Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

3D LiDAR Map Compression Using Deep Neural Network

Xuanmeng Zhang, Yang Li, Huan Yin and Rong Xiong
Institute of Cyber-Systems and Control
Zhejiang University
Hangzhou, China
rxiong@zju.edu.cn

***Abstract*— 3D LiDAR based maps provide basic foundation for localization and navigation of mobile robots. However, in outdoor scenarios, too many map points bring a huge computational burden to the processor, resulting in both slow calculation speed and poor real-time performance. In this paper, we propose a deep learning method to reduce the point cloud maps by applying and modifying the structure of PointNet. The recent PointNet architecture proposes a new approach to process 3D point clouds in map, which has made great performance in many tasks. In addition, we conduct experiments on self-collected dataset and evaluate the localization performance of compressing maps. The final results show that our model can implement map compression while keeping the localization error within an acceptable range.**

***Index Terms*— 3D point clouds, mobile robot, map compression**

I. Introduction

Map is a key component of mobile robots on road. 3D LiDAR map, or 3D point cloud map, is one of most widely used maps, and is also quite significant to localization tasks in outdoor scenarios. Generally, 3D point clouds stores massive data in the medium, aiming to achieving fine refinement of feature coverage. Recently this kind of representation of map is commonly used . But the trouble is point cloud database is too large for mobile robots. Millions of points result in high computational cost. More seriously, real-time localization can not be implemented. One possible solution is to remove the redundancy and noisy data. In this way, the computation speed of localization algorithm can be accelerated while managing to implement localization as well as not affecting the accuracy.

The most simplest method to compress 3D LiDAR map is random sampling. It reduces 3D data according to specific rules and ratio. However, how to implement this method is confusing, for the rules to keep one point or not may be difficult to formulate. Another common method is VoxelGrid filter. Based on bounding box algorithm, this approach [1] is originally used to solve the optimal enclosing space of discrete point sets. VoxelGrid filter method divides the entire space into many voxel grids. Points inside one grid are replaced by the one center point, approximatively implementing downsampling method. The searching algorithm [2] commonly used in random sampling and VoxelGrid filter is called octree.

In vision community, map compression can be implemented based on optimization approach. Specifically, some researchers [3]–[7] solved map compression using programming methods. They use as fewer points as possible to match at least the fixed number of features or points for localization. This method takes the binary relations between map points and the number of points that can be matched into account. And they also considered giving different weights to different points, which can intuitively represent the importance of each point. However, bad things may happen with the number of point clouds increasing because it is not a feasible solution for huge optimization.

For 3D point clouds, deep learning technique can be applied to many tasks, object classification [8], [9] and place recognition [10], [11] etc. Charles R.Qi and Hao Su [12] released a novel and effective deep learning architecture PointNet to process 3D point clouds. This typical feature-based deep neural network (DNN) is able to learn deep point set features efficiently and robustly. Taking point clouds' coordinates directly as input, PointNet outputs both class labels of the entire point clouds and per point segment labels for each point. Every point in input data is represented by a three-dimensional coordinate without other features. The overall idea of PointNet is to fit a symmetric function. Using convolution and max pooling layers, the properties of point clouds can be preserved successfully.

In this paper, considering the existing works, we refer to the structure of PointNet to achieve our goal of compressing. Based on the idea of point cloud segmentation, the compression problem can be formulated as a classification task: evaluate the importacne of every point in the map. In the experimental section, we test the localization accuracy of the proposed compressed maps. Overall, the contributions of this paper are as follows:

- We transform map compression problem to the point cloud segmentation, which can be implement with PointNet.
- Taking the result of linear programming as the supervision material, we construct a deep learning network for map compression in new environments.
- After training, we conduct experiments on our self-collected dataset and validate the effectiveness of the

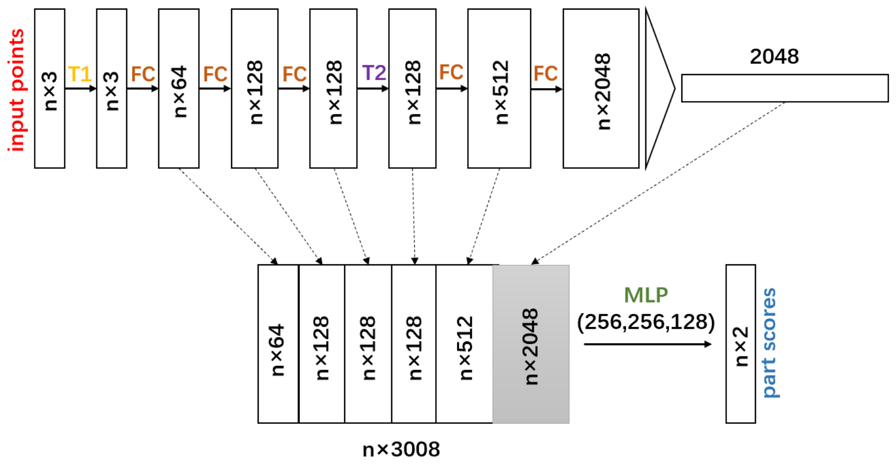


Fig. 1 Network architecture for learning method. n is the number of points in the map. T1 and T2 are transformation networks for input points and features. FC is fully connected layer and MLP is multi-layer perceptron. Dimensions of each layer are also presented in the figure.

learning model.

II. Method

In this section, deep learning method is used to process 3D point clouds and implement LiDAR map compression.Taking Pointnet as the backbone, we propose a framework to implement compression on point cloud maps.

A. Data annotation

We use integer linear programming (ILP) method to solve map selection problem as the data annotation for the following learning step. The formulation of the map compression problem in this paper is same as [4], as follows:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{q}^{\mathsf{T}}\mathbf{x} + \lambda \mathbf{1}^{\mathsf{T}}\boldsymbol{\zeta} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \boldsymbol{\zeta} \geq b\mathbf{1} \\ & \mathbf{x} \in \{0,1\}^{N_m} \\ & \boldsymbol{\zeta} \in \{0, \mathbb{Z}_+\}^{N_p} . \end{aligned} \tag{1}$$

Since the quadratic term brings much computational complexity and small gain in the experiments [4], we discard the quadratic term to simplify the problem, same as [3], [5]. We release the relative source code online [1], which can help understanding the implementation of ILP. Finally, the final optimized result $\mathbf{x}$ from ILP is used to annotate the 3D point clouds in the map.

B. Network structure

PointNet, the backbone in this paper, provides an efficient architecture to process unordered point sets while keeping the permutation invariance and transformation invariance of the point clouds. To implement the classification and segmentation task, the last fully connected layers of the network aggregate all the learned values into the feature descriptor. More importantly, this network can learn a set of optimization functions that select informative points from the input point cloud, which arouses our great interest. This property is consistent with our compression task because key points can preserve environmental characteristics as much as possible.

For our purpose, we take a point cloud as input and find a subset of it. Apparently, the order of points, the rotation and translation transformation of the point cloud are supposed to be independent of the final compression result. From the front part structure, our model can learn a spatial encoding of every point and then aggregate all individual point features to one global feature. After processing the previous point cloud, we make use of extracted global features and local feature vectors to solve the final compression problem. A simple idea is to view the task as a segmentation problem by scoring every point indicated how important the point is. Points with higher scores are supposed to more imporatnt to summerize the features. Give a certain compression ratio, we select the correspond points with higher ranks.

In our stucture, modifications occur in the back end of segmentation network. Firstly, we remove the output result of classification, which is originally used to identify point cloud types. Beacuse in map compression, we don't have to classify the categories of point clouds,. The thing we need to do is to find the key points. Secondly, for the feature descriptor, we reduce the dimensions to 3008, containing features extract from different stage of intermediate process. Consequently, the 16-dimension one-hot vector indicating category of input point clouds is removed from the feature descriptor. The entire structure of our model is shown in Fig. 1. As we

[1]https://github.com/ZJUYH/map_compression

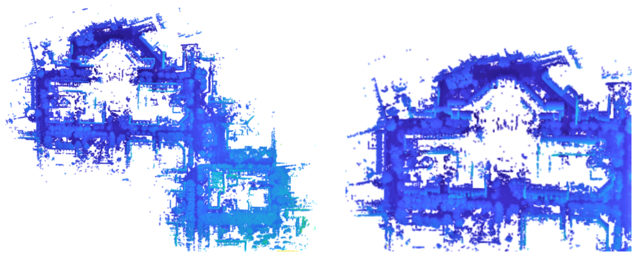

(a) Initial full map (b) Part of the inital map

Fig. 2 We present the 3D point cloud map generated from our dataset, visualized by matlab toolbox.

can see, the transformation networks T1 and T2 are use to eliminate the scale and rotate invairance of the model. Fully connected layers map the lower dimensions feature to higher dimentions space. Collecting these feaatures with different dimensions, we can finally give every point a score through the classifier multi-layer preceptron. Overall, we remove the unnecessary part of the original network and adjust the input part and output part to meet the demands.

C. Applying PointNet

Considering that input requires a standard number of point clouds , we divide the whole map data into several frames with the same number of points. After processing every single frame, we reunited all the sparse point cloud into a complete global map. In this way, we can improve the model's ability of generalization. Detailed descriptions are as follows:

Firstly, to generate points in each frame, we make use of the mobile robot's path points segment the 3D point cloud map data. For every pose in path points, we want to select points around it from the original map. In this way, every point will be allocated one nearest path point. According to the sequence of path points, we rearrange the order of point cloud data.

After that, we take 1024 points in the order of the new data continuously. One single frame consists of 1024 points. In case there are less than 1024 points in one frame, we randomly duplicate several points to add up to the number of 1024. The segment frames are stored respectively.

Then, to reduce the bias caused by different scales, we convert the global coordinates to local coordinates. For every generated frame, we set a virtual center point as the new origin by computing the average of all points' coordinates in the frame. The local coordinates can be get by subtracting the virtual center's global coordinate. Another thing we need to take into consideration is the scale of point cloud may affect the quality of compression. A solution is to normalize all the coordinates. To be more specific, mapping the coordinates into a unit cubic so the absolute value of all coordinates are limited within a range of $[0, 1]$.

Having finished these steps mentioned above, the input data for the deep neural network are prepared. For the training step, we take the point clouds' coordinates and their corresponding labels as the input to the network. Each label is either 1 or 0, indicating preserving the point or not. We regard the annotation of programming method as the groundtruth and minimize the cross entropy between the true labels and predicted values. At the end of every training epoch, we evaluate the new model's performance in order to get the best one with minimal loss. Once the training is finished, the learned model can be applied to map compression in new environments.

III. Experiments

In this section, we use our model to compress the point cloud on a test data set and visualize the output results. Additionally, the similarity between the predict point sets and the ground truth from ILP method are measured. Lastly, we evaluate the localization performance on the compressed maps.

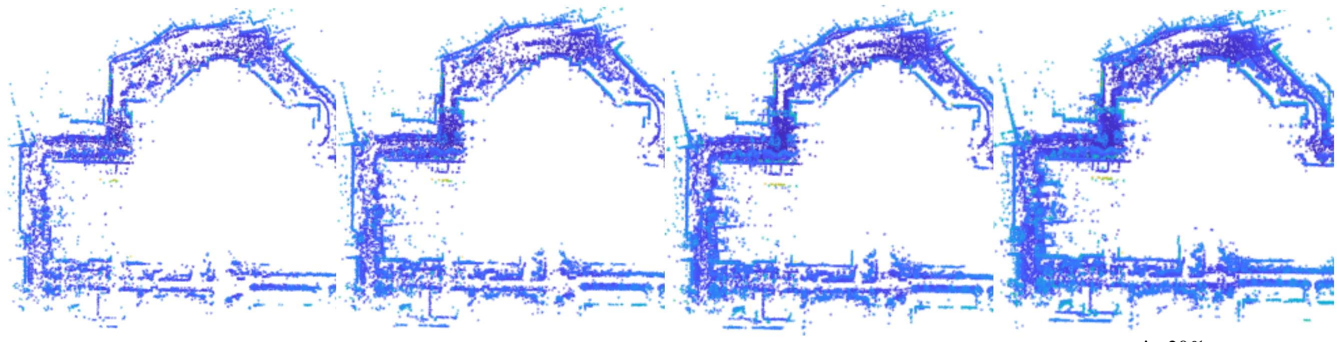
A. Dataset

In the experiment section, we use a multi-session dataset called YQ, which is collected in a university campus by a mobile robot platform [13]. The platform is equipped with Velodyne VLP-16 sensor for robot perception. We generate a global map and robot poses using simultaneous localization and mapping (SLAM). The SLAM Technique [14] with loop closures [11] can construct precise the global 3D LiDAR map, shown in Fig. 2. In this paper, we reorganize the dataset and divide the map into two parts, the first half is used for network training and another is used for testing.

B. Similarity evaluation

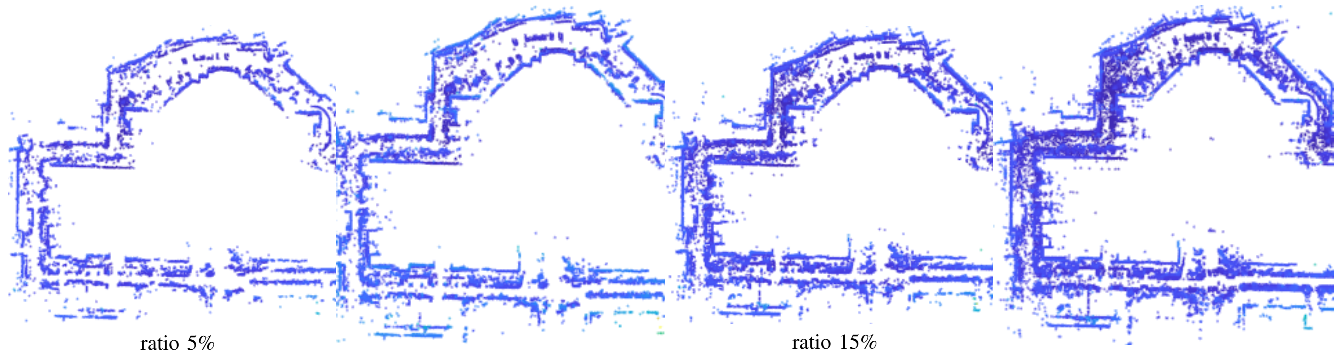
Test results will return a collection of labels, indicating which points should be preserved. With the compression ratio of 20%, 15%, 10% and 5%, four maps compressed by ILP are shown in Fig. 3 (a), the sparse maps generated by our models are shown in Fig. 3 (b) respectively. For example, the 20% map means that about 20% points are reserved after selection using map compression.

We compute the nearest distances between the results from our model and the ILP to evaluate the similarity. In order to obtain the distances, we build KD-tree using points which are reserved after programming. KD-tree can accelerate the process of finding the nearest points and calculating distances. In this way, the smallest distance between our predict results and grountruth can be computed. Counting the ratios of different distances, we can present the distribution of all distances to compare the similarities of the point clouds. As shown in Fig. 4, most of the distances are below $0.1m$, which indicates that the learned results are similar with the compressed maps from ILP.



ratio 5% ratio 10% ratio 15% ratio 20%

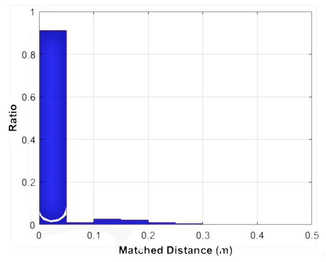
(a) Map compression results from ILP with compression ratios 5%, 10%, 15% and 20% respectively.



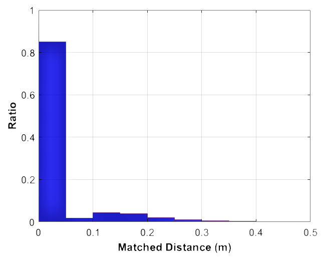
ratio 5% ratio 10% ratio 15% ratio 20%

(b) Map compression results from learning method with compression ratios 5%, 10%, 15% and 20% respectively.

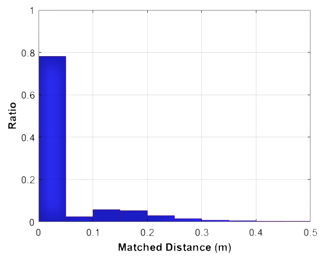
Fig. 3 Four pairs of visualization results of learning method and ILP with different ratios. The results of two methods are partially similar. Learning method obtains a more clear image in fact.



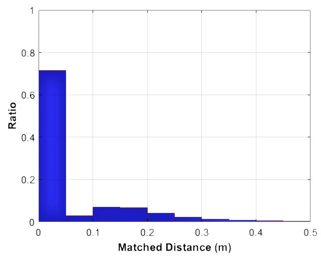
(a) ratio 5%



(b) ratio 10%



(c) ratio 15%



(d) ratio 20%

Fig. 4 Histograms of distances between results of our method and ILP method, with ratios 5%, 10%, 15% and 20% respectively.

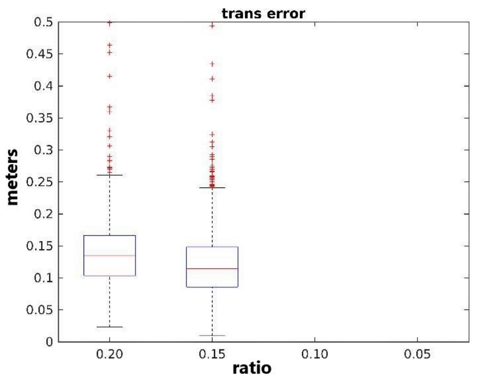
C. *Localization performance*

To evaluate the quality of compressed map, we apply ICP registration to the four different-ratio maps. As we can see, the translation errors are presented in Fig. 5 (a) and the rotation errors in Fig. 5 (b). For the ratio of 20%, the average of translation error is $0.14m$ and rotation error is $0.11°$. When it comes to 15%, the two errors are $0.11m$ and $0.11°$. However, pose tracking failed for ratio 5% and 10%. After all, the sparse maps will lose some important information after map compression. In general, our model can select the most important points from the whole map, because both the translation error and rotate error are acceptable for mobile robot navigation.

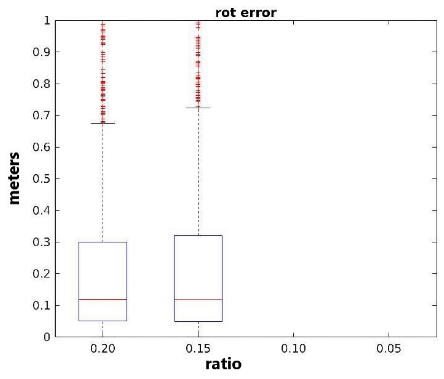
D. *Efficiency*

In order to analyze the efficiency of our learning method, we record the training and testing time cost of the four ratios. A device is used for computation of training and testing, equipped with Intel(R) Xeon(R) CPU E5-2696 v3 2.30GHz and 64GB RAM.

1) Train: The time cost for minimizing loss function is the main part of the time cost, which is evaluated using the work



(a) translation errors



(b) rotation errors

Fig. 5 Localization errors by applying ICP registration.

station. We train the data with 1500 epochs to ensure the convergence of model. The time costs of four compression ratios are similar, with the training time roughly equal to $3.2h$. Because the loss function definition is the same for the four ratios, training time is only depend on the number of epochs and the size of dataset.

2) Test: When the test is performed, we first use the trained model to score all points in the original map, and then select those points whose scores are higher than others. So the time cost for testing is depend on data size and the model size. The time costs are $15.634s$, $16.202s$, $15.597s$, $15.290s$, corresponding to the ratio of 20%, 15%, 10% and 5%. Obviously, the difference of time cost among four ratios is small, because the ratio is not the main reason for time cost and their initial data sizes are the same.

IV. Conclusion

In this paper, we propose a model taking PointNet as backbone to implement map compression. The key point is how to use the path information to segment the map and keep points in different proportions. In addition, it is also critical to adjust the network according to our demand and make it work in our task. In the experiments, we demonstrate the effectiveness of the proposed learning method. The results show that we can compress the point cloud to a certain ratio with comparable localization precision. Finally, we also evaluate the time costs with different compression ratios.

Acknowledgment

This work was supported by the National Key R&D Program of China (2018YFB1309300), Science and Technology Project of Zhejiang Province (2019C01043) and National Nature Science Foundation of China (61903332).

References

[1] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*. IEEE, 2018, pp. 4490–4499.

[2] C. Moreno, Y. Chen, and M. Li, "A dynamic compression technique for streaming kinect-based point cloud data," in *2017 International Conference on Computing, Networking and Communications(ICNC)*. IEEE, 2017, pp. 550–555.

[3] H. Soo Park, Y. Wang, E. Nurvitadhi, J. C. Hoe, Y. Sheikh, and M. Chen, "3d point cloud reduction using mixed-integer quadratic programming," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2013, pp. 229–236.

[4] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart, "Keep it brief: Scalable creation of compressed localization maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2015, pp. 2536–2542.

[5] D. Van Opdenbosch, T. Aykut, N. Alt, and E. Steinbach, "Efficient map compression for collaborative visual slam," in *Proc. IEEE Winter Conf. Applic. Comput. Vis. (WACV)*. IEEE, 2018, pp. 992–1000.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: a versatile and accurate monocular slam system," in *IEEE Transactions on Robotics, Volume: 31, Issue: 5*. IEEE, 2015, pp. 1147–1163.

[7] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3d lidar point clouds," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2798–2805.

[8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.

[9] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 918–927.

[10] M. Angelina Uy and G. Hee Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4470–4479.

[11] H. Yin, Y. Wang, X. Ding, L. Tang, S. Huang, and R. Xiong, "3d lidar-based global localization using siamese neural network," *IEEE Trans. Intell. Transp. Syst.*, 2019.

[12] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 652–660.

[13] L. Tang, Y. Wang, X. Ding, H. Yin, R. Xiong, and S. Huang, "Topological local-metric framework for mobile robots navigation: a long term perspective," *Autonomous Robots*, vol. 43, no. 1, pp. 197–211, 2019.

[14] X. Ding, Y. Wang, H. Yin, L. Tang, and R. Xiong, "Multi-session map construction in outdoor dynamic environment," in *Proc. IEEE Int. Conf. Real-time Computing Robot. (RCAR)*. IEEE, 2018, pp. 384–389.