Supplementary materials 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

1 Vector Neuron Translation Invariant

In this section we verify that our VNT layers are indeed translation and rotation equivariant, as well as explicitly present other layers, not included in the paper.

1.1 Verifying SE(3)-equivariant linear layer

We verify that the linear module $f_{\text{lin}}(\cdot;\mathbf{W})$, defined via a weight matrix $\mathbf{W} \in \mathcal{W}^{C'\times C}$, acting on a vector-list feature $\mathbf{V} \in \mathbb{R}^{C\times 3}$, such that

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

is $\mathbf{SE}(3)$-equivariant.

Let $\mathbf{w_j} \in \mathbb{R}^{C'\times C}$ be the j column of $\mathbf{W}$, and 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:

$$\begin{aligned} f_{\text{lin}}(\mathbf{V}R + \mathbb{1}_C T) &= \mathbf{W}(\mathbf{V}R + \mathbb{1}_C T) = \mathbf{W}\mathbf{V}R + \mathbf{W}\mathbb{1}_C T = \\ &= \mathbf{W}\mathbf{V}R + \left(\sum_{j=1}^{C} \mathbf{w}_j\right) T = (\mathbf{W}\mathbf{V})R + \mathbb{1}_{C'} T, = f_{\text{lin}}(\mathbf{V})R + \mathbb{1}_{C'} T, \end{aligned} \tag{2}$$

where $\mathbb{1}_C = [1, 1, \ldots, 1]^T \in \mathbb{R}^{C\times 1}$ is a column vector of length C, and $\sum_{j=1}^{C} \mathbf{w_j} = \mathbb{1}_{C'}$, since $\left(\sum_{j=1}^{C} \mathbf{w_j}\right)[i] = \sum_{j=1}^{C} w_{ij} = 1$ for $i = 1, \ldots, C'$

1.2 Verifying SE(3)-equivariant ReLU

We verify that the ReLU layer is $\mathbf{SE}(3)$-equivariant.

Let $\mathbf{V}, \mathbf{V'} \in \mathbb{R}^{C\times 3}$ be the input and output of a ReLU layer,

$$\mathbf{V'} = f_{\text{ReLU}}(\mathbf{V}). \tag{3}$$

Let $\mathbf{v'} \in \mathbb{R}^{1\times 3}$ be a single vector, such that $\mathbf{v'} \in \mathbf{V'}$. As explained in Section 3.1 of the paper, we learn three translation equivariant linear maps, $\mathbf{Q}, \mathbf{K}, \mathbf{O} \in \mathcal{W}^{1\times C}$

projecting the input to $\mathbf{q}, \mathbf{k}, \mathbf{o} \in \mathbb{R}^{1\times 3}$, yielding an origin $\mathbf{o}$, a feature $\mathbf{q_o} = \mathbf{q} - \mathbf{o}$ and a direction $\mathbf{k_o} = \mathbf{k} - \mathbf{o}$. The ReLU layer for a single vector neuron is then defined via

$$\mathbf{v}' = \begin{cases} \mathbf{o} + \mathbf{q_o} & \text{if} \quad \langle \mathbf{q_o}, \mathbf{k_o} \rangle \geq 0, \\ \mathbf{o} + \mathbf{q_o} - \left\langle \mathbf{q_o}, \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|} \right\rangle \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|}, & \text{otherwise.} \end{cases} . \tag{4}$$

$\mathbf{q}, \mathbf{k}, \mathbf{o}$ are $\mathbf{SE}(3)$-equivariant and according to Eq .(7) of the paper, $\mathbf{q_o}, \mathbf{k_o}$ are translation invariant (and rotation equivariant) as they are the subtraction of two $\mathbf{SE}(3)$-equivariant vector neurons, thus, the condition term $\langle \mathbf{q_o}, \mathbf{k_o} \rangle$ is also translation invariant. As shown in VNN [6], the inner product of two rotation equivariant vector-neurons is rotation invariance. Similarly here, assume the input $\mathbf{V}$ is rotated with a rotation matrix $R \in \mathbb{R}^{3\times 3}$, then

$$\langle \mathbf{q_o} R, \mathbf{k_o} R \rangle = \mathbf{q_o} R R^T \mathbf{k_o} = \mathbf{q_o} \mathbf{k_o}^T = \langle \mathbf{q_o}, \mathbf{k_o} \rangle \tag{5}$$

To conclude, the condition term $\langle \mathbf{q_o}, \mathbf{k_o} \rangle$ is $\mathbf{SE}(3)$-invariant.

When $\langle \mathbf{q_o}, \mathbf{k_o} \rangle \geq 0$, the output vector neuron $\mathbf{v}' = \mathbf{o} + \mathbf{q_o} = \mathbf{q}$, and thus it is $\mathbf{SE}(3)$-equivariant.

When $\langle \mathbf{q_o}, \mathbf{k_o} \rangle < 0$ the output vector neuron is

$$\mathbf{o} + \mathbf{q_o} - \left\langle \mathbf{q_o}, \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|} \right\rangle \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|}, \tag{6}$$

Similarly to Eq .(5) the term

$$\left\langle \mathbf{q_o}, \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|} \right\rangle \frac{1}{\|\mathbf{k_o}\|} = \frac{\langle \mathbf{q_o}, \mathbf{k_o} \rangle}{\langle \mathbf{k_o}, \mathbf{k_o} \rangle}, \tag{7}$$

is also $\mathbf{SE}(3)$-invariant.

We can now easily prove that if the input $\mathbf{V}$ is rotated with a rotation matrix $R \in \mathbb{R}^{3\times 3}$ and translation vector $T \in \mathbb{R}^{1\times 3}$, then

$$\begin{aligned} \mathbf{v}' =& \mathbf{o}R + \mathbb{1}_C T + \mathbf{q_o} R - \left\langle \mathbf{q_o}, \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|} \right\rangle \frac{\mathbf{k_o} R}{\|\mathbf{k_o}\|} \\ =& \left(\mathbf{o} + \mathbf{q_o} - \left\langle \mathbf{q_o}, \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|} \right\rangle \frac{\mathbf{k_o}}{\|\mathbf{k_o}\|} \right) R + \mathbb{1}_C T = \mathbf{v}' R + \mathbb{1}_C T, \end{aligned} \tag{8}$$

Thereby completing the proof.

1.3 VNT-LeakyReLU

LeakyReLU is defined in a similar manner to the ReLU layer, with slight modification to the output vector neuron, given by

$$\mathbf{v}' = \alpha \mathbf{q} + (1 - \alpha)\, \mathbf{v}'_{\mathrm{ReLU}}, \tag{9}$$

where $\alpha \in \mathbb{R}$

Easy to see that the $\mathbf{v}'$ is $\mathbf{SE}(3)$-equivariant.

1.4 VNT-MaxPool

Given a set of vector-neuron list $\mathcal{V} \in \mathbb{R}^{N\times C\times 3}$, we learn two linear maps $\mathbf{K}, \mathbf{O} \in \mathcal{W}^{C\times C}$, shared between $\mathbf{V_n} \in \mathcal{V}$.

We obtain a translation invariant direction

$$\mathcal{K} = \{\mathbf{K}\mathbf{V_n} - \mathbf{O}\mathbf{V_n},\}_{n=1}^{N} \tag{10}$$

and a translation invariant features

$$\mathcal{Q} = \{\mathbf{V_n} - \mathbf{O}\mathbf{V_n}\}_{n=1}^{N}. \tag{11}$$

The VNT-MaxPool is defined by

$$f_{MAX}\left(\mathcal{V}\right)[c] = \mathbf{V}_{n^*}[c] \tag{12}$$

$$\text{where} \quad n^* = \arg\max_{n} \langle \mathbf{Q_n}[c], \mathbf{K_n}[c] \rangle, \tag{13}$$

where $\mathbf{Q_n} \in \mathcal{Q}$ and $\mathbf{K_n} \in \mathcal{K}$. Since $\mathbf{Q_n}$, $\mathbf{K_n}$ are translation invariant, and their inner product is also rotation invariant the selection process of n^* for every channel c is invariant to $\mathbf{SE}(3)$. We note that both $\mathbf{K}, \mathbf{O}$ can be shared across vector-neurons.

2 Implementations Details

2.1 Encoder architecture

In this section we elaborate on our encoder architecture. Our encoder contains VNT layers following with VNN layers as reported in Table 1. LinearLeakyReLU stands for the leakyReLU with feature learning $\mathcal{Q}$. For the exact VNN layers definition (and specifically STNkd) we refer the reader to VNN [6].

3 Implicit reconstruction

Occupancy network reconstruction from a point clouds $X \in \mathbb{R}^{N\times 3}$, with a learned embedding $\mathbf{z} \in \mathcal{X}$, learns a mapping function $f_\theta(p, \mathbf{z}) : \mathbb{R}^3 \times \mathcal{X} \to [0, 1]$. In occupancy network completion experiment, the point cloud is sampled from the watertight mesh, and the mesh is used as supervision to sample M training point $\{p_i\}_{i=1}^{M}$ inside and outside the mesh, indicated by $\{o_i\} \in [0, 1]^M$. We follow the same experiment, with slight changes. We feed our learned pose-invariant encoding $\mathbf{Z_s}$ through f_θ, and project the points $\{p_i\}$ from the input pose to the learned canonical pose by:

$$\tilde{p}_i = (p_i - \tilde{T})\tilde{R}^T \quad i = 1, \ldots, M \tag{14}$$

Therefore, our reconstruction loss is

$$\mathcal{L}_{rec} = \sum_{i=1}^{M} \mathcal{L}_{\text{BCE}}\left(f_\theta(\tilde{p}_i, \mathbf{Z_s}), o_i\right), \tag{15}$$

Table 1. Our shape-pose disentangling encoder architecture.

Name	Input channel	Output channel	Type	Split
LinearLeakyReLU	3	64//3	VNT	
LinearLeakyReLU	64//3	64//3	VNT	
T-invariant	64//3	64//3	VNT	Translation branch
LinearLeakyReLU	64//3	64//3	VNN	
STNkd+concat	64//3	2·(64//3)	VNN	
LinearLeakyReLU	2·(64//3)	2·(64//3)	VNN	
LinearLeakyReLU	2·(64//3)	170	VNN	
BatchNorm	170	170	VNN	Rotation branch
Meanpool+concat	170	340	VNN	
R-invariant	340	340	VNN	
Flatten	$340 \cdot 3$	1020	Regular	
Max-pool	1020	1020	Regular	

where $\mathcal{L}_{\text{BCE}}$ is binary cross entropy loss. For implicit reconstruction we have found it beneficial to train the network in an alternating approach, where at the first phase we backward w.r.t $\mathcal{L}_{rec}$ and in the second phase we backward w.r.t

$$\mathcal{L}_2 = \lambda_1 \mathcal{L}_{ortho} + \lambda_2 \mathcal{L}^{aug}_{consist}. \tag{16}$$

4 VNT vs VNN with center subtraction

Our VNT layers are novel extensions to Vector Neuron architecture enabling **SE**(3)-equivariant feature learning. Actually, **SE**(3)-equivariance can be simply achieved by subtracting the mean of the input point cloud or its' bounding box center. However, such a procedure is prone to errors in the presence of noise and partly occluded objects and does not produce actual **SE**(3)-equivariant network components. To demonstrate the advantages of using our VNT, we repeat the classification experiment on ModelNet40 conducted in [6], focusing on **SE**(3)transformation. We use VNN-pointnet as proposed in [6] (for brevity we will omit the suffix pointnet as all our classifiers are based on the pointnet architecture) and variants of it: VNN-mean - a VNN classifier where at the forward pass the input point cloud is mean-centered and VNN(2)-mean which is similar to VNN-mean but with slightly more capacity. We implement our VNT-pointnet with the same encoder design presented in the main paper. We report in Table 2 the first few layers for all encoders (the remaining layers are the same as in VNN-pointnet [6] for all encoders). At test time, every point cloud is presented with a single rotation and translation (drawn randomly in the range $[-0.1, 0.1]$). As shown in Table 3, the accuracy of our VNT classifier achieves the highest classification accuracy by a margin. Please note also that our accuracy of 82.5 is superior to the original VNN-pointnet tested on aligned data (74.1 according to our experiment and 77.5 according to [6]).

Table 2. architecture differences between VNN-pointnet and VNT-pointnet. We briefly report the first few layers for each encoder. Please refer to [6] for further details.

VNN	VNN-mean	VNN(2)-mean	VNT	In ch	Out ch	Sym
-	mean-subtraction	mean-subtraction	-	3	3	
VN-LLReLU	VN-LLReLU	VN-LLReLU	VNT-LLReLU	3	64//3	
mean-pool	mean-pool	mean-pool	mean-pool	64//3	64//3	x
-	-	VN-LLReLU	VNT-LLReLU	64//3	64//3	
-	-	VN-LLReLU	VNT-LLReLU	64//3	1	x_c
-	-	Residual	Residual	64//3	64//3	
x	x	$x - x_c$	$x - x_c$			

Table 3. Accuracy comparison of VNT and VNN on ModelNet40 (higher is better). All classifiers were trained on **SE**(3)augmented data and tested with the same **SE**(3)augmentation (translation was drawn randomly in the range $[-0.1, 0.1]$)

	VNN	VNN-mean	VNN(2)-mean	VNT
Accuracy	0.708	0.626	0.667	**0.825**

In addition, since reducing the mean is a global, none-learned, procedure, it can easily be affected by the presence of noise. We demonstrate this weakness by adding 5 points located at $(0, 0, 0)$ and then gradually translating the 5 points to $(1, 1, 1)$. For each translation step, we evaluate the accuracy of VNT and VNN-mean as shown in Fig. 1. Intuitively, both methods will be slightly affected since such data is not present in the training set, but as for VNT such noise is local, affecting mainly through the features of the noisy points, VNN-mean is much more vulnerable, as the points move further away from the origin, all of the points' features are affected. This phenomenon can be seen in the drop in accuracy in Fig. 1.

5 Discussion regarding the differences to Li et al. [13]

Li et al. [13] proposed a clever category level pose estimation utilizing **SE**(3) equivariant backbone. While our work shares similarities with Li et al., key differences exist between the two methods. First, Li et al. rely on the "laziness" of the network to generate a consistent shape using only one mlp layer as a decoder. Similarly, the input point clouds are scaled to have a unit diagonal length, a procedure that is prone to error in the presence of noisy data. On the other hand, we introduce a consistency loss by altering the input point cloud with a variety of simple shape augmentations and do not scale the input point cloud. Second, our network is **SE**(3)-equivariant to all orientations while Li et al. propose **SE**(3)-equivariance network only within a pre-defined subgroup of orientations,

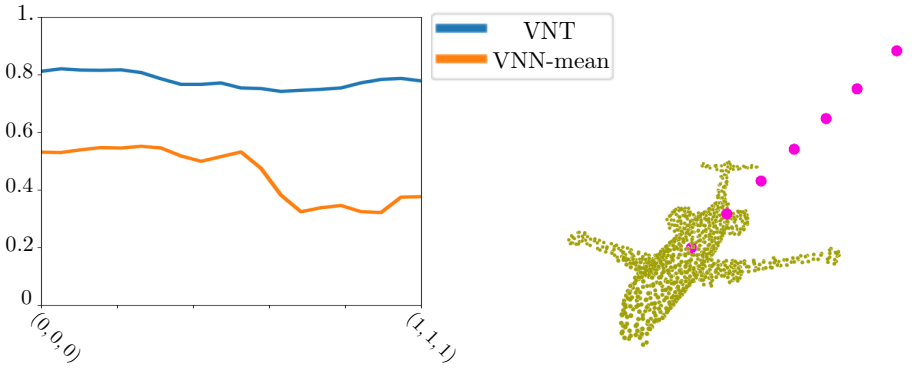

Fig. 1. Robustness of VNN-mean centered and VNT. We introduce a noise to every point cloud by adding 5 points placed at the origin. The points are gradually translated from the origin to $(1, 1, 1)$ as visualized in magenta on the right. The accuracy of both methods is evaluated for every translation step as shown on the left, where the horizontal axis indicates an (x, x, x) translation. VNT accuracy outperforms VNN-mean and is relatively unaffected by the translation of the points. On the other hand, VNN-mean is susceptible to translation, as can be seen in the drop at around $(0.5, 0.5, 0.5)$.

followed by a learned non-equivariant deviation. While this clever design generate a very consistent shape (consistency of 34.4/8.1 compared to our 49.9/24.3, for planes/chairs), it comes on the expense of stability of the method (3.2/3.1 compared to ours 0.02/0.04 for planes/chairs), as shown in Fig. 2. It should also be noted that Li et al. rely on a relatively complex design of equivariant layers, while we extend VNN, a simple equivariant architecture, which is easier to integrate with existing pipelines for point cloud processing.

6 Additional results

6.1 Augmentations ablation

In this section we specify in more details our augmentations, which can be seen in Fig. 3, and ablate their individual donation to the consistency of our canonical representation. Our augmentations are Furthest point sampling (FPS), with random number of points $N_{\text{FPS}} = \text{U}(300, 500)$ per batch, K-NN removal (KNN), where a point is randomly selected on the point cloud, and its $N_{\text{KNN}} = 100$ points are removed, Gaussian Noise added to the point clouds with $\mu = 0$ and $\sigma = 0.025$, a re-sampling augmentations (Resample) where we re-select which $N = 1024$ to sample from the original point cloud, and canonical rotation (Can), where the point cloud reconstruction in its canonical representation is rotated and transformed to create a supervised version of itself. Since our method is pose-invariant by construction, different augmentations have no effect on the stability, thus, we ablate only w.r.t the consistency as reported in Table 4. We

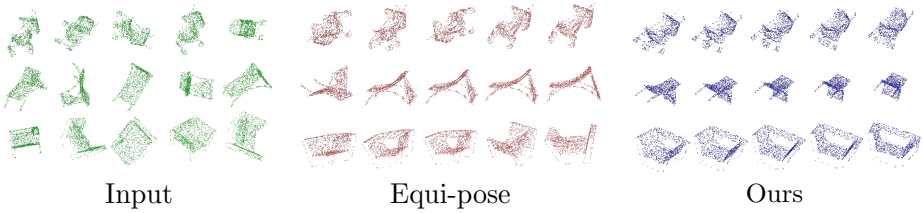

Fig. 2. Stability of our method compared to Equi-pose. Our method (right) is fully **SE**(3)-equivariant, thus changes in the orientation of the same input point cloud (Different columns on the left) does not change our canonic pose. Equi-pose (middle) does not exhibit the same level of stability, as can be seen by the canonic pose changes for different orientation of the input point cloud.

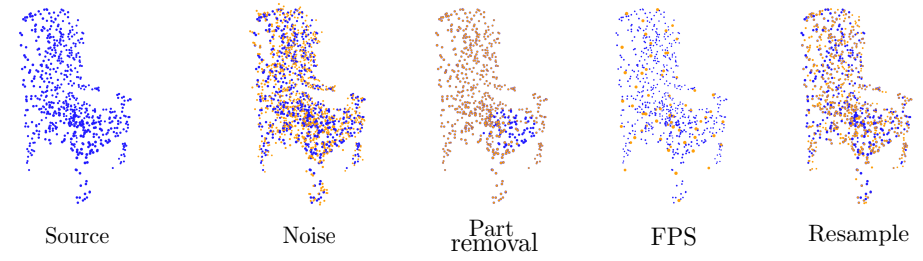

Fig. 3. Point cloud augmentations. The source point cloud on the left (blue), is modified (brown) to supervise a rotation and translation learning invariant of the shape.

ablate the donation of each augmentation by removing it from the training process and measuring the consistency as defined in the paper. Evidently, the lesser factor is the noise addition augmentation, while KNN and FPS donate the most to the consistency metric.

Table 4. Consistency of different augmentations composition (lower is better). Each column but the last represents the absence of an augmentation, indicating its importance to the consistency metric.

	-FPS	- Noise	-KNN	-Resample	-Can	All
Airplanes	52.31	50.0	66.4	50.5	53.7	**49.9**
Chairs	27.31	24.41	27.31	25.3	25.1	**24.31**

6.2 Augmentations pose Invariance

In Tab. 5 we measure the median deviation in canonical pose due to the different augmentations (measured in degrees). As can be seen the deviation for both airplanes and chairs is lower than 3.65.

Table 5. Median deviation of canonical pose due to augmentation

	FPS	Noise	Patch	Resample
Airplanes	3.01	1.35	2.17	1.67
Chairs	3.65	1.42	2.38	2.09

6.3 Pose consistency

We present more canonical alignment results for point clouds and implicit function reconstruction Fig. 4, Fig. 5, Fig. 6 Fig. 7 and Fig. 8.

In addition, we experiment with partial dataset for shape completion derived from ShapeNet (See Yuan et al. "Pcn: Point completion network"). The partial points clouds are a projection of 2.5D depth maps of the model into 3D point clouds. The dataset contains 8 such partial point clouds per model. As can be seen in Fig. 9 while learning a consistent canonical pose is difficult for partial shapes, our canonical pose is reasonable and mostly consistent. Although, misalignment is apparent in the consistency histogram, please note that no complete point cloud is present in this setting, and no hyper-parameter tuning was done.

6.4 Stability

Please see the attached videos for stability visualization, divided to two subfolders for chairs and airplanes. In each video, we sample a single point cloud and rotate it with multiple random rotation matrices. We feed the rotated point cloud (see on the left of each video) through Compass, Canonical Capsules and Our method, and show the input point cloud in canonical pose for all methods. Our method reconstruct a $\mathbf{SE}(3)$-invariant canonical representation and a $\mathbf{SE}(3)$-equivariant pose estimation, thus, almost no changes are observable in the canonical representation, while both Canonical Capsules and Compass exhibit instability in the canonical pose estimation.

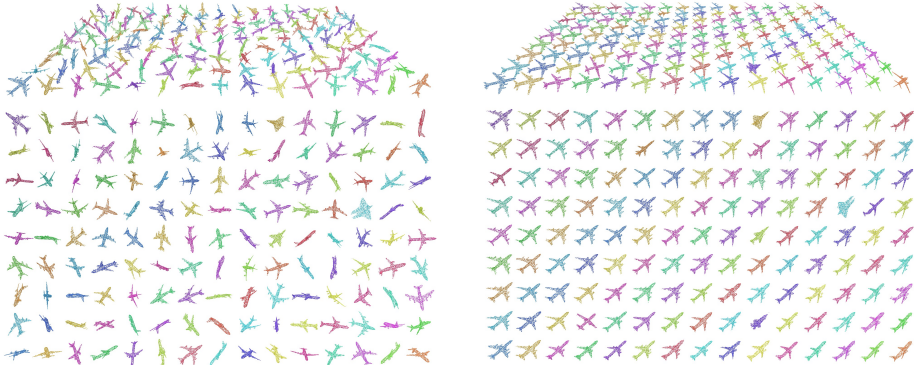

Fig. 4. More results of planes in canonic representations. The planes on the left are randomly translated and rotated, as seen from a side view (first row) and top view (second row). Our canonical representation, on the right, exhibit good alignment across different instances both in orientation and position.

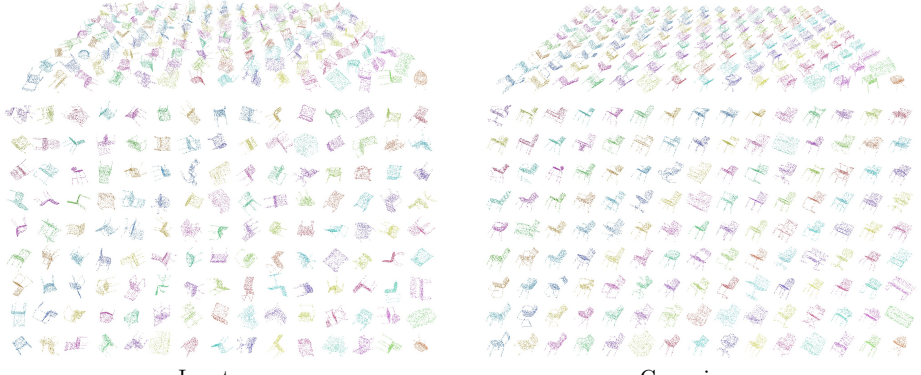

Fig. 5. More results of chairs in canonic representations. The chairs on the left are randomly translated and rotated, as seen from a side view (first row) and top view (second row). Our canonical representation, on the right, exhibit good alignment across different instances both in orientation and position.

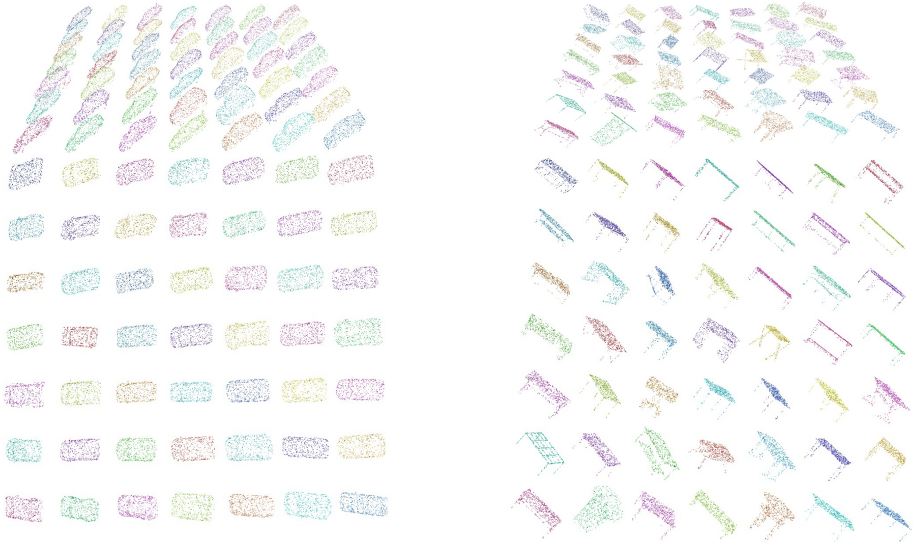

Fig. 6. More results of canonic representations for tables and cars from shapenet. Our canonical representation, exhibit good alignment across different instances both in orientation and position.

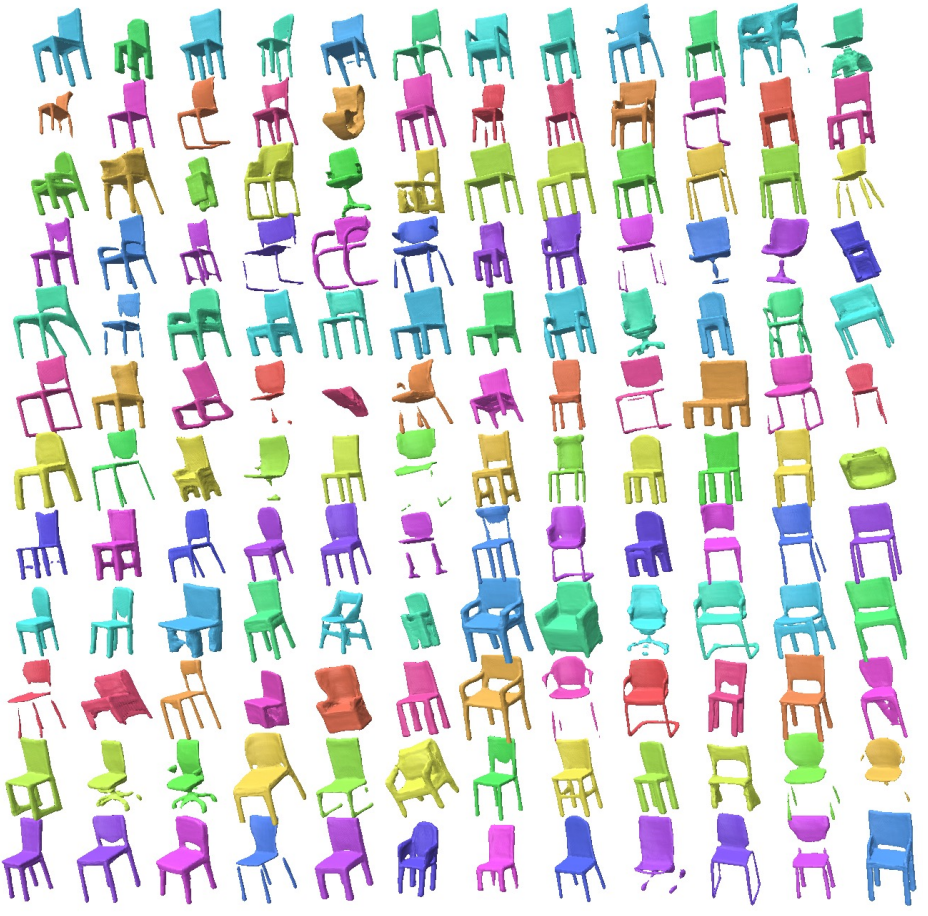

Fig. 7. More results of implicit function reconstruction for chairs in canonic representations. Our canonical representation exhibit good alignment across different instances both in orientation and position.

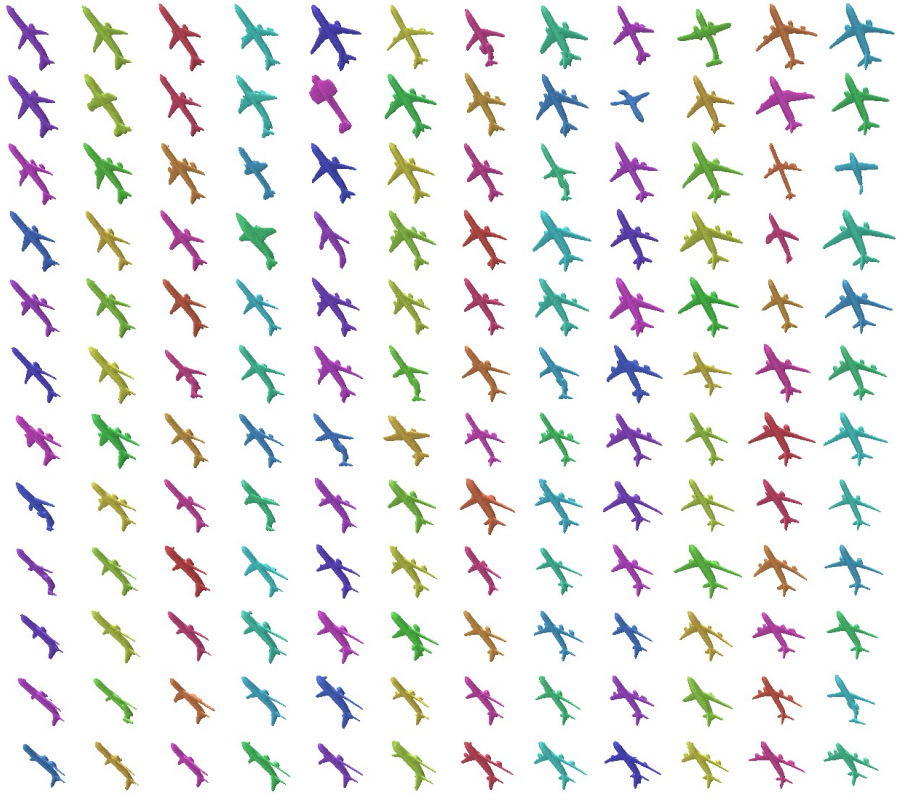

Fig. 8. More results of implicit function reconstruction for planes in canonic representations. Our canonical representation exhibit good alignment across different instances both in orientation and position.

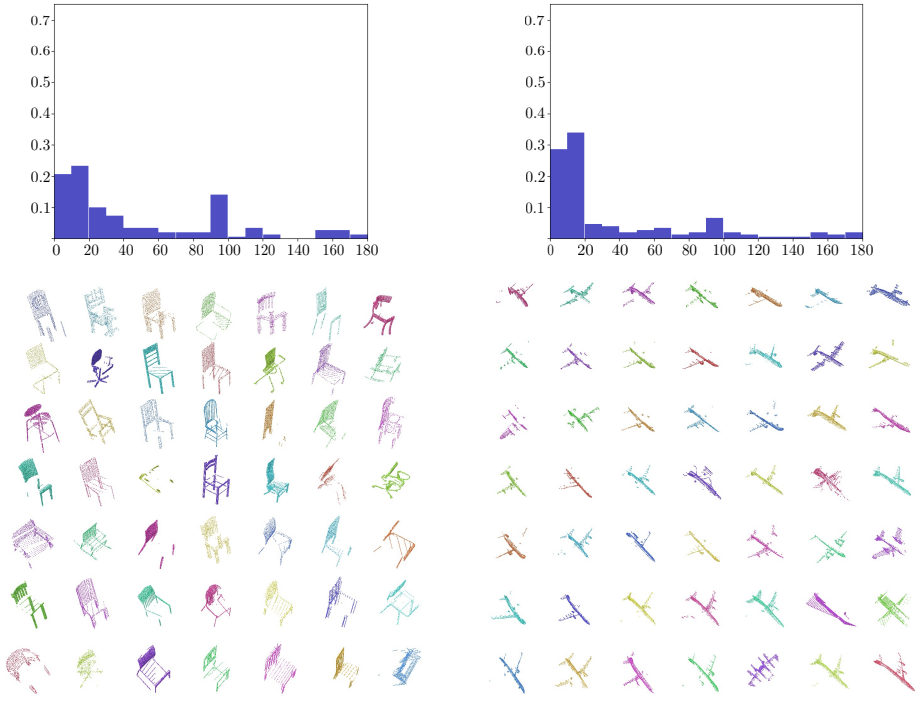

Fig. 9. Consistency of the canonical pose for partial shapes of planes and chairs. While our method is not directly optimized to achieve canonic alignment of partial shapes due to occlusions, it has reasonable performances has can be seen in the histogram of the first row and from samples in the second row.