Loop Closure Detection in SLAM by Combining Visual CNN Features and Submaps

Hao Qin, May Huang
AI Research Laboratory
International Technological University
San Jose, CA, USA
e-mail: awww797877@gmail.com

Jian Cao, Xing Zhang
School of Software and Microelectronics
Peking University
Beijing, China
e-mail: caojian@ss.pku.edu.cn

***Abstract*—Using simultaneous localization and mapping (SLAM) with 2D LIDAR is an efficient approach for robots to build a floor plan, but it is sensitive to the environment. For improving the accuracy, we match LIDAR data with sub-maps. Furthermore, we convert LIDAR data to images and merge with camera data for image matching. Combining the two approaches, we achieve robust and accurate loop closure detection. The descriptors generated by CNN model will be used as features to image matching for accuracy improvement.**

Keywords-SLAM; loop closure detection; CNN; submaps; robotic

I. Introduction

SLAM contains the concurrent construction of a representation of the environment, and the estimation of the location and orientation of the robot. In the last decade, SLAM has made great amazing progress [1]. Depending on the sensor, SLAM technology is divided into LIDAR SLAM and visual SLAM. Because the data acquired by LIDAR sensors is relatively more accurate, the SLAM technique with LIDAR sensors can be more easily applied to the actual scene. Up to present, LIDAR SLAM with wheel encoders and IMU sensors enabled high-precision and robust results at a low computational cost in a simple environment. On the other hand, LIDAR SLAM is also easily misleading by the environment due to the sparseness of LIDAR data. In addition, the dynamic environment is also a big challenge for SLAM.

Earlier classic SLAM technology is EKF [2] SLAM which does not solve the problem of accumulated errors. In order to solve the problem of accumulation of errors, graph-based SLAM technology was proposed by Lu and Milios in 1997[3], in which loop closure detection is one of the core technologies. However, because of the computational complexity of the graph-based SLAM optimization algorithm, it took several years to become popular. Graph-based SLAM can be divided into two parts: the front-end and the back-end. The function of the front-end is acquiring and processing sensor data with feature extraction and data association. The duty of the back-end is performing inference with the front-end on the processed data. The first estimation of location and map takes place in the front-end. To reduce the accumulative errors of estimation by the front-end, the back-end uses loop closure detection to find the re-visited place, which is combined with optimization algorithms to reduce accumulative errors.

Computing the similarities of probability grids between each laser scan is the classical loop closure detection algorithm with 2D LIDAR [4][5]. If the environment is dynamic, open, or not flat, it is a huge challenge for robots to find the re-visited place through the matching of probability grids with sparse 2D LIDAR data. In view of this situation, we proposed a novel approach which combines 2D LIDAR and camera. As the robot moves, the program builds submaps with a few consecutive scans in real time. At each step, the robot will quickly and accurately find the loop closure by comparing new scan and submaps. To exclude as much as possibly the environmental disturbances mentioned above, we introduced image matching. As the robot moves, the program stores a keyframe of images for loop closure detection when a scan is inserted into a submap. Firstly, we turn the LIDAR data into images and generate descriptors using the CNN algorithm in conjunction with the camera images. Then we train this network to build a stable and recognizable model with pose and orientation distance. When the program founds that the current scan and a few submaps matched, it sub-depth searches for the exact location of the match, and matches the current image with the corresponding keyframes to remove the error matches and find the exact match location. Finally, we optimize the result of SLAM based on the loop closure.

II. Related Work

LIDAR sensors are characterized by high speed, low computational complexity, and high accuracy. In ordinary circumstances, 2D LIDAR SLAM has enough accuracy and robustness[1]. Position recognition and loop closure detection are extremely crucial for self-driving robot in unknown environment. Graph-based SLAM with 2D LIDAR uses probability grids to find the similarities between each scan. Compared with the traditional matching algorithm, CNN algorithm has more and more obvious advantages in image matching.

A. 2D LIDAR SLAM

Robots use 2D LIDAR to build robust and accurate map for navigation indoor. C. Cadena et al[1] have discussed the application and development of 2D LIDAR SLAM. Robots acquire LIDAR data and pre-process it, then estimate the status of robot by matching and create a grid map. Some algorithms assume Kalman filters(KF) as their main algorithm to predict their phase estimation[6]. Particle filters(PF) is another popular algorithm to apply localization for robots[7]. This method has the advantage of representing uncertainty through multimodel distributions and dealing with non-Gaussian noise. However, it still requires complicated calculation because of plenty of particles. To solve the cumulative error problem, the idea of optimization is more and more important in SLAM. Currently, graph-based SLAM is the best way to achieve this aim.

In the past decade, a large number of SLAM algorithms have been developed for applications. LagoSLAM [8] uses linear optimizer to reduce errors quickly. HectorSLAM[9] optimizes laser scan points with built map and predicts the probability grids at each cell. KartoSLAM[10] is also a graph-based model, which design its solver with noniterative Cholesky matrix decomposition for sparse linear systems. The ROS package Gmapping[11] is the most prevalent SLAM package for mapping, which makes use of RBPF algorithm to improve the accuracy of location.

B. *Loop Closure Detection for Graph-Based SLAM*

Successful Loop closure detection is the premise of graph optimization. The most prevalent loop closure detection algorithm is Bag-of-Visual-Words (BoVW)[1] which is used for visual SLAM based on hand-crafted descriptors. For LIDAR SLAM, the most common practice is to search for matching candidate grids in each scan, which is computationally complex[12]. Another way is to quickly find the loop by matching the features extracted from the laser scan[13]. M. Himstedt[14] used histogram for position matching which is fast but not robust enough. W. Hess[5] proposed a scan-to-submap approach to improve the precision of localization and mapping and speed up its operation with sparse laser scan, this method is sensitive to the environment because of the sparseness of the laser scan.

C. *CNN-Based Descriptors*

It is one of the key technologies of visual SLAM to find the correlation between the images through feature descriptors. Prior to the prevalence of CNN, the way of image matching relied entirely on manual definition, and people sought to find better image features for image matching by mining different feature of the image. CNN present breakthrough in the last years[15] which has proved that it could generate more robust and abundant descriptors than traditional algorithms. With linear and nonlinear neural networks this method could automatically generate appropriate descriptors for matching. A series of achievements convinced us to use CNN to generate descriptors for loop closure detection. Instead of solely focusing on pairs for image matching, recent work on deep learning for learning feature embeddings examines the use of triplets of samples shows the triplets of samples have outweighed performs [16].

III. Descriptors Generation and Matching

A. *CNN Architecture*

With the inspiration by[17][16], we use a new method to generate descriptors with triplets. Learning representations with triplets will produce better results than learning with pairs using the same network. We extract three features from {a, p, n} at a time, where a is the anchor, p is a feature that is close but different from a, and n is a feature that is completely different from a. Our aim is to better distinguish between images in different scenes by training CNN networks to extract stable distinguishable features. In our case, the specific approach is to define a loss function which is as formulation (1)(2)(3). The loss value is made as small as possible by training, that is, the difference between a and p is as small as possible and the difference between a and n is as large as possible. Where μ is a margin parameter.

$$\gamma_+ = \|f(a) - f(p)\|_2 \tag{1}$$

$$\gamma_- = \|f(a) - f(n)\|_2 \tag{2}$$

$$\sigma(\gamma_+,\ \gamma_-) = \max(0, \mu\ +\ \gamma_+ -\ \gamma_-) \tag{3}$$

The architecture to generate descriptors is briefly summarized in Table I. To ensure the efficiency of feature descriptor generation, we use shallow convolution neural network. For activation layer, Tanh is used as activation function because we can acquire robust descriptors through its quick convergence and high saturability. Instead of using regular max pooling layer in this architecture, adaptive max pooling function is used to improve the balance and robustness of this network, this function could adaptively re-weights the contributions of each pixel based on their observed losses.

B. *Fine-Tuning for Local Data*

At first, we train our model with Photo Tourism [18] which provides large number of patches for learning-based descriptors generators. After this work, we have a pre-trained model which could be fine-tuned with our recorded dataset which come from MIT Stata Center Data Set [19]. For 2D LIDAR data, we convert them to images in Cartesian scan with formulation (4)(5). In which r means the range between the LIDAR sensor and some obstacles. For θ the laser scan data in ROS topic is separated with increase angle, so we can acquire the angle θ of each range. In order to locate the converted points to the field near the image center we add 16 in the formulation. By entering these converted images and camera images into the pre-trained model, increasing the learning rate of the last layer of this network, and reducing the learning rates of other layers, we have acquired a new model that accommodates the data we collect.

$$x = r * \cos\theta + 16 \tag{4}$$

$$y = r * \sin\theta + 16 \tag{5}$$

In order to label our own training data, we use their location and orientation to distinguish. As formulation (6)(7), we calculate their matching value by calculating their transform distance and rotation distance, the matching value is close to 0 or 1. Where 1 means the label of each match between any two images. x_1, y_1 and θ_1 is the pose of the first image, and x_2, y_2 and θ_2 is the pose of the second image. t_m is the maximum value of distance, and θ_m is the maximum value of rotation. That's mean

$$0 \leq \frac{\sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2} + |\theta_1 - \theta_2|}{t_m + \theta_m} \leq 1$$

TABLE I. THE MODEL ARCHITECTURE TO GENERATE DESCRIPTORS

Layer	Conv2d	Tanh	Pool2d	Conv2d	Tanh	Linear	Tanh
Kernel	(7, 7)			(6, 6)			
Out	(26, 26, 32)	(26, 26, 32)	(13, 13, 32)	(8, 8, 64)	(8, 8, 64)	128	128

$$l = \left(\frac{1}{1 + e^{-\frac{\sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2} + |\theta_1 - \theta_2|}{t_m + \theta_m}}} - 0.5 \right) * 2 \tag{6}$$

$$l = \begin{cases} 1, \ l < 0.1 \\ -1, l \ \geq 0.1 \end{cases} \tag{7}$$

Because we have two sets of images of LIDAR images and camera images, there are two sets of feature descriptors. we fuse the two sets of descriptors to calculate their distance with Euclidean distance. The whole model as shown in fig 1. First, a shallow CNN is trained to produce robust feature descriptors, where two models are generated, and then the feature descriptors are fused for loop closure detection.

C. *Loop Closure Detection*

Inspired by [5] submaps are used to quickly and accurately complete the task of loop closure detection. As the robot moves, we need to constantly insert the latest scan into the newest submap or build a new submap. As for the conditions for creating a new submap, we need to match the latest submap and the latest scan at each step. If the match is below the threshold, we think it is time to create a new submap, which is different from W. Hess' method, He specifies how many scans each submap should contain.

While inserting a new scan into the nearest submap or creating a new submap, the most recent image is saved as a keyframe for image-based loop detection.

Because a scan is smaller than a submap, when we match a scan with a submap, we coincide the origin of the scan with the origin of the submap, and then compute their match. Whenever there is a match between a scan and a submap, we treat this scan as a sliding window. Unlike ordinary sliding window algorithm, this window also needs to rotate. In fact, this match will go through 3 layers for loop, the time complexity is higher.

Considering the regularity of the matching score of this window in different location of a submap, and in combination with the above submap creation method, we will find that this rule is roughly as shown in Figure 2, and the matching score will peak at a certain position, While the farther away from this position will match the lower score. Depending on the conditions under which a new submap is created, the matching score's vertex position usually falls within the target submap. We just need to find this vertex position, without having to fully experience the 3-layer for loop. The DFS algorithm is used to find this vertex position as shown in algorithm 1.

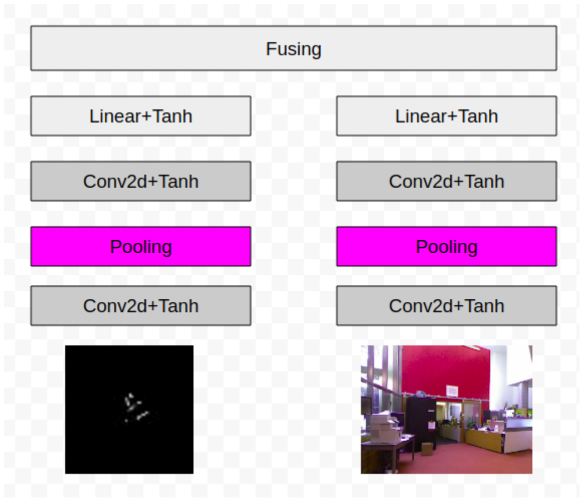


Figure 1. The model to generate fused descriptors with LIDAR images and RGB images.

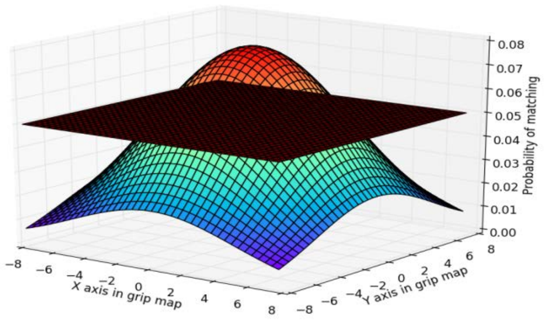


Figure 2. The general distribution rule of matching score between scan and submap.

In the meantime, the image at the current moment is used to match the keyframes of several nearby scans, and when the scores of these matches are below the threshold, we do not consider the location to be a loop closure position. If both of these methods treat this position as a loop closure point, we put the state of the current position into the graph as a node and then optimize it with the Newton's method in optimization in G2O.

IV. Experiments and Results

In this section, we present some results of our approach computed from some open data packets. We demonstrate that our approach works well in general personal computers.

A. Descriptors Generation

PyTorch is used to build the CNN architecture for generating feature descriptors. PyTorch is a very powerful Python framework for deep learning that easily loads data, defines model architecture, saves trained models, and debugs. The training data come from Photo Tourism which contains large groups patches with their matching information, as shown in figure. After training, we use HPatches[20] to evaluate our matching model, there, images in each sequence are sorted by increasing transformation, resulting in increased detector noise with easy, hard and tough groups. As shown in table II, our modified model from TConv has better mean Average Precision(mAP) than SIFT and DeepCompare[17]. It consumes a little more time than ORB, but is less than SIFT.

TABLE II. mAP in HPathces Data Set

Model	SIFT	DeepComp	TConv
Easy	0.453178	0.467729	0.501201
Hard	0.193015	0.224921	0.281239
Tough	0.0862911	0.129381	0.141921
Mean	0.244162	0.274010	0.308120

We test our approach with MIT Stata Center Data Set. These data sets are provided as ROS bag. In order to test our model and show the test results, we read this data from ROS topic and write them into the corresponding files. 5 poses are chosen randomly, and they are used to calculate pose distance and descriptor distance, results are shown in table III, these two distances have the same trend, and pose distance is about twice as much as descriptor distance. That's mean, when the descriptor distance is small enough, we can think the pose distance is small enough, in other words, the test two places is the same place, and one loop closure is detected.

TABLE III. Compare between Pose Distance and Descriptor Distance

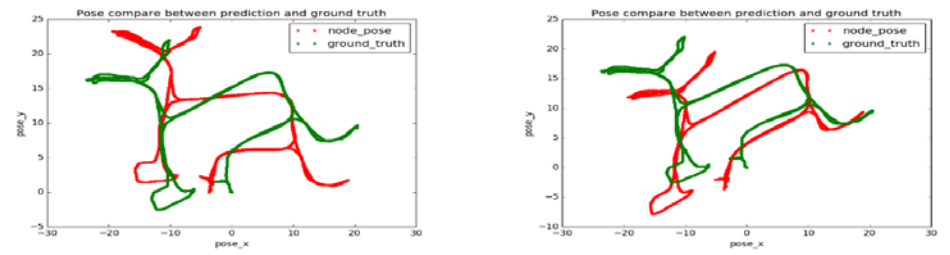
Pair	(0,1)	(0,2)	(0, 3)	(0, 4)	(1, 2)	(1, 3)	(1, 4)	(2, 3)	(2, 4)	(3, 4)
Pose Distance	0.00	17.92	22.15	28.58	17.85	22.14	28.59	4.22	12.47	9.71
Descriptor Distance	1.33	8.83	10.44	11.37	8.84	10.59	11.48	2.95	5.11	3.99

B. Pose Evaluation

In this section, a complete SLAM project containing our loop closure detection algorithm was run for testing. In order to evaluate the results of our approach, we still use MIT Stata Center Data Set as our test data set, which provides 2d ground truth (x, y, θ) of each frame in the frequency of 20 HZ. The data is about 667 seconds, containing more than 10000 frames, for the sake of drawing and analysis, we acquire these frames by sampling in the frequency of 4 HZ.

The results are shown in fig 3 and fig 4, we compare our approach with KartoSLAM in squared distances between estimation and ground truth. As you can see, the transform and rotation could be a little more accurate than KartoSLAM. When our robot is running with our approach, the high efficiency of this algorithm guarantees it keep real-time.

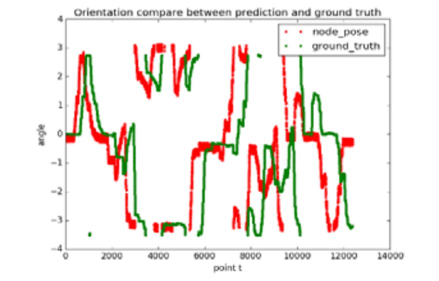
```
    Algorithm 1: Loop Closure Detection in Submaps
            score_best ← score_init
                stack ← score_init
        while stack is not empty do:
            pop point from the stack
            find the next point in x, y, θ.
            if score_current > score_best
then
                    update match
                    score_best ←
score_current
            else
                    stack ← score_current
```



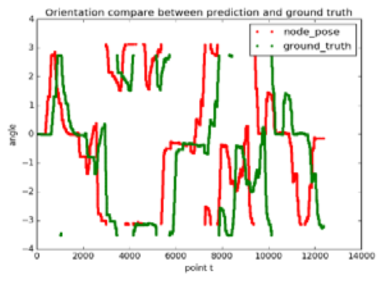
(a) Transform compare between KartoSLAM prediction and ground truth

(b) Transform compare between our prediction and ground truth

Figure 3. Transform compare between prediction and ground truth.



(a) Rotation compare between KartoSLAM prediction and ground truth



(b) Rotation compare between our prediction and ground truth

Figure 4. Rotation compare between prediction and ground truth.

V. CONCLUSIONS

In this paper, we propose a new loop closure detection algorithm that combines both camera and LIDAR sensors. In the image matching session, we proposed an improved learning based descriptors generator with triplets and an adaptive max pooling layer to generate more robust and accurate descriptors. In order to improve the accuracy of loop detection, we first use the sliding window to find the exact matching position of scan in the corresponding submap, and then use image matching to remove errors and increase the optimization conditions. Experimental results show that our method is helpful to improve the accuracy of SLAM algorithm.

ACKNOWLEDGMENT

The authors would like to thank International Technological University (ITU) for its support on AI Research Lab and this research. We would also like to show our gratitude to Dr. Huang, Dr. Cao for their assistance.

REFERENCES

[1] C. Cadena et al., Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age, in IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309-1332, Dec. 2016.

[2] Julier, S.J.; Uhlmann, J.K. (2004). Unscented filtering and nonlinear estimation. Proceedings of the IEEE: 401–422. doi:10.1109/jproc.2003.823141.

[3] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. Autonomous Robots, 4:333–349, 1997.

[4] S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. Intl. J. of Robotics Research, 25(5/6):403-430, 2005.

[5] W. Hess, D. Kohler, H. Rapp, and D. Andor, Real-Time Loop Closure in 2D LIDAR SLAM, in Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016. pp. 1271–1278.

[6] S. Thrun., W. Burgard, D. Fox., Probabilistic Robotics, MIT Press,2005.

[7] D. Fox, W. Burgard, F. Dellaert, S. Thrun. Monte carlo localization: Efficient position estimation for mobile robotsProc. AAAI-99, Orlando, FL (1999).

[8] L. Carlone, R. Aragues, J.A. Castellanos, and B. Bona. A linear approximation for graph-based simultaneous localization and mapping, In Proc. of the Int. Conf. Robotics: Science and Systems, 2011.

[9] S. Kohlbrecher, J. Meyer, O. Von Stryk, U. Klingauf. A Flexible and Scalable SLAM System with Full 3D Motion Estimation, In the Int. Symp. on Safety, Security and Rescue Robotics (SSRR), Nov. 2011.

[10] R. Vincent, B. Limketkai, M. Eriksen. Comparison of indoor robot localization techniques in the absence of GPS, In Proc. of SPIE: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV of Defense, Security, and Sensing Symposium, April 2010.

[11] G. Grisetti, C. Stachniss, W. Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters, In Trans. On Robotics , 23(1), Feb. 2007.

[12] E. Olson, M3RSM: Many-to-many multi-resolution scan matching, in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), June 2015.

[13] Li, J.; Zhong, R.; Hu, Q.; Ai, M. Feature-Based Laser Scan Matching and Its Application for Indoor Mapping. Sensors 2016, 16, 1265.

[14] M. Himstedt, J. Frost, S. Hellbach, H.-J. Böhme, and E. Maehle, Large scale place recognition in 2D LIDAR scans using geometrical landmark relations, in Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. IEEE, 2014, pp. 5030–5035..

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems (NIPS), pages 1097–1105, 2012.

[16] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. Proc. BMVC, 2016.

[17] Sergey Zagoruyko, Nikos Komodakis. Learning to Compare Image Patches via Convolutional Neural Networks. In Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, June 2015.

[18] S. Winder and M. Brown. Learning local image descriptors. In Proc. CVPR, 2007.

[19] Fallon, Maurice F.; Johannsson, Hordur; Kaess, Michael; Leonard, John J. The MIT Stata Center Dataset, IJRR Dataset Paper, under review.

[20] Vassileios Balntas*, Karel Lenc*, Andrea Vedaldi and Krystian Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors, CVPR 2017.