_t}$. We expect to have a consistent pose for all aligned input shapes, thus, we quantify for each instance i the angular deviation $d_i^{consist}$ of its estimated pose $\tilde{R}_i$ from the Chordal L2-mean [12] pose $d_i^{consist} = \angle\Big(\tilde{R}_i, \frac{1}{N_t}\sum_i \tilde{R}_i\Big)$. We present an histogram of $\{d_i^{consist}\}_{i=1}^{N_t}$ in Fig. 5. As can be seen, our method results in a more aligned canonical shapes as indicated by the peak around the lower deviation values. We visualize the misalignment of Canonical Capsules by sampling objects with small, medium and large deviation, and compare them to the canonical representation achieved by Compass and our method for the same instances. The misalignment of Canonical Capsules may be attributed to the complexity of matching unsupervised semantic parts between chairs as they exhibit high variation (size, missing parts, varied structure). In Table 1 we quantify the consistency by the standard deviation of the estimated pose $\sqrt{\frac{1}{N_t}\sum_i d_i^2}$ and we present the Instance-Level Consistency (IC) and Ground Truth Consistency (GC) as defined in [22]. Evidently, Compass falls short for all object classes, while our method preforms overall in a consistent manner. Canonical Capsules preforms slightly better than our method for planes, though most of the miss-consistency is rooted in the symmetry of the object as indicated by the low GC. The table class is a unique example of a canonic pose which is not well-defined due to the rotation symmetry of tables (especially round tables). Despite the relatively low quantitative performance, in our supplementary files we show that the canonic shape of the tables is generally aligned.

4.3 Stability

A key attribute in our approach is the network construction, which outputs a purely **SE**(3)-invariant canonical shape. Since we do not require any optimization for such invariance, our canonical shape is expected to be very stable compared with Canonical Capsules and Compass. We quantify the stability, as proposed by Canonical Capsules, in a similar manner to the consistency metric. For each instance i, we randomly rotate the object $k = 10$ times, and estimate the canon-

Table 1. Comparison of consistency and stability (lower is better).

	Stability/Consistency			IC/GC ($\times 10^3$)		
	Capsules	Compass	Ours	Capsules	Compass	Ours
Airplanes	7.42 /**45.76**	13.81/71.43	**0.02**/49.97	2.1/**6.4**	6.3/20.1	**1.9**e−**4**/8.1
Chairs	4.79/68.13	12.01/68.2	**0.04**/**24.31**	2.4/13.1	8.4/51.2	**2**e−**4**/**12.1**
Cars	81.9/**11.1**	19.2/87.5	**0.03**/35.6	34.5/3.53	5.7/10.5	**1.6**e−**4**/**1.2**
Tables	14.7/119.3	74.8/115.3	**0.02**/**106.3**	10.5/78.1	14/92.9	**1.2**e−**3**/**15.3**

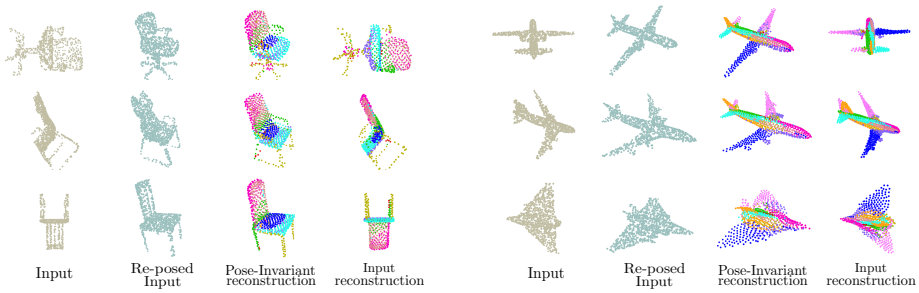

Fig. 7. Reconstruction of chairs and planes under SE(3) transformations. The input point cloud (left) is disentangled to shape (second from the right) and pose, which together reconstruct the input point cloud, as shown in the right most column. The inverse pose is applied to the input point cloud to achieve a canonical representation (second image from the left). The colors of the reconstructed point cloud indicate different decoders of AtlasNetV2 [7].

ical pose for each rotated instance $\{\tilde{R}_{ij}\}_{j=1}^{k}$. We average across all N_t instances the standard deviation of the angular pose estimation as follows,

$$d^{stability} = \frac{1}{N_t} \sum_i \sqrt{\sum_j \angle\Big(\tilde{R}_{ij}, \frac{1}{k}\sum_j \tilde{R}_{ij}\Big)^2}.$$

The results are reported in Table 1. As expected, Canonical Capsules and Compass exhibit non-negligible instability, as we visualize in Fig. 6.

4.4 Reconstruction quality

We show qualitatively our point cloud reconstruction in Fig. 7. Please note that our goal is not to build a SOTA auto-encoder in terms of reconstruction, rather we learn to disentangle pose from shape via auto-encoding. Nonetheless, our auto-encoder does result in a pleasing result as shown in Fig. 7. Moreover, since we employ AtlasNetV2[7] which utilizes a multiple patch-based decoder, we can examine which point belongs to which decoder. As our shape-encoding is both invariant to pose and consistent across different shapes, much like in the aligned scenario, each decoder assume some-what of semantic meaning, capturing for example the right wing of the airplanes. Please note that we do not enforce any structuring on the decoders.

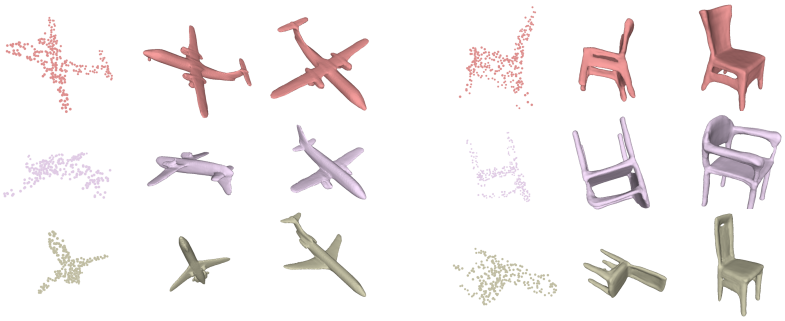

Fig. 8. Reconstruction results of OccNet [16] via shape-pose disentanglement. An input point cloud on the left is disentangled to shape encoding and pose. OccNet decodes only the shape encoding yielding a canonical shape on the right column. The reconstruction is then transformed by the estimated pose as seen in the middle column. Meshes are extracted via Multiresolution IsoSurface Extraction (MISE) [16]

4.5 3D implicit reconstruction

We show that our encoder can be attached to a different reconstruction task by repeating OccNet [16] completion experiment. We replace OccNet encoder with our shape-pose disentagling encoder. The experiment is preformed with the same settings as in [16]. We use the subset of [3], and the point clouds are sub-sampled from the watertight mesh, containing only 300 points and applied with a Gaussian noise. We have trained OccNet for $600K$ iterations and report the results of the best (reconstruction wise) checkpoint. We show in Fig. 8 a few examples of rotated point clouds (left), its implicit function reconstruction (middle) and the implicit function reconstruction in the canonical pose (right).

5 Conclusions

We have presented a stable and consistent canonical representation learning. To achieve a pose-invariant represenation, we have devised an $\mathbf{SE}(3)$-equivairant encoder, extending the VNN framework, to meet the requirements of canonical pose learning, i.e., learning rigid transformations. Our experiments show, both qualitatively and quantitatively, that our canonical representation is significantly more stable than recent approaches and has similar or better consistency, especially for diverse object classes. Moreover, we show that our approach is not limited to specific decoding mechanism, allowing for example to reconstruct canonical implicit neural field. In the future, we would like to explore the potential of our canonical representation for point cloud processing tasks requiring aligned settings, such as completion and unsupervised segmentation, where the canonical representation is learned on-the-fly, along with the task.

Acknowledgments: This work was supported in part by the Israel Science Foundation (grants no. 2492/20, 3441/21 and 3611/21)

References

1. Bar-Itzhack, I.Y.: Iterative optimal orthogonalization of the strapdown matrix. IEEE Transactions on Aerospace and Electronic Systems (1), 30–37 (1975)
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
3. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: European conference on computer vision. pp. 628–644. Springer (2016)
4. Cohen, T.S., Geiger, M., Köhler, J., Welling, M.: Spherical cnns. arXiv preprint arXiv:1801.10130 (2018)
5. Cohen, T.S., Welling, M.: Steerable cnns. arXiv preprint arXiv:1612.08498 (2016)
6. Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., Guibas, L.J.: Vector neurons: A general framework for so (3)-equivariant networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12200–12209 (2021)
7. Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., Aubry, M.: Learning elementary structures for 3d shape generation and matching. In: Advances in Neural Information Processing Systems. pp. 7433–7443 (2019)
8. Esteves, C., Allen-Blanchette, C., Makadia, A., Daniilidis, K.: Learning so (3) equivariant representations with spherical cnns. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 52–68 (2018)
9. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3d surface generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 216–224 (2018)
10. Gu, J., Ma, W.C., Manivasagam, S., Zeng, W., Wang, Z., Xiong, Y., Su, H., Urtasun, R.: Weakly-supervised 3d shape completion in the wild. In: European Conference on Computer Vision. pp. 283–299. Springer (2020)
11. Hamdi, A., Giancola, S., Ghanem, B.: Mvtn: Multi-view transformation network for 3d shape recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1–11 (2021)
12. Hartley, R., Trumpf, J., Dai, Y., Li, H.: Rotation averaging. International journal of computer vision **103**(3), 267–305 (2013)
13. Li, X., Weng, Y., Yi, L., Guibas, L.J., Abbott, A., Song, S., Wang, H.: Leveraging se (3) equivariance for self-supervised category-level object pose estimation from point clouds. Advances in Neural Information Processing Systems **34**, 15370–15381 (2021)
14. Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., Bai, X.: Tanet: Robust 3d object detection from point clouds with triple attention. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11677–11684 (2020)
15. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In: International Conference on Learning Representations (2022)
16. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
17. Novotny, D., Ravi, N., Graham, B., Neverova, N., Vedaldi, A.: C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7688–7697 (2019)

18. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019)
19. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
20. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
21. Rempe, D., Birdal, T., Zhao, Y., Gojcic, Z., Sridhar, S., Guibas, L.J.: Caspr: Learning canonical spatiotemporal point cloud representations. Advances in neural information processing systems **33**, 13688–13701 (2020)
22. Sajnani, R., Poulenard, A., Jain, J., Dua, R., Guibas, L.J., Sridhar, S.: Condor: Self-supervised canonicalization of 3d pose for partial shapes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16969–16979 (2022)
23. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 770–779 (2019)
24. Spezialetti, R., Stella, F., Marcon, M., Silva, L., Salti, S., Di Stefano, L.: Learning to orient surfaces by self-supervised spherical cnns. arXiv preprint arXiv:2011.03298 (2020)
25. Sun, W., Tagliasacchi, A., Deng, B., Sabour, S., Yazdani, S., Hinton, G.E., Yi, K.M.: Canonical capsules: Self-supervised capsules in canonical pose. Advances in Neural Information Processing Systems **34** (2021)
26. Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., Riley, P.: Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. arXiv preprint arXiv:1802.08219 (2018)
27. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019)
28. Weiler, M., Geiger, M., Welling, M., Boomsma, W., Cohen, T.: 3d steerable cnns: Learning rotationally equivariant features in volumetric data. arXiv preprint arXiv:1807.02547 (2018)
29. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. Advances in Neural Information Processing Systems **32** (2019)
30. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 206–215 (2018)
31. Yang, Z., Sun, Y., Liu, S., Jia, J.: 3dssd: Point-based 3d single stage object detector. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11040–11048 (2020)