# RoReg: Pairwise Point Cloud Registration with Oriented Descriptors and Local Rotations Supplementary Material

Haiping Wang*, Yuan Liu*, Qingyong Hu, Bing Wang, Jianguo Chen, Zhen Dong†, Yulan Guo, Wenping Wang, and Bisheng Yang†

## APPENDIX A
## PROOF OF PROPERTIES

In this section, we provide the proof of *rotation-equivariance* and *rotation-invariance* as discussed in Sec. 3.2.

**Property 1**. *If we rotate $N_p$ with a rotation $\alpha \in G$ and extract a $f_0'$ on the rotated version, then the $f_0$ and $f_0'$ are associated by $f_0' = P_\alpha \circ f_0$, where $P_\alpha$ means a row-wise permutation.*

**Proof**: The initial icosahedral feature map $f_0$ on $N_p$ is written as

$$f_0(\beta) = \varphi(T_\beta \circ N_p), \tag{1}$$

where $\beta \in G$. Using a rotation $\alpha \in G$ in the icosahedral group to rotate the input point set, the corresponding icosahedral feature is

$$f_0'(\beta) = \varphi(T_\beta \circ T_\alpha \circ N_p) \tag{2}$$

Due to the closure property of a group and the composition $\beta\alpha \in G$, Eq. 2 can be expressed by

$$f_0'(\beta) = \varphi(T_{\beta\alpha} \circ N_p) = f_0(\beta\alpha) \tag{3}$$

Since $f_0$ is represented by a $60 \times n_0$ matrix and different row indices represent different rotations $\beta \in G$, Eq. 3 means that the row corresponding to an arbitrary rotation $\beta$ of $f_0'$ will be the same as the row corresponding to the rotation $\beta\alpha$ of $f_0$. Thus, $f_0$ and $f_0'$ are associated with a row-wise permutation $P_\alpha$

$$f_0' = P_\alpha \circ f_0. \tag{4}$$

**Property 2**. *If $f_l$ has the rotation-equivariance of Property 1, then the feature $f_{l+1}$ extracted by Eq. 2 of the main paper also holds the rotation-equivariance. The subsequent batch normalization and ReLU layers also preserve such rotation-equivariance.*

**Proof**: If $f_l$ holds Eq. 3 with $f_l'(\beta) = f_l(\beta\alpha)$, we apply the group convolution proposed in main paper on $f_l$ to get $f_{l+1}$

$$\begin{aligned}\left[f_{l+1}'(\beta)\right]_j &= \sum_i^{13} w_{j,i}^T f_l'(h_i\beta) + b_j \\ &= \sum_i^{13} w_{j,i}^T f_l(h_i\beta\alpha) + b_j \\ &= [f_{l+1}(\beta\alpha)]_j.\end{aligned} \tag{5}$$

The first equation is the definition of the group convolution. The second equation holds because $f_l'(\beta) = f_l(\beta\alpha)$. The third equation also uses the definition of the group convolution which treats $\beta\alpha$ as a whole element. Since $f_{l+1}'(\beta) = f_{l+1}(\beta\alpha)$ holds for any $\beta \in G$, $f_{l+1}' = P_\alpha \circ f_{l+1}$. The batch normalization and ReLU layers only modify the values of elements so that they also preserve the rotation-equivariance.

**Property 3**. *If we rotate $N_p$ with a rotation $\alpha \in G$, then the resulted descriptor $d'$ will be the same as the descriptor $d$ extracted on the original $N_p$.*

**Proof**: Given $d$ and $d'$ of the original and rotated point cloud by $m \in G$, we have

$$d' = AvgPool(F') = AvgPool(P_\alpha \circ F) = AvgPool(F) = d, \tag{6}$$

where the second equation holds because AvgPool is unaffected by permutations.

## APPENDIX B
## IMPLEMENTATION DETAILS

In this section, we provide implementation details for each component of RoReg.


- *H. Wang, Z. Dong, and B. Yang are with the Department of State Key Laboratory of information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, China. E-mail: {hpwang, dongzhenwhu, bshyang}@whu.edu.cn.*
- *Y. Liu is with the Computer Science Department, The University of Hong Kong, China. E-mail: yuanly@connect.hku.hk.*
- *Q. Hu and B. Wang are with the Department of Computer Science, University of Oxford, United Kingdom. E-mail: qingyong.hu, bing.wang@cs.ox.ac.uk.*
- *J. Chen is with DiDi Chuxing, China. E-mail: jerrychen@didiglobal.com.*
- *Y. Guo is with the the School of Electronics and Communication Engineering, the Shenzhen Campus, Sun Yat-sen University, China. E-mail: guoyulan@sysu.edu.cn.*
- *W. Wang is with the Department of Visualization, Texas A&M University, USA. E-mail: wenping@tamu.edu.*
- * *The first two authors contribute equally.*
- † *The corresponding authors: Zhen Dong (dongzhenwhu@whu.edu.cn) and Bisheng Yang (bshyang@whu.edu.cn).*

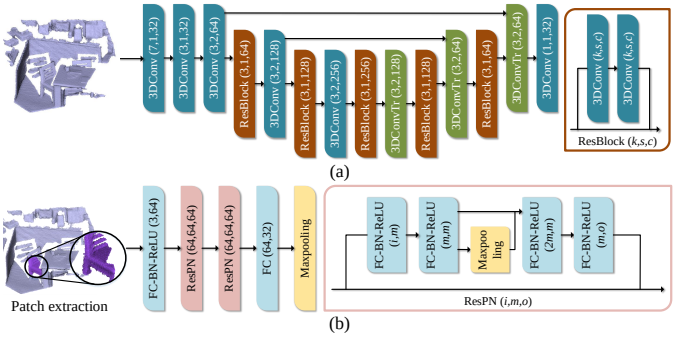


Fig. 1. (a) FCGF backbone used in RoReg. "3DConv (*k,s,c*)" denotes a 3D sparse convolution layer with a kernel size of *k*, a stride value of *s*, and *c* channels. "3DConvTr (*k,s,c*)" means the transpose convolution layer for upsampling. All convolution layers except the last one are stacked with a batch normalization layer and a ReLU layer. More details can be found in [1]. (b) PointNet backbone used in RoReg. "'FC (*i,o*)" means the fully connected layer mapping the input *i* dimensional features to *o* dimensional features. "BN" and "ReLU" denote the batch normalization layer and the ReLU layer.

## B.1 Oriented descriptor extraction

### B.1.1 Architecture

The architecture of backbones used in RoReg is shown in Fig. 1. For the default FCGF backbone, we adopt the same architecture as [1], which compute features on every point by a single forward pass on multiple 3D convolution layers [2]. We follow [1] to first downsample the input point cloud with a voxelsize of 0.025m for the 3DMatch and 3DLoMatch datasets and 0.15m for the ETH dataset.

The PointNet [3] backbone used in Sec. 4.4 is a 10-layer $MLPs$ which outputs a 32-dimension feature vector on a given point set. Given a point in the point cloud, we search for its neighboring points within a given radius, randomly sample 1024 neighboring points, feed it to the PointNet, and obtain the 32-dim feature vector by a max pooling operation. We set the radius of neighborhood as 0.3m for 3DMatch and 3DLoMatch datasets and 1m for the ETH dataset following [4], [5].

The *group feature extractor* constructs initial group features utilizing the given backbone. For the FCGF backbone, we rotate the whole input point cloud with rotations in the icosahedral group and process the rotated point clouds in parallel to obtain all point features simultaneously. For the PointNet backbone, we rotate each exacted point patch with the rotations and construct features on each point in parallel. The *group feature embedder* contains 4 group convolution layers with channl numbers of $[32 \rightarrow 256, 256 \rightarrow 512, 512 \rightarrow 256, 256 \rightarrow 32]$.

### B.1.2 Loss

Given a batch of true point pairs $\{(p, p^+)\}$ we randomly add rotations $\{\tilde{R}_p\}$ on them and compute the RoReg-Desc $\{(F_p, F_p^+)\}$. The rotation-invariant parts $\{(d_p, d_p^+)\}$ are supervised by the batch-hard loss [4]

$$\ell_{inv}(d, d^+, D^-) = \frac{e^{||d-d^+||_2} - \min\limits_{d^- \in D^-} e^{||d-d^-||_2}}{e^{||d-d^+||_2} + \sum\limits_{d^- \in D^-} e^{||d-d^-||_2}} \quad (7)$$

where $p$ is omitted for simplicity. $d$ is a rotation invariant descriptor, $d^+$ is its matched descriptor, $D^-$ are the other negative descriptors in the batch, and $||.||_2$ is the L2 norm.

Moreover, though the equivariance property of RoReg-Desc is theoretically guaranteed, noise or density variations may break the equivariance. Thus, we further supervise the rotation-equivariant part to make the network more robust. Specifically, we encourage a pair of RoReg-Desc descriptors to satisfy the row-permutation brought by the corresponding coarse rotations. For the rotation equivariant parts $(F, F^+)$ of a true correspondence $\{(p, p^+)\}$, we compute the ground truth coarse rotations $\tilde{g}_p = argmin_{g \in G} \delta(g, \tilde{R})$ for each pair and compute a equivariance loss

$$\ell_{eqv}(F, F^+, \tilde{g}) = -log(\frac{e^{\langle flat(F), flat(P_{\tilde{g}} \circ F^+ \rangle})}{\sum\limits_{g \in G} e^{\langle flat(F), flat(P_g \circ F^+ \rangle})}), \quad (8)$$

where $flat$ means flatten into vectors, $\langle ., . \rangle$ means the vector dot product.

The final loss for the *group feature embedder* is computed by

$$\ell_d = \lambda * \ell_{inv}(d, d^+, D^-) + \ell_{eqv}(F, F^+, g^+), \quad (9)$$

where $\lambda$ is set to 5.

### B.1.3 Training details

We train the FCGF backbone following the same setting as [1] and train the PointNet backbone following [5], but only use the rotation data argumentation in $[0°, 45°]$.

We follow [1], [6], [7] to use 46 scenes in 3DMatch dataset [8], 8 scenes for validation, and 8 scenes for testing, which is also adopted for the implementation of all other RoReg components. On each point cloud pair with an overlap ratio larger than 30% in the training/validation split, we randomly select 32 ground-truth point pairs as a batch. The distance of each ground-truth point pair is less than 0.03m under the ground-truth transformation. We apply the rotation data augmentation of all SO(3) to it. We train the *group feature embedder* for 10 epochs and use an Adam optimizer with a learning rate of 1e-4 for training. The learning rate is exponentially decayed by a factor of 0.5 for every 2 epochs.

## B.2 Local rotation estimation

### B.2.1 Architecture

The *rotation residual regressor* contains 3 group convolution layers with channel numbers of $[128 \rightarrow 256, 256 \rightarrow 512, 512 \rightarrow 256]$, an average pooling layer along the group channel, and a 3-layer MLPs of $(256 \rightarrow 512 \rightarrow 128 \rightarrow 4)$ with batch normalization and ReLU layers.

### B.2.2 Training details

We utilize the same training batches in Sec. B.1.3. For each point pair in the batch, we obtain its ground truth rotation residual $\tilde{R}_\epsilon = \tilde{R}\tilde{g}^T$ for the supervision of Eq. 6 in the main paper. We train the *rotation residual regressor* for 10 epochs using Adam optimizer with a learning rate of 1e-3, which exponentially decays by a factor of 0.5 every 2 epochs.

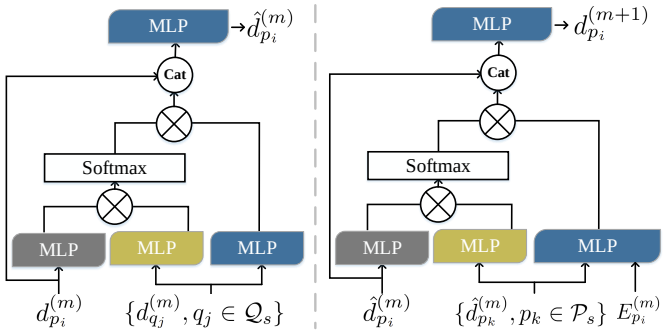


Fig. 2. (a) Cross attention layer; (b) Self attention layer.

### B.3 Rotation-guided detection

#### B.3.1 Architecture

The detection network $\phi$ contains 2 group convolution layers with channel numbers of $[32 \rightarrow 64, 64 \rightarrow 1]$, and a 2-layer $MLPs(60 \rightarrow 32 \rightarrow 1)$ with batch normalization and ReLU layers. In the evaluation, we set neighboring point number $k$ in non-maximal suppression (Sec.3.4 of the main paper) to 5 according to the validation set.

#### B.3.2 Training details

We construct training batches on each point cloud pair as Sec. 3.4 in the main paper and use a batch size of 64. The hyperparameters in the rotation-guided loss function are: $\theta_d = 0.3m$, $\theta_R = 45°$ and $\epsilon = 5°$. We train the rotation-guided detection network $\phi$ for 5 epochs. We use the Adam optimizer with an initial learning rate of 1e-4. The learning rate is exponentially decayed by a factor of 0.8 per epoch.

### B.4 Rotation coherence matcher

#### B.4.1 Attention layers

The architecture of attention layers in the rotation-guided attention block is shown in Fig. 2. Following previous methods [7], [9], [10], the cross attention layer is in charge of processing input rotation invariant feature $d^{(m)}$ to produce updated invariant features $\hat{d}^{(m)}$ with messages aggregated from the other point cloud. Specifically, given rotation invariant features $d_{p_i}^{(m)}$ and $\{d_{q_j}^{(m)} | q_j \in \mathcal{Q}_s\}$, we construct query, key, value vectors following:

$$Q_{p_i}^{(m)} = \eta_y(d_{p_i}^{(m)}),\ K_{q_j}^{(m)} = \eta_k(d_{q_j}^{(m)}),\ V_{q_j}^{(m)} = \eta_v(d_{q_j}^{(m)}), \tag{10}$$

where $\eta$ denotes a two-layer MLPs with the output dimension $n_l$. Then we update $d_p^{(m)}$ to $\hat{d}_p^{(m)}$ following:

$$\begin{aligned} \hat{d}_{p_i}^{(m)} =& \eta_u([d_{p_i}^{(m)}; \sum_{\mathcal{Q}_s} w_{q_j}^{(m)} V_{q_j}^{(m)}]), \\ w_{q_j}^{(m)} =& Softmax(Q_{p_i}^{(m)T} K_{q_j}^{(m)})/\sqrt{n_l}. \end{aligned} \tag{11}$$

The updating of $d_{q_j}^{(m)} \rightarrow \hat{d}_{q_j}^{(m)}$ is conducted similarly.

Then, $\hat{d}_{p_i}^{(m)}$ is further processed with a self-attention layer to output $d_{p_i}^{(m+1)}$ using the messages from $\mathcal{P}_s$. For the value vectors, we further concatenate the $E_{p_i}^{(m)}$ from rotation coherence layer

$$V_{p_i}^{(m)} = \eta_v([\hat{d}_{p_i}^{(m)}; E_{p_i}^{(m)}]), \tag{12}$$

where the $E_{p_i}^{(m)}$ provides features about rotation coherence. Such rotation coherence features pull features of inlier pairs closer while push away features of outlier pairs. Meanwhile, $\hat{d}_{q_j}^{(m)}$ is updated to $d_{q_j}^{(m+1)}$ similarly.

#### B.4.2 Architecture

The *rotation coherence matcher* contains $M = 2$ *rotation-guided attention block*s. In each block, intermediate feature channel is 64 and the output channel number is 32. We use 4 heads in attention layers [11].

#### B.4.3 Loss

We also apply an additional loss to supervise the learning of intermediate updated descriptors. We compute score matrix $S^{(m)}(p, q)$ of intermediate updated descriptors by $C^{(m)}(p, q) = d_p^{(m)} d_q^{(m)T}, m = 1, ..., M$ and dual-softmax operator [12] to compute $S^{(m)}$. Afterwards, given the ground truth matches $\mathcal{M}_{gt} = \{(p \in \mathcal{P}_s, q \in \mathcal{Q}_s)\}$, we compute a loss $l_{ds}^{(m)}$ from $S^{(m)}$ by

$$\ell_{ds}^{(m)} = -\frac{1}{|\mathcal{M}_{gt}|} \sum_{(p,q)\in\mathcal{M}_{gt}} \log S^{(m)}(p, q), \tag{13}$$

The final loss for the matcher is

$$\ell_m = M * \ell_{ot} + \sum_{m}^{M} \ell_{ds}^{(m)}. \tag{14}$$

#### B.4.4 Training details

A batch is constructed by randomly sampling $200 \sim 1600$ points on each point cloud of a point cloud pair with an overlap ratio larger than $5\%$. For each point in the source point cloud, we search its spatial nearest point in the target point cloud under the ground truth transformation. The point pair is regarded as a ground truth match if it satisfies the spatial mutual check [4] and distance check (less than 0.2m if aligned), and the point will be set to unmatched otherwise. We set the batch size to 1. We train the rotation coherence matcher with an Adam optimizer for 8 epochs with an initial learning rate of 1e-3. The learning rate is exponentially decayed by a factor of 0.8 every 2 epochs.

## APPENDIX C
## COMPARISON WITH EPN [13]

Both EPN [13] and RoReg are based on the icosahedral convolutions proposed in EMVN [14] to extract rotation-equivariant features. However, our network design is different from EPN and we extend to use the rotation equivariance in the whole registration pipeline, which are stated in details in the following.

a) *Network design.* EPN applies KP-Conv [15] as the translation-equivariant layer and the icosahedral convolution as the rotation-equivariant layer. The architecture of EPN interleaves the translation-equivariant layer with the rotation-equivariant layer, as shown in Fig. 3 (a). In comparison, RoReg-Desc first uses a backbone (FCGF [1]) to extract translation-equivariant features in the original Euclidean space and the applies the icosahedral convolutions, as shown in Fig. 3 (b).

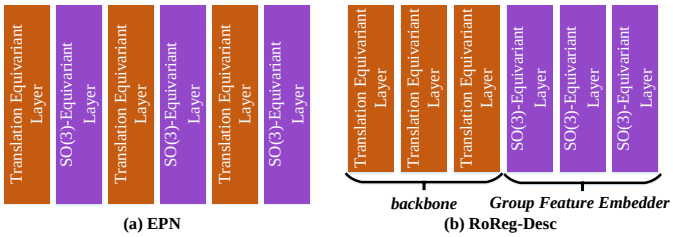


Fig. 3. Architecture difference between EPN and RoReg-Desc.

The design difference brings two consequences. First, training of EPN is much more memory-consuming than RoReg-Desc. In EPN, interleaving two translation-equivariant and rotation-equivariant layers forces EPN to maintain all 60 feature vectors on the icosahedral group for all layers including the translation-equivariant layers, which consumes lots of GPU memory. This limits the network size and the batch size in training an EPN, both of which are essential for learning a discriminative descriptor with metric learning. In RoReg, we are able to first train the backbone, which does not require the storage of 60 feature vectors on the icosahedral group. Then, we train the rotation-equivariant part based on the fixed backbone. In the implementation, on a 2080Ti GPU, we are only able to generate 5 EPN features in a training batch but allow constructing 128 RoReg-Desc in a training batch.

Second, the interleaving design in EPN is less compatible with advanced fast 3D feature extraction. For example, FCGF [1] is a well-designed efficient feature extractor with stacked SparseConv layers using MinkowskiEngine. Applying the sparse convolution of FCGF in EPN to replace the KP-Conv brings difficulty in implementation. Because the data structure of SparseConv layers and the icosahedral convolution layers are different. An additional data structure conversion would be required. In comparison, RoReg can be freely combined with FCGF or other advanced feature extractors in the future.

b) *How to use the rotation equivariance.* Though EPN also proposed to extract rotation-equivariant features, they only utilized it by pooling on the equivariant feature maps to get invariant descriptors while neglecting potential local rotation estimated by aligning two rotation-equivariant features. In comparison, RoReg not only applies the invariant part for matching but also uses the estimated local rotations in the feature detection, feature matching and transformation estimation, which greatly improves the registration performance.

c) *Results.* RoReg-Desc outperforms EPN on both efficiency and effectiveness. Under the same setting (randomly sampling 2.5k keypoints, using nearest neighbor searching with mutual check for feature matching and a vanilla RANSAC for transformation estimation), RoReg-Desc achieves $90.9\%/65.1\%$ RR on the 3DMatch/3DLoMatch dataset and surpasses EPN by $4.2\%/8.8\%$. Moreover, constructing RoReg-Desc is $\sim 190\times$ times faster than EPN due to the usage of advanced FCGF backbone.

## APPENDIX D
## ABLATION STUDY ON THE GROUP CHOICE

To further study the performances of RoReg with different SO(3) discrete groups, we re-implement RoReg with the tetrahedral group containing 12 group elements and the octahedral group containing 24 group elements. The time consumption and registration performances on the 3DMatch dataset and the 3DLoMatch dataset are reported in Table 1. It can be seen that when using a smaller group, RoReg requires much less time but yields worse registration performances, especially on the challenging 3DLoMatch dataset. Because a smaller group leads to sparse discretization of the SO(3) space and the rotation equivariance is limited on the elements in the group.

## APPENDIX E
## ABLATION STUDY ON COMPONENTS

We conduct ablation studies on each component of RoReg and RoReg-PN on the 3DMatch and the 3DLoMatch datasets as briefly discussed in Sec.4.4. The detailed results are shown in Table 2.

**Baseline model**. The model 0 is the baseline model which directly adopts the features output by the backbone (FCGF or PointNet) as descriptors. The keypoints used in the model 0 are randomly selected among all points. Descriptors of two point clouds are matched by the nearest neighborhood matcher with a mutual nearest test, which is the same as in [4]. Finally the vanilla RANSAC with edge-length-check and distance-check is applied for 50k iterations to find the final transformation.

**RoReg-Desc description**. In the model 1, we construct RoReg-Descs using the model 0 as the backbone and use the rotation-invariant parts of RoReg-Descs as descriptors instead. It can be seen that RoReg-Desc improves the registration performances by a large margin comparing to the model 0 on both IR and RR metrics. This demonstrates that the proposed feature embedder is able to extract discriminative features on the icosahedral feature map and produces robust rotation-invariant descriptors.

**Rotation-guided detection**. Based on the model 1, we further introduce the proposed rotation-guided detection in the model 2 to detect keypoints. On IR metric, the rotation-guided detection brings a $3.4\% \sim 19.4\%/2.0\% \sim 20.3\%$ improvements to the model 1. The improved correspondences result in $1.5\% \sim 11.8\%$ and $2.4\% \sim 22.3\%$ improvements on RR metric. The improvements are more significant as the keypoints become sparser, which demonstrates the repeatability and matchability of detected keypoints.

**Rotation coherence matcher**. Based on the model 1, we replace the nearest neighborhood matcher with the proposed rotation coherence matcher to construct the model 3. The model 3 is able to estimate accurate correspondences, which improves the IR metric by $4.6\% \sim 19.0\%$ with the FCGF backbone and $5.2\% \sim 25.5\%$ with the PointNet backbone. Using the rotation coherence matcher improves the RR metric by $7.3\%$ on average.

**OSE-RANSAC**. Based on the model 1, we utilize the proposed OSE-RANSAC to replace the vanilla RANSAC, which results in the model 4. Though the model 4 uses

TABLE 1
The ablation study on the SO(3) discrete group choice.

| Group | Time Consumption | | | | | Performance on 3DMatch | | | Performance on 3DLoMatch | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Des.(s) | Det.(s) | Mat.(s) | Est.(s) | Total(min) | FMR(%) | IR(%) | RR(%) | FMR(%) | IR(%) | RR(%) |
| Tetrahedral (12 elements) | 0.40 | 0.02 | 0.22 | 0.06 | **19.2** | 97.8 | 78.4 | 91.3 | 78.9 | 36.1 | 66.8 |
| Octahedral (24 elements) | 0.74 | 0.02 | 0.24 | 0.07 | 23.7 | 97.8 | 78.6 | 92.0 | 79.7 | 36.9 | 68.0 |
| Icosahedral (60 elements) | 1.81 | 0.03 | 0.27 | 0.12 | 37.1 | **97.9** | **80.2** | **93.2** | **82.1** | **39.6** | **71.2** |

TABLE 2
Ablation studies on components of RoReg. "Des." denotes the descriptor, where blank means using the features from the backbone FCGF/PointNet as the descriptor and ✓means using RoReg-Desc as the descriptor. "Det." denotes the feature detection, where blank means using randomly-sampled points and ✓means using keypoints detected by the proposed rotation guided detection. "Mat." denotes the feature matching, where blank means using the nearest neighborhood matcher and ✓uses the proposed rotation coherence matcher. "Est." denotes the final transformation estimation step, where blank means the vanilla RANSAC and ✓means the proposed OSE-RANSAC.

| | | | | | FCGF (fully convolution based backbone) | | | | | | | | PointNet (local-patch based backbone) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 3DMatch | | | | 3DLoMatch | | | | 3DMatch | | | | 3DLoMatch | | | |
| | | #kps | | | 2500 | 1000 | 500 | 250 | 2500 | 1000 | 500 | 250 | 2500 | 1000 | 500 | 250 | 2500 | 1000 | 500 | 250 |
| Id | Des. | Det. | Mat. | Est. | *Registration Recall (%)* | | | | | | | | *Registration Recall (%)* | | | | | | | |
| 0 | | | | | 76.3 | 75.7 | 73.9 | 70.0 | 36.0 | 34.9 | 31.9 | 28.5 | 49.3 | 49.5 | 43.0 | 36.9 | 15.0 | 13.7 | 10.5 | 8.1 |
| 1 | ✓ | | | | 90.9 | 89.6 | 88.4 | 83.7 | 65.1 | 63.7 | 57.4 | 47.7 | 83.1 | 78.8 | 71.6 | 56.6 | 41.1 | 34.9 | 26.2 | 14.5 |
| 2 | ✓ | ✓ | | | 92.4 | 91.9 | 91.1 | 90.3 | 67.5 | 66.7 | 63.1 | 59.5 | 85.5 | 82.8 | 80.8 | 77.4 | 47.1 | 43.3 | 39.5 | 36.8 |
| 3 | ✓ | | ✓ | | 92.0 | 91.6 | 89.8 | 89.6 | 67.7 | 65.2 | 63.0 | 51.5 | 86.2 | 84.8 | 81.1 | 71.2 | 55.6 | 49.8 | 42.1 | 28.4 |
| 4 | ✓ | | | ✓ | 91.6 | 89.9 | 88.7 | 85.9 | 67.5 | 65.3 | 60.2 | 51.6 | 84.1 | 81.5 | 76.0 | 64.0 | 52.4 | 44.4 | 36.3 | 18.8 |
| 5 | ✓ | ✓ | ✓ | ✓ | 93.2 | 92.7 | 93.3 | 91.2 | 71.2 | 69.5 | 67.9 | 64.3 | 86.8 | 86.1 | 85.7 | 81.6 | 56.7 | 54.3 | 51.0 | 43.8 |
| Id | Des. | Det. | Mat. | | *Feature Matching Recall (%)* | | | | | | | | *Feature Matching Recall (%)* | | | | | | | |
| 0 | | | | | 89.2 | 89.2 | 88.5 | 87.2 | 51.1 | 49.8 | 47.3 | 43.9 | 70.2 | 68.1 | 65.7 | 62.5 | 25.1 | 22.8 | 19.7 | 17.5 |
| 1/4 | ✓ | | | | 97.6 | 98.1 | 97.0 | 96.2 | 77.9 | 76.0 | 73.9 | 68.7 | 91.7 | 89.7 | 86.7 | 79.4 | 49.4 | 44.4 | 38.4 | 30.7 |
| 2 | ✓ | ✓ | | | 97.6 | 98.2 | 97.5 | 96.3 | 80.4 | 78.8 | 78.1 | 76.4 | 93.7 | 92.7 | 92.1 | 92.4 | 57.1 | 51.6 | 52.5 | 57.1 |
| 3 | ✓ | | ✓ | | 98.1 | 98.3 | 97.9 | 97.5 | 81.8 | 80.4 | 78.8 | 74.0 | 95.8 | 96.1 | 94.2 | 92.1 | 69.6 | 67.0 | 64.5 | 56.6 |
| 5 | ✓ | ✓ | ✓ | | 97.9 | 98.2 | 97.8 | 97.2 | 82.1 | 81.7 | 81.6 | 80.2 | 95.5 | 95.8 | 95.3 | 94.7 | 69.2 | 69.7 | 68.4 | 64.1 |
| Id | Des. | Det. | Mat. | | *Inlier Ratio (%)* | | | | | | | | *Inlier Ratio (%)* | | | | | | | |
| 0 | | | | | 53.0 | 47.4 | 41.6 | 34.9 | 14.7 | 12.2 | 10.3 | 8.7 | 21.8 | 19.1 | 16.6 | 13.7 | 4.3 | 3.7 | 3.3 | 2.7 |
| 1/4 | ✓ | | | | 61.4 | 54.6 | 47.1 | 37.4 | 23.7 | 19.5 | 16.0 | 14.3 | 27.9 | 23.5 | 19.6 | 15.2 | 8.1 | 6.7 | 5.5 | 4.2 |
| 2 | ✓ | ✓ | | | 65.4 | 59.3 | 55.5 | 56.8 | 27.1 | 23.5 | 20.5 | 22.4 | 35.3 | 29.2 | 30.7 | 35.5 | 10.6 | 8.7 | 9.0 | 11.0 |
| 3 | ✓ | | ✓ | | 79.3 | 73.5 | 66.1 | 55.2 | 36.5 | 30.6 | 25.1 | 18.9 | 53.4 | 46.0 | 38.6 | 29.4 | 18.0 | 14.9 | 12.3 | 9.4 |
| 5 | ✓ | ✓ | ✓ | | 80.2 | 75.1 | 74.1 | 75.2 | 39.6 | 34.0 | 31.9 | 34.5 | 56.4 | 51.2 | 49.2 | 49.9 | 19.2 | 17.5 | 16.3 | 16.4 |

the same estimated correspondences as the model 1, the model 4 not only converges faster but also achieves a higher RR metric. Improvements on the RR metric brought by the OSE-RANSAC are more significant on the 3DLoMatch dataset which contains low-overlap scan pairs. The results demonstrate that the proposed OSE-RANSAC is able to generate more robust and accurate hypotheses with only one correspondence, which can handle low overlap and noisy correspondences.

**RoReg**. By incorporating all proposed components in the model 5, RoReg with the FCGF backbone achieves SOTA performances on both datasets. Moreover, applying RoReg on the PointNet brings $13.0\% \sim 36.2\%$ improvements on IR metric and $35.7\% \sim 44.7\%$ improvements on RR metric, which already outperforms well-designed registration algorithms [1], [4], [5], [13], [16] on the RR metric, demonstrating the compatibility of RoReg with different backbones.

# APPENDIX F
# EVALUATION ON KITTI DATASET

## F.1 KITTI

The KITTI odometry benchmark [17] is collected by a Velodyne-64 Lidar sensor. It contains 11 sequences of street scenarios including cars, buildings, and trees. We follow [1], [5], [6] to use sequence 0-5 for training, 6-7 for validation and 8-10 for testing. We use point cloud pairs that are at most 10m away for evaluation same as [5]–[7], yielding 1358 point cloud pairs for training, 180 point cloud pairs for validation, and 555 point cloud pairs for testing.

## F.2 Metrics

We adopt *Rotation error* (RE), *Translation Error* (TE), and *Transformation Recall* (TR) for performance evaluation. We follow [1], [7] to refine the ground truth transformations using the ICP algorithm [18]. The thresholds of rotation error and translation erro in TR are set to 5° and 0.2m same as the previous methods.

## F.3 Results

The quantitative results on KITTI dataset are reported in Table. 3. RoReg performs on-par with the previous methods and achieves a final $99.8\%$ success rate with less than 1k RANSAC iterations.

# APPENDIX G
# EVALUATION ON ROTATION-SYMMETRICAL OBJECTS

Rotational symmetrical objects such as vases are difficult for reliable correspondence estimation with RoReg because the symmetry brings ambiguity for the rotation estimation. We conduct an experiment on ModelNet40 to study the performance of RoReg on rotation-symmetrical objects to show how the rotation ambiguity affects the performances.

TABLE 3
Quantitative results on KITTI. "TE" and "RE" are the average translation error and average rotation error respectively. "TR" is the transformation recall.

| Methods | TE(cm)↓ | RE(°)↓ | TR(%)↑ |
|---|---|---|---|
| 3DFeat-Net [19] | 25.9 | **0.25** | 96.0 |
| FCGF [1] | 9.5 | 0.30 | 96.6 |
| D3Feat [16] | 7.2 | 0.30 | **99.8** |
| SpinNet [5] | **6.8** | 0.32 | **99.8** |
| Predator [6] | **6.8** | 0.27 | **99.8** |
| CoFiNet [7] | 8.2 | 0.41 | **99.8** |
| RoReg | 7.0 | 0.32 | **99.8** |

### G.1 ModelNet40

ModelNet40 [20] contains CAD models from 40 man-made object categories. Most of the objects are designed mirror-symmetrical or axis-symmetrical, including tables, bowls, vases etc. We select 11 objects from each category of ModelNet40, i.e., 440 objects in total. For each object, 10k points are uniformly sampled on the object surface as the source point cloud. To generate a target point cloud for each source point cloud, we randomly rotate the source point cloud in [0°, 180°) and add uniformly-sampled noise in [-0.03,0.03]. To this end, we obtain 440 pairs of point clouds for the registration experiment.

### G.2 Settings

Without any finetuning on ModelNet40, we direct evaluate RoReg-PN trained on the 3DMatch dataset for this experiment. For each point cloud, we extract RoReg-Desc on 512 uniformly sampled points. Then, for each aforementioned point cloud pair, the RoReg-Desc of the source and the target point cloud are matched to build correspondences with the rotation coherence matcher. A local transformation matrix is estimated on each correspondence using the proposed local rotation estimation module.

### G.3 Results

In Fig. 4, we report the inlier ratio (with $\tau_1$ set to 0.1) and average angular error of estimated local rotations on each category. For rotation-symmetric objects like bowls, the established correspondences are typically unreliable with lower inlier ratio due to the matching ambiguity and thus the local rotations estimated on the correspondences are inaccurate with higher rotation errors due to the rotation ambiguity. In Fig. 5, we show some typical examples. For rotation-symmetric objects (Fig. 5(a-c)), it is hard for RoReg to construct reliable correspondences and estimate correct local rotations, unless the correspondences lie in distinctive areas (the handle in Fig. 5(d)). For objects without such symmetries, the estimated correspondences and local rotations from RoReg are much more reliable (Fig. 5(e-f)).

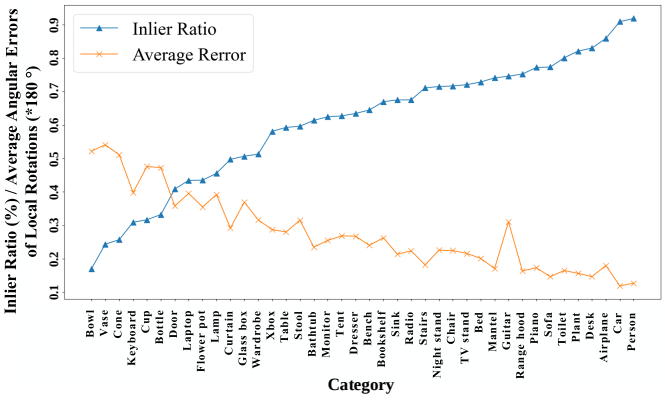


Fig. 4. Inlier ratio and average local rotation error on each object category.

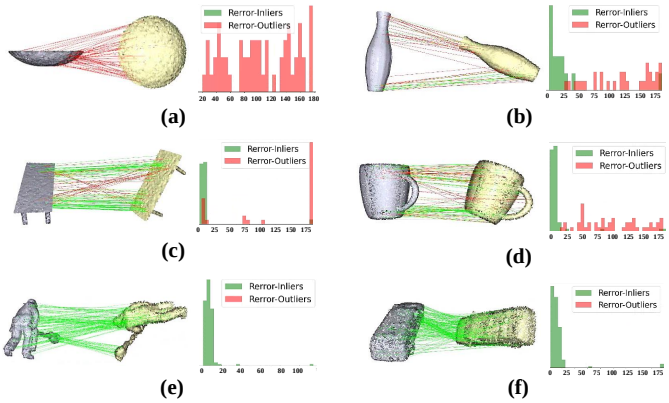


Fig. 5. Qualitative results on ModelNet40. (a) comes from 'bowl'. (b) comes from 'vase'. (c) comes from 'table'. (d) comes from 'cup'. (e) comes from 'person'. (f) comes from 'car' with asymmetrical interiors. On each object, we report the error distribution of local rotations estimated by the correct correspondences (green) and incorrect correspondences (red).

## REFERENCES

[1] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *ICCV*, 2019.
[2] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *CVPR*, 2019.
[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.
[4] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *CVPR*, 2019.
[5] S. Ao, Q. Hu, B. Yang, A. Markham, and Y. Guo, "Spinnet: Learning a general surface descriptor for 3d point cloud registration," in *CVPR*, 2021.
[6] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler, "Predator: Registration of 3d point clouds with low overlap," in *CVPR*, 2021.
[7] H. Yu, F. Li, M. Saleh, B. Busam, and S. Ilic, "Cofinet: Reliable coarse-to-fine correspondences for robust point cloud registration," in *NeurIPS*, 2021.
[8] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *CVPR*, 2017.
[9] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *CVPR*, 2020.
[10] Y. Li and T. Harada, "Lepard: Learning partial point cloud matching in rigid and deformable scenes," *arXiv preprint arXiv:2111.12591*, 2021.
[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.
[12] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "Loftr: Detector-free local feature matching with transformers," in *CVPR*, 2021.
[13] H. Chen, S. Liu, W. Chen, H. Li, and R. Hill, "Equivariant point network for 3d point cloud analysis," in *CVPR*, 2021.

[14] C. Esteves, Y. Xu, C. Allen-Blanchette, and K. Daniilidis, "Equivariant multi-view networks," in *ICCV*, 2019.
[15] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *CVPR*, 2019.
[16] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai, "D3feat: Joint learning of dense detection and description of 3d local features," in *CVPR*, 2020.
[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.
[18] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE TPAMI*, no. 5, pp. 698–700, 1987.
[19] Z. J. Yew and G. H. Lee, "3dfeat-net: Weakly supervised local 3d features for point cloud registration," in *ECCV*, 2018.
[20] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015.