Efficient 3D LIDAR based loop closing using deep neural network

Huan Yin, Xiaqing Ding, Li Tang, Yue Wang *Member, IEEE*, and Rong Xiong *Member, IEEE*

***Abstract*—Loop closure detection in 3D LIDAR data is an essential but challenging problem in SLAM system. It is important to reduce global inconsistency or re-localize the robot that loses the localization, while is difficult for the lack of prior information. We present a semi-handcrafted representation learning method for LIDAR point cloud using siamese convolution neural network, which states the loop closure detection to a similarity modeling problem. With the learned representation, the similarity between two LIDAR scans is transformed as the Euclidean distance between the representations respectively. Based on it, we furthermore establish kd-tree to accelerate the searching of similar scans. To demonstrate the performance and effectiveness of the proposed method, the KITTI dataset is employed for comparison with other LIDAR loop closure detection methods. The result shows that our method can achieve both higher accuracy and efficiency.**

I. Introduction

Loop closure detection, or place recognition, is a key part in the area of SLAM (simultaneous localization and mapping) for mobile robotics. The drift is unavoidable for mobile robots without loop closure detection or global localization. In order to reduce the accumulated error taken by the odometry, loop closure detection is quite necessary, especially for long-term running robots. By finding loop closures, the robot understands the real topology of the environment [1].

Actually, loop closure is the problem of recognizing a previously-visited place with low or no prior information. Most existing loop closure detection algorithms measure the similarity of the sensor data between places essentially. For vision-based SLAM, sequence of images provide a rich data source, from which it is relatively easier for robots to identify the re-observed places by using developed image processing technologies. Generally, the most popular loop closure algorithm for visual SLAM is based on point features or global descriptors [2]; besides, deep neural network (DNN) has been applied in image recognition, which is recently employed to place recognition [3].

As for laser-based SLAM, numerous laser sensors provide accurate range and intensity information of discrete laser points. Some works achieve the place recognition in 3D LIDAR data based on the segmentation of point clouds and handcrafted descriptors [4] [5]. However, the research on features in point clouds is not as mature as that in vision community. The popularity of neural network in images is extended to laser these days, but representation of LIDAR data is task oriented. For example, Li et al. [6] utilize a fully convolution network techniques in 3D LIDAR data based detection tasks by projecting range scans to 2D images. But the projected 2D images could not be used for loop closure detection directly.

All authors are with the State Key Laboratory of Industrial Control and Technology, Zhejiang University, Hangzhou, P.R. China. Yue Wang is with iPlus Robotics Hangzhou, P.R. China. Yue Wang is the corresponding author wangyue@iipc.zju.edu.cn. Rong Xiong is the co-corresponding author.

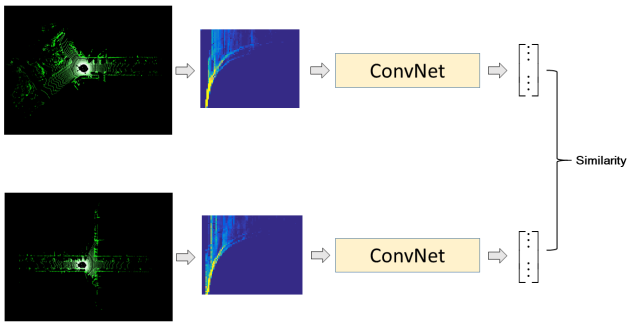


Fig. 1: A brief illustration of our detection process. The 3D point cloud (left) is first transformed to an image-like 2D representation (left two) with rotation invariance. The representation is colored for showing. A pair of such images are put into a symmetric DNN (right two), where each side of it is the same. The network is trained by using the ground truth and 3D LIDAR data. The final representation of point cloud is a feature vector (right) and the similarity is easily achieved in Euclidean space for further loop closure detection.

As one can see, the main obstacle preventing the DNN being employed in loop closure detection is the lack of representation of 3D LIDAR data. In this paper, we use siamese convolution neural network, a popular metric learning tool, to learn a representation which is suitable for loop closure detection.

In summary, we propose a method to transform the LIDAR data to a DNN accessible representation with rotation invariance. This handcrafted representation is further learned by a DNN to identify loop closures. As a result, the output of the network becomes low-dimensional representaion, which also makes the Euclidean metric meaningful. It finally provides an efficient way to detect loop closures in 3D LIDAR data. This paper presents following contributions:

- A statistics on the range information converts a 3D point cloud to one-channel image. The representation is suitable for neural network and is in prior rotation invariance.
- The loop closure detection problem is turned into an identity verification problem. A siamese convolution neural network is introduced to model the similarity between

978-1-5386-3742-5/17/$31.00 © 2017 IEEE

LIDAR scans which can be trained using small amount of samples.

- The Euclidean metric in the final representation is meaningful on which a kd-tree is established to improve the search efficiency in huge data significantly, compared to other distance metrics.

The reminder of this paper is organized as follows: Section II describes the related work of loop closure detection in 3D point clouds and the methods of visual loop closure detection. In Section III, we introduce the details of our methods and distance metrics. We make comparison on KITTI odometry benchmark in Section IV. In Section V, we conclude with a brief discussion on our work.

II. Related Works

A. Visual Place Recognition

A widely-used visual loop closure algorithm is Bag-of-Words (BoW) [7]. The BoW model clusters the visual features descriptors in images [2], and builds the dictionary to achieve place recognition. The SLAM system in [8] builds a database incrementally with index for keyframes, so that querying the database can be done very efficiently. Cummins et al. [9] proposed a method that takes into account of the probabilities of features and is able to work out the probability that two images show the same region of the world.

In recent years, convolution neural networks are used as robust feature extractors for visual place recognition [10]. Chen et al. [11] proposed a system combining the powerful features learned by networks with a spatial and sequential filter. In [12], real world datasets are presented to evaluate the specific challenges in place recognition by using different kinds of networks.

While these techniques above cannot be applied to loop closure detection of 3D LIDAR data directly. Extracting features in 3D point clouds is hard because of the point structure. Point cloud is not suitable to be the input of DNN for the lack of representation.

B. Loop Closing in 3D point clouds

Loop closure detection in 3D LIDAR data remains an open problem in the field of mobile robotics, which needs to be solved with poor or no prior information. A number of papers propose different solutions on loop closure detection from different views in the recent years.

Previous works extract local features from keypoints and perform matches on the basis of these features. Nieto and Ramos [13] used features that capture important geometric and statistical properties of point clouds. The features are used as input to the machine learning algorithm $AdaBoost$s to build a non-linear classifier capable of detecting loop closure from pairs of point clouds. While Bosse and Zlot [5] used a 3D $Gestalt$ descriptor to describe the keypoints extracted from point clouds. The vote scores with keypoints and thresholds are critical for this kind of place recognition.

Some works achieve loop closure detection at the semantic level. $SegMatch$, presented by Dube et al. [4], is a segment based algorithm to perform place recognition with 3D point clouds. Algorithms like this rely on mature point cloud segmentation technologies [14]. Fernandez et al. [15] detected planes in 3D environments to perform place recognition.

Considering laser sensor provides sufficient range information, global feature is useful for detecting loop closures in 3D LIDAR data. Magnusson et al. [16] proposed an appearance based loop detection method using NDT surface representation. Röhling et al. [17] proposed a 1-D histogram to describe the range distribution of a point cloud. A large number of histograms are compared each other using Wasserstein metric to measure the similarity between point clouds. In this paper, we compare our method with this metric distance, but the 1D histogram turns to be the 2D handcrafted representation instead.

III. Method

The whole detection framework is shown in Fig. 1. First, by using the ring structure and range distribution information of LIDAR data, we transform the point cloud to a handcrafted representation with rotation invariance. Secondly, we utilize the pre-trained convolution neural network to achieve a learned low-dimensional representation. The Euclidean distance between the final representations is the similarity for loop closure detection. Based on the low-dimensional final representations, a kd-tree is established to increase the searching efficiency.

A. Rotational Invariant Representation

For a point cloud P obtained by LIDAR sensor, according to the discrete elevation angle in spherical coordinates, it could be divided into N rings S_N^i, where i is the index of rings from 1 to N.

In order to transform the entire 3D point cloud to a 2D representation, we first transform each S_N^i to a 1D histogram H^i. We set a constant bucket count b and a distance range $I = [v_{min}, v_{max}]$, then divide I into subintervals of size:

$$\Delta I_b = \frac{1}{b}\left(v_{max} - v_{min}\right)$$

Each bucket corresponds with one of the disjunct intervals, show as follows:

$$I_b^k = [v_{min} + k \cdot \Delta I_b, v_{min} + (k+1) \cdot \Delta I_b]$$

So all the range values $v(p)$ of points in the ring can find which bucket it belongs to. And the histogram for a ring S_N^i of point cloud P can be written as

$$H_b^i = \left(h_b^0, \cdots, h_b^{b-1}\right)$$

with

$$h_b^k = \frac{1}{\left|S_N^i\right|}\left|\left\{p \in S_N^i : v\left(p\right) \in I_b^k\right\}\right|$$

Theoretically, the number of LIDAR measurements per frame is a constant value. While in practise, some surface types (e.g. glass) may not return any meaningful measurements at all for some reason. The normalization ensures that the histograms remain comparable under these unexpected conditions.

Finally, we stack N histograms H_b^i together in the order of the rings from top to bottom. Then a $N \times b$ one-channel image-like representation $X = (H_b^0, \cdots, H_b^{N-1})$ is produced with a

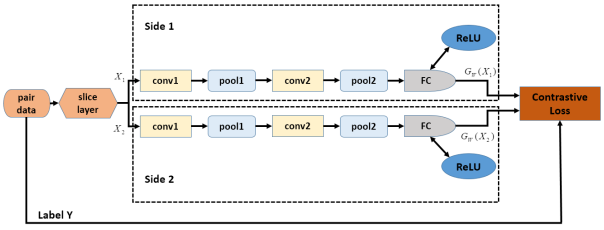


Fig. 2: The framework of our siamese DNN. The two sides of the network are same and share the same parameters. In the frame, *conv* stands for a convolution layer, *pool* stands for a MAX pooling layer and FC stands for a fully convolution layer. For training this network, a pair of images X_1, X_2 and the label Y are needed.

point cloud P by our method. Using the 2D representation, if the robot rotates at the same place, the representation always keeps constant, thus rotational invariant.

One disturbance in place recognition is the moving objects, as it may cause unpredictable changes of range distributions. By utilizing the ring information, the disturbance can be tolerated to some extent, since the moving objects usually occur near the ground, the rings corresponding to higher elevation angles are decoupled form these dynamics. Besides, the range values are reliable in LIDAR data. In a word, that's why we transform the point cloud in this way.

B. Learned Representation

With the handcrafted representation X, we transform the loop closure detection to an identity verification problem, which can be solved with a siamese neural network. We propose a DNN with contrastive loss to learn the final representation and implement it with caffe[1]. The structure of the neural network is shown in Fig. 2.

For a siamese convolution neural network, the most critical and interesting part is the contrastive loss function, proposed by Lecun et al. [18], shows as follows:

$$L(Y, X_1, X_2) = Y\frac{1}{2}(D_W)^2 + (1-Y)\frac{1}{2}max(0, m - D_W)^2$$

In the training step, the loss function needs a pair of samples to calculate the final loss. Let label Y be assigned to a pair X_1 and X_2 : $Y = 1$ if X_1 and X_2 are similar, representing the two places are loop closed in this paper; and $Y = 0$ if X_1 and X_2 are dissimilar, representing the two places are not loop closed, which is the common situation in SLAM system for mobile robotics.

In the contrastive loss function, $m > 0$ is a margin value, which has an influence on the role that dissimilar pairs play in the training step. Actually, the greater the margin value is, the harder the neural network is trained. In this paper, most dissimilar pairs of images cannot be differentiated from similar pairs easily, so we set $m = 8$, a relative high value.

Assume the output of one side of siamese convolution neural network is a d dimensional vector $G_W(X) = \{x_1, \cdots, x_d\}$. Define the parameterized distance function to be learned from the neural network between X_1 and X_2 is $D_W(X_1, X_2)$, which represents the euclidean distance between the outputs of G_W. Show as follows:

$$D_W(X_1, X_2) = \|G_W(X_1) - G_W(X_2)\|_2$$

The purpose of contrastive loss is to decrease the value of D_W for similar pairs and increase that value for dissimilar pairs. In a word, the final representation $G_W(X)$ is actually a low-dimensional vector, which represents the point cloud P and the pose it assigned to.

C. Distance Metrics

Actually, for a pair of input images X_1 and X_2, to achieve the similarity in the test step, we assume all places are loop closed and set the label $Y = 1$, then the contrastive loss is as follows

$$L(X_1, X_2) = \frac{1}{2}(D_W)^2$$

As one can see, if the two places are real loop closed or the final representations are similar, the calculated contrastive loss should be a low value. If not loop closed, it should be a higher value, while the second part of contrastive loss should be lower. So the final representation $G_W(X)$ of point cloud P is meaningful in Euclidean space and we propose the metric W_s as follows:

$$\mathcal{W}(X_1, X_2)_S = D_W(X_1, X_2)$$

We also employ another two classical distance metrics to measure the similarity between X_1 and X_2.

The first one is discrete Wasserstein metric between two histograms, which is also known as $Earth\ Mover's\ Distance$ (EMD). For multi-dimensional histograms, this computation involves the solution of a transportation problem, typically solved with the Hungarian Algorithm, which is quite complex. As our image is made up by 1-D histograms, we define the final metric is the mean value of the Wasserstein metric of N histograms, so the formula of EMD metric is as follows:

$$\mathcal{W}(X_1, X_2)_E = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{b}\sum_{i=1}^{b}\left|\sum_{j=1}^{i} x_{1_{n,j}} - x_{2_{i,j}}\right|$$

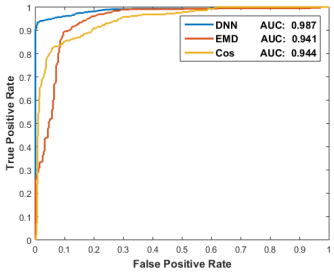
Another distance metric is $Cosine\ Distance$ (Cos). Cosine distance is a simple and common measurement of similarity between two non-zero vectors. It measures the cosine of the angle between them, which could also be used to measure the similarity between images. The definition of the similarity in this case is as follows:

$$\mathcal{W}(X_1, X_2)_C = 1 - \frac{1}{N}\sum_{n=1}^{N}\frac{X_{1_n} \cdot X_{2_n}}{|X_{1_n}|\,|X_{2_n}|}$$

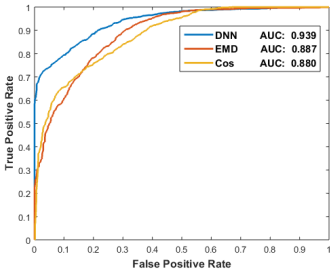
Essentially, the three different methods mentioned in this section measure the distance between a pair of images X_1 and X_2. Both $\mathcal{W}_C$ and $\mathcal{W}_E$ are at the range $[0, 1]$, while $\mathcal{W}_S$ is not because of the definition of the final representation G_W.

However, for all metrics mentioned in this paper, something is common: the higher the metric is, the dissimilar the two

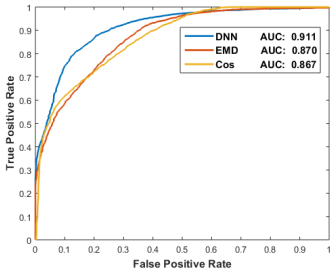
[1] http://caffe.berkeleyvision.org/



(a) $p = 1m$



(b) $p = 2m$



(c) $p = 3m$

Fig. 3: Experiment 1. ROC curves of sequence 08 in KITTI odometry benchmark with different thresholds p and different metrics mentioned in this paper.

images are and vice versa. If the metric is under a certain threshold τ or close to zero, we could conclude that two images or features are similar, and the two poses that the point cloud assigned are loop closed.

D. KD-Tree Search Implementation

If the network is pre-trained, we are able to transform a 3D point cloud P to a d dimensional feature vector $G_W(X)$. The final representation is semi-handcrafted because of the artificial statistics and DNN. We could build a kd-tree T based on the large number of final low-dimensional vectors.

If a new point cloud comes and is transformed to a certain vector through the pre-trained DNN, the feature vector is able to find m closest vectors in the established tree under the threshold τ. Essentially, each vector is associated with a certain and exact pose in the SLAM system, but not all the closest vectors are the loop closed poses. Some closest poses and the newcome pose are continuous on the time axis. After filtering these poses, the remains of closest poses are the real loop closures we want.

The use of kd-tree on the outputs of network takes advantage of our final representation $G_W(X)$. More crucially, kd-tree based search significantly increases the searching efficiency when we look for loop closures in huge data, for most of calculation is unneeded and ignored under the constraint of threshold τ.

IV. Evaluation

KITTI dataset [19] is a public and popular benchmarking dataset, and has the odometry datasets with Velodyne 64 rings LIDAR sensor and the ground truth. In fact, 5 of these sequences (00, 02, 05, 06 and 08) contain the loop closures to evaluate our methods. Among them, sequence 08 is the only one that can test the rotation invariance of loop closure algorithms.

We evaluate the accuracy of our methods in Experiment 1 and Experiment 2 and test the efficiency in Experiment 3. In Experiment 1, we test the loop closure detection with different distance thresholds. In Experiment 2, we set the distance threshold as a constant value, and test the method on different sequences. In Experiment 3, we record the time costs with different distance metrics. In Experiment 4, case study is shown for further analysis.

A. Experiment 1: Different Distance Threshold

As for ground truth provided by KITTI, we consider that two places are the same if their Euclidean distance is below p. So different threshold p determines the requirement for loop closure and the samples we prepared for network training. The higher threshold p is, the more training and testing sample we will have. Some samples are repeated so the down-sampling is realized. The details of sample size is in TABLE I (Size: positive sample size / negative sample size).

In Experiment 1, we test different values of p on sequence 08 and other sequences provide the training samples. To evaluate the result of training, we select the Receiver Operating Characteristic (ROC) curves [17], which plots the False Positive Rate against the True Positive Rate, and calculate the Area Under Curve (AUC) for the correlative curve. We compare the training result of neural network with other two metrics: EMD metric and Cos metric. (see Fig. 3)

As the results show, obviously, our method presents a better result than the other metrics under different distance threshold p. For loop closure detection, the larger p is, the greater the difficulties, and AUC value of each metric becomes lower. Besides, sequence 08 is the only one that the robot turns 180° when the loop closure appears, which proves that our representation is also rotational invariant for loop closure detection in 3D LIDAR data.

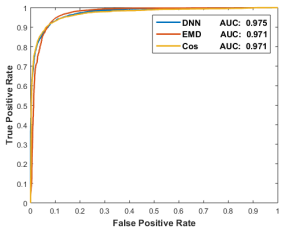
TABLE I: Sample Size and AUC of sequence 08

Sequence	p	Size	DNN	EMD	Cos
08	1 m	284 / 552	**0.987**	0.941	0.944
08	2 m	1114 / 1959	**0.939**	0.887	0.880
08	3 m	1994 / 4141	**0.911**	0.870	0.867

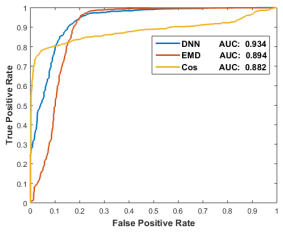
B. Experiment 2: Leave One Out

In Experiment 2, we set threshold $p = 3m$ as a constant value for different sequences in KITTI odometry benchmark. Each time, we test on one sequence and use training samples of the others. We compute and plot similarity matrices to evaluate our results for loop closure detection. A similarity matrix is able to reflect the detection results directly.

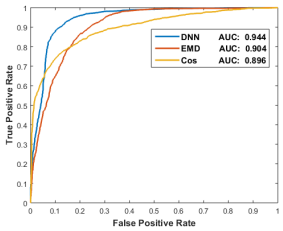
In order to compute a similarity matrix, we need to determine threshold τ to decide whether two places are the same. We select the widely-used F_β score to achieve threshold τ.



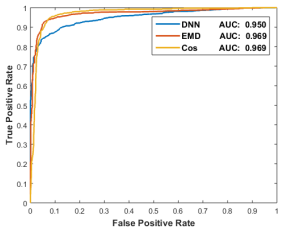
(a) ROC curve of sequence 00



(b) ROC curve of sequence 02

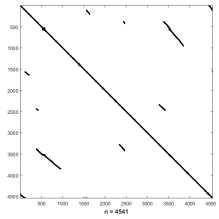


(c) ROC curve of sequence 05

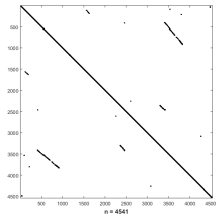


(d) ROC curve of sequence 06

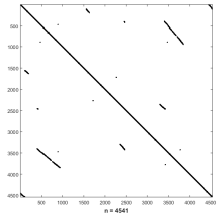
Fig. 4: Experiment 2. ROC curves of sequences in KITTI odometry benchmark with a certain threshold $p = 3m$ and different metrics mentioned in this paper.



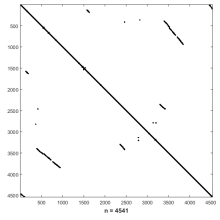
(a) Ground Truth



(b) Our Algorithm



(c) EMD Metric



(d) Cos Metric

Fig. 5: Experiment 2. Similarity matrices of sequence 00 in KITTI odometry benchmark with different metrics mentioned in this paper. Threshold τ is determined by F_β for different metric: (b) $\tau = 1.300$ (c) $\tau = 0.261$ (d) $\tau = 0.046$.

Using the ground poses provided by KITTI, we are able to determine True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) with different thresholds τ. We select threshold τ with the best threshold F_β.

The formula of F_β shows as follow:

$$F_1 = \frac{(1+\beta^2) \cdot TP}{(1+\beta^2) \cdot TP + \beta^2 \cdot FN + FP}$$

However, the ratio of positive sample and negative sample in training and testing samples is much higher than in the real circumstance, where most places are not similar after all. To compute the similarity matrices, we select a relative high value for β to balance samples we select and the real environment.

We plot ROC curves of different sequences in KITTI odometry benchmark in Fig. 4 and compute the similarity matrices of sequence 00 in Fig. 5. The details of sample size and detection results for each seqeunce is in TABLE II. Our method performs a better result at most of the sequences.

TABLE II: Sample Size and Detection Results of Sequences

Sequence	p	Size	DNN	EMD	Cos
00	3 m	3777 / 7362	**0.975**	0.971	0.971
02	3 m	842 / 1551	**0.934**	0.894	0.882
05	3 m	1810 / 3809	**0.944**	0.904	0.896
06	3 m	789 / 1692	0.950	**0.969**	**0.969**

C. Experiment 3: Efficiency

Our evaluation of computational time is based on the similarity matrix. Actually, since the similarity matrix is symmetric, we only need to compute half of the similarity matrix. And if a new point cloud comes, the loop closure detection of it is based on the previous clouds, so our computing process is from the first row to the last with the robot moves on the road.

In Experiment 3, we first achieve the final representation of the newcome point cloud with the pre-trained DNN, and build the kd-tree based on the previous vectorized representations. Then, we find possible loop closures of it with a certain threshold τ. The time cost of our method to compute the similarity matrix of a sequence is from the first transformation step to the final matrix. For the other two metrics, EMD and Cos, to compute the similarity matrix, we have to compute the similarity between each handcrafted image and judge the loop closures with threshold τ.

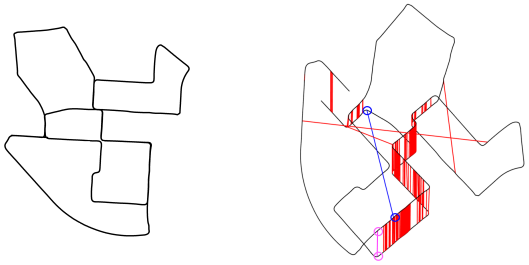
We record some relative time costs for sequence 00, as shown in TABLE III. Actually, for a pair of point clouds, our method takes a longer time to achieve the similarity than the other two metrics. But the difference of time cost for different metrics to build a similarity matrix is large. The advantage of transforming the point clouds to the final representation is obvious. The time cost of computing similarity matrix by using our algorithm is much less than the others. It is unpractical for EMD and Cos metrics to build kd-tree, while our representations are meaningful in Euclidean space, and the kd-tree accelerates the searching speed significantly.

TABLE III: Time Costs of sequence 00

Methods	Single Similarity	Similarity Matrix
Our Method	0.016 s	**81s s**
EMD Metric	0.0046 s	>26338 s
Cos Metric	**0.0001 s**	>6330 s

D. Experiment 4: Case Study

In Experiment 4, we give examples of correct and failed detections of loop closure using our method. Actually, the dynamic objects moving around the robot is unavoidable



(a) sequence 00 (Bird View) (b) Loop Closure Detection

Fig. 6: Experiment 4. (a) A bird view of the trajectory of sequence 00. (b) Loop Closure detections are presented with with certain thresholds $\tau = 1.300$ and $p = 3m$. It is convenient to show the results with a slowing change on Z-axis of the trajectory. Red lines stand for the detection results. We select one false positive (blue line) for case study, and one false negative (magenta line) example is also presented.

in urban areas, which may cause false negative detections using our methods. Besides, a few scenes are hard to be distinguished from each other if we only utilize the range distribution information of point clouds, which may cause false positive results. (See Fig. 6)

For example, in Fig. 7, scenes like these confuse us a lot. Scene 1 and Scene 2 are the point clouds captured the poses marked blue in Fig. 6. The two poses are far away and point clouds are totally different from bird view, but are considered to be loop closed, because of the range distribution are similar, and the final vectorized representations are closed by our methods.

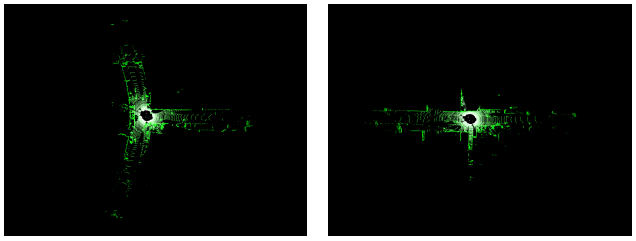
V. Conclusion

The loop closure detection in 3D LIDAR data requires a fast and high precision in real time SLAM system for mobile robots. Our approach shows that it is possible and convenient to use convolution neural network to transform the point cloud to a rotational invariant representation. The transformation relies on the range distribution information and ring structure of 3D LIDAR captured by Velodyne sensor. A siamese DNN is also involved in the transformation. Besides, based on the final representations, in our algorithm, a kd-tree is proposed to increase the search efficiency.

The dynamic changes in SLAM system cause lots of trouble in our methods. So it is supposed to filter the dynamic objects for loop closure detection in the future work. The intensity information and the normal vectors of point clouds can also be utilized for building handcrafted representations.

Acknowledgment

This work was supported by the National Nature Science Foundation of China (Grant No. U1609210 and Grand No. 61473258).



(a) Scene 1 (b) Scene 2

Fig. 7: Experiment 4. Two point clouds from bird view in sequence 00 of KITTI odometry benchmark. Scene 1 and Scene 2 are marked with blue circles in Fig.6.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard, "Simultaneous localization and mapping: Present, future, and the robust-perception age," vol. 32, no. 6, 2016.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.

[4] R. Dub, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based loop-closure for 3d point clouds," 2016.

[5] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3d lidar datasets," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 2677–2684.

[6] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," in *Robotics: Science and Systems*, 2016.

[7] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3921–3926.

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2017.

[9] M. J. Cummins and P. M. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[10] S. Lowry, N. Snderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2016.

[11] Z. Chen, O. Lam, A. Jacobson, and M. Milford, "Convolutional neural network-based place recognition," *Computer Science*, 2014.

[12] N. Sunderhauf, S. Shirazi, F. Dayoub, and B. Upcroft, "On the performance of convnet features for place recognition," in *Ieee/rsj International Conference on Intelligent Robots and Systems*, 2015, pp. 4297–4304.

[13] J. I. Nieto and F. T. Ramos, "Learning to close loops from range data," *International Journal of Robotics Research*, vol. 30, no. 14, pp. 1728–1754, 2011.

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

[15] E. Fernandez-Moral, W. Mayol-Cuevas, V. Arevalo, and J. Gonzalez-Jimenez, "Fast place recognition with plane-based maps," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 2719–2724.

[16] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, "Appearance-based loop detection from 3d laser data using the normal distributions transform," *Journal of Field Robotics*, vol. 26, no. 11-12, p. 892?914, 2009.

[17] T. Röhling, J. Mack, and D. Schulz, "A fast histogram-based similarity measure for detecting loop closures in 3-d lidar data," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 736–741.

[18] R. Hadsell, S. Chopra, and Y. Lecun, "Dimensionality reduction by learning an invariant mapping," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1735–1742.

[19] R. Urtasun, P. Lenz, and A. Geiger, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.