SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2023, Vol. 66 152102:1–152102:15
https://doi.org/10.1007/s11432-021-3570-5

Cy-CNN: cylinder convolution based rotation-invariant neural network for point cloud registration

Hengwang ZHAO[1†], Zhidong LIANG[2†], Yuesheng HE[1], Chunxiang WANG[1] & Ming YANG[1*]

[1]*Department of Automation, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[2]*Research Institute of Robotics, Shanghai Jiao Tong University, Shanghai* 200240*, China*

Received 4 July 2021/Revised 14 November 2021/Accepted 18 April 2022/Published online 23 April 2023

Abstract Point cloud registration is a challenging problem in the condition of large initial misalignments and noises. A major problem encountered in the registration algorithms is the definition of correspondence between two point clouds. Point clouds contain rich geometric information and the same geometric structure implies the same feature even if they are in different poses, which motivates us to seek a rotation-invariant feature representation for calculating the correspondence. This work proposes a rotation-invariant neural network for point cloud registration. To acquire rotation-invariant features, we firstly propose a rotation-invariant point cloud representation (RIPR) at the input level. Instead of using the original coordinates, we propose to use point pair features (PPF) and the transformed coordinates in the local reference frame (LRF) to represent a point. Then, we design a new convolution operator named Cylinder-Conv which utilizes the symmetry of cylinder-shaped voxels and the hierarchical geometry information of the surface of 3D shapes. By specifying the cylinder-shaped structures and directions, Cylinder-Conv can better capture the local neighborhood geometry of each point and maintain rotation-invariance. Finally, we combine RIPR and Cylinder-Conv to extract normalized rotation-invariant features to generate the correspondence and perform a differentiable singular value decomposition (SVD) step to estimate the rigid transformation. The proposed network presents state-of-the-art performance on point cloud registration. Experiments show that our method is robust to initial misalignments and noises.

Keywords deep learning, point cloud registration, rotation invariant, computer vision

Citation Zhao H W, Liang Z D, He Y S, et al. Cy-CNN: cylinder convolution based rotation-invariant neural network for point cloud registration. Sci China Inf Sci, 2023, 66(5): 152102, https://doi.org/10.1007/s11432-021-3570-5

1 Introduction

Point clouds have recently gained increasing relevance in the field of computer vision as a powerful 3D representation model [1–3]. Point cloud registration is one of the fundamental problems in many tasks such as 3D reconstruction [4], pose estimation [5], and object tracking [6]. Registration aims to find a rigid transformation matrix that aligns two point clouds. Although remarkable progress has been achieved, it is still a challenging problem for point clouds in the condition of large initial misalignments and noises.

The iterative closest point (ICP) algorithm [7,8] and its variants [9] are widely used methods for point cloud registration. But they are known to easily converge to a local minimum. Some methods [10, 11] introduce probability models to the registration framework, but they are also vulnerable to local minima under large initial misalignment. To improve the robustness to initial misalignment, some methods [12,13] use hand-crafted rotation-invariant features [14] for correspondence generation. Although these feature-based methods are more robust to large misalignments, they are sensitive to noises, because the hand-crafted features are not stable on noisy point clouds.

* Corresponding author (email: MingYANG@sjtu.edu.com)
† Zhao H W and Liang Z D have the same contribution to this work.

© Science China Press 2023

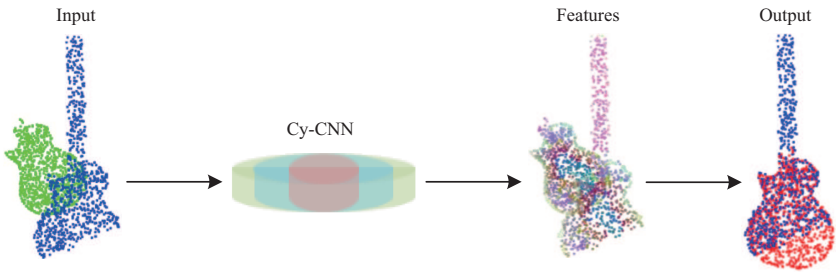

Figure 1 (Color online) The illustration of Cy-CNN for point cloud registration. The inputs are the source (blue) point cloud and the target (green) point cloud with partial occlusion and noises. We use Cy-CNN to extract point-wise features for the two point clouds. The features are visualized using t-SNE [18]. It clearly shows that the features extracted by Cy-CNN are rotation-invariant and conducive to correspondence generation, and the final output (red) is well aligned with the source point cloud.

Recently, deep learning is gradually applied to point cloud registration for its powerful capability of feature representation [15–17]. Most of the learning-based registration methods try to learn rotation-invariant features for correspondence generation. Although significant progress has been made, these methods cannot maintain rotation-invariance during both the input stage and the feature learning stage, making them hard to generalize to unseen initial misalignment. Besides, how to effectively utilize the surface geometry information of 3D shapes for stable rotation-invariant feature learning is also to be explored.

In this paper, we propose Cy-CNN, a cylinder convolution based rotation-invariant neural network for point cloud registration, robust to initial misalignments and noises, as shown in Figure 1 [18]. In order to acquire rotation-invariant features, we firstly propose a rotation-invariant point cloud representation (RIPR) in the input stage. We use point pair features (PPF) and the transformed coordinates in the local reference frame (LRF) to represent a point, instead of using the original coordinates. Then, a new convolution operator is designed to directly compute the convolution on point clouds. This new convolution operator, which is named Cylinder-Conv, leverages the symmetry of cylinder-shaped voxels and the hierarchical geometry information of the surface of 3D shapes. By specifying the cylinder-shaped structures and directions, Cylinder-Conv can better capture the local neighborhood geometry of each point and maintain rotation-invariance. Finally, we combine RIPR and Cylinder-Conv to extract normalized rotation-invariant features. By analyzing the similarity of the normalized features, we generate the correspondence of the two point clouds. A differentiable singular value decomposition (SVD) step is performed to estimate the final rigid transformation. The key contributions of our work are as follows:

- We propose a RIPR, which combines PPF and the transformed coordinates in the LRF to represent a point.
- We propose a new rotation-invariant convolution operator named Cylinder-Conv. It can help to capture the hierarchical geometry information of the surface of 3D shapes and maintain rotation-invariance.
- We propose Cy-CNN, a cylinder convolution based rotation-invariant neural network for point cloud registration. It combines RIPR and Cylinder-Conv to extract rotation-invariant features for point cloud registration. The proposed network presents state-of-the-art performance on point cloud registration.

2 Related work

Classical point cloud registration. The classical point cloud registration algorithms can be classified into two categories: correspondence based registration algorithms and direct registration algorithms. The correspondence based registration algorithm consists of establishing the correspondence of two point clouds and optimizing the distance of such correspondence. ICP [7, 8] establishes correspondences by searching the closest point in another point cloud and uses the Euclidean metric to optimize the distance between correspondences. Some variants of ICP use point-to-plane [19], point-to-line [20], or plane-to-plane [9] to establish the correspondences or improve the optimization algorithm [21]. However, these methods are sensitive to initial misalignments. Probabilistic models [10, 11] are applied to handle uncertainty and partiality, but they are still vulnerable to local minima under large initial misalignments. The hand-crafted rotation-invariant features [14, 22, 23] based methods establish correspondences using local feature descriptors, and remove outliers with RANSAC [14] or other techniques [12, 24–26]. These registration methods are robust to initial misalignments, but the hand-crafted features are easily affected

by noises and cause messy correspondences. For direct registration methods, point-wise corresponding points are not assigned. Refs. [27–29] represented a point cloud as a Gaussian mixture model (GMM) where the centroids are points in the point set. But these methods are affected by partial occlusions and cannot guarantee real-time performance.

Learning based point cloud registration. (1) Deep learning on point cloud. The point cloud data is unordered, so it is challenging to perform the convolution operation directly on the point cloud data. Some methods convert raw point clouds to regular representations. View-based methods [30, 31] project point clouds to a collection of images and utilize the 2D convolution. Volumetric methods [32, 33] convert point clouds to 3D voxels and apply the 3D convolution. PointNet [34] is a pioneer directly processing raw point clouds addressing the problem of permutation invariance. Refs. [35–37] considered the local relationship of point sets. Refs. [38–40] fitted graph convolution [41] to point cloud for better learning localized features. However, these methods are not specially designed on data representation and feature extraction for achieving rotation-invariant features. Ref. [42] applied spherical harmonics kernels to achieve rotation-invariant convolution. But it does not design a rotation-invariant representation of the point cloud in the input stage. Refs. [43,44] designed rotation-invariant representations based on reference points of the complete point cloud. Ref. [45] mapped a point cloud into the unit spherical space (relative to the centroid of the complete point cloud) and extracted features using spherical voxel convolution. But the reference points of the complete point cloud (like the centroid) are not stable in partially visible point clouds, which causes these rotation-invariant representations to be easily affected by the partial occlusions. At the same time, these approaches are designed for classification and segmentation tasks and cannot be directly applied to registration tasks.

(2) Learning based point cloud registration. Recently, learning-based methods are gradually attempting to apply to point cloud registration. Refs. [46, 47] proposed to learn 3D descriptors utilizing deep neural networks and PPF. A concurrent study [48] to our method proposes to learn 3D descriptors using a 3D cylindrical convolution layer. Although all extract rotation-invariant features utilizing the properties of the cylinder, the voxel division and convolution mechanism are different from each other. Besides, these methods [46–48] only focus on descriptor learning, in which obtaining registration results is not the final goal. To achieve final registration, approaches such as RANSAC are still necessary to iteratively remove outliers. Some recent studies [15–17,49–51] presented end-to-end point cloud registration methods. Refs. [15, 50] applied PointNet to learn the global features of point clouds, and iteratively estimated transformations using neural networks. However, such methods cannot capture local information. To exploit local information, Ref. [49] detected the keypoints through a point weighting deep neural network. DCP [16] extracts features for each point for point cloud registration. RPM-Net [17] also learns point-wise features and establishes the correspondences with a differentiable Sinkhorn layer. However, these end-to-end point cloud registration methods [15–17, 49–51] cannot maintain rotation-invariance both in the input stage and the feature learning stage. In our work, the proposed RIPR and Cylinder-Conv ensure the property of rotation-invariance in the input stage and the feature learning stage, respectively.

3 Methodology

In this study, we propose a rotation-invariant convolutional neural network for point cloud registration leveraging the symmetry of cylinder-shaped voxels and the hierarchical geometry information of the surface of 3D shapes. In this section, we introduce the main technical components of the proposed method that include: rotation-invariant point cloud representation, convolutions on cylinder-shaped voxels, and architecture.

3.1 Rotation-invariant point cloud representation

To eliminate the influence of rotations at the input stage and leverage the local structure of point clouds, we propose RIPR combining PPF [52] and the transformed coordinates in the LRF [23], as is shown in Figure 2. In the existing approaches [23, 52], PPF and LRF are always used independently. To the best of our knowledge, this is the first time combining PPF and LRF as a rotation-invariant representation for point cloud feature learning, considering the combination of them provides the network with richer rotation-invariant information at the input stage. Let X denote the input point cloud, where $X = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^3$. For each point $x_i \in X$, we define a local neighborhood $\mathcal{N}(x_i)$ containing K

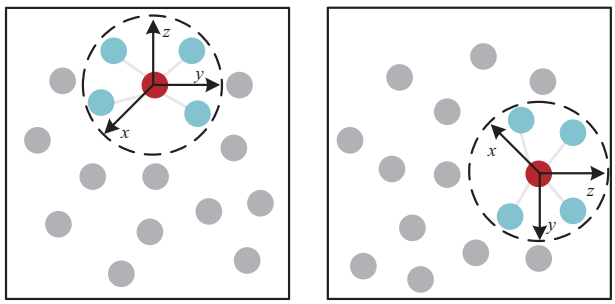

Figure 2 (Color online) Illustration of the RIPR. The circle (shaped in 3D) with radius r means the influence region for a RIPR. The coordinate system is the LRF in the local region. The query point x_i (red) and its K nearest neighbor points x_j (blue) form K pairs. (a) Part of the points in X; (b) part of the points in X after rotations.

nearest neighbor points. The RIPR of x_i is given by

$$\mathrm{RIPR}(x_i, x_j) = \left\{\mathrm{PPF}(x_i, x_j) \in \mathbb{R}^4, x'_j \in \mathbb{R}^3\right\} \in \mathbb{R}^7, \tag{1}$$

where $x_j \in \mathcal{N}(x_i)$, x'_j is the transformed coordinates in the LRF, $\mathrm{PPF}(x_i, x_j)$ are point pair features [52] which describe relative distance and angle between x_i and each neighbor points x_j.

$$\mathrm{PPF}(x_i, x_j) = (\angle(n_i, n_j), \angle(n_i, \Delta x_{i,j}), \angle(n_j, \Delta x_{i,j}), \|\Delta x_{i,j}\|) \in \mathbb{R}^4, \tag{2}$$

where $\|\cdot\|$ denotes the Euclidean norm, $\angle(\cdot, \cdot)$ denotes the angle between two vectors, n_i and n_j are the normals of points x_i and x_j, and $\Delta x_{i,j}$ denotes the offset between x_i and x_j:

$$\Delta x_{i,j} = x_j - x_i. \tag{3}$$

To calculate the rotation-invariant transformed coordinates of x_j, we construct the LRF by the normal vector and the average normalized $\Delta x_{i,j}$:

$$\begin{aligned} e_x &= \frac{\sum_{j=1}^{K} \frac{\Delta x_{i,j}}{\|\Delta x_{i,j}\|}}{K}, \\ e_y &= e_x \times n_i, \\ e_z &= n_i, \end{aligned} \tag{4}$$

where $\times$ denotes the cross product, e_x, e_y and e_z are the coordinate axes of the LRF. The transformed coordinates $x'_j = (x'_{jx}, x'_{jy}, x'_{jz}) \in \mathbb{R}^3$ of x_j are calculated as

$$\begin{aligned} x'_{jx} &= {\Delta x_{i,j}}^{\mathrm{T}} e_x, \\ x'_{jy} &= {\Delta x_{i,j}}^{\mathrm{T}} e_y, \\ x'_{jz} &= {\Delta x_{i,j}}^{\mathrm{T}} e_z. \end{aligned} \tag{5}$$

To obtain the RIPR of x_i, we concatenate the representations of each neighboring points $x_j \in \mathcal{N}(x_i)$ into a vector $\mathrm{RIPR}_{x_i} \in \mathbb{R}^{K\times 7}$, which is invariant to any rotation in SO(3). In our design, PPF describes the relative distance and angles between the query point and neighbor points, and the transformed coordinates in the LRF describe the relative position in the local region. While maintaining rotation-invariance, the combination of them makes the point cloud representation contain rich geometric information, conducive to subsequent point cloud feature learning.

3.2 Convolutions on cylinder-shaped voxels

To learn point-wise rotation-invariant features for point cloud registration, we define a rotation-invariant convolution operator named Cylinder-Conv that is able to directly work with point clouds. The convolution operator is applied at each point in a point cloud to extract point-wise features. Figure 3 shows an example of the Cylinder-Conv operator focusing on one point. Benefiting from the symmetry of the cylinder, we can obtain the same points before and after rotation in each cylinder-shaped voxel, which

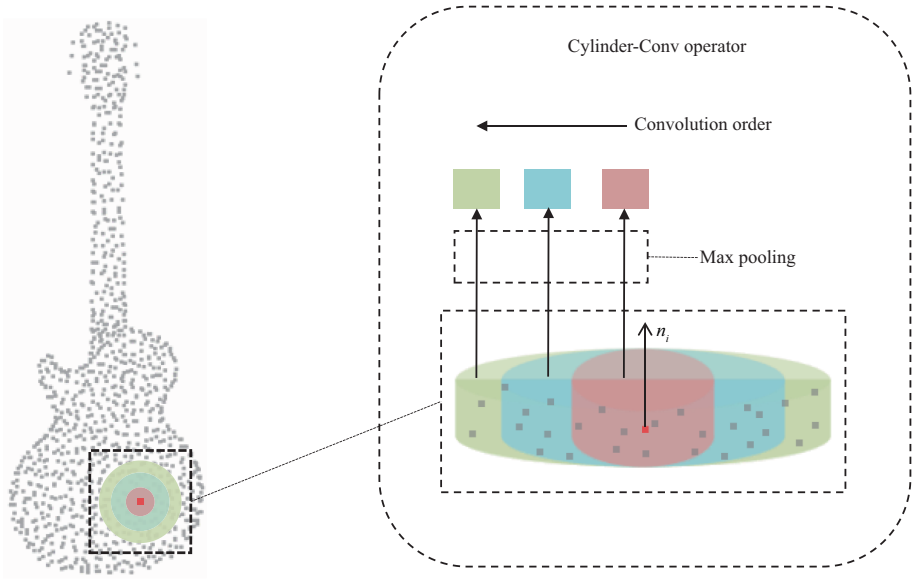

Figure 3 (Color online) Illustration of the proposed rotation-invariant convolution operator Cylinder-Conv. The convolution operator is applied at each point in a point cloud. This is an example of a convolution operator with three cylinder-shaped voxels focusing on one point. The red point is where the Cylinder-Conv operator is applied, and the gray points in the cylinder-shaped voxels are the neighbor points of the red point. n_i is the calculated normal direction of the red point. The cylinder-shaped voxels have a ring-like shape. Colors denote different cylinder-shaped voxels without overlapping. These voxels are naturally ordered from the innermost to the outermost. We use max-pooling to aggregate features of the points in each voxel.

makes Cylinder-Conv have the property of rotation invariance. Besides rotation-invariance, the distinguishment of the features in different local areas is also very important for finding exact corresponding points. To better capture the local neighborhood geometry of each point and make the feature more distinguishable, we define a series of cylinder-shaped voxels. The so-called cylinder-shaped voxel has a ring-like shape except the center one (the red cylinder shown in Figure 3) and these cylinder-shaped voxels have no overlaps.

To define a cylinder-shaped voxel, we specify the radius, height, and direction. The radius and height of a cylinder-shaped voxel determine the receptive field of the proposed convolution operator. Considering that the normal direction of the local area is rotation-invariant relative to the object and can represent the surface geometry information of the object, we use the normal direction as the axis direction of the cylinder-shaped voxel. Principal component analysis (PCA) is used to calculate the normal of local area [53]. Figure 3 shows a Cylinder-Conv operator with three cylinder-shaped voxels. The union of these cylinder-shaped voxels yields the domain Ω_p. We use $|\Omega_p|$ to represent the number of voxels in the domain Ω_p. We can define the convolution as

$$F(p)^{(n)} = \sum_{C \in \Omega_p^{(n)}} \omega_C^{(n)} F(C)^{(n-1)}, \tag{6}$$

where $\omega_C \in \mathbb{R}^{O \times I}$ denotes the weights applied on one of the cylinder-shaped voxel $C \in \Omega_p$, $\omega = \{\omega_C \mid C \in \Omega_p\} \in \mathbb{R}^{|\Omega_p| \times O \times I}$ denotes the weights of Cylinder-Conv, O denotes the dimension of output features, superscript (n) denotes the data or parameters of layer n. $F(C) \in \mathbb{R}^I$ denotes the feature of one cylinder-shaped voxel $C \in \Omega_p$, where I stands for the dimension of input features. $F(p) \in \mathbb{R}^O$ denotes the features of the point p. Because the cylinder-shaped voxels are naturally ordered (from innermost to outermost), there is no ambiguity among the cylinder-shaped voxels. To make the features more robust to noise, we apply max-pooling to aggregate features of the points in each cylinder-shaped voxel:

$$F(C) = \text{maxpooling}(\{F(q) : q \in \Omega_C\}), \tag{7}$$

where $F(q) \in \mathbb{R}^I$ denotes the features of the point q, Ω_C denotes the domain of one cylinder-shaped voxel C. The convolution operator is applied at each point in a point cloud to generate point-wise features. Figure 3 shows an example of focusing on one point.

The core of the proposed Cylinder-Conv is to aggregate hierarchical local surface information in a rotation-invariant way, and Cylinder-Conv is applied to each point in a point cloud to extract features.

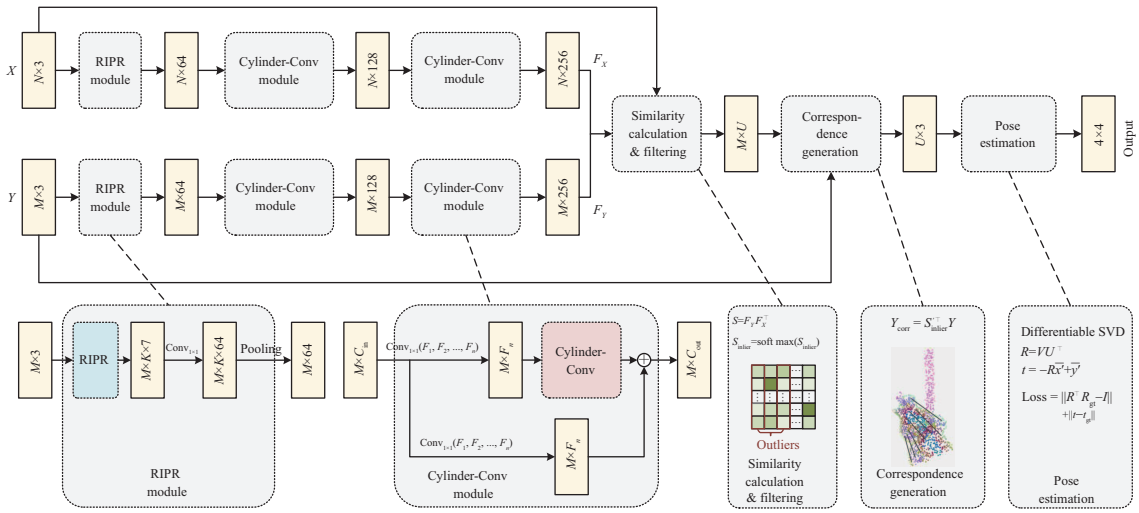

Figure 4 (Color online) The architecture of Cy-CNN. The inputs of our network are the source point cloud X and the target point cloud Y. The output is a 4×4 transformation matrix that transforms Y to X. C_{in} and C_{out} respectively denote the input channel and output channel of the Cylinder-Conv module. $\text{Conv}_{1\times1}(F_1, F_1, \ldots, F_n)$ stands for the 1×1 convolution applied sequentially with corresponding channels $F_i, i \in 1, \ldots, n$. F_X and F_Y stand for the normalized features of X and Y. The similarity matrix is displayed on a grid. The closer the value is to 1, the darker the square is. The columns marked in red are the outliers that will not appear in S_{inlier}. $\bar{x}'$ and $\bar{y}'$ are the mass centers of X_{inlier} and Y_{corr}. R_{gt} and t_{gt} are the ground truth transformation.

Most of the learning-based point cloud processing methods [16, 34, 35] need to select the local neighbor points (e.g., the ball query) to aggregate features. Compared with simply using the ball to define neighbor points, the cylinder-shaped voxels can better fit the surfaces of the local areas. The neighbor points of the cylinder-shaped voxels are spread from the surface of the objects. Combined with the multi-scaled structure of the cylinder-shaped voxels, Cylinder-Conv can capture hierarchical geometry information of the objects. Such kind of hierarchical geometry information makes the learned features of the different local areas distinguishable, which is conducive to correspondence generation. In summary, the properties of the proposed Cylinder-Conv are as follows: (1) Cylinder-Conv is invariant to rotation, owing to the axial symmetry of cylinder-shaped voxels and rotation-invariance of the normal direction. Neighbor points within the kernel remain consistent after rotation. Besides, the normal direction flipping issue does not influence our convolution operator. (2) Cylinder-Conv can exploit the surface geometry information of objects. That is because the normal represents the direction of the local area of the object. When Cylinder-Conv is applied to one of the points in the point cloud, the cylinder-shaped voxels can be fitted to the surface of the local area. Therefore, we can learn features that are distinguishable from different components of the object. (3) Cylinder-Conv is robust to noise, owing to the multi-scale cylinder-shaped voxels and the pooling operation in each cylinder-shaped voxel. Such kind of structure also allows a larger and more receptive field without increasing the number of network layers and contributes to capturing the hierarchical geometry information of the object.

3.3 Architecture

The architecture of our proposed Cy-CNN is illustrated in Figure 4. The network consists of two major parts: encoder and registration. The encoder part is composed of RIPR module, Cylinder-Conv module, and point-wise convolution. The source X (N points) and the target Y (M points) are fed into the RIPR module which consists of RIPR, a 1×1 convolution, and a pooling operation. The pooling operation aggregates the information of neighbor points. The output of the RIPR module is fed into a series of Cylinder-Conv modules to extract high-level features. Cylinder-Conv module contains a Cylinder-Conv operator and a series of 1×1 convolutions. To combine hierarchical features, we use residual connections in the Cylinder-Conv module, as shown in Figure 4. With the encoder, we generate point-wise rotation-invariant normalized features, which are $F_X \in \mathbb{R}^{N\times256}$ and $F_Y \in \mathbb{R}^{M\times256}$.

The registration part is composed of similarity calculation & filtering, correspondence generation, and pose estimation. The similarity matrix S is calculated with

$$S = F_Y {F_X}^{\mathrm{T}} \in \mathbb{R}^{M\times N}. \tag{8}$$

Let us focus on $f_{Xi} \in \mathbb{R}^{256}$ which is the i-th row of matrix F_X, and $f_{Yj} \in \mathbb{R}^{256}$ which is the j-th row of matrix F_Y. Since we have normalized the features, the closer their inner product of f_{Xi} and f_{Yj} is to 1, the more similar the two vectors are. f_{Yi} has different levels of similarity with each element in F_X. Then we focus on $s_i \in \mathbb{R}^M$ which is the i-th column of S, which indicates the similarity between f_{Xi} and each element in F_Y. Intuitively, if the maximum of s_i is less than a threshold, which means there is no point in Y that is similar enough to the i-th point in X, the i-th point of X is likely to be a point in the non-overlapping area. We set a threshold τ, and if the maximum of s_i is less than τ, the i-th point of X is regarded as an outlier. Specifically, we remove the i-th column of S and the i-th row of X, then we obtain $S_{\text{inlier}} \in \mathbb{R}^{M \times U}$ and $X_{\text{inlier}} \in \mathbb{R}^{U \times 3}$. The similarity matrix S_{inlier} can be interpreted as the correspondence probability between the points in the overlapped area. We use softmax to calculate the normalized similarity:

$$S'_{\text{inlier}} = \text{softmax}(S_{\text{inlier}}), \tag{9}$$

where $S'_{\text{inlier}} \in \mathbb{R}^{M \times U}$. With such normalized similarity matrix S'_{inlier}, we generate averaged correspondence points in Y for each point in X_{inlier}:

$$Y_{\text{corr}} = S'^{\text{T}}_{\text{inlier}} Y, \tag{10}$$

where $Y_{\text{corr}} \in \mathbb{R}^{U \times 3}$. The points in X_{inlier} and Y_{corr} correspond to each other. Given the corresponding pairs $\{x'_l \in X_{\text{inlier}}, y'_l \in Y_{\text{corr}}\}, l = 1, \ldots, U$, we perform a differentiable SVD step to estimate the relative pose R and t. For the differentiable SVD, we use the implement in Pytorch. We use the $L2$ distance in the Euclidean space as our loss function:

$$\text{Loss} = \left\| R^{\text{T}} R_{\text{gt}} - I \right\|^2 + \left\| t - t_{\text{gt}} \right\|^2, \tag{11}$$

where R and t are the predicted transformation, R_{gt} and t_{gt} are the ground truth transformation, $\|\cdot\|^2$ stands for $L2$ distance.

4 Experiments

We evaluate the proposed Cy-CNN on the ModelNet40 [54] dataset following the experimental setting of [15, 16], and we evaluate Cy-CNN on the Stanford Bunny [55], the 3DMatch dataset [56] for real-world point cloud registration. The ModelNet40 dataset consists of 40 object categories. We leave out 9 rotationally symmetrical categories such as cone. We randomly sample 1024 points from each model. Points are centered and rescaled to fit in the unit sphere. We compare our model to ICP [8], Go-ICP [21], FGR [12], TEASER++ [26], PointNetLK [15], and DCP-v2 [16]. For ICP and FGR, we use the implementation provided by Intel Open3D. For Go-ICP, TEASER++, PointNetLK, and DCP-v2, we use the authors' released codes. Note that FPFH [14] features are used in FGR and TEASER++.

The architecture of the proposed network is shown in Figure 4. We end up using three cylinder-shaped voxels in the Cylinder-Conv operator because our early experiments show that it is suitable for most of the datasets. The height and the radii of the three cylinder-shaped voxels on the ModelNet40 dataset and the Stanford Bunny dataset are 0.05 and [0.05, 0.1, 0.15] m, and that of the three cylinder-shaped voxels on the 3DMatch dataset are 0.15 and [0.15, 0.20, 0.45] m. The number of cylinder-shaped voxels, the radius, and the height, which control the resolution of the features, can be adjusted to fit other data. Two Cylinder-Conv modules are used in our architecture. The parameters of 1×1 convolutions in the first Cylinder-Conv module are [32, 64, 128], and the parameters in the second are [128, 128, 256]. The similarity threshold τ for filtering outliers is set as 0.95. We use Adam to optimize the network parameters. The initial learning rate is 0.001 and divided by 10 at epochs 50 and 150. The network is trained for a total of 200 epochs. All models in this paper are trained with a GeForce GTX 1080 Ti GPU, an Intel Xeon E5-2620 CPU, and 32G memory. Root mean squared error (RMSE) and mean absolute error (MAE) are used to measure the distance between the predicted values and the ground truth values following the evaluation metric of [15, 16]. These metrics should be close to zero if registration is perfect. All of the angle measurements in the results are in degrees.

4.1 Full dataset train and test

Our first experiment is to train Cy-CNN on the training set and test on the test set in ModelNet40, with no knowledge of the category label. The point clouds used during training and testing are different. As

Table 1 Test on unseen point clouds

Model	Initial misalignments $[0°, 15°]$				Initial misalignments $[15°, 30°]$				Initial misalignments $[30°, 45°]$			
	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)
ICP [8]	0.235162	0.165293	0.000028	0.000021	10.47001	9.591403	0.001196	0.001063	26.063487	24.930045	0.002928	0.002629
Go-ICP [21]	0.159222	0.123556	0.000012	0.000010	0.399097	0.317900	**0.000033**	**0.000029**	11.139241	8.858796	0.000930	0.000819
FGR [14]	0.125132	0.088014	0.000483	0.000395	0.150846	0.113263	0.000664	0.000578	1.467942	1.134116	0.001754	0.001521
TEASER++ [26]	0.103731	0.072216	0.000351	0.000326	0.132562	0.093437	0.000533	0.000482	1.134051	0.852316	0.001141	0.000842
PointNetLK [15]	**0.009819**	**0.006787**	**0.000005**	**0.000004**	**0.071381**	**0.062731**	0.000432	0.000361	2.291099	1.966454	0.015259	0.012407
DCP-v2 [16]	0.653601	0.549208	0.001278	0.001051	0.732468	0.632514	0.001300	0.001133	0.869550	0.767109	0.001282	0.001136
Cy-CNN (ours)	0.286316	0.240256	0.000036	0.000028	0.295949	0.254710	0.000037	0.000029	**0.321398**	**0.284175**	**0.000037**	**0.000029**

in previous studies [15, 16], we use random transformation with rotation angles in [0°, 45°] along each axis and translation in [−0.5, 0.5] m during training. The target point cloud Y is a rigid transformation of the source point cloud X, and we add additive Gaussian noise ($\mu = 0, \sigma = 0.02$) to X and Y during training.

To evaluate the performance of our method on clean data during testing, we did not add Gaussian noise in our first experiment. In Table 1, we evaluate the performance of our method and several state-of-the-art registration methods with different initial misalignments, where R indicates rotation, t indicates translation, and the bold entries indicate the best performance. ICP performs well in small initial misalignments but converges to a local minimum in large initial misalignments. FGR and TEASER++ perform well even at large initialization misalignments. Cy-CNN is comparable to other methods in the case of small initial misalignments and Cy-CNN outperforms other methods under all the performance metrics in the case of initial misalignments [30°, 45°]. This verifies that our network learns rotation-invariant features that contribute to correspondence generation even with large misalignments. The average end-to-end inference time of our network is 25.2 ms.

4.2 Robustness to noise

We explore the robustness of Cy-CNN against noise, which always exists in real-world point clouds. We add Gaussian noises of different certain standard deviations to the source and target point clouds. This experiment is more challenging than that without noise due to non one-to-one correspondence. We use the model from Subsection 4.1 trained on the full Dataset. The rotation and translation settings are the same with Subsection 4.1. Table 2 shows the results with Gaussian noise ($\mu = 0, \sigma = 0.01$). Table 3 shows the results with Gaussian noise ($\mu = 0, \sigma = 0.02$). Compared with the results in Table 1, we find that FGR and TEASER++ are sensitive to noise. The performance of TEASER++ is better than FGR but also affected under Gaussian noise ($\mu = 0, \sigma = 0.02$). We suppose that the manually designed features used in FGR and TEASER++ are unstable under noise. ICP, Go-ICP, and Cy-CNN remain robust to noise. This validates that our use of the Cylinder-Conv operator captures hierarchical geometry information which can alleviate noise interference. The first three columns of Figure 5 show registration results and feature visualizations of Cy-CNN on some 3D shapes with Gaussian noise ($\mu = 0, \sigma = 0.01$). In order to better show the rotation-invariant features, point clouds are colored with low dimensional embeddings of the local feature using t-SNE [18]. We can see that the same part of the point clouds in different poses has similar features, which contributes to calculating the correspondence.

To further explore the generalizability of Cy-CNN to unseen noise levels, we test our model (trained in Subsection 4.1, $\sigma = 0.02$) under different noise levels ($\sigma = 0/0.01/0.02/0.03$). We compare our method with three local feature-based methods. Figure 6 shows the results with [30°, 45°] initial misalignments. We can see that the performance of FGR, TEASER++, and DCP-v2 is heavily affected when the noise level is larger than $\sigma = 0.01$. The proposed method shows stronger noise resilience, because the multi-scale cylinder-shaped voxels and the pooling operation in each cylinder-shaped voxel make the rotation-invariant feature more stable.

4.3 Robustness to initial misalignments

We also explore the robustness of Cy-CNN against initial misalignments. In this subsection, we use the model from Subsection 4.1 trained on full dataset with rotation angles [0°, 45°] along each axis and translation [−0.5, 0.5] m. We test with initial angles [0°, 90°) along each axis and translation [−0.5, 0.5] m. Figure 7 shows the experiment results. Cy-CNN achieves better performance than other point cloud registration methods with large initial misalignments. This also indicates that our model has the

Table 2 Test on objects with Gaussian noise ($\mu = 0, \sigma = 0.01$)

Model	Initial misalignments $[0°, 15°]$				Initial misalignments $[15°, 30°]$				Initial misalignments $[30°, 45°]$			
	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)
ICP [8]	0.363799	0.269937	0.000423	0.000367	10.964116	10.095423	0.001403	0.001241	26.603537	25.497968	0.003047	0.002733
Go-ICP [21]	**0.156194**	**0.123945**	**0.000409**	**0.000354**	**0.354194**	**0.277793**	**0.000430**	**0.000374**	11.034553	8.870747	**0.001267**	**0.001115**
FGR [14]	1.661972	1.361509	0.007024	0.005993	2.403952	1.969845	0.009068	0.007899	8.577731	7.430874	0.016649	0.014575
TEASER++ [26]	1.131462	0.926672	0.003571	0.002963	1.829841	1.427276	0.004742	0.003935	4.364125	3.588270	0.011548	0.009436
PointNetLK [15]	0.633874	0.539278	0.006550	0.005648	0.729381	0.631044	0.007245	0.006251	2.983881	2.556072	0.020383	0.016900
DCP-v2 [16]	5.588781	4.508863	0.002797	0.002299	5.742204	4.864097	0.002849	0.002485	6.982967	6.057312	0.002821	0.002498
Cy-CNN (ours)	0.409451	0.335424	0.002211	0.001826	0.430314	0.359782	0.002185	0.001894	**0.482591**	**0.426159**	**0.002212**	**0.001940**

Table 3 Test on objects with Gaussian noise ($\mu = 0, \sigma = 0.02$)

Model	Initial misalignments $[0°, 15°]$				Initial misalignments $[15°, 30°]$				Initial misalignments $[30°, 45°]$			
	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)
ICP [8]	0.891415	0.696285	0.000825	0.000714	11.856294	10.981529	0.001694	0.001495	27.278722	26.185233	0.003239	0.002893
Go-ICP [21]	**0.378321**	**0.311603**	**0.000807**	**0.000698**	**0.643440**	**0.529441**	**0.000826**	**0.000717**	11.204717	9.037265	**0.001602**	**0.001405**
FGR [14]	9.886202	7.731857	0.023819	0.020084	12.825663	10.741107	0.027244	0.023707	24.364371	21.233212	0.038089	0.033443
TEASER++ [26]	4.260962	3.451379	0.022981	0.018845	10.774972	8.835481	0.026146	0.023513	18.762249	15.241326	0.031860	0.026763
PointNetLK [15]	1.222220	1.035671	0.012446	0.010756	1.393394	1.202675	0.013825	0.011909	3.891121	3.349679	0.027494	0.022897
DCP-v2 [16]	13.600126	11.213411	0.005121	0.004233	14.171057	12.027894	0.005144	0.004493	15.310481	13.330386	0.005114	0.004521
Cy-CNN (ours)	0.714914	0.577209	0.002876	0.002330	0.743338	0.629433	0.002870	0.002491	**0.858950**	**0.760266**	**0.002859**	**0.002513**

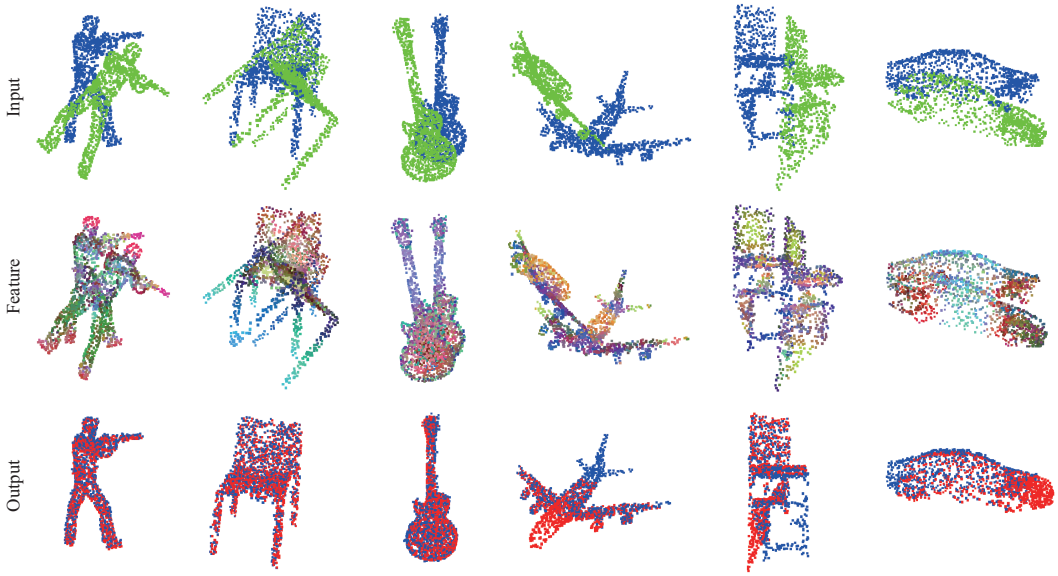

Figure 5 (Color online) Registration results and feature visualizations of Cy-CNN. The first three columns are noisy point clouds. The last three columns are partial point clouds with noises. Top: the source (blue) and target (green) point clouds. Middle: t-SNE [18] visualization of features. It clearly shows that the features of Cylinder-Conv are rotation invariant. Bottom: the registration results (red) of our method.

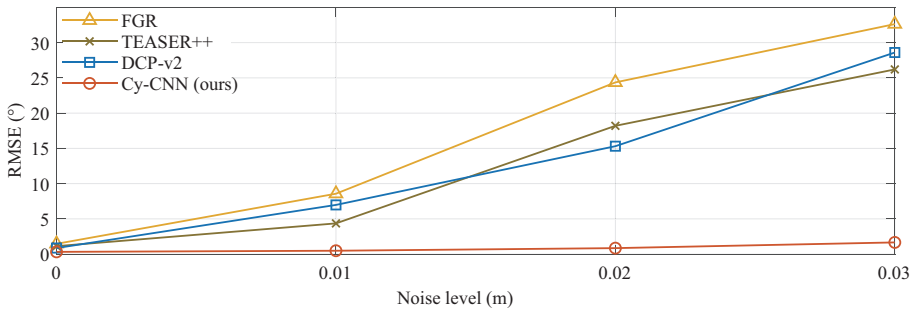

Figure 6 (Color online) Results with different noise levels. The abscissa is the mean of Gaussian noises, and the ordinate is the rotation RMSE.

generalizability of unseen initial angles, supporting our use of symmetrical Cylinder-shaped voxels and RIPR.

In our experiments, ICP fails because of lacking a good initial guess. However, ICP can be used as

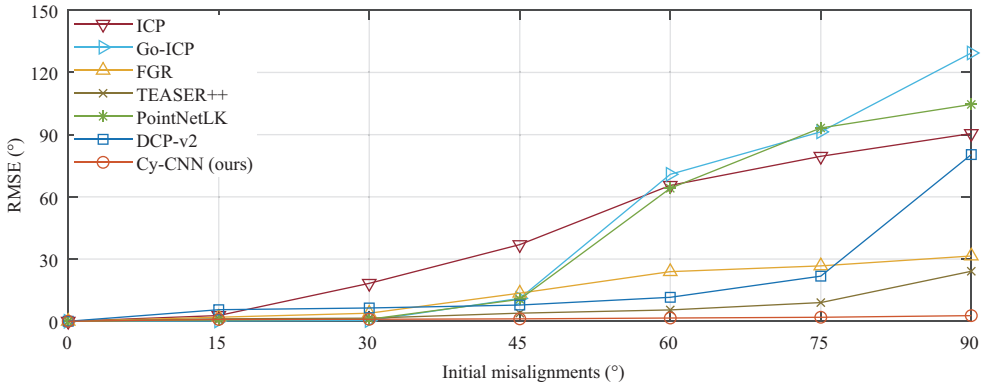

Figure 7 (Color online) Results with different initial misalignments. The proposed model achieves reliable registration even with large and unseen initial misalignments.

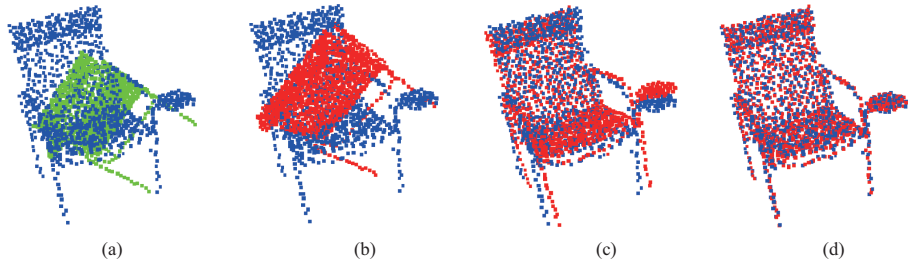

Figure 8 (Color online) The demonstration of the two-step registration procedure. (a) The source (blue) and target (green) point cloud. The initial angles are generated in [0°, 90°) along each axis. (b) The result (red) of ICP with large initial misalignments and the value of RMSE(R) is 87.533°. (c) The initial transformation (red) is provided by Cylinder-Conv and the value of RMSE(R) is 2.412°. (d) The result (red) of ICP initialized with Cy-CNN. With an initial transformation provided by Cy-CNN, ICP converges to the global optimum. The value of RMSE(R) is 0.023°.

Table 4 Test on unseen categories

Model	Initial misalignments [0°, 15°]				Initial misalignments [15°, 30°]				Initial misalignments [30°, 45°]			
	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)
ICP [8]	0.229181	0.166492	0.000026	0.000022	12.849926	11.985556	0.001507	0.001348	30.984059	30.099091	0.003650	0.003273
Go-ICP [21]	0.115038	0.088632	0.000009	0.000008	0.566037	0.418517	**0.000050**	**0.000044**	13.072995	10.563528	0.001154	0.001014
FGR [14]	0.013766	0.011688	0.000146	0.000123	**0.020753**	**0.017284**	0.000152	0.000134	1.226873	1.092993	0.001102	0.000997
TEASER++ [26]	0.092356	0.059531	0.000313	0.000276	0.112732	0.082953	0.000462	0.000401	1.012623	0.745820	0.001083	0.000877
PointNetLK [15]	**0.001889**	**0.001660**	**0.000011**	**0.000009**	0.110262	0.096065	0.000945	0.000776	5.920093	4.981530	0.0373815	0.031065
DCP-v2 [16]	2.118484	1.583357	0.018179	0.013319	3.614282	2.696888	0.027249	0.019143	6.400655	4.666704	0.044429	0.029158
Cy-CNN (ours)	0.307489	0.261533	0.000041	0.000032	0.322620	0.281512	0.000042	0.000034	**0.359220**	**0.318854**	**0.000042**	**0.000034**

a local registration algorithm by initializing with a rigid transformation from the proposed Cy-CNN. Figure 8 shows an example of this two-step procedure. Although ICP fails in the condition of large initial misalignments, with an initialization provided by Cy-CNN, it converges to the global optimum. This experiment shows that, in some scenes, ICP can be an effective way to fine-tune the output of Cy-CNN.

4.4 Category split

In order to test the generalizability on unseen categories, we train our model on 20 categories, then test on the held-out categories which have not been seen during training. Other methods are also tested on the held-out categories. Table 4 shows the results of this experiment. Compared with Table 1, we find ICP, Go-ICP, FGR, and TEASER++ are not sensitive to categories. The learning-based method PointNetLK performs well when the initial misalignments are small ([0°, 15°]) on unseen categories, but it degenerates when the rotation becomes large ([30°, 45°]). Another learning-based method DCP-v2 shows the sensitivity to unseen categories, especially at large initialization misalignments. Cy-CNN exhibits similar results to Table 1. Because our network learns rotation-invariant features that are unrelated to categories, Cy-CNN has the ability to generalize for registration on categories that are unseen during training.

Table 5 Test on partially visible data with noise

Model	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)
ICP [8]	27.469155	26.230035	0.054282	0.048805
Go-ICP [21]	11.641246	9.321784	0.026926	0.021493
FGR [14]	8.712343	7.664377	0.020346	0.018321
TEASER++ [26]	6.931573	5.621506	0.019642	0.015930
PointNetLK [15]	9.234156	8.124132	0.037327	0.033414
DCP-v2 [16]	7.121321	6.234231	0.034124	0.031571
Cy-CNN (ours)	**2.762431**	**2.251381**	**0.016126**	**0.014541**

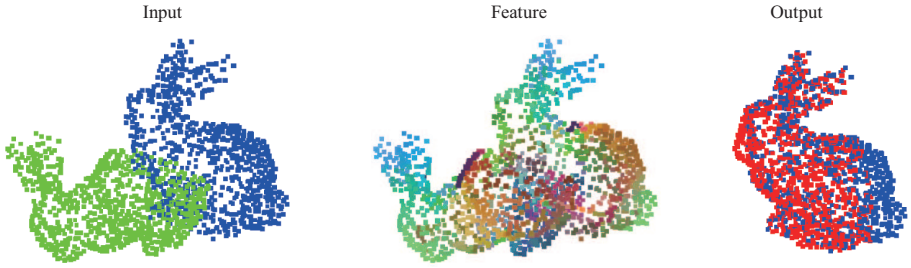

Figure 9 (Color online) The registration results of partially visible data with noise on the Stanford Bunny dataset.

4.5 Partially visible data

We evaluate Cy-CNN on the common registration scenario of aligning partially visible data, in which the ranges of the two point clouds do not completely overlap. Depending on the data collection method, point cloud data is usually partially visible in the real world. For example, the RGB-D camera only collects the points visible to the camera. We simulate the partially visible data on ModelNet40 as follows. For each point cloud, we randomly place a point in space and compute the 768 nearest neighbors. This operation can be thought of as placing a sensor in a location and sampling data of the 3D shapes. Next, we add Gaussian noise ($\mu = 0, \sigma = 0.01$) to the partial point cloud. The random transformation is similar to Subsection 4.2 with rotation angles [0°, 45°] along each axis and translation [−0.5, 0.5] m during training. We train PointNetLK [15], DCP-v2 [16] and our model on partially visible data sampled from ModelNet40 dataset [54]. We evaluate the models with initial misalignments [30°, 45°] and translation [−0.5, 0.5] m. The results are shown in Table 5. The performance of ICP, Go-ICP, FGR, and TEASER++ is similar to that in Subsection 4.2. Our method outperforms other methods in this experiment. This indicates that our method can resist partial deletion of point clouds to a certain extent. The last three columns of Figure 5 show the example results of partially visible data on ModelNet40.

4.6 Real-world data

To evaluate the performance of our method on real-world point clouds, we first test our model on the Stanford Bunny dataset [55]. Because the dataset only has 10 real scans, we fine-tune the model used in Table 5 for 30 epochs with a learning rate of 0.0001. For each scan, we generate 100 training examples by randomly transforming and sampling the scan as we do on ModelNet40. Table 6 shows the quantitative registration results, and Figure 9 shows the qualitative registration results and feature visualization on the Stanford Bunny dataset. We can see that even if the target point cloud and the transformed point cloud are not completely overlapped, our method can align the two point clouds. This result can be fine-turned with ICP as mentioned in Subsection 4.3.

To evaluate the performance of our method on 3D scene registration, we perform experiments on the 3DMatch dataset [56], which consists of 62 real-world indoor scenes. Each scene contains some partially overlapped fragments and has the ground truth transformations that can be used for evaluation. We follow the official specifications and split the data into 54 scenes for training and 8 for testing. As a preprocessing step, we apply voxel-grid downsampling and randomly sample 4000 points for each point cloud fragment. During training and testing, we add additional Gaussian noise ($\mu = 0, \sigma = 0.01$), rotation angles [0°, 45°], and 70% partial occlusions. Table 6 shows the quantitative registration results, and Figure 10 shows the qualitative registration results and feature visualization on the 3DMatch dataset. We can see that the

Table 6 Ablation experiments of parameter setting

Dataset	Height (m)	Radius (m)	τ	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)
Stanford Bunny	0.03	[0.03, 0.06, 0.09]	0.95	3.129834	2.563814	0.017112	0.015321
Stanford Bunny	0.05	[0.05, 0.10, 0.15]	0.95	**2.845304**	**2.328922**	**0.016610**	**0.014977**
Stanford Bunny	0.10	[0.10, 0.20, 0.30]	0.95	3.319361	2.716920	0.017364	0.015747
Stanford Bunny	0.05	[0.05, 0.10, 0.15]	0.99	2.883757	2.352211	0.016932	0.015121
Stanford Bunny	0.05	[0.05, 0.10, 0.15]	0.90	2.930663	2.398790	0.017013	0.015051
Stanford Bunny	0.05	[0.05, 0.10, 0.15]	0.80	3.077196	2.518729	0.017121	0.015342
Stanford Bunny	0.05	[0.05, 0.10, 0.15]	0.70	3.231055	2.644665	0.017236	0.015551
Stanford Bunny	0.05	[0.05, 0.10, 0.15]	0.60	3.263367	2.671112	0.017283	0.015628
3DMatch	0.10	[0.10, 0.20, 0.30]	0.95	4.182783	3.483261	0.017732	0.015984
3DMatch	0.15	[0.15, 0.30, 0.45]	0.95	**3.764543**	**3.133166**	**0.017413**	**0.015603**
3DMatch	0.20	[0.20, 0.40, 0.60]	0.95	4.015261	3.341835	0.017647	0.015861
3DMatch	0.15	[0.15, 0.30, 0.45]	0.99	3.879905	3.237347	0.017585	0.015819
3DMatch	0.15	[0.15, 0.30, 0.45]	0.90	3.893215	3.245125	0.017613	0.015833
3DMatch	0.15	[0.15, 0.30, 0.45]	0.80	4.087876	3.407381	0.017721	0.015912
3DMatch	0.15	[0.15, 0.30, 0.45]	0.70	4.272261	3.543676	0.017829	0.016236
3DMatch	0.15	[0.15, 0.30, 0.45]	0.60	4.293256	3.579113	0.017831	0.016358

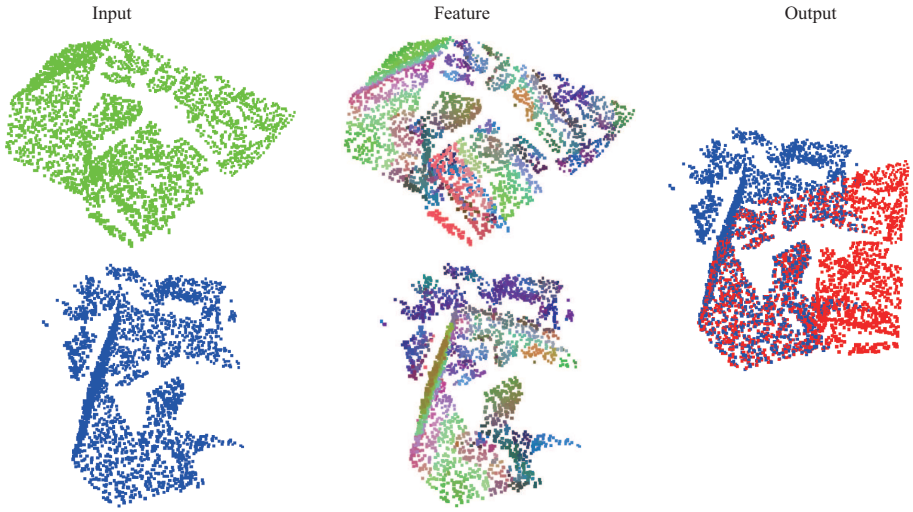

Figure 10 (Color online) The registration results of 3D scenes on the 3DMatch dataset.

features of the same geometric structure of the two point cloud fragments are similar (e.g., the chair), which is conducive to correspondence generation. And the predicted transformation makes the two point clouds align well. This verifies that Cy-CNN is able to handle 3D scene reconstruction applications.

4.7 Ablation study

Effectiveness of different components. In the proposed Cy-CNN architecture, RIPR and Cylinder-Conv are two components that can be replaced. To study the effectiveness of RIPR in Subsection 3.1, we replace RIPR with PPF or coordinates in LRF. That is because only using PPF or LRF respectively is also a kind of rotation-invariant representation. To study the effect of the symmetry of cylinder-shaped voxels in Cylinder-Conv, we replace Cylinder-Conv with a uniform grid voxel based convolution [36] which is aligned with the world coordinate system without considering local surface direction. Both the direction and the grid shape of the voxel [36] cause it to be not rotation-invariant. To study the ability of Cylinder-Conv to capture hierarchical geometric information of the surface of objects, we replace Cylinder-Conv with DGCNN [38] which learns local features via constructing the k-NN graph. We train the models with grid voxel based convolution [36] using and not using RIPR, the models with DGCNN using and not using RIPR, and the models with Cylinder-Conv not using RIPR as the experiment setting in Subsection 4.1. The model with Cylinder-Conv using RIPR is the model which has been trained in

Table 7 Ablation experiments on ModelNet40 dataset

Model	PPF	LRF	RMSE(R)	MAE(R)	RMSE(t)	MAE(t)
Hua et al. [36]	✔	×	3.938751	3.269163	0.004578	0.003708
Hua et al. [36]	×	✔	3.817526	3.034327	0.004432	0.003661
Hua et al. [36]	✔	✔	2.393821	1.938614	0.004312	0.003536
DGCNN [38]	✔	×	3.067987	2.485069	0.004815	0.003903
DGCNN [38]	×	✔	2.617781	2.113263	0.004641	0.003732
DGCNN [38]	✔	✔	1.399740	1.147787	0.004252	0.003487
Cylinder-Conv (ours)	✔	×	0.810549	0.655653	0.003681	0.002982
Cylinder-Conv (ours)	×	✔	0.732503	0.591325	0.003242	0.002413
Cylinder-Conv (ours)	✔	✔	**0.482591**	**0.426159**	**0.002212**	**0.001940**

Subsection 4.1. We test with initial misalignments [30°, 45°] and translation [−0.5, 0.5] m. We also add Gaussian noises ($\mu = 0, \sigma = 0.01$) to the point clouds.

Table 7 shows the experiment results. The models using PPF or LRF are marked with a "✔", and the models without using PPF or LRF are marked with a "✘". When a model simultaneously uses PPF and LRF, it can be seen as using RIPR in Subsection 3.1. By comparing the models only using PPF (lines 1, 4, 7) with the models only using LRF (lines 2, 5, 8), we find that the performance of LRF is slightly better than that of PPF. We infer that is because using the coordinates in LRF as input is beneficial to feature learning, compared to only using PPF as input. By comparing the models using RIPR with the models not using RIPR, we find that the performance of the combination of PPF and LRF (lines 3, 6, 9) is better than only using either one of them (lines 1, 2, 4, 5, 7, 8). That is because RIPR provides richer information combining surface structure and relative positions in the local region than only using PPF or LRF as a representation. By comparing the models using grid voxel based convolution [36] (lines 1–3) with the models using Cylinder-Conv (lines 7–9), Tabel 7 shows that the models using Cylinder-Conv perform better. Neighbor points in grid voxels will change after rotation while neighbor points in cylinder-shaped voxel maintain invariance. By comparing models using DGCNN (lines 4–6) with Cylinder-Conv (lines 7–9), we find that Cylinder-Conv outperforms DGCNN. Cylinder-Conv can better exploit the surface information of objects owing to the usage of the normal direction as mentioned in Subsection 3.2. Moreover, the multi-scale cylinder-shaped voxels and the pooling operation in each cylinder-shaped voxel make the neighbor relationship more robust to jitter than the k-NN graph used in DGCNN. Using the proposed Cylinder-Conv, we learn robust and rotation-invariant features from different components of the object, which contributes to the registration results.

Parameter setting. To study the influence of the size of cylinder-shaped voxels and the similarity threshold τ on the performance of our method, we set different parameters to train and test our model on the Stanford Bunny dataset and the 3DMatch dataset. These two datasets introduced in Subsection 4.6 are real-world datasets, which can verify the value of our method on real-world applications. For the Stanford Bunny dataset, we retrain our model on Subsection 4.1 using different parameters and fine tune it on the Stanford Bunny dataset. For the 3DMatch dataset, we directly train our model on the 3DMatch dataset using different parameters. Other experimental settings are the same with Subsection 4.6. Table 6 shows the experiment results. We observe that the proper sizes of the cylinder-shaped voxels on the Stanford Bunny dataset (object-level point cloud) are 0.05 m of height and [0.05, 0.10, 0.15] m of radii, the proper sizes of the cylinder-shaped voxels on the 3DMatch dataset (3D scene point cloud) are 0.15 m of height and [0.15, 0.30, 0.45] m of radii. We infer that when the size is too small the captured geometric information is not sufficient for correspondence generation, and when the size is too large, too much contextual information is encoded, thus weakening the distinctiveness of features. For the choice of similarity threshold τ, we adjust its value in [0.6, 0.99] and observe the results. We find that if τ is too large (tending to 1), too few correspondences are generated which impacts the final pose estimation; if τ is too small (tending to 0), too many outliers are in the correspondences which also impacts the final pose estimation. We find that when τ is less than 0.7, the results change very slightly. We think that is because nearly no point is filtered as an outlier when τ is less than 0.7, which means the value of τ gradually makes no sense when τ is tending to be small. So we end up reporting the results of $\tau \in [0.6, 0.99]$. As shown in Table 6, we find that 0.95 is a proper choice, and the jitter of τ only has a slight influence on the results.

Table 8 Inference time (s)

Number of points	ICP	Go-ICP	FGR	TEASER++	PointNetLK	DCP-v2	Cy-CNN (ours)
512	0.008	14.713	0.026	0.017	0.045	0.007	0.012
1024	0.019	15.531	0.083	0.046	0.056	0.009	0.025
2048	0.032	15.942	0.143	0.093	0.148	0.021	0.051

4.8 Computational efficiency

We analyze the inference time of various methods with the computing platform mentioned before. The averaged inference time using different numbers of points is shown in Table 8, which is averaged over the entire test set. Note that ICP, Go-ICP, FGR, and TEASER++ are executed on the CPU and the remaining algorithms on the GPU. Our algorithm is significantly faster than Go-ICP, FGR, TEASER++, and PointNetLK, but it is slower than ICP and DCP-v2.

5 Conclusion

In this paper, we propose Cy-CNN, a novel cylinder convolution based rotation-invariant neural network for point cloud registration. The core of Cy-CNN is to learn point-wise rotation-invariant features for calculating the correspondence between two point clouds, where RIPR and Cylinder-Conv ensure the property of rotation-invariance in the input stage and the learning stage, respectively. Extensive experiments show that our end-to-end trainable model is reliable to predict high-quality registration results in a single pass, even with large initial misalignments and noise. Our method also shows the ability to generalize to unseen categories and the compatibility with ICP for more accurate estimation.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 62173228, 61873165).

References

1 Liu B S, Chen X M, Han Y H, et al. Accelerating DNN-based 3D point cloud processing for mobile computing. Sci China Inf Sci, 2019, 62: 212102
2 Peng H T, Zhou B, Yin L Y, et al. Semantic part segmentation of single-view point cloud. Sci China Inf Sci, 2020, 63: 224101
3 Zhao H, Zhu C Y, Xu X, et al. Learning practically feasible policies for online 3D bin packing. Sci China Inf Sci, 2022, 65: 112105
4 Han L, Gu S, Zhong D, et al. Real-time globally consistent dense 3D reconstruction with online texturing. IEEE Trans Pattern Anal Mach Intell, 2022, 44: 1519–1533
5 Fang S, Li H, Yang M. LiDAR SLAM based multivehicle cooperative localization using iterated split CIF. IEEE Trans Intell Transp Syst, 2022. doi: 10.1109/TITS.2022.3174479
6 Holz D, Ichim A E, Tombari F, et al. Registration with the point cloud library: a modular framework for aligning in 3-D. IEEE Robot Automat Mag, 2015, 22: 110–124
7 Chen Y, Medioni G. Object modelling by registration of multiple range images. Image Vision Computing, 1992, 10: 145–155
8 Besl P J, McKay N D. Method for registration of 3-D shapes. In: Proceedings of Sensor Fusion IV: Control Paradigms and Data Structures, 1992. 586–606
9 Segal A, Haehnel D, Thrun S. Generalized-ICP. In: Proceedings of Robotics: Science and Systems, Seattle, 2009. 435
10 Agamennoni G, Fontana S, Siegwart R Y, et al. Point clouds registration with probabilistic data association. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016. 4092–4098
11 Magnusson M, Lilienthal A, Duckett T. Scan registration for autonomous mining vehicles using 3D-NDT. J Field Robotics, 2007, 24: 803–827
12 Zhou Q Y, Park J, Koltun V. Fast global registration. In: Proceedings of European Conference on Computer Vision. Berlin: Springer, 2016. 766–782
13 Ma J, Jiang X, Fan A, et al. Image matching from handcrafted to deep features: a survey. Int J Comput Vis, 2021, 129: 23–79
14 Rusu R B, Blodow N, Beetz M. Fast point feature histograms (FPFH) for 3D registration. In: Proceedings of IEEE International Conference on Robotics and Automation, 2009. 3212–3217
15 Aoki Y, Goforth H, Srivatsan R A, et al. PointNetLK: robust & efficient point cloud registration using pointnet. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019. 7163–7172
16 Wang Y, Solomon J M. Deep closest point: learning representations for point cloud registration. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019. 3523–3532
17 Yew Z J, Lee G H. RPM-Net: robust point matching using learned features. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 11824–11833
18 van der Maaten L, Hinton G. Visualizing data using t-SNE. J Mach Learning Res, 2008, 9: 2579–2605
19 Censi A. An ICP variant using a point-to-line metric. In: Proceedings of IEEE International Conference on Robotics and Automation, 2008. 19–25
20 Pomerleau F, Colas F, Siegwart R, et al. Comparing ICP variants on real-world data sets. Auton Robot, 2013, 34: 133–148
21 Yang J, Li H, Campbell D, et al. Go-ICP: a globally optimal solution to 3D ICP point-set registration. IEEE Trans Pattern Anal Mach Intell, 2015, 38: 2241–2254
22 Rusu R B, Blodow N, Marton Z C, et al. Aligning point cloud views using persistent feature histograms. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. 3384–3391

23 Yang J, Quan S, Wang P, et al. Evaluating local geometric feature representations for 3D rigid data matching. IEEE Trans Image Process, 2019, 29: 2522–2535
24 Ma J, Wu J, Zhao J, et al. Nonrigid point set registration with robust transformation learning under manifold regularization. IEEE Trans Neural Netw Learn Syst, 2018, 30: 3584–3597
25 Ma J, Zhao J, Jiang J, et al. Locality preserving matching. Int J Comput Vis, 2019, 127: 512–531
26 Yang H, Shi J, Carlone L. TEASER: fast and certifiable point cloud registration. IEEE Trans Robot, 2020, 37: 314–333
27 Myronenko A, Song X B. Point set registration: coherent point drift. IEEE Trans Pattern Anal Mach Intell, 2010, 32: 2262–2275
28 Jian B, Vemuri B C. Robust point set registration using Gaussian mixture models. IEEE Trans Pattern Anal Mach Intell, 2011, 33: 1633–1645
29 Li L, Yang M, Wang C, et al. Robust point set registration using signature quadratic form distance. IEEE Trans Cybern, 2018, 50: 2097–2109
30 Su H, Maji S, Kalogerakis E, et al. Multi-view convolutional neural networks for 3D shape recognition. In: Proceedings of the IEEE International Conference on Computer Vision, 2015. 945–953
31 Wu B, Wan A, Yue X, et al. SqueezeSeg: convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D lidar point cloud. In: Proceedings of IEEE International Conference on Robotics and Automation, 2018. 1887–1893
32 Tchapmi L, Choy C, Armeni I, et al. SEGCloud: semantic segmentation of 3D point clouds. In: Proceedings of International Conference on 3D Vision, 2017. 537–547
33 Maturana D, Scherer S. VoxNet: a 3D convolutional neural network for real-time object recognition. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2015. 922–928
34 Qi C R, Su H, Mo K, et al. PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 652–660
35 Qi C R, Yi L, Su H, et al. PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 5099–5108
36 Hua B S, Tran M K, Yeung S K. Pointwise convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 984–993
37 Lin L Q, Huang P D, Xue F Y, et al. Hausdorff point convolution with geometric priors. Sci China Inf Sci, 2021, 64: 210105
38 Wang Y, Sun Y, Liu Z, et al. Dynamic graph CNN for learning on point clouds. ACM Trans Graph, 2019, 38: 1–12
39 Liang Z, Yang M, Deng L, et al. Hierarchical depthwise graph convolutional neural network for 3D semantic segmentation of point clouds. In: Proceedings of International Conference on Robotics and Automation, 2019. 8152–8158
40 Valsesia D, Fracastoro G, Magli E. Learning localized representations of point clouds with graph-convolutional generative adversarial networks. IEEE Trans Multimedia, 2020, 23: 402–414
41 Cao W M, Zheng C T, Yan Z Y, et al. Geometric deep learning: progress, applications and challenges. Sci China Inf Sci, 2022, 65: 126101
42 Poulenard A, Rakotosaona M J, Ponty Y, et al. Effective rotation-invariant point CNN with spherical harmonics kernels. In: Proceedings of International Conference on 3D Vision, 2019. 47–56
43 Li X, Li R, Chen G, et al. A rotation-invariant framework for deep point cloud analysis. IEEE Trans Visual Comput Graph, 2021. doi: 10.1109/TVCG.2021.3092570
44 Chen C, Li G, Xu R, et al. ClusterNet: deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 4994–5002
45 You Y, Lou Y, Liu Q, et al. Pointwise rotation-invariant network with adaptive sampling and 3D spherical voxel convolution. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2020. 12717–12724
46 Deng H, Birdal T, Ilic S. PPFNet: global context aware local features for robust 3D point matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 195–205
47 Deng H, Birdal T, Ilic S. PPF-FoldNet: unsupervised learning of rotation invariant 3D local descriptors. In: Proceedings of the European Conference on Computer Vision (ECCV), 2018. 602–618
48 Ao S, Hu Q, Yang B, et al. SpinNet: learning a general surface descriptor for 3D point cloud registration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021. 11753–11762
49 Lu W, Wan G, Zhou Y, et al. DeepVCP: an end-to-end deep neural network for point cloud registration. In: Proceedings of the IEEE International Conference on Computer Vision, 2019. 12–21
50 Khazari A E, Que Y, Sung T L, et al. Deep global features for point cloud alignment. Sensors, 2020, 20: 4032
51 Gojcic Z, Zhou C, Wegner J D, et al. Learning multiview 3D point cloud registration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 1759–1769
52 Drost B, Ulrich M, Navab N, et al. Model globally, match locally: efficient and robust 3D object recognition. In: Proceedings of 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010. 998–1005
53 Huang H, Li D, Zhang H, et al. Consolidation of unorganized point clouds for surface reconstruction. ACM Trans Graph, 2009, 28: 1–7
54 Wu Z, Song S, Khosla A, et al. 3D ShapeNets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. 1912–1920
55 Turk G, Levoy M. The Stanford 3D Scanning Repository. Stanford University Computer Graphics Laboratory, 2005. http://graphics.stanford.edu/data/3Dscanrep
56 Zeng A, Song S, Nießner M, et al. 3DMatch: learning local geometric descriptors from RGB-D reconstructions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 1802–1811