A Closer Look at Rotation-invariant Deep Point Cloud Analysis

Feiran Li[1*] Kent Fujiwara[2] Fumio Okura[1] Yasuyuki Matsushita[1]
[1]Osaka University [2] LINE Corporation

Abstract

We consider the deep point cloud analysis tasks where the inputs of the networks are randomly rotated. Recent progress in rotation-invariant point cloud analysis is mainly driven by converting point clouds into their respective canonical poses, and principal component analysis (PCA) is a practical tool to achieve this. Due to the imperfect alignment of PCA, most of the current works are devoted to developing powerful network structures and features to overcome this deficiency, without thoroughly analyzing the PCA-based canonical poses themselves. In this work, we present a detailed study w.r.t. the PCA-based canonical poses of point clouds. Our investigation reveals that the ambiguity problem associated with the PCA-based canonical poses is handled insufficiently in some recent works. To this end, we develop a simple pose selector module for disambiguation, which presents noticeable enhancement (i.e., 5.3% classification accuracy) over state-of-the-art approaches on the challenging real-world dataset.[1]

1. Introduction

Deep learning is thriving in point cloud analysis owing to its excellent performance in various tasks. As a primitive representation of 3D data that can be directly obtained from sensors, point clouds are widely employed as the direct inputs of modern neural networks [17, 30, 31].

However, there exists a fundamental problem when point clouds are used in high-level applications, such as classification, retrieval, and segmentation. In such applications, we expect the networks to present consistent inferences w.r.t. varying affine transformations on the point clouds. While the effects of scaling and translation can be eliminated effectively by normalization and centralization [17], achieving rotational invariance remains an open problem.

Various methods have been proposed to tackle this issue. There are attempts to learn to align shapes in an optimal pose [7, 9], or robustify the networks via equivariant properties [6, 20, 21]. However, these methods are not strictly rotation-invariant and require the rotational space to be densely sampled for data augmentation. Some works also attempt to handcraft rotation-invariant geometric features [4, 24, 36], although they inevitably suffer from information loss compared to directly using the Cartesian coordinates of the point clouds.

Some recent works propose to use intrinsically determined canonical poses to avoid information loss. As a practical tool, PCA enables us to intrinsically determine 3 orthogonal bases (*i.e.*, the principal axes) of a given point cloud and align them to the world Cartesian coordinate. Despite the effectiveness of PCA-based canonical poses, we observe that some recent works are devoted to developing more powerful networks without clearly studying the canonical poses themselves [10, 34, 35, 37]. Consequently, these works have not explicitly tackled the ambiguity problem of the PCA-based canonical poses, which hinders their performance.

In this work, we analyze in detail the PCA-based canonical poses in point cloud analysis. We study the actual number of ambiguities of the PCA-based canonical poses, explore the effects of PCA in point cloud analysis, and propose a pose selector module for disambiguation. Our investigation reveals that the accurate identification of ambiguities can lead to noticeable performance boosts than the recently developed networks and features. *I.e.*, state-of-the-art approaches would be outperformed by a large margin on the challenging real-world dataset if ambiguities are properly identified. We summarize our contributions as follows:

- We study the actual number of ambiguities of the PCA-based canonical poses, showing that it has not been fully addressed in recent works.
- We demonstrate the actual ability of PCA-based canonical poses by distinct experiments and detailed analysis, hoping to prompt future works in the context of rotation-invariant point cloud analysis.
- We propose a pose selector that can effectively disambiguate the canonical poses and boost performance.

*Work partially done during an internship at LINE.
[1]Source code can be found at https://github.com/SILI1994/rotation-invariant-pointcloud-analysis.

2. Related works

In this section, we briefly review the developments of deep learning techniques on point cloud data as well as the past efforts on rotation-robust and rotation-invariant deep point cloud analysis.

2.1. Deep learning on point clouds

Point clouds are difficult for neural networks to handle due to their irregularity. As solutions, conventional methods propose to render them to images [5, 23] or conduct quantization to obtain volumetric grids [14, 18] to facilitate convolution. However, such approaches either cannot handle tasks, such as segmentation, that require point-wise labeling, or generalize poorly to dense point clouds due to high memory consumption.

Some recent research explores inputting point clouds directly to networks. Pioneered by PointNet [17], which uses point-wise convolution followed by a max-pooling layer to achieve permutation-invariant representations, the following works explore aggregating local information to boost performance. For example, DGCNN [30] employs graph convolution by mapping point clouds to k-nearest-neighbor graphs. PointConv [31] propose to approximate the convolution kernel with a multi-layer perceptron network. KPConv [25] uses a set of kernel points to define the convolution area. However, all these methods assume intra-category point clouds to be already aligned to the same pose, without which the performance would decline significantly.

2.2. Rotation-robust point cloud analysis

Some works aim at robustifying the networks w.r.t. randomly rotated point clouds. The primary philosophy lies in designing modules that are equivariant to rotations. For example, the pioneering work of Esteves *et al*. [6] propose to map the point clouds into spherical functions and introduces a spherical convolution operator equivariant to rotations. SFCNN [20] replaces the manually designed mapping with a trainable neural network to learn to project the point clouds onto discretized spheres. Spezialetti *et al*. [22] propose a self-supervised strategy to learn canonical poses via spherical convolution. Shen *et al*. [21] define a transformation that maps both the inputs and the intermediate-layer features into unit quaternions to realize rotational equivariance. RotPredictor [7] proposes to transform the Cartesian coordinate system into a cylindrical one, on which rotations are represented as translations. Consequently, rotational equivariance can be achieved due to the translation-equivariant property of convolutional neural networks [31]. Although these works present significant robustness w.r.t. rotations, they are still not strictly rotation-invariant. Moreover, since most approaches in this thread require test-time augmentation (*a.k.a.*, voting) and such augmentations are often conducted by randomly sampling rotations from $\mathrm{SO}(3)$, their performances are not entirely stable and heavily depend on the sampling efficiency.

2.3. Rotation-invariant point cloud analysis

Another thread of works explores achieving strict rotational invariance. For example, SPH-Net [16] proposes to extend the point clouds to volumetric functions and designs a rotation-invariant convolution kernel based on spherical harmonics. Its expression ability is, however, limited due to the calculation of feature norms. There also exist methods that focus on handcrafting rotation-invariant features based on intrinsic geometries. For example, based on the relative locations, distances, and angles among points, RIConv [36], Triangle-Net [33], Li *et al*. [11], and SRI-Net [24] manually design different forms of rotation-invariant features and propose their respective network structures. Despite the well-designed networks, these handcrafted geometric features inevitably lead to information loss.

On the other hand, some approaches propose to convert rotated point clouds to their PCA-based canonical poses to achieve rotational invariance. Compared to the feature-based methods, these canonical poses can ultimately preserve the shape information of the input point clouds. For example, Fujiwara and Hashimoto [8] leverage the distance field to disambiguate the signs of canonical poses and concatenate them to formulate the inputs. Xiao *et al*. [34] and Yu *et al*. [35] claim that there are 8 sign ambiguities of the canonical poses and employ trainable attention-based selection modules for disambiguation. Kim *et al*. [10] propose to conduct PCA from local patches to the entire point clouds. Zhao *et al*. [37] combine the canonical poses with handcrafted features to formulate the inputs. In summary, although these works have developed varying network structures to suit the canonical poses better, the associated ambiguity problem is unfortunately not clearly studied.

3. Ambiguities of the canonical pose

In this section, we explore the exact number of ambiguities of the PCA-based canonical poses. To make the paper self-contained, we first briefly review how the canonical pose is calculated. Specifically, for a given point cloud $\mathbf{P} \in \mathbb{R}^{n\times 3}$, PCA is performed by:

$$\frac{\sum\left(\mathbf{P}_i-\bar{\mathbf{P}}\right)\left(\mathbf{P}_i-\bar{\mathbf{P}}\right)^T}{n}=\mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T, \tag{1}$$

where $\mathbf{P}_i \in \mathbb{R}^3$ is the i^{th} point of $\mathbf{P}$, $\bar{\mathbf{P}} \in \mathbb{R}^3$ is the center of $\mathbf{P}$, $\mathbf{E}$ is the eigenvector matrix composed of eigenvectors $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ (*a.k.a.*, principal axes), and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \lambda_3)$ are the corresponding eigenvalues (*a.k.a.*, principal values). By aligning the principal axes to the three axes of the world coordinate, we obtain the canonical pose as $\mathbf{P}_{\text{can}} = \mathbf{P}\mathbf{E}$. The rotation-invariant property of $\mathbf{P}_{\text{can}}$ can be facilely derived as follows:

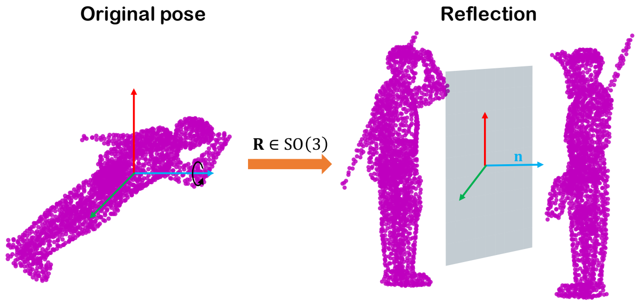

Figure 1: A conceptual explanation of improper rotation, which consists of: *1)* rotation along an axis; *2)* reflection along the plane perpendicular to this axis. Pure planar reflection can be considered as a special case of improper rotation where the rotation stage vanishes.

Proof. By applying a random rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ on the point cloud $\mathbf{P}$, we can obtain its rotated version $\mathbf{P}\mathbf{R}^T$. In the same manner as shown in Eq. (1), we can conduct PCA on it in the form of

$$\begin{aligned} &\frac{\sum \left(\mathbf{R}\mathbf{P}_i - \mathbf{R}\bar{\mathbf{P}}\right)\left(\mathbf{R}\mathbf{P}_i - \mathbf{R}\bar{\mathbf{P}}\right)^T}{n} \\ =&\mathbf{R}\left(\frac{\sum \left(\mathbf{P}_i - \bar{\mathbf{P}}\right)\left(\mathbf{P}_i - \bar{\mathbf{P}}\right)^T}{n}\right)\mathbf{R}^T \\ =&\left(\mathbf{R}\mathbf{E}\right)\mathbf{\Lambda}\left(\mathbf{R}\mathbf{E}\right)^T, \end{aligned} \tag{2}$$

where $\mathbf{RE}$ becomes the new principal axes. Therefore, as mentioned above, the canonical pose can be computed as

$$\left(\mathbf{P}\mathbf{R}^T\right)_{\mathrm{can}} = \mathbf{P}\mathbf{R}^T \cdot \mathbf{R}\mathbf{E} = \mathbf{P}\mathbf{E}, \tag{3}$$

on which the rotation $\mathbf{R}$ has no effect. □

3.1. Sign Ambiguity

As pointed out by some previous works [8, 34, 35], the PCA-based canonical pose contains sign ambiguities. Specifically, for a certain eigenvector $\mathbf{e}$, both $+\mathbf{e}$ and $-\mathbf{e}$ can satisfy the rule of eigen-decompositions. Consequently, by assigning different signs to each of the three eigenvectors, there are 8 possible poses when calculating the canonical pose of a given point cloud.

While this ambiguity problem is ignored in some existing works [10, 37], it is also explicitly pointed out and handled via trainable modules [34, 35] or analytical methods [8]. However, we argue that this 8-ambiguity declaration contains non-rotational transformations. In detail, assume that a specific combination of eigenvectors $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$ is with determinant 1. Consequently, only 4 out of the 8 ambiguities are with determinant 1, representing rotations, and the others are with -1, characterizing improper rotations (*i.e.*, the combination of rotation and reflection, as illustrated in Fig. 1). A detailed justification is presented in Table 1.

	Determinant	Geometric meaning
$[+\mathbf{e}_1, +\mathbf{e}_2, +\mathbf{e}_3]$	1	Rotation
$[-\mathbf{e}_1, -\mathbf{e}_2, +\mathbf{e}_3]$	1	Rotation
$[+\mathbf{e}_1, -\mathbf{e}_2, -\mathbf{e}_3]$	1	Rotation
$[-\mathbf{e}_1, +\mathbf{e}_2, -\mathbf{e}_3]$	1	Rotation
$[-\mathbf{e}_1, +\mathbf{e}_2, +\mathbf{e}_3]$	-1	Improper rotation
$[+\mathbf{e}_1, -\mathbf{e}_2, +\mathbf{e}_3]$	-1	Improper rotation
$[+\mathbf{e}_1, +\mathbf{e}_2, -\mathbf{e}_3]$	-1	Improper rotation
$[-\mathbf{e}_1, -\mathbf{e}_2, -\mathbf{e}_3]$	-1	Improper rotation

Table 1: Determinants and geometric meanings of the eigenvector matrix w.r.t. different assignment of signs.

3.2. Order ambiguity

The aforementioned claim of 8 ambiguities is also insufficient. Specifically, this claim takes for granted that the eigenvectors are sorted according to some rules (*e.g.*, by arranging the eigenvectors in the same order w.r.t. ascending/descending eigenvalues). However, we argue that such sorting rules are just manually defined for convenience. *I.e.*, there is no mathematical support for taking the eigenvector corresponding to the largest/smallest eigenvalue as the x-axis and the smallest/largest one as the z-axis. As we will show later in Sec. 6.2, this sorting-dependent order in fact obviously hinders the performances for aligning intra-category objects to different poses. Consequently, apart from the sign ambiguities, we claim that there also exist 6 order ambiguities by permuting the 3 eigenvectors.

By summarizing the above discussions of ambiguities together, we can conclude that there in fact should be

$$4\,(\text{sign ambiguities}) \times 6\,(\text{order ambiguities}) = 24$$

ambiguities in total speaking of the PCA-based canonical poses of point clouds. A detailed visualization can be found in the supplementary material.

4. Pose selector: Learning to disambiguate the canonical poses

In this section, we propose a novel function to cope with the ambiguities, which can effectively boost the performances of point cloud analysis tasks.

A straightforward strategy to incorporate the aforementioned ambiguities for rotation-invariant analysis is to consider every possible pose as an independent training instance, so that the network would learn to associate each possible pose to the label of the corresponding shape. Although this strategy works well, as we will show in the following sections, it is more effective to let the model learn to select the optimal canonical pose by incorporate pose selection into the training process. To this end, we design a trainable module to learn to use the 24 poses.

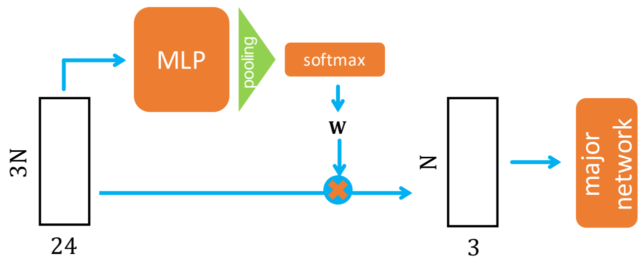

Figure 2: Structure of the pose selector, which predicts the weights of the 24-dimensional inputs. We implement the module with 256-512 1D convolutional filters followed by max-pooling, 24-dimensional linear mapping, and softmax.

We denote this module as "pose selector." As illustrated in Fig. 2, it is a lightweight trainable network that can be incorporated into any network developed for point cloud analysis. Specifically, by flattening the point cloud $\mathbf{P} \in \mathbb{R}^{n\times 3}$ to a vector and concatenating all the 24 ambiguities together, we obtain a $3n \times 24$ representation of all the canonical poses. Pose selector then takes this representation as input and axis-wise converts it into high dimensional features, which are pooled among the $3n$ dimensions to reach a single feature vector. This feature is further linearly mapped back to 24 dimensions and activated by softmax, resulting in a 24-dimensional weight vector $\mathbf{w}$. To obtain the selected point cloud $\mathbf{P}_{\text{sel}} \in \mathbb{R}^{n\times 3}$ in the optimal pose, we take the weighted sum:

$$\mathbf{P}_{\text{sel}} = \sum_{d=1}^{24} \mathbf{w}_d \mathbf{P}_d \,, \tag{4}$$

where $\mathbf{w}_d$ and $\mathbf{P}_d$ are the weight and corresponding canonical pose, respectively.

There remain ambiguities in the orders of the 24 poses during concatenation (*i.e.*, 24! possible orders for concatenation). However, owing to the closedness of $\mathrm{SO}(3)$, we can always generate the remaining 23 poses in a deterministic manner from an arbitrary canonical pose. Therefore, the order ambiguity that arises from concatenation can in fact be reduced to 24.

5. Performance study by using the correctly identified ambiguities

We claim that the proper identification of ambiguities can lead to more significant accuracy enhancements than well-designed networks or handcrafted rotation-invariant features. For verification, we conduct experiments on popularly used classification and segmentation benchmarks to compare with the performance of related works.

5.1. Implementation details

We employ the well-known DGCNN [30] network structure (with the spatial transformer network [9]) as our backbone without any modification. All the hyper-parameters and training setups remain the same as suggested in [30], except that the initial learning rate of the SGD optimizer is reduced from 0.1 to 0.01 for faster convergence and fine-tuning purpose. In the training phase, we randomly sample from all the possible 24! different concatenation patterns to robustify the model. For testing, we report two kinds of results: with and without test-time augmentation (TTA), which takes all the 24 concatenation patterns generated along a specific root as inputs and uses their mean value as the final prediction.

For experimental setup, we follow the related works [11, 24, 35, 36] and compare different algorithms in 3 modes:

- Both training and testing sets are rotated around the z-axis: z/z.
- Training set is rotated around the z-axis and testing set is randomly rotated: $z/\mathrm{SO}(3)$.
- Both training and testing sets are randomly rotated: $\mathrm{SO}(3)/\mathrm{SO}(3)$.

In our implementation, rotation matrices are randomly sampled with the *special_ortho_group* function of *Scipy* [29].

5.2. Object classification

We first carry out comparisons on the synthetic ModelNet40 dataset [32], which consists of 12311 meshes from 40 categories with 9843 for training and 2468 for testing. In practice, we use the data released by Qi *et al.* [17], where the point clouds are already pre-processed.

We employ both rotation-robust and rotation-invariant approaches for comparison. Peer methods characterized as rotation-robust all aim to learn rotation-equivariant representations by converting the Cartesian coordinates to either spherical or cylindrical ones. For the rotation-invariant counterparts, they mainly use the PCA-based canonical poses [10] or handcrafted geometric features [11, 24, 33, 36], or both of them [35, 37] as the inputs. Within our knowledge, LGR-Net [37] is currently the most competitive algorithm that leads the scoreboards of various tasks.

The results are presented in Table 2. As shown, it is facile to champion the accuracy providing correctly identified ambiguities. Among the peer algorithms, RI-GCN (xyz only) is especially meaningful in comparison since it also uses the PCA-based canonical poses as inputs and develops powerful networks to better aggregate local features. However, it did not address the ambiguity problem, resulting in a lower accuracy than our method.

	Method	Inputs	z/z	$z/\mathrm{SO}(3)$	$\mathrm{SO}(3)/\mathrm{SO}(3)$	Acc. drop
Rotation-sensitive	PointNet [17]	xyz	88.5	16.4	70.5	54.1
	DGCNN [30]	xyz	92.2	20.6	81.1	60.5
	PointNet++ [19]	xyz	89.3	28.6	85.0	56.4
	PointConv [31]	xyz	91.6	-	85.6	-
Rotation-robust	Shen *et al.* [21]	xyz	83.0	83.0	83.0	0.0
	Spherical CNN [6]	voxel	88.9	76.9	86.9	10.0
	a^3SCNN [13]	voxel	89.6	87.9	88.7	0.8
	SFCNN [20]	xyz	91.4	84.8	90.1	5.3
	SFCNN [20]	xyz + normal	**92.3**	85.3	91.0	5.7
	RotPredictor [7]	xyz	92.1	-	90.8	-
Rotation-invariant	RIConv [36]	feature	86.5	86.4	86.4	0.0
	Triangle-Net [33]	feature	-	-	86.7	-
	SRI-Net [24]	feature	87.0	87.0	87.0	0.0
	SPH-Net [16]	xyz	87.7	86.6	87.6	1.0
	Yu *et al.* [35]	xyz + feature	89.2	89.2	89.2	0.0
	Li *et al.* [11]	feature	89.4	89.4	89.3	0.1
	RI-GCN [10]	xyz	89.5	89.5	89.5	0.0
	RI-GCN [10]	xyz + normal	91.0	91.0	91.0	0.0
	LGR-Net [37]	xyz + normal + feature	90.9	90.9	91.1	0.2
	Ours (w/o TTA)	xyz	90.2	90.2	90.2	0.0
	Ours (w/ TTA)	xyz	91.6	**91.6**	**91.6**	0.0

Table 2: Classification accuracy (%) on the ModelNet40 dataset. Point clouds for all the methods are with density 1024. Within the inputs column, xyz denotes the Cartesian coordinates of the point clouds and features stand for handcrafted geometric features. The last column records the differences between $z/\mathrm{SO}(3)$ and $\mathrm{SO}(3)/\mathrm{SO}(3)$.

5.3. Object part segmentation

We also evaluate different methods on the part segmentation task, which aims to predict point-wise segmentation labels for the input point clouds. For experimental setup, we employ the ShapeNet part segmentation dataset [2] for benchmarking and use the pre-processed data released by Qi *et al.* [17], which consists of 16881 point clouds from 16 categories partitioned with 50 part labels in total. We follow the standard train-test split with 14007 objects for training and 2874 for testing. The experiments are again conducted in 3 modes as mentioned above. For evaluation, we calculate the mean intersection of union (mIoU, %) for each shape and report the mean value over all instances as the final results. The per-category mIoU is presented in the supplemental material for interested readers.

Table 3 presents the results. Some of the methods mentioned in the classification experiments are omitted since their networks are not modified for the segmentation task in their respective proposals. We emphasize that the enhancement over LGR-Net is achieved without including normals and geometric features in the input as it does. In fact, such complementary inputs are known to boost the performances (*i.e.*, as shown in Table 2, even just adding normals can lead to approximately 1% enhancements for both SFCNN and RI-GCN). Moreover, by comparing the performances on both classification and segmentation tasks, we can observe that the abilities of some specifically designed features and network structures [10, 11] are not very consistent among different point cloud analysis tasks, showing as the competitive performances on a particular application but limit results on the other one.

5.4. Classification on real-world dataset

A major concern of applying PCA to real-world point clouds lies in its sensitiveness w.r.t. various nuisances such as noise, incompleteness, and deformations. Therefore, in this section, we study how the PCA-based canonical poses perform in such scenarios. For experimental setup, we follow LGR-Net [37] and employ the *OBJ_BG* dataset from ScanObjectNN [28], which contains scans of 2890 indoor objects classified into 15 categories with 2312 for training and 578 for testing[2]. All objects are manually registered to the closest poses w.r.t. their CAD models, leading to analogously the same pose within each category. In general, this dataset is much more challenging than the synthetic ModelNet40 for containing noise, holes, deformations, and backgrounds.

We report the results in Table 4. Compared to LGR-Net,

[2]The author-released dataset contains 2890 point clouds although it is claimed as 2902 in [28].

	Method	Inputs	z/SO(3)	SO(3)/SO(3)	Drop of mIoU
Rotation-sensitive	PointCNN [12]	xyz	34.7	71.4	36.7
	DGCNN [30]	xyz	37.4	73.3	35.9
	PointNet [17]	xyz	37.8	74.4	36.6
	PointNet++ [19]	xyz	48.2	76.7	28.5
Rotation-robust & invariant	Triangle-Net [33]	feature	-	72.5	-
	RIConv [36]	feature	75.3	75.5	0.2
	RI-GCN [10]	xyz	77.2	77.3	0.1
	SRI-Net [24]	feature	80.0	80.0	0.0
	Li *et al.* [11]	feature	82.2	82.5	0.3
	LGR-Net [37]	xyz + normal + feature	-	82.8	-
	Ours (w/o TTA)	xyz	81.7	81.7	0.0
	Ours (w/ TTA)	xyz	**83.1**	**83.1**	0.0

Table 3: Mean IoU (%) over all instances on ShapeNet. The inputs of each algorithm are the same as mentioned in Table 2. All point clouds are with density 2048 except for RI-GCN [10] and Triangle-Net [33], whose inputs are with density 1024.

	z^*/z^*	z^*/SO(3)	SO(3)/SO(3)
PointNet [17]	79.4	16.7	54.7
PointNet++ [19]	87.8	15.0	47.4
PointCNN [12]	**89.9**	14.6	63.7
DGCNN [30]	87.3	17.7	71.8
RIConv [36]	-	78.4	78.1
LGR-Net [37]	-	81.2	81.4
Ours (w/o TTA)	84.3	84.3	84.3
Ours (w/ TTA)	86.7	**86.7**	**86.7**

Table 4: Classification accuracy (%) on the real-world ScanObjectNN dataset. z^* denotes the original pre-aligned dataset without any rotations. All point clouds are with density 1024. Our approach can noticeably outperform LGR-Net, which is the best peer method on synthetic datasets.

which is the best peer method on both ModelNet40 and ShapeNet, our method presents a noticeable outperformance on the challenging real-world dataset (*i.e.*, 5.3% enhancement in accuracy). This result indicates that the PCA-based canonical poses with properly identified ambiguities are in fact more effective than some handcrafted features and related network structures. Furthermore, by comparing the results under the z/z setup, we can observe that the result presented by PCA-based canonical poses is only slightly worse than the one obtained with the manual alignment, demonstrating the robustness of PCA-based canonical forms w.r.t. various nuisances in the real world.

5.5. Ablation study of the pose selector

In this section, we study the effectiveness of our proposed pose selector module. For comparison, in the training phase, we randomly select 1 from the 24 ambiguities and insert it into the vanilla DGCNN network. For testing, we conduct TTA over all the ambiguities. Experiments are carried out

	ModelNet40	ShapeNet
w/o pose selector	91.3	82.8
w/ pose selector	91.6	83.1

Table 5: Pose selector can increase the accuracy of different tasks compared to the vanilla network, demonstrating its effectiveness in disambiguating the canonical poses.

on both classification and segmentation tasks for clear comparison. Results are reported in Table 5. As shown, our pose selector can effectively enhance the performance on different tasks by merging the information of all the possible canonical poses. Furthermore, we observe that the pose selector can also accelerate the convergence compared to using the vanilla DGCNN network.

6. Further explorations regarding the PCA-based canonical poses

Despite its effectiveness, the capacities of PCA-based canonical poses are not clearly studied. Therefore, we desire to explore the following questions with the hope to prompt future research in the context of rotation-invariance point cloud analysis:

- How well can the PCA-based canonical poses perform?
- Can we reduce the number of ambiguities in the light of disambiguation methods?
- To what extent do the inaccurate ambiguities hinder the performance?
- What does PCA conduct on the point clouds? In what conditions does it perform the best?

	Pre-aligned pose	Selected canonical pose	All possible canonical poses
Acc.	92.9	92.0	91.6
mIoU	85.2	84.7	83.1

Table 6: Classification accuracy on ModelNet40 and mIoU (%) on ShapeNet w.r.t. different input pose on the classification and part segmentation tasks. Results from left to right are obtained with: the original datasets; manually selected canonical pose via Eq. (5); and all the 24 ambiguities.

All the experiments mentioned hereafter are with exactly the same setup as mentioned in Sec. 5. TTA is conducted for all the experiments containing ambiguities. For experiments that do not contain ambiguities, we switch to use the vanilla DGCNN without the pose selector.

6.1. How well can the canonical poses perform?

Regardless of the ambiguity problem, PCA also cannot perfectly align intra-category objects to the same canonical pose due to their varying shapes. Consequently, some existing works [7, 10, 35, 37] consider these imperfect alignments as the primary reason that hampers the performance when canonical poses are used for rotation-invariant point cloud analysis. However, we argue that such a declaration is biased due to the improperly identified ambiguities.

In this section, we test the maximum capability of the canonical poses. Specifically, we desire to explore how well the canonical poses can perform if the ambiguity problem is ideally solved. For experimental setup, we manually select 1 pose from the 24 ambiguities to make the poses of intra-category objects as similar as possible. In the implementation, since the original point clouds within the ModelNet40 dataset are already precisely aligned, we use them as reference and select 1 from all the 24 possible canonical poses that minimizes the rotational residual:

$$\mathbf{P}_{\text{selected}} = \operatorname*{argmin}_{\mathbf{P}\in\mathcal{A}} \|\mathbf{R}_{\mathbf{P}} - \mathbf{I}\|_F \,, \tag{5}$$

where $\mathcal{A}$ consists of the 24 canonical poses, $\mathbf{R}_{\mathbf{P}}$ is the relative rotation from the current candidate canonical pose $\mathbf{P}$ to the reference calculated by the method of Umeyama [27], and $\mathbf{I}$ is the 3D identity matrix.

We conduct experiments on both classification and part segmentation tasks to extensively demonstrate the potential of the PCA-based canonical poses. Results are reported in Table 6. As presented, both classification and segmentation accuracies can be further enhanced if the ambiguities are ideally handled. For the segmentation task, the accuracy even approaches the full capacity of the network structure. Consequently, we can conclude that contrary to the common blame on the imperfect alignment, it is in fact the incorrect number of ambiguities that primarily lowers the accuracy.

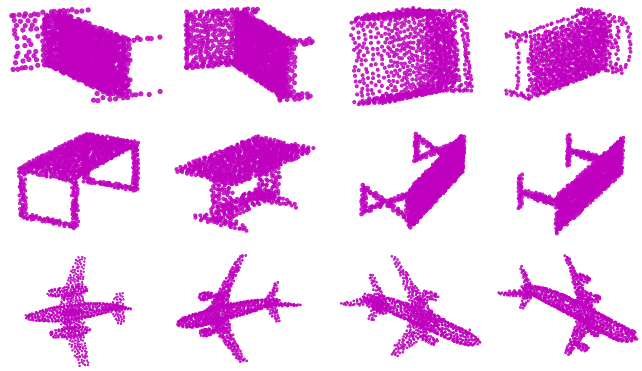

Figure 3: Disambiguation methods may assign objects within the same category to different poses.

	Strat. I	Strat. II	Strat. III	All possible canonical poses
Acc. (%)	89.5	90.3	87.2	91.6

Table 7: Classification accuracy (%) w.r.t. distinct disambiguation strategies on ModelNet40. The number of ambiguities from left to right are 4, 6, 0, 24, respectively.

6.2. Can disambiguation methods help?

Although the ambiguity cannot be addressed mathematically, there do exist some conventions on disambiguation. In this section, we explore whether these methods can benefit point cloud analysis tasks. Specifically, we conduct experiments with 3 disambiguation strategies:

- Strategy I: Determine the order of eigenvectors by sorting the corresponding eigenvalues in ascending order. This operation reduces the number of ambiguities to 4.
- Strategy II: Determine the sign of eigenvectors by letting more data lie on the positive half-axes. This strategy states that the principal axes should keep the same signs with the majority of the data vectors [1, 26]. Accordingly, the number of ambiguities is reduced to 6.
- Strategy III: Combine the aforementioned Strategies I and II together to eliminate all the ambiguities.

As shown in Table 7, these disambiguation methods are unideal for point cloud analysis tasks, as there is a noticeable drop in accuracy compared to the case where all the ambiguities are considered. This is because objects within the same category may still lie in different poses after such disambiguation procedures. A detailed illustration is presented in Fig. 3, where poses are obtained with Strategy III.

6.3. Do inaccurate ambiguities have any effect?

We use Strategy I from Sec. 6.2 to determine the order of eigenvectors and assign all the 8 possible combinations

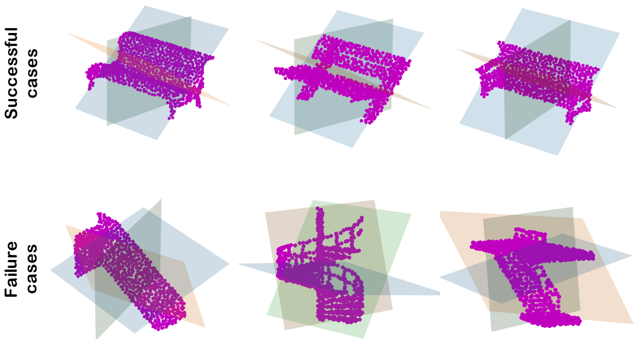

Figure 4: Conceptual illustrations of the PCA-based alignments on non-symmetric objects. Planes characterize the three principal axes detected by PCA.

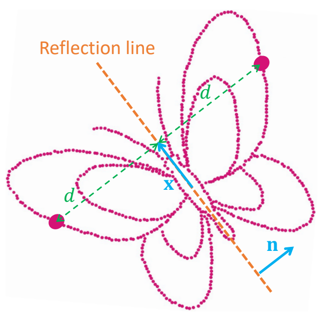

Figure 5: A point cloud can be denoted as $\mathbf{x}+d\mathbf{n}$ and $\mathbf{x}-d\mathbf{n}$ given known symmetry.

of signs to it. By doing so, we obtain an accuracy of 89.2%, which is 2.4% lower than the one obtained with correctly identified ambiguities. Moreover, this result is also worse than the one obtained with correct but insufficient 4 sign ambiguities (Strategy I of Table 7), demonstrating that the incorrectly identified ambiguities do harm the performance.

6.4. When does PCA perform the best?

We claim that PCA is effective for aligning objects that contain orthogonal plane-reflective symmetries. Specifically, for a symmetric object with orthogonal reflection planes, (part of) its principal axes would always be parallel to the normals of these planes [3, 15]. As simple proof, let us assume that a given centralized point cloud $\mathbf{P}$ is symmetric along a plane with unit normal $\mathbf{n}$. Then, as shown in Fig. 5, we can denote each pair of symmetric points as $\mathbf{x}+d\mathbf{n}$ and $\mathbf{x}-d\mathbf{n}$, where $\mathbf{x}$ belongs to the reflection plane and d is the point-to-plane distance. Thus the covariance matrix of $\mathbf{P}$ can be calculated as:

$$\begin{aligned}\mathbf{C} &= \sum_i \left(\mathbf{x}_i + d_i\mathbf{n}\right)\left(\mathbf{x}_i + d_i\mathbf{n}\right)^T + \left(\mathbf{x}_i - d_i\mathbf{n}\right)\left(\mathbf{x}_i - d_i\mathbf{n}\right)^T \\ &= \sum_i \mathbf{x}_i\mathbf{x}_i^T + 2d_i^2\mathbf{n}\mathbf{n}^T. \end{aligned} \tag{6}$$

By multiplying $\mathbf{n}$ on both side of Eq. (6), we can obtain the following formulation with the property that $\mathbf{x}$ is always perpendicular to $\mathbf{n}$:

$$\mathbf{C}\mathbf{n} = \left(\sum_i \mathbf{x}_i\mathbf{x}_i^T + 2d_i^2\mathbf{n}\mathbf{n}^T\right)\mathbf{n} = 2\sum_i d_i^2\mathbf{n}, \tag{7}$$

which is an eigen-problem identical to the one in PCA.

Since Eq. (7) holds for all reflection planes, a point cloud with more than 2 orthogonal plane-reflections would have its principal axes uniquely determined up to the 24 sign and order ambiguities. Therefore, it is facile for PCA to perform excellent intra-class alignment on objects that consist of orthogonal plane-symmetries, such as desks, tables, and bathtubs. Owing to their man-made properties, these alignments are often presented as upright on the tabletop. Moreover, we observe that PCA can still achieve acceptable alignment on objects where the symmetries are only approximately satisfied, such as guitars, cups, and cars. However, this does not imply PCA would fail if orthogonal symmetries do not exist. As shown in the top row of Fig. 4, intra-class objects can still be aligned to similar poses owing to their similar structures. However, the alignments would be arbitrary if the intra-class shapes are not even similar in structure, as shown in the bottom row of Fig. 4, the aligned poses of the stairs are random due to their completely different structures.

7. Discussion and conclusions

This work explores the identification of proper pose ambiguities and its effects when PCA is used to achieve rotation-invariant point cloud analysis. A pose selector module is also developed for disambiguation. Although many efforts had been spent on developing powerful network structures or handcrafting descriptive features to complement the PCA-based canonical poses, our experiments indicate that it is the correctly identified ambiguities that can lead to more significant performance boosts, evident from the noticeable accuracy enhancements on various datasets.

We hope our analyses can prompt further rethinking and future network designs regarding rotation-invariant point cloud analysis. For example, we plan to study whether a more effective disambiguation module can be developed. Another thread of work lies in achieving more precise alignments by robustifying the vanilla PCA.

Acknowledgment

This work was supported by NII CRIS collaborative research program operated by NII CRIS and LINE Corporation. We would like to thank Menandro Roxas for fruitful discussions.

References

[1] Rasmus Bro, Evrim Acar, and Tamara G Kolda. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics*, 22(2):135–140, 2008. 7

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[3] Mohamed Chaouch and Anne Verroust-Blondet. Alignment of 3d models. *Graphical Models*, 71(2):63–76, 2009. 8

[4] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[6] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 5

[7] Jin Fang, Dingfu Zhou, Xibin Song, Shengze Jin, Ruigang Yang, and Liangjun Zhang. Rotpredictor: Unsupervised canonical viewpoint learning for point cloud classification. In *Proceedings of International Conference on 3D Vision (3DV)*, 2020. 1, 2, 5, 7

[8] Kent Fujiwara and Taiichi Hashimoto. Neural implicit embedding for point cloud analysis. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 3

[9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*, 2015. 1, 4

[10] SeoHyun Kim, JaeYoo Park, and Bohyung Han. Rotation-invariant local-to-global representation learning for 3d point cloud. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, 2020. 1, 2, 3, 4, 5, 6, 7

[11] Xianzhi Li, Ruihui Li, Guangyong Chen, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. A rotation-invariant framework for deep point cloud analysis. *arXiv preprint arXiv:2003.07238*, 2020. 2, 4, 5, 6

[12] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on χ-transformed points. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, 2018. 6

[13] Min Liu, Fupin Yao, Chiho Choi, Ayan Sinha, and Karthik Ramani. Deep learning 3d shapes using alt-az anisotropic 2-sphere convolution. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 5

[14] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2015. 2

[15] Predrag Minovic, Seiji Ishikawa, and Kiyoshi Kato. Symmetry identification of a 3-d object represented by octree. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(5):507–514, 1993. 8

[16] Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov. Effective rotation-invariant point cnn with spherical harmonics kernels. In *Proceedings of International Conference on 3D Vision (3DV)*, 2019. 2, 5

[17] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 4, 5, 6

[18] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[19] 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. 5, 6

[20] Yongming Rao, Jiwen Lu, and Jie Zhou. Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 5

[21] Wen Shen, Binbin Zhang, Shikun Huang, Zhihua Wei, and Quanshi Zhang. 3d-rotation-equivariant quaternion neural networks. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 5

[22] Riccardo Spezialetti, Federico Stella, Marlon Marcon, Luciano Silva, Samuele Salti, and Luigi Di Stefano. Learning to orient surfaces by self-supervised spherical cnns. *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, 2020. 2

[23] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 2

[24] Xiao Sun, Zhouhui Lian, and Jianguo Xiao. Srinet: Learning strictly rotation-invariant representations for point cloud classification and segmentation. In *Proceedings of International Conference on Multimedia (ACMMM)*, 2019. 1, 2, 4, 5, 6

[25] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019. 2

[26] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2010. 7

[27] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *Computer Architecture Letters*, 13(04):376–380, 1991. 7

[28] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019. 5

[29] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 4

[30] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Transactions on Graphics (TOG)*, 38(5):1–12, 2019. 1, 2, 4, 5, 6

[31] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 5

[32] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4

[33] Chenxi Xiao and Juan Wachs. Triangle-net: Towards robustness in point cloud learning. In *Proceedings of Winter Conference on Applications of Computer Vision (WACV)*, 2021. 2, 4, 5, 6

[34] 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 *Proceedings of International Conference on Multimedia and Expo (ICME)*, 2020. 1, 2, 3

[35] Ruixuan Yu, Xin Wei, Federico Tombari, and Jian Sun. Deep positional and relational feature learning for rotation-invariant point cloud analysis. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 4, 5, 7

[36] Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *Proceedings of International Conference on 3D Vision (3DV)*, 2019. 1, 2, 4, 5, 6

[37] Chen Zhao, Jiaqi Yang, Xin Xiong, Angfan Zhu, Zhiguo Cao, and Xin Li. Rotation invariant point cloud classification: Where local geometry meets global topology. *arXiv preprint arXiv:1911.00195*, 2019. 1, 2, 3, 4, 5, 6, 7