2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP) | 978-1-6654-3153-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/M2VIP49856.2021.9665098

PLReg3D: Learning 3D Local and Global Descriptors Jointly for Global Localization

Zhijian Qiao[1], Hanwen Wang[1], Yu Zhu[1] and Hesheng Wang[1]

***Abstract*— In the application of autonomous driving, 3D global map helps to reduce the accumulated error, thus it becomes a crucial precondition for accurate and robust localization. However, in the absence of GPS, it is difficult for the robot to locate in the global map at the initial step. In this paper, an efficient learning-based pipeline is proposed to learn 3D local and global descriptors of large-scale point cloud jointly. Specifically, point-wise local descriptors are learned to predict correspondence for global registration and subsequently aggregated into a compact global descriptor by a simple but efficient pooling layer for 3D place recognition. To this end, proposed network adapts FPN and U-Net to extract multi-scale feature and 3D sparse convolution is applied to encode voxelized large-scale point cloud fastly. Evaluation on public dataset KITTI and our robot show that robust global localization on a fixed global map can be achieved accurately and efficiently. Our source code is available at https://github.com/IRMVLab/PLReg3D.git**

I. INTRODUCTION

In recent years, deep learning has attracted more and more interest from researchers in the field of localization and mapping, with numerous related researches popping up. [1] divides the tasks of deep learning in localization into two parts, namely odometry estimation and global localization. Although there have been many recent works on LiDAR odometry [2], [3] and Visual-Inertial Odometry (VIO) [4], which have gradually improved the accuracy of odometry estimation, it still cannot avoid the defect in the nature of such algorithms, which is the accumulated error as the distance increases. Although loop detection can correct this error to some extent, it also greatly limits the freedom of the robot path planning.

Therefore, global localization is the best choice for long-term condition. According to the type of the sensor, global localization can be divided into 2D-to-2D and 3D-to-3D. Compared with image, point cloud has more structure information and more accurate depth information. More importantly, the environment at the time of global map construction can very different from that at the time of localization due to season, weather, lighting and other factors, but point cloud data

This work was supported in part by the Natural Science Foundation of China under Grant 62073222 and U1913204, in part by "Shu Guang"project supported by Shanghai Municipal Education Commission and Shanghai Education Development Foundation under Grant 19SG08, in part by Shenzhen Science and Technology Program under Grant JSGG20201103094400002, in part by the Science and Technology Commission of Shanghai Municipality under Grant 21511101900, in part by grants from NVIDIA Corporation. Corresponding Author: Hesheng Wang.

The authors are with Department of Automation, Key Laboratory of System Control and Information Processing of Ministry of Education, Key Laboratory of Marine Intelligent Equipment and System of Ministry of Education, Shanghai Engineering Research Center of Intelligent Control and Management, Shanghai Jiao Tong University, Shanghai 200240, China.

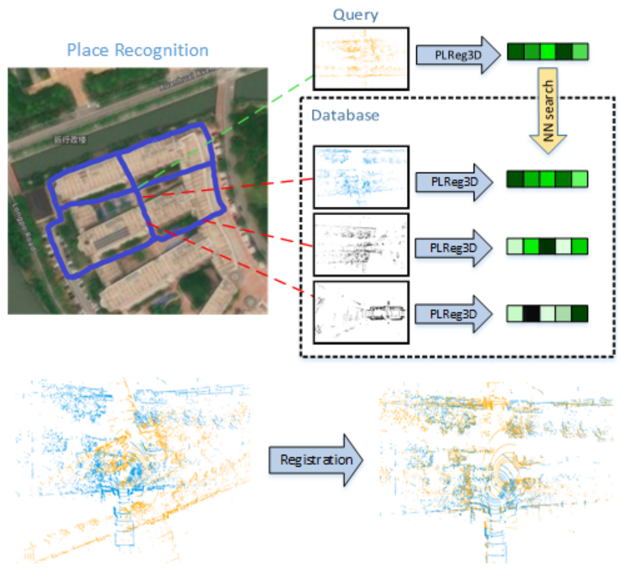

Fig. 1. The global localization framework is shown above. For a query point cloud (yellow), we use KNN to find the most similar point cloud (blue) based on the global descriptor of the point cloud, and exclude those that are not similar (grey). This gets a coarse pose in the global coordinate system. Then, based on the local descriptors, we register the query point cloud (yellow) and its most similar point cloud (blue) to further obtain an accurate pose.

are naturally more robust for these environmental changes. Usually, a built map (or database) is used to provide globally consistent information, including geometric, semantic, and so on. As is shown in Fig. 1, the algorithm takes the real-time point cloud obtained from the sensor as input. It first obtains a coarse pose by retrieving the place in the map that is most similar to the input point cloud, and then fines the pose estimation by point cloud registration. Since the global information does not change, the localization error depends only on how well the point cloud collected from the sensor register with the global map, which is not affected by previous localization results.

In this paper, we propose a efficient network named PLReg3D(place recognition and registration) to follow this coarse-to-fine pipline. In order to encode point clouds more effectively, we voxelize point cloud and apply convolution operation on 3D structure. Compared with projecting point cloud onto an pseudo image and then applying 2D convolution, the method we use retains 3D structure information, and the application of 3D sparse convolution allows us to achieve processing speed comparable to 2D convolution. In addition, regardless of point-wise local descriptors or global descriptors, we need to not only obtain a large enough or even global receptive field, but also learn local fine structure information. To this end, we adopt U-Net and FPN in our network, whose upsampling layer and lateral connection make extracted feature both high-semantic and high-resolution. More importantly, the learned local descriptors are reused to generate global descriptor, which allows our network to avoid redundant computation consumption in inference and become more real-time. Finally, we use the model trained by point cloud registration as a pretrained model for the backbone of our network, so that the model can benefit from structure information learned from registration at the beginning of training. Overall, the contribution of this article consists of three points:

First of all, we propose a network that can jointly learn local and global descriptors, and several of the most effective point cloud learning methods are integrated into the system to learn multi-scale and discriminative point cloud feature.

Secondly, we propose an effective training pipline, so that global descriptor learning benefits from local geometric features without any effort at the initial step.

Finally, the proposed method was evaluated in both public dataset KITTI [5] and our dataset collected by a robot, and perform well. We also evaluate a variety of outlier removing methods to make the final localization results more robust in order to enhance applicability.

II. Related Work

3D-to-3D localization is usually divided into two steps: given an built map and query(or anchor) point cloud frame, first use the place recognition algorithm [6] to retrieve the most similar point cloud frame to obtain a coarse global pose, and then register the anchor frame with the retrieved frame to get a more accurate pose transformation.

A. Place Recognition

PointNetVLAD [7], the first learning-based large-scale point cloud global descriptor extraction network, uses the NetVLAD layer [8] to aggregate the local point cloud feature that PointNet [9] learns, compressing into a compact low-dimensional global descriptor. The most similar point cloud to the query frame is found by K Nearest Neighbor (KNN), which retrieve the point cloud from database whose global descriptor has the smallest Euclidian distance from that of the query frame. However, PointNet ignores the local structure information of point cloud, so LPD-Net [10] uses KNN graph neural network to aggregate neighbor points in both Cartesian space and feature space to obtain more local information-rich feature, achieving far better performance than PointNetVLAD [7]. Recently, MinkLoc3D [11] applies 3D sparse convolution to extracting multi-scale feature from point cloud, achieving both the best result and the fastest inference speed.

B. Global Registration

Deep learning-based point cloud registration [12]–[15], with excellent feature extraction capabilities, is also better than most traditional registration methods with hand-crafted feature, such as ICP [16] and NDT [17], in global registration tasks that do not provide initial pose. DCP [18], for example, is the first network to apply Transformer to point cloud registration. Based on DCP [18], PRNet [19] works with partial-to-partial registration in an iterative way, which is more practical for point cloud collected by sensors in different views. Instead of only learning descriptos, D3Net [20] jointly learns of dense detection and description of 3d local descriptors. Different from above point-based methods, FCGF [21] voxelizes large-scale point cloud and uses a 3D sparse convolution-based U-Net [22] network to extract more discriminative point-wise feature. Based on this, DGR [23] then evaluates the weight of each matching pair, and proposes a more robust global registration method. While methods based on deep learning have achieved excellent results on the point cloud registration, some traditional methods still have great reference value. TEASER++ [24] can solve the rigid body transformation problem well even if the input correspondences have an extremely large number of outliers.

However, none of the above methods explores the internal relationship between global descriptors and local descriptors, so as to design a reasonable method for joint learning. As far as we know, there is little work to learn two descriptors together and use them for place recognition and point cloud registration. DH3D [25] is the first approach that unifies global place recognition and local 6DoF pose refinement by designing a Siamese network and an attention-based aggregation layer.

III. Method

Given the input point cloud $P_b \in R^{N\times 3}$ in body coordinates, as well as the global point cloud map $M = \{P_i \mid i = 0, \ldots, m\}$ in the form of a set of submaps P_i, our goal is to output the global position T_b^w of P_b. To do this, we need to learn from each point cloud P its local descriptors $f_l \in R^{N\times c}$ (c is the local feature dimension), and its global descriptor $f_g \in R^d$ (d is the global descriptor dimension). Based on the Euclidean distance between f_g, we can find the submap P_{pos} that is most similar to P_b. Then, through KNN in the local descriptors f_l, we can expose the correspondence between the two point clouds, and thus compute the relative pose transformation between the two point clouds, T_b^w, by directly applying SVD decomposition.

A. Network architecture

As shown in Fig. 2, our network mainly consists of four main parts: the voxelization layer, the U-Net-based feature extraction layer, the FPN-based feature extraction layer, and the aggregation layer. Like most methods that use 3D sparse convolution, we perform voxelization to turn the input point cloud $\mathrm{P} = \{(\mathrm{x_i}, \mathrm{y_i}, \mathrm{z_i})\}$ into a single-channel sparse tensor $\hat{P} = \{(\widehat{x_l}, \widehat{y_l}, \widehat{z_l}, r)\}$, where r is the reflection intensity, which acts as a voxel feature. When the reflection intensity is missing, the feature can also be directly set to 1, indicating that each point has the same feature.

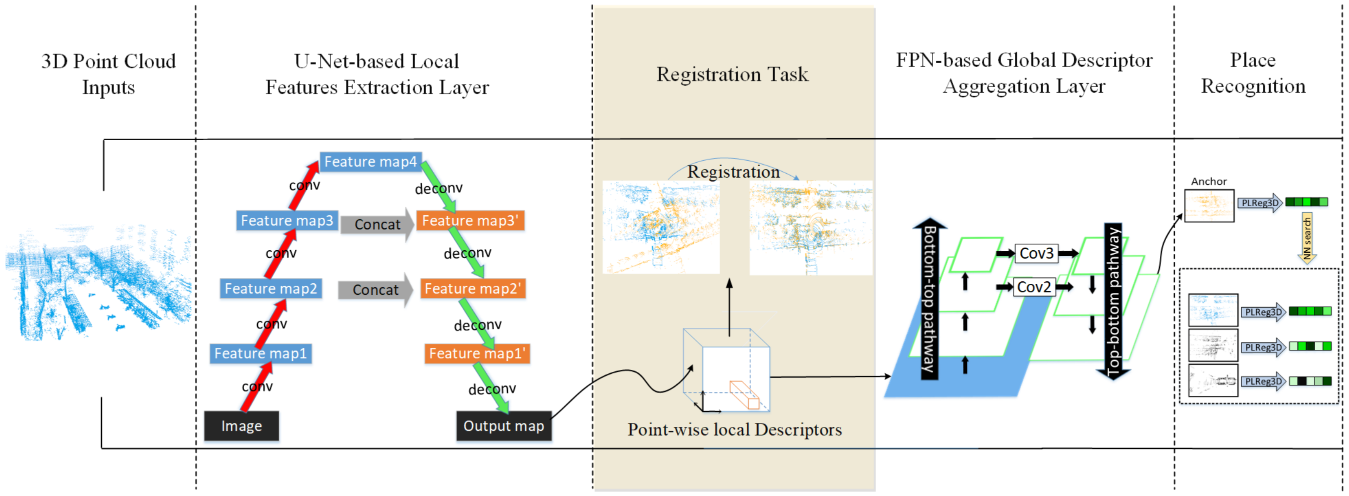

Fig. 2. Overview of PLReg3D. For the input point cloud, we first use U-Net to extract the point-wise local descriptors for point cloud registration, and refill them into a sparse feature cuboid. Then, FPN is adopted to aggregate feature cuboid on different scales, and finally a compact global descriptor can be compressed by a pooling layer for place recognition.

FCGF [21] is a fast and accurate point cloud registration network, so we are inspired to use a U-Net structure with skip connection and residual block to extract sparse fully-convolutional feature. Residual block allows us to have a deeper network structure, and skip connection enable the local descriptors f_l to combine semantic information with high-resolution feature from shallow network.

At this point, we directly use the sparse fully-convolutional features f_l to learn the global descriptors f_g, rather than relearn from the original point cloud P. This not only saves computation, but also allows the learned global descriptor to naturally contain spatial structure information. FPN is used in the field of object detection. With the increase of convolution layer, the semantic information of feature map becomes richer, but on the contrary, the spatial resolution of feature map becomes lower. Therefore, the feature map with rich semantic information but low resolution is up-sampled through interpolation by FPN, and then the resulting feature map is fused with the feature map of shallow network through lateral connection to obtain high resolution feature map with rich semantic information. As depicted in Figure 2, local descriptors f_l go through an FPN-based network structure, which consists of both bottom-up and top-down parts, thus gaining feature with a greater receptive field, denoted as f_lg.

MinkLoc3D [11] proved that the simple and effective aggregation layer GeM performs better than NetVLAD in both efficiency and performance, so we chose the same structure as our aggregation layer to generate the final global descriptor f_g:

$$f_g^{(k)} = \left(\frac{1}{N} \sum_{j=1 \ldots N} \left(f_{lg}^{(j,k)} \right)^p \right)^{\frac{1}{p}} \tag{1}$$

Where $f_{lg}^{(j,k)}$ is k-th feature of j-th point feature in f_lg and p is the relaxation factor of GeM to generalize between global max pooling and global average pooling operators.

B. Network training

Our training process mainly consists of two processes: point cloud registration and place recognition. We can get the relative pose transformation between two point clouds through SLAM [3] and use this as the ground truth. Pose-based losses and matched point-based losses are two common losses, which can be expressed as follows respectively:

$$\mathcal{L}_{pose} = \left\| \mathbf{R}_{pred}^T \mathbf{R}_{gt} - \mathbf{I} \right\|^2 + \|\mathbf{t}_{pred} - \mathbf{t}_{gt}\|^2 \tag{2}$$

$$\mathcal{L}_{cor} = \frac{1}{N} \sum_{j}^{N} |(\mathbf{R}_{gt}\mathbf{x}_j + \mathbf{t}_{gt}) - (\mathbf{R}_{pred}\mathbf{x}_j + \mathbf{t}_{pred})| \tag{3}$$

$\mathbf{x}_j$ is an origin point in source(or anchor) point cloud. $\mathbf{R}_{pred}$ and $\mathbf{t}_{pred}$ can be simply solved by SVD. Define centroids of two point cloud as

$$\bar{x} = \frac{1}{K} \sum_{i=1}^{K} x_j, j \in \mathbb{K}, \tag{4}$$

$$\bar{y} = \frac{1}{K} \sum_{i=1}^{K} \hat{y}_i, i \in \mathbb{K}. \tag{5}$$

$\hat{y}_i$ is the matching point of x_j. The cross-covariance matrix H can be obtained as

$$H = \sum_{i=1}^{K} (x_i - \bar{x})(\hat{y}_i - \bar{y}). \tag{6}$$

Similar to [18], the traditional singular value decomposition (SVD) can be used to decompose the matrix $H = USV^T$, then the pose estimation result can be acquired by

$$\begin{aligned} \mathbf{R}_{pred} &= VU^T \\ \mathbf{t}_{pred} &= -\mathbf{R}_{pred}\bar{x} + \bar{y} \end{aligned} \tag{7}$$

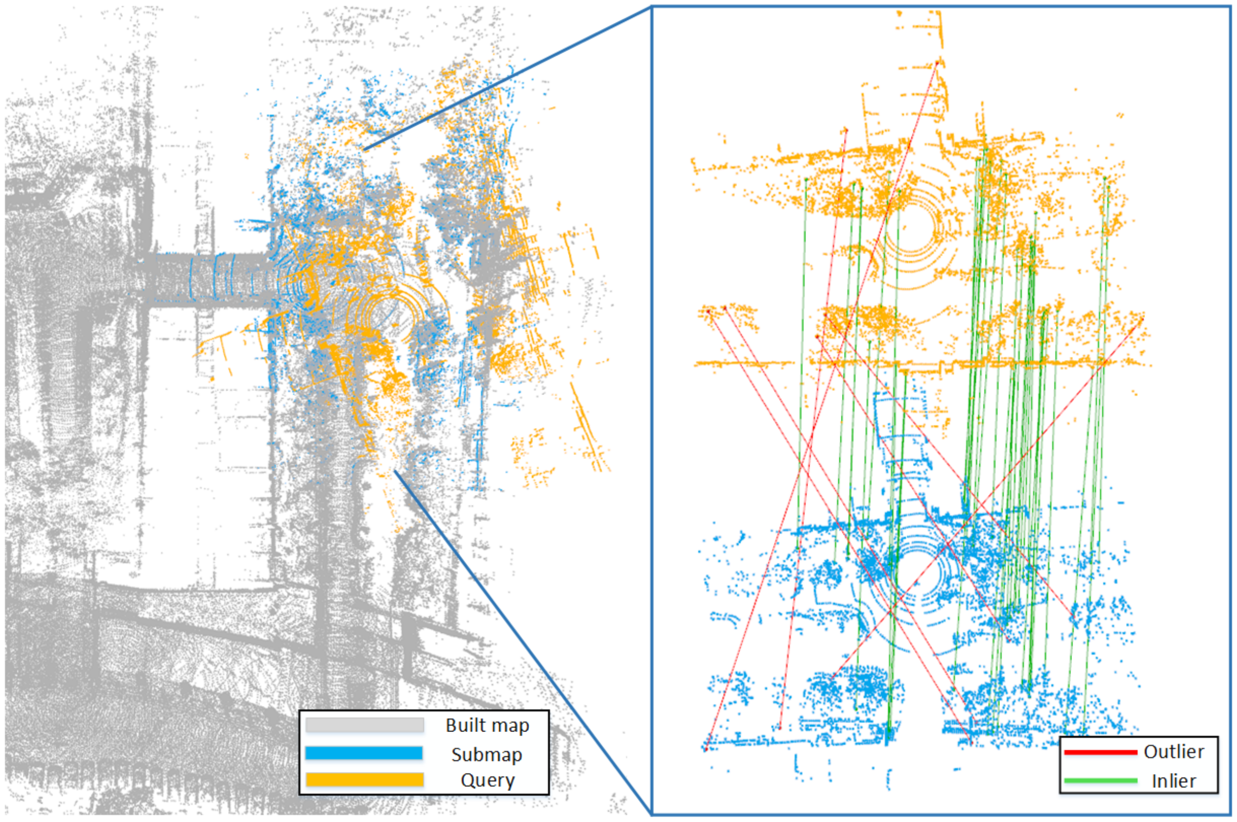

Fig. 3. We show the global localization based on a built map(gray). Given the query point cloud (yellow), based on the KNN on global descriptors, we find the most similar submap (blue) to query. The robot has different positions and poses when collecting these two point cloud frames, which is marked by the coordinate axes shown in the figure. With red, green, and blue axis corresponding to X,Y and Z axis respectively, the coordinate system on the left refers to the submap, while the right refers to query. Through KNN on the local descriptors as well as the outlier removing methods such as RANSAC, we can expose the correspondence between two point clouds so as to compute their relative pose transformation.

$\mathbf{R}_{gt}$ and $\mathbf{t}_{gt}$ represent ground truth from LiDAR SLAM. However, considering that the accuracy of SLAM is not guaranteed in some scenarios, Chamfer distance metric and Earth Mover's Distance metric are also our alternatives:

$$\mathcal{L}_{EMD} = \min_{\psi:\mathbf{X}_t \to \mathbf{Y}} \frac{1}{|\mathbf{X}_t|} \sum_{x \in \mathbf{X}_t} \|x - \psi(x)\|_2 \tag{8}$$

$$\mathcal{L}_{CD} = \frac{1}{|\mathbf{X}_t|} \sum_{\mathbf{x} \in \mathbf{X}_t} \min_{\mathbf{y} \in \mathbf{Y}} \|\mathbf{x} - \mathbf{y}\|_2^2 + \frac{1}{|\mathbf{Y}|} \sum_{\mathbf{y} \in \mathbf{Y}} \min_{\mathbf{x} \in \mathbf{X}_t} \|\mathbf{x} - \mathbf{y}\|_2^2 \tag{9}$$

Where $\mathbf{X}_t$ is the transformed source point cloud by predicted transformation and ψ is a bijection function that minimizes the distance between corresponding pairs. However, we find the losses based on Chamfer distance and Earth Mover's Distance can not lead to a right direction for our network training. This is probably because both of these losses require correspondence in some sense, which is difficult to find in the case of predicted and true poses that are significantly different. Finally, we choose both the matched point-based loss and the pose-based loss with the same weight.

In order not to lose the spatial structure information learned by the point cloud registration task, we chose to fix the parameters of the U-Net feature extraction module when training the place recognition task, and optimize only the FPN-based module and GeM. The training methods and losses are set to be mini-batch and triplet loss used by MinkLoc3D [11], respectively. Given a triplet (P_a, P_pos, P_neg) representing anchor, positive sample and negative sample respectively, a triplet margin loss is defined as follows:

$$\mathcal{L}_{tri} = \max\left\{d\left(a_i, p_i\right) - d\left(a_i, n_i\right) + m, 0\right\} \tag{10}$$

Where i represents the index of training triplet, d is a distance metric function which is Euclidean distance in this paper and m is the margin hyperparameter to make different clusters far away from each other.

C. Experiment

To verify the effectiveness of our approach in global localization, we experimented with the public dataset KITTI [5] and our own dataset, respectively. Our own data set was collected by the eight-wheel robot shown in Fig. 4. Since we first apply place recognition to get a coarse pose, and then use point cloud registration to compute an accurate pose, the experiment part will focus on these two parts.

TABLE I
COMPARISON RESULTS OF THE SUCCESS RATE WITH DIFFERENT METHODS

	Time(s)	RTE(m)	RRE(degree)	Success rate(%)
DGR	0.5652	1.0104	9.1237	86.9565
RANSAC100	0.5244	2.4756	8.2820	78.2608
DGR+RANSAC100	0.5449	2.1231	10.9417	84.7826
RANSAC1000	0.5846	2.5505	11.4518	82.6086
DGR+RANSAC1000	0.5901	2.1418	9.84186	84.7826
RANSAC10000	1.1127	2.4559	9.7808	86.9565
DGR+RANSAC10000	0.9689	2.2830	9.6080	80.4347
TEASER++	1.9545	1.0128	9.0371	86.9565
DGR+TEASER++	1.0684	1.7190	14.2415	82.6086
DGR+ICP	**0.6570**	**1.0020**	**8.4859**	**89.1304**
DGR+ICP+TEASER++	1.1712	1.3295	12.8767	82.6086
DGR+ICP+RANSAC10000	**1.0660**	**0.6860**	**8.19139**	**91.3043**

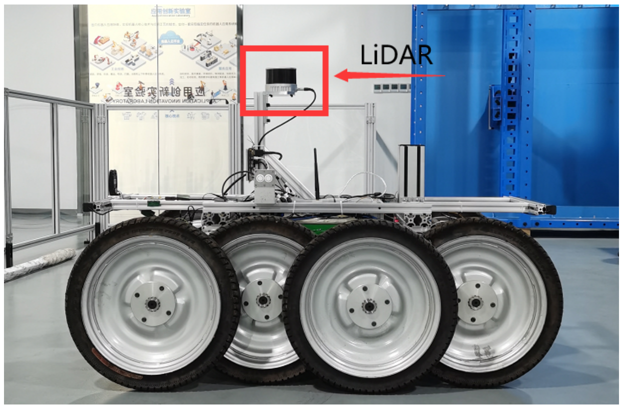

Fig. 4. Eight-wheel off-road robot. The robot can walk on uneven roads, such as slopes, grass, and even stairs. LiDAR is placed on top of the robot to collect point cloud data.

1) Place Recognition: We chose the KITTI 00 series as our training set and the 01 series as the test set. First, we sample the series equidistantly based on the global pose of each frame in the series. For training sets, the sampling distance is 1m, and for test sets, the sampling distance is 1.5m. Then, for each frame of point cloud, we limit it to a cube of 30m edge length. For each frame of point cloud, we mark the point clouds within 5m distance from it as positive samples and the point clouds more than 30m away as negative samples. Furthermore, in the training process, we used data augumentation, including random rotation, random flip, and random translation. For our own dataset, ground truth is obtained by SLAM, and the subsequent steps are the same as those used to process KITTI datasets.

We use recall rate as the evaluation metric. That is, find n most similar places in the database, if one is correct, count the success of the retrieve, and finally count the success rate of all samples. To prove that our method works, we did the following ablation experiment, as shown in Table II. FPN represents the global descriptor extraction network. "Fix" means that U-Net network was added in front of FPN and the parameters are fixed after registration training, while "noFix" means parameters of U-Net would be update. "KITTI" and "Robot" represents the dataset on which training or testing was performed. In the end, we counted the Top 5 recall rate and the Top 1% recall rate.

TABLE II
COMPARISON RESULTS OF THE AVERAGE RECALL (%) UNDER DIFFERENT NETWORKS AND SETTINGS

	Top 5 /KITTI	Top 1% /KITTI	Top 5 /Robot	Top 1% /Robot
FPN+KITTI	**97.82**	**99.63**	91.50	97.95
Fix+FPN+KITTI	**97.80**	**99.53**	90.82	**98.36**
noFix+FPN+KITTI	97.65	99.59	89.58	97.4
FPN+Robot	96.39	99.11	94.24	96.03
Fix+FPN+Robot	95.86	98.93	**96.98**	**98.08**
noFix+FPN+Robot	95.96	98.86	96.30	97.95

From the above results, it can be seen that in most cases, point cloud registration improved the performance of place recognition to some extent, and that fixing the parameters of local feature extraction network U-Net not only simplified the training process, but also made our results even better. But we also observed that in some cases, the addition of registering for the network did not lead to better results. From our original viewpoint, each point input to the FPN would have the same feature (i.e., 1), the only difference being their three-dimensional coordinates. Adding the registration module and learning the local descriptor as the initial feature of each point is at worst better than setting it up the same way. While this is not always the case, we suspect that the deepening of the network may make FPN less easy to train. Therefore, how to adjust the depth of U-NET and FPN needs to be further studied.

2) Point Cloud Registration: We paired adjacent frames of point cloud 5m apart for the training of point cloud registration. Similarly, we use random rotation and random translation for data augumentation. We followed the DGR training process. However, since the parameters of voxelization have some influence on the network, we compared two reasonable parameter settings, as shown in Table III.

TABLE III
COMPARISON RESULTS OF THE SUCCESS RATE UNDER DIFFERENT SETTINGS

	Time(s)	RTE(m)	RRE(degree)	Success rate(%)
v0.2+p4	0.79	1.10	11.64	84.78
v0.3+p3	**0.56**	**1.01**	**9.12**	**86.95**

"v0.2+p4" means that the size (edge length) of each voxel is 0.2m, and that the nearest 4 voxels are marked as positive samples. We can see that when the voxel is 0.3m in size, the effect is better.

After we get the most similar point cloud, we'll align query to it. Even though the two point clouds are collected in the same place, differences in translation and rotation still exist. More importantly, since the time of collecting may be a long time apart, objects in the scene might change, such as pedestrians, vehicles, etc., which makes the point clouds we collect have mismatched parts. Therefore, we should apply appropriate strategies to remove outliers during registration. We used RANSAC, DGR [23], and TEASER++ [24] to remove outliers and compared the registration success rates in Table I. We regard a registration error within 1m and 10 degrees as success. RANSAC100 represents that the maximum number of iterations is 100. ICP is used to refine the registration result. Figure 3 shows an example of registration.

It can be seen from the table that, in terms of individual methods, TEASER++ and DGR have better registration accuracy, but TEASER++ takes much longer than DGR. This is because TEASER++ uses the truncated least squares method, which requires several iterative estimation, but DGR directly predicts the uncertainty of each matching pairs, thereby screening out the outliers. In theory, for RANSAC, the larger the max iteration number, the better the result should be. But through experiments, we found that this seems difficult to guarantee, yet registration time increases while the max iteration number becomes larger. Although ICP is easy to fall into a local optimal solution, we found that if we can give a good initial value, then ICP is very helpful to improve the success rate. Finally, we verified that DGR is used for registration first, RANSAC is used when there are too many outliers, and finally ICP is used to refine the registration results, which has the best effect.

IV. CONCLUSIONS

We propose a simple and efficient network to learn local and global descriptors jointly, and simplify the training process by fixing the parameters of local feature extraction network. At the same time, we have verified the validity of our algorithm on real dataset collected by our robot, and obtained good results. In the future, we will test the effectiveness and generalization capability of our work in global localization based on larger maps.

REFERENCES

[1] C. Chen, B. Wang, C. X. Lu, N. Trigoni, and A. Markham, "A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence," *arXiv preprint arXiv:2006.12567*, 2020.

[2] X. Zhao and H. Wang, "Self-localization using point cloud matching at the object level in outdoor environment," in *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2019, pp. 1447–1452.

[3] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[4] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] W. Zhang and C. Xiao, "Pcan: 3d attention map learning using contextual information for point cloud based retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 436–12 445.

[7] M. Angelina Uy and G. Hee Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4470–4479.

[8] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[10] Z. Liu, S. Zhou, C. Suo, P. Yin, W. Chen, H. Wang, H. Li, and Y.-H. Liu, "Lpd-net: 3d point cloud learning for large-scale place recognition and environment analysis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2831–2840.

[11] J. Komorowski, "Minkloc3d: Point cloud based large-scale place recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1790–1799.

[12] H. Wei, Z. Qiao, Z. Liu, C. Suo, P. Yin, Y. Shen, H. Li, and H. Wang, "End-to-end 3d point cloud learning for registration task using virtual correspondences," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2678–2683.

[13] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust efficient point cloud registration using pointnet," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7156–7165.

[14] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler, "Predator: Registration of 3d point clouds with low overlap," *arXiv preprint arXiv:2011.13005*, 2020.

[15] Z. J. Yew and G. H. Lee, "3dfeat-net: Weakly supervised local 3d features for point cloud registration," in *European Conference on Computer Vision*. Springer, 2018, pp. 630–646.

[16] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-icp: A globally optimal solution to 3d icp point-set registration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2241–2254, 2015.

[17] H. Merten, "The three-dimensional normal-distributions transform," *threshold*, vol. 10, p. 3, 2008.

[18] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3523–3532.

[19] ——, "Prnet: Self-supervised learning for partial-to-partial registration," in *Advances in Neural Information Processing Systems*, 2019, pp. 8814–8826.

[20] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6359–6367.

[21] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8958–8966.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[23] C. Choy, W. Dong, and V. Koltun, "Deep global registration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2514–2523.

[24] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration," *IEEE Trans. Robotics*, 2020.

[25] J. Du, R. Wang, and D. Cremers, "Dh3d: Deep hierarchical 3d descriptors for robust large-scale 6dof relocalization," in *European Conference on Computer Vision*. Springer, 2020, pp. 744–762.