Have I been here before? Learning to Close the Loop with LiDAR Data in Graph-Based SLAM

Tim-Lukas Habich[1], Marvin Stuede[1], Mathieu Labbé[2] and Svenja Spindeldreier[1]

***Abstract*—This work presents an extension of graph-based SLAM methods to exploit the potential of 3D laser scans for loop detection. Every high-dimensional point cloud is replaced by a compact global descriptor, whereby a trained detector decides whether a loop exists. Searching for loops is performed locally in a variable space to consider the odometry drift. Since closing a wrong loop has fatal consequences, an extensive verification is performed before acceptance. The proposed algorithm is implemented as an extension of the widely used state-of-the-art library RTAB-Map, and several experiments show the improvement: During SLAM with a mobile service robot in changing indoor and outdoor campus environments, our approach improves RTAB-Map regarding total number of closed loops. Especially in the presence of significant environmental changes, which typically lead to failure, localization becomes possible by our extension. Experiments with a car in traffic (KITTI benchmark) show the general applicability of our approach. These results are comparable to the state-of-the-art LiDAR method LOAM. The developed ROS package is freely available.**

I. Introduction

One key challenge in developing autonomous mobile robots is the SLAM (Simultaneous Localization And Mapping) problem [1]. Graph-based solutions are often based on a static environment, which is unrealistic: objects move, seasons influence the appearance of the surroundings and – depending on the lighting – images of the same scene differ. For long-term autonomy in dynamic environments two main requirements must be met. First, the computational complexity of graph optimization, the back-end, must be limited. Second, the process has to be robust against changes in the environment. The processing of sensor data, the front-end, must be developed in such a way that, despite changes, already visited places are recognized again. Answering the question *Have I been here before?* belongs to the most important tasks during SLAM, since in presence of a loop closure, the odometry drift can be corrected retrospectively and the map quality can be improved.

Loop detection approaches can be classified according to the sensor technology used. Due to the high information density in an image and the multitude of effective techniques, visual methods are widely used. Where such methods are stable for static environments, the illumination change can already lead to failure within a few hours. Actively illuminated sensors such as LiDAR (Light Detection And Ranging) scanners are robust in such situations, as generated point clouds are illumination-invariant. This work deals with the combination of the two approaches to realize robust mapping and localization. The basis for this is the widespread library RTAB-Map (Real-Time Appearance-Based Mapping) [2]. Although the method can be used with a LiDAR, the data is only marginally utilized due to the simplified method of loop search within a constant radius using 2D Iterative Closest Point (ICP), making an extension suitable. Fig. 1 illustrates our approach. Loops are searched based on scan descriptors, and within a variable radius r depending on the largest eigenvalue $\lambda_{\max}$ of the position covariance matrix to take the odometry drift into account. If a loop exists, the respective scans are registered and the graph is optimized with the calculated transformation $\boldsymbol{T}$.

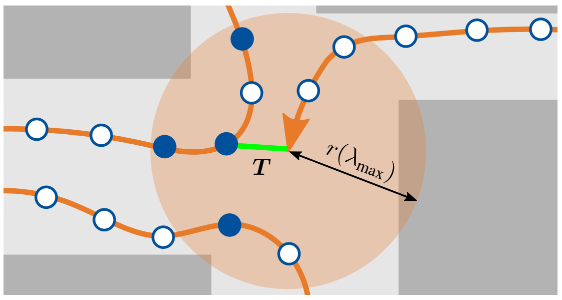

Fig. 1. Loops with discrete nodes (blue points) are searched for in a variable radius of the current position (orange arrow). The search space depends on the position uncertainty. Due to the use of different memories not all nodes in the radius are used as loop candidates.

The paper is structured as follows. Sec. II presents related work. Based on this, Sec. III describes our contributions:

- further development of a loop detection method to enable the necessary scan registration,
- introduction of several loop verification steps to reject false positives,
- elaboration of a robust, open-source ROS package[1] to close loops with LiDAR data during graph-based SLAM.

This is followed by various validations in Sec. IV and conclusions in Sec. V.

II. Related Work

Approaches to mapping changing environments are manifold. Examples are filtering moving objects and assuming a

[1]T.-L. Habich, M. Stuede and S. Spindeldreier are with the Leibniz University Hannover, Institute of Mechatronic Systems, D-30823 Garbsen, Germany, tim-lukas.habich@imes.uni-hannover.de
[2]M. Labbé is with the Université de Sherbrooke, Interdisciplinary Institute of Technological Innovation, Sherbrooke, Québec, J1K 0A5 Canada

[1]https://github.com/MarvinStuede/cmr_lidarloop

static environment [3], modeling the dynamics in the frequency domain [4] and integrating new data to adapt the map [5]. The continuous adaptation to changes is also possible by removing old nodes at loop closure [6] or by representing the environment in several time frames simultaneously [7].

RTAB-Map [2] is suitable for large-scale and long-term online SLAM in changing environments due to its memory management [8]: The bounded Working Memory (WM) ensures a bounded demand on computing which is realized by transferring nodes to a database. A buffer of constant size ensures that recently created nodes are not considered for loop detection, since the odometry drift cannot be significantly corrected. Whereas most state-of-the-art methods are either visual or LiDAR-based, RTAB-Map supports both. This key advantage enables the use of many sensor configurations, comparison of results and easy integration into different systems. The availability in the ROS framework, multi-session operation and a high accuracy [2] further promote its deployment.

RTAB-Map's LiDAR-based *proximity detection* module was developed for challenging situations, such as a significant illumination change, in which the visual loop detection is not promising. However, the used 2D ICP algorithm is a simplified method for loop detection. The high amount of data within the three-dimensional point cloud is not used and a fast loop search with several hundred potential pairs is not possible. Beyond that, this module only searches within a constant radius for loops which is problematic, since the error of the estimated position increases due to odometry drift. The search space should thus be permanently adjusted. Further, a local search is useless at the beginning of multi-session operation, because the position in the old map is unknown. Instead, a global search must be performed here so that a link can be established between the two sessions, and the robot can locate itself in the old map. The potential of scans to close loops is thus not exploited, making an extension necessary.

To detect loops with LiDAR data, there are a variety of methods which are based on histograms to reduce the scan dimension. Representing a scanning area as a piecewise continuous function using Normal Distribution Transform (NDT) and detecting loops by matching feature histograms realizes a high accuracy [9]. The time to create such histograms is an important property which is a disadvantage of NDT. Instead, by using simpler histrogram types such as the range or height of each point, a fast histogram generation [10] is possible with results comparable to NDT. The azimuthal and radial division of the scan into bins and use of the maximum height of the points in each bin also allows a fast computation of a global descriptor called Scan Context [11]. However, besides height and range, there is further information in the raw scan, which can be obtained with low computational effort. In [12], each point cloud is described with 41 rotationally invariant geometric features, so that a trained classifier decides whether a loop exists. The extensive scan description with small, rapidly computed features and fast prediction make this approach suitable for efficient loop search. However, the method does neither consider the scan registration nor the important loop verification to reject false positives. We thus extend the method of Granström and Schön [12] with these essential elements and integrate it into RTAB-Map, so that a robust loop search with laser scans is realized.

III. SLAM with Learned LiDAR Loop Detector

The developed LiDAR extension is presented in Fig. 2 and described in this section.

A. *Loop Classification*

Loop detection is a binary problem: either the robot has already visited the current location in the past or not. To solve this problem based on LiDAR data, a classifier is trained. A point cloud $\boldsymbol{P}_i = \{\boldsymbol{p}_i^k\}_{k=1}^N$ from node i in the graph contains N points $\boldsymbol{p}_i^k \in \mathbb{R}^3$ representing the environment. The rotation invariant classifier presented in [12] is used in our work and is briefly introduced in the following.

Depending on the LiDAR, the number of points differs. For example, the scanner of the mobile robot Sobi used for evaluation (see Sec. IV) generates $N \approx 45{,}000$ points per cloud, requiring a dimensionality reduction for comparison. For this, each cloud is described by global features [12]

$$\boldsymbol{f}_i = (\boldsymbol{f}_i^{\mathrm{I}}, \boldsymbol{f}_i^{\mathrm{II}})^{\mathrm{T}}. \tag{1}$$

Features $\boldsymbol{f}_i^{\mathrm{I}}$ of type I map the point cloud to a real number, e.g. simple geometric quantities such as the mean range or the range's standard deviation. More complex geometric quantities, such as the center and radius of a sphere fitted to the point cloud, are also calculated. A total of 32 features of type I are computed. In addition, there are nine features $\boldsymbol{f}_i^{\mathrm{II}}$ of type II. These are range histograms with nine different container sizes $b_1, \ldots, b_9$. For each histogram, starting at the sensor origin, the scan is divided into annular container of constant width and the points lying in each container are counted. Each feature is a vector whose dimension depends on the container size. Before feature computation, every scan is processed in a way that all points with range r_k greater than a maximum range $r_{\max}$ are moved in the direction of the sensor origin so that $r_k \leq r_{\max}$ applies for all points. This limitation allows range histograms of one container size to have the same dimension in any case. A total of 41 features are calculated consisting of 843 real numbers during SLAM with Sobi (see Sec. IV). Thus, the dimension of any scan is significantly reduced.

If during mapping and localization the features are calculated and stored for each node, the descriptor $\boldsymbol{f}_{\mathrm{c}}$ of the current position can be efficiently compared to descriptors of map nodes $\boldsymbol{f}_i$ with respect to a possible loop closure. An AdaBoost Classifier [13] [14] predicts if a loop is present and the entity ${}^{i}p_{\mathrm{c}}$ indicates the probability that the corresponding scans are from the same location. The input of the classifier is generated from the respective feature vectors by comparing them appropriately [12]. Type I features are compared via the absolute difference and histogram comparison for features of type II is done using Pearson's correlation coefficient.

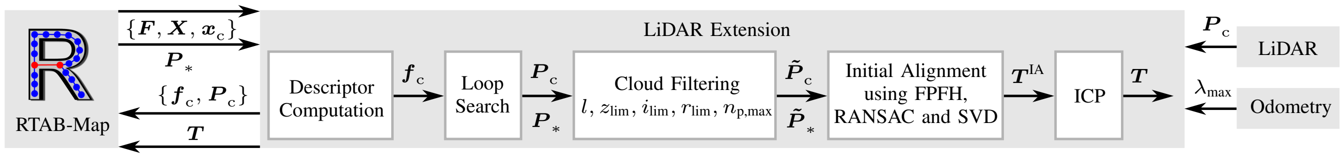
RTAB-Map
$\{\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{x}_\mathrm{c}\}$
$\boldsymbol{P}_*$
$\{\boldsymbol{f}_\mathrm{c}, \boldsymbol{P}_\mathrm{c}\}$
$\boldsymbol{T}$
LiDAR Extension
Descriptor Computation
$\boldsymbol{f}_\mathrm{c}$
Loop Search
$\boldsymbol{P}_\mathrm{c}$
$\boldsymbol{P}_*$
Cloud Filtering
l, z_lim, i_lim, r_lim, $n_\mathrm{p,max}$
$\tilde{\boldsymbol{P}}_\mathrm{c}$
$\tilde{\boldsymbol{P}}_*$
Initial Alignment using FPFH, RANSAC and SVD
$\boldsymbol{T}^\mathrm{IA}$
ICP
$\boldsymbol{T}$
$\boldsymbol{P}_\mathrm{c}$
LiDAR
λ_max
Odometry

Fig. 2. LiDAR extension of RTAB-Map. Loops are permanently searched for using the scan descriptors. In case of a detected loop, the registration of the scans takes place in a parallel thread. After the point clouds have been pre-processed, first a rough registration and then a refinement takes place.

AdaBoost classification [15] belongs to the class of boosting algorithms and uses T weak learners, which together form a strong classifier. In each learning round, the data is reweighted depending on the current prediction error. Incorrectly classified and thus difficult cases are prioritized higher, so that the number of training rounds T significantly influences the accuracy. For $T \geq 50$ weak classifiers, the error was observed to stop decreasing for the used LiDAR descriptors [12]. Accordingly, in the present work $T = 50$ rounds are performed for learning.

B. Scan Registration

In case of a detection, a further step is necessary to close a loop: the scan registration. This aims at determining the homogeneous transformation $\boldsymbol{T}$ between the current point cloud $\boldsymbol{P}_\mathrm{c}$ and the point cloud $\boldsymbol{P}_*$, with which a loop was detected. The transformation is then used to add a link between the two nodes in the map in order to take the loop into account. The entire registration process is illustrated in Fig. 2 and is implemented with the point cloud library [16].

Registering raw point clouds is computationally expensive and the scans also contain regions that prevent successful registration. Examples are points on the ground or regions generated by reflection of the laser beam on glass fronts. To encounter this, the following filters are executed consecutively:

- voxel grid filter with voxel side length l,
- height filter to remove points with height lower than z_lim,
- intensity filter to remove points with intensity lower than i_lim and therefore points which laser beams were reflected,
- range filter to remove points with range larger than r_lim and therefore to remove scan regions which are far away and do not describe the environment in sufficient detail,
- random downsampling filter to randomly remove points until the point cloud consists of $n_\mathrm{p,max}$ points.

The pre-processed scans $\tilde{\boldsymbol{P}}_\mathrm{c}$, $\tilde{\boldsymbol{P}}_*$ used for registration can have a large translational and rotational offset. Using a local method like ICP [17] would not be expedient due to a convergence into a local optimum. Consequently, a rough registration to compute an initial alignment takes place first according to [18]. Thereby, Fast Point Feature Histograms (FPFH) [19] are used as robust multidimensional features that describe local geometry around a point and only persistent features are used [20]. The latter increases robustness, since only unique regions are taken into account. With these keypoints, the correspondences can then be estimated by performing a nearest neighbor search in the feature space.

Because of a partial overlap, not every keypoint has a match and a large number of incorrect correspondences would be possible. These outliers negatively influence the registration and must be rejected. By means of outlier rejection based on RANdom SAmple Consensus (RANSAC) [21] a transformation is computed over a multiplicity of iterations with different subsets of correspondences. Only those correspondences are classified as correct whose Euclidean distance, after application of the respective transformation, is smaller than a defined threshold. The outliers that are present during the transformation with the most inliers are rejected.

With the filtered correspondences the initial alignment can be completed using Singular Value Decomposition (SVD) [22]. The transformation $\boldsymbol{T}^\mathrm{IA}$ of this global method is the initial guess for subsequent refinement, so that the output of the ICP registration $\boldsymbol{T}$ is used to correct the odometry drift. Since adding a wrong transformation has drastic consequences for the SLAM procedure, some verification criteria must be met. First, both processed scans must consist of at least $n_\mathrm{p,min}$ points so that enough data is included. Second, a transformation is only accepted if at least n_inliers correspondences after outlier rejection are present which ensures that enough matching areas are found. Third, the transformation has a translational offset of at most t_max. This condition increases the robustness since a larger translational offset leads to a smaller scan overlap, and thus the registration can become less accurate.

C. Extending the RTAB-Map Framework

The presented loop detection and scan registration are integrated as an extension into RTAB-Map which is illustrated in Fig. 2. For each scan, the corresponding descriptor is calculated and sent to RTAB-Map. Saving the scans with associated features enables compatibility with multi-session operation. The extension is continuously supplied with current map data which contains required information of the graph such as the position $\boldsymbol{x}_\mathrm{c}$ of the current node. Further, the extension receives the positions $\boldsymbol{X} = \{\boldsymbol{x}_i | \forall i\}$ of all map nodes and associated descriptors $\boldsymbol{F} = \{\boldsymbol{f}_i | \forall i\}$, which is the basis for the actual task to detect loops.

When receiving new map data, a loop search is done with the trained detector. Similar to *proximity detection*, the current node is compared with other WM nodes within a certain radius only. For example, if the robot is located in a corridor and the entire WM is checked, a loop detection with a node in a similar corridor at a completely different location in the building would be possible due to similar scans. Adding such a loop would be fatal for the SLAM process, so the local restriction is useful for increasing robustness.

In determining the search space, the present work differs from RTAB-Map's approach. For large loops, a large odometry drift is present and therefore, a large uncertainty in the estimated position. The search carried out in *proximity detection* in a constant radius is problematic here because the estimated position is highly inaccurate and loops are searched for in the wrong map area. Accordingly, we follow a different philosophy: the use of a variable search space depending on the position inaccuracy. The search radius

$$r(\lambda_{\text{max}}) = r_{\text{min}} + \beta g_{\text{max}}(\lambda_{\text{max}}) \tag{2}$$

consists of two parts. In the constant radius r_{min}, loops are searched for if the current position can be assumed to be without errors. This is the case at the start of the process as well as at loop closures, where the inaccuracy is corrected. In all other cases, the estimated position is subject to error, whereby we assume the odometry error to be steadily increasing. This property is taken into account by the second component of (2). Thereby, $g_{\text{max}}(\lambda_{\text{max}}) = 2\sqrt{5.991\lambda_{\text{max}}}$ is the length of the longest major axis of the 95% confidence ellipse of the two-dimensional position. This quantity can be obtained from the largest eigenvalue λ_{max} of the covariance matrix, for which the odometry data is processed by the extension. The parameter β serves as a scaling factor. Fig. 3 visualizes the variable search space. The approach enables to include nodes close to the exact position in the loop search despite a large position inaccuracy.

As already mentioned, adding a wrong loop is fatal for the integrity of the map. Hence, further precautions are taken to reduce the number of false positives. Only when ${}^{i}p_{\text{c}} > p_{\text{min}}$ applies to the predicted loop probability ${}^{i}p_{\text{c}}$, the pair is treated as a potential loop pair. Further information on the adjustment of the threshold p_{min}, i.e. the fine tuning of precision and recall, is presented in Sec. IV-A.

If the search was successful, a final verification is performed with the node of highest loop probability ${}^{*}p_{\text{c}}$ at the location i^{*}. The current node is compared with the surrounding nodes of the potential loop candidate. If at least one loop with the current node is detected in the immediate neighborhood consisting of $2n_{\text{v}}$ verification nodes, this candidate is accepted.

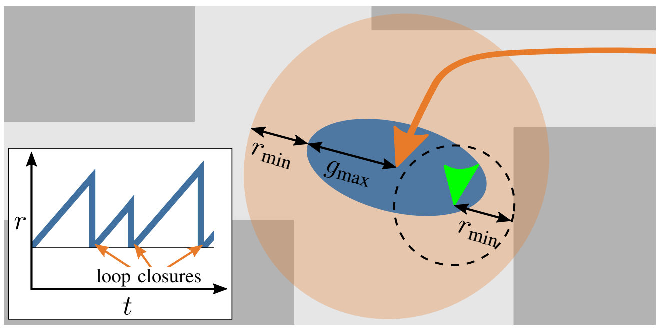

Fig. 3. Despite the difference between the actual (green arrow) and estimated position (orange arrow), the search space (orange circle) contains the desired circular area with the radius r_{min} around the actual position. The schematic course shows the increase of the radius with increasing time t due to odometry error. If a loop closure occurs, the uncertainty is reset.

In each case n_{v} nodes with $i < i^{*}$ and n_{v} nodes with $i > i^{*}$ are considered. For the strictest variant ($n_{\text{v}} = 1$) there must be either another loop with the node created before or after the loop candidate. Otherwise, the loop is rejected and the search starts again when new map data is received. If the verification is successful, the scan registration takes place in a parallel thread and the link between the involved nodes is added with the accepted transformation $\boldsymbol{T}$. After further verification in RTAB-Map, the loop can be closed by graph optimization.

Two final adjustments remain to be made: First, loops must be searched robustly even in multi-session operation, which is impossible with the previous approach of a local search if the position is unknown. Especially at the beginning of a new session the relative position in the old map is in general not given – the initial state problem. A position-independent search must be carried out here, so that nodes of the entire WM are examined. As alternative local restriction during multi-session, the size n_{ms} is introduced which is the number of consecutive loop candidates that must lie within a radius r_{ms}. Only one loop is accepted if several loop pairs are from the same place. This type of loop search is done in multi-session mode for the first n_{start} accepted loops, after which it has to be checked if the localization in the old map was successful. The criterion for this is the ratio $\alpha = \frac{n_{\text{local}}}{n_{\text{WM}}}$, where n_{local} is the number of nodes in the local map and n_{WM} is the number of all WM nodes. If this ratio is smaller than a defined threshold α_{min}, there are few nodes in the local map, so the relative positions of many WM nodes to the robot are unknown. In this case, the entire WM should continue to be searched for loops, since localization is not yet satisfactory. Otherwise, the relative position of many WM nodes with regard to the robot is known and the local search in the variable radius can be started. To meet online requirements, the second adjustment is to limit the number of nodes used in the search. For this, $n_{\text{n,max}}$ nodes are randomly selected from the possible candidates. Due to the fast detector, hundreds of nodes can be used.

IV. Experimental Validation

The evaluation is divided into three parts: First, a loop detector is trained and tested in an independent environment (see Sec. IV-A). Second, multi-session experiments were performed with this detector under challenging conditions (see Sec. IV-B). The mobile service robot Sobi [23] is used for both validations, which is a ROS-based information and guidance system equipped with a differential drive base (Neobotix MP-500), two RGBD cameras (Intel Realsense D435, front and back) and a 3D LiDAR (Velodyne VLP-16). An extended Kalman filter [24] is used to fuse the wheel odometry with the IMU data (XSens MTi-30). Third, the general applicability of our approach is shown with the widely used KITTI [25] dataset in Sec. IV-C.

A. Detector Performance

To train the detector, indoor and outdoor data consisting of descriptors with corresponding coordinates was collected on an university campus. There are large distances between objects in outdoor areas of the campus, so $r_{\text{max}} = 40\text{m}$ is chosen. With the container sizes $b_1, \ldots, b_9$ suggested in [12], each feature vector consists of 843 entries. There is no ground truth position, so the optimized poses were obtained from RTAB-Map to generate the most accurate positions of the 1248 nodes with a path length of 697m.

Comparing each node with itself and with all others gives 779,376 pairs. As the distance at which the detector should treat a pair as positive, we choose 3m. With this distinction, the set is divided into 11,458 positive and 767,918 negative pairs. Training with this set would not be expedient, as the number of pairs for the classes is clearly unbalanced. As in [12], a random subset of negative pairs is used, where we select 11,458 pairs to obtain the same amount of data for both classes. However, it is noticeable that the performance varies considerably depending on the subset, since half of the data is randomly taken from a large set. Thus, an optimization was carried out, whereby 50 detectors were trained with different subsets. For comparison, common criteria consisting of detection rate D and false alarm rate FA are used:

$$D = \frac{\text{\# Positive data pairs classified as positive}}{\text{\# Positive data pairs}}, \quad (3)$$

$$FA = \frac{\text{\# Negative data pairs classified as positive}}{\text{\# Negative data pairs}}. \quad (4)$$

An ideal detector would detect every loop ($D = 100\%$) and would also not detect any loop incorrectly ($FA = 0\%$). Since our extension still verifies every possible loop pair, $FA < 1\%$ is set as target. The decisive parameter for this is the introduced threshold p_{min}, which is fine tuned with a training-independent test dataset consisting of 181 nodes. The path of length 72m is a loop in an indoor hall, which is illustrated in Fig. 4a). For each of the 50 detectors, p_{min} was incrementally increased so that the requirement $FA < 1\%$ is met. This process can be visualized with the ROC (Robot Operating Curve) of the best performing detector in Fig. 4b). Increasing the threshold results in a more restrictive detector, which is less erroneous but also detects fewer loops. The best classifier ($D = 47.3\%$, $FA = 0.8\%$, $p_{\text{min}} = 52.4\%$) detects approximately half of all loops with only few false positives in the unknown environment.

The test results can be illustrated in the form of two matrices. The classification matrix Fig. 4c), in which each node pair is examined by the detector for a loop, is compared with the distance matrix Fig. 4d). The latter represents the underlying ground truth, since all pairs with a distance less than 3m are treated as loop and the rest as negative pairs. The orange area on the diagonal represents the successive nodes that are close enough together. The off-diagonal orange area is the loop that is driven, since the robot returned to an old position at a later point in time and the nodes concerned are a short distance apart. For the trained detector, these matrices are roughly similar and all detected loops are located in areas where the pairs are not far away from each other. However, pairs with a distance minimally greater than the selected 3m are classified as positive as well. This does not pose a problem, as these loop closures are also desirable.

B. SLAM under Challenging Conditions

The presented method is furthermore validated in an experiment under challenging conditions. Since the introduced extension shall support the loop detection, we compare RTAB-Map extended with our work against the default operation. The extension is termed LL (LiDAR Loop). First, all required sensor data was recorded while driving. The evaluation was performed afterwards on a 2.60GHz Intel Core i7-4720HQ CPU with 8 GB of RAM running Linux, so that both methods had the same data available. Predicting with the detector takes on average 2ms and RTAB-Map's time threshold was set to 0.75s. Depending on the cloud sizes, registration of a loop pair takes 2s–10s. Due to the use of parallel threads, the longer time of registration is unproblematic. The parameters of our extension must be adjusted depending on the environment and robot and were chosen as follows:

- $r_{\text{min}} = t_{\text{max}} = 3\text{m} \rightarrow$ loop distance during training,
- $n_{\text{p,max}} = 10,000$, $n_{\text{p,min}} = 7,000$, $n_{\text{inliers}} = 1,000$, $i_{\text{lim}} = 5$, $z_{\text{lim}} = 0.3\text{m}$, $r_{\text{lim}} = 30\text{m}$, $l = 0.03\text{m} \rightarrow$ fast and robust scan registration,
- $n_{\text{n,max}} = 200 \rightarrow$ many nodes during loop search and computation time less than RTAB-Map's time threshold,
- $\beta = 0.25 \rightarrow$ reduces the increase of the search radius,
- $n_{\text{v}} = 1 \rightarrow$ strict verification,
- $\alpha_{\text{min}} = 0.5, n_{\text{ms}} = 2, r_{\text{ms}} = 5\text{m}, n_{\text{start}} = 3 \rightarrow$ global loop search until localization is satisfactory.

According to odometry, the path for the outdoor map has a length of 423m and consists of 769 nodes. When mapping with RTAB-Map, there were 6 visual loop closures, compared to 15 loops closed with the extended method. The potential of the extension is already apparent in single session operation, as significantly more loops were detected. However, there are no major differences in terms of map quality. The main reason is the precise odometry used, which means that with such a path, a few loops are sufficient for an acceptable map quality. Due to similar maps, Fig. 5 presents only the map generated by RTAB-Map+LL with the corresponding nodes in orange.

The added value becomes evident when taking the usual task of a mobile robot into account: *map the environment and extend the map on another day*. The central requirement for such a multi-session operation is a successful localization in the old map. For this, the paths shown in Fig. 5 in light and dark blue were driven on another day at a different time. It was checked individually whether a localization succeeded or failed in the map generated with the respective method. Due to the dynamic environment, RTAB-Map did not detect any loops and, therefore, a localization for all of the five paths failed, preventing to extend the map. In contrast, with RTAB-Map+LL, a localization in the map of the old session was possible in four out of five paths, and the mapping could be continued. Loops were detected in the third path, but localization failed due to a wrong transformation matrix. The participating scans are from a location on the campus

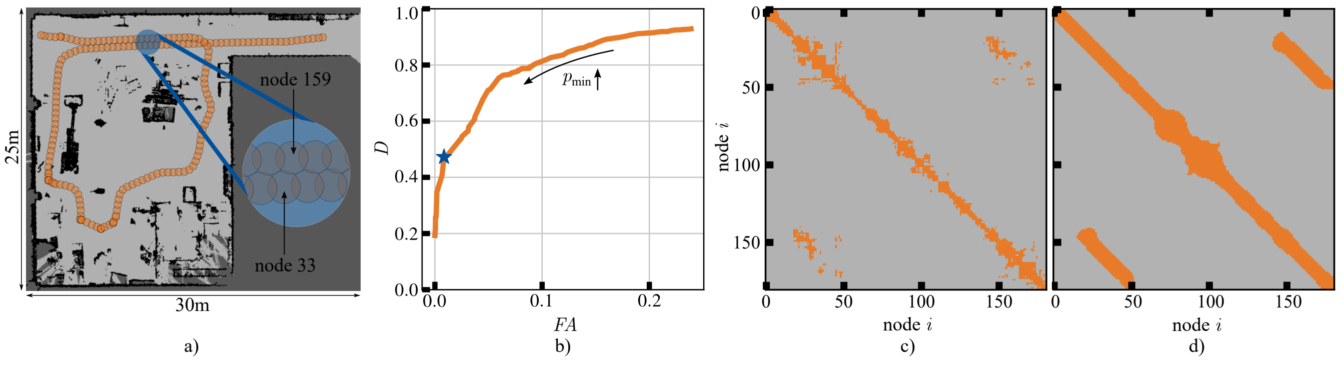

Fig. 4. Detector performance for training-independent test environment a). ROC b) shows performances of best detector at different thresholds $p_{\min}$. Detectors can be compared using their detection rates at the desired boundary of $FA < 1\%$ (blue star). Corresponding classification matrix c) compared to distance matrix d) visualizes performance of this detector at $p_{\min} = 52.4\%$ with $D = 47.3\%$ and $FA = 0.8\%$. Orange areas represent loops and grey areas no loops.

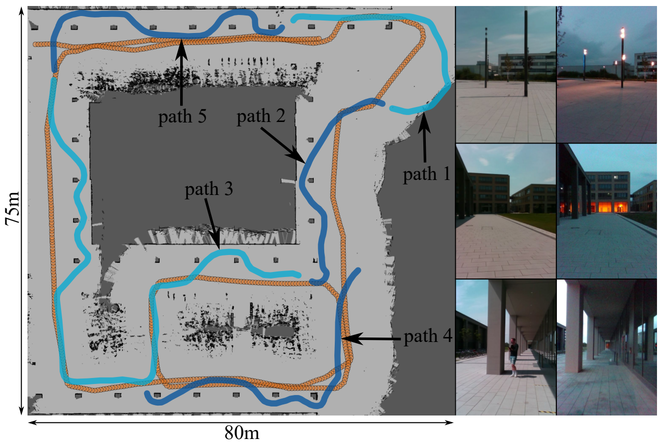

Fig. 5. Outdoor map with associated nodes (orange circles), five paths (light and dark blue) that were driven on another day and images of three exemplary loop pairs detected by RTAB-Map+LL.

which has few descriptive elements except for a multitude of repetitive pillars. Due to the repetitiveness, pillars of one scan were registered with different pillars of the other scan, so that a translational error led to a mapping failure.

To illustrate the environmental dynamics, Fig. 5 shows pictures of loop pairs successfully added in RTAB-Map+LL operation. A change in sunlight, switched on lights, a person in the picture and a different field of view are the reasons why the visual loop search of RTAB-Map failed. Only by observing the three-dimensional surrounding structure, a localization is still possible. Even if the robot travels to an already visited location with a different orientation and therefore the image comparison returns no match, loop detection succeeds due to the 360° view and the rotation invariant features used.

Similar experiments were conducted in two challenging indoor environments: an entrance hall with many glass fronts and an office environment with corridors. According to odometry, the mapping paths have a length of 196m and 299m respectively. Analogous to the outdoor experiment, three paths were driven at a different time in each environment. RTAB-Map operation realizes successful localization in two out of six paths, whereby in the respective areas, a change in sunlight has a small influence due to a large amount of artificial light. Using RTAB-Map+LL, a successful localization is present in all six cases. Combining the results of the indoor and outdoor validation, RTAB-Map realizes a successful localization in two out of eleven cases. In contrast to that, with the LiDAR extension it was possible to localize in ten out of eleven paths.

C. *KITTI Dataset*

Finally, our extension was evaluated with the widely used KITTI [25] odometry benchmark. We continued to use the detector trained in Sec. IV-A, so that the training data recorded in indoor and outdoor environments on our campus differs significantly from the test data acquired in road traffic. The sequences containing loops are mapped with our proposed extension using the RTAB-Map parameters for KITTI [2]. Odometry is calculated with the front stereo camera image sequences Frame-To-Map (F2M) wise. The following parameters of our package have been adjusted for the car and road traffic: $r_{\min} = 7.5\text{m}$, $r_{\max} = 50\text{m}$, $n_{\text{v}} = 3$, $z_{\text{lim}} = 1\text{m}$, $l = 0.2\text{m}$, $t_{\max} = 10\text{m}$. Since the intensities of the point clouds were not given, we set $i_{\text{lim}} = 0$.

Our extended version of RTAB-Map is compared to the LiDAR-based approach LOAM [26] which is currently ranked #2 on KITTI's odometry leaderboard. Table I shows the results of the two methods using the average translational error as the performance metric. The number of visual and the number of additional LiDAR-based loop closures in RTAB-Map+LL operation are also included. The results show that the performance of our extension is comparable to LOAM. In four out of seven sequences, the error is lower with our method than with LOAM. For sequence 09, RTAB-Map+LL performs significantly worse. This can be explained by the small number of loop closures. Moreover, the potential of the LiDAR extension can be seen in sequence 08. Since the camera is oriented in the opposite direction when traversing back, preventing the detection of visual loops. By using LiDAR data, 30 loops can still be closed in this case.

Table I
RESULTS FOR THE KITTI ODOMETRY DATASET. AVERAGE TRANSLATIONAL ERROR IN %.

Sequence	00	02	05	06	07	08	09
RTAB-Map+LL	**0.70**	0.99	**0.39**	**0.60**	**0.58**	1.18	1.39
# visual loops	156	51	88	54	18	0	3
# LiDAR loops	126	5	72	38	18	30	1
LOAM [26]	0.78	**0.92**	0.57	0.65	0.63	**1.12**	**0.77**

This experiment demonstrates the general applicability of our method. Despite large deviation between training and test environment, loops are still detected successfully. To improve loop detection in road traffic, training could be carried out directly with part of the KITTI data. In this case, the loop distance of 3m defined during training could also be increased, since detection is desired at greater distances on the road.

V. CONCLUSIONS

We presented a module to close loops based on laser scans to extend graph-based SLAM methods. By global description of a point cloud with rotation invariant features and a trained classifier, the question *Have I been here before?* can be answered under challenging conditions, which is essential for the long-term autonomy of mobile robots. Experiments show that the classifier detects 47.3% of loops with a small number of false positives which can be filtered out by verification. In dynamic environments, localization with RTAB-Map succeeded only in two out of eleven cases. A connection between all different sessions could be established using our extension. Except for a registration error, in ten out of eleven experiments, the localization succeeded. The potential of our module was also demonstrated by a validation on the KITTI dataset.

Despite all the positive results, new tasks arise: Since registration can fail with many repeating elements, the acceptance of a calculated transformation should be further restricted, e.g. by considering the percentage of overlapping regions. Further, we will integrate the extension into a map management approach [23]. With different SLAM configurations, it would be possible to use a different classifier and registration parameters for a short range than for a long range environment. The main potential for improvement lies in the use of absolute positions using WiFi and GPS data. When starting multi-session operation instead of a loop search in the entire map, loops could be searched for directly in the local environment.

REFERENCES

[1] S. Thrun and J. J. Leonard, "Simultaneous Localization and Mapping," *Springer Handbook of Robotics*, pp. 871–889, 2008.

[2] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 110, no. 3, p. 346, 2018.

[3] C.-h. Hua, L.-h. Dou, H. Fang, and H. Fu, "A novel algorithm for SLAM in dynamic environments using landscape theory of aggregation," *Journal of Central South University*, vol. 23, no. 10, pp. 2587–2594, 2016.

[4] T. Krajnik, J. P. Fentanes, J. M. Santos, and T. Duckett, "FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments," *IEEE Transactions on Robotics*, vol. 33, pp. 964–977, 2017.

[5] T. Krajnik, J. Pulido Fentanes, M. Hanheide, and T. Duckett, "Persistent localization and life-long mapping in changing environments using the Frequency Map Enhancement," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 4558–4563.

[6] M. T. Lazaro, R. Capobianco, and G. Grisetti, "Efficient Long-term Mapping in Dynamic Environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 153–160.

[7] P. Biber and T. Duckett, "Dynamic Maps for Long-Term Operation of Mobile Service Robots," in *Robotics: Science and Systems I*, 2005.

[8] M. Labbé and F. Michaud, "Long-term online multi-session graph-based SPLAM with memory management," *Autonomous Robots*, vol. 42, no. 6, pp. 1133–1150, 2018.

[9] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, "Appearance-based loop detection from 3D laser data using the normal distributions transform," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 23–28.

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

[11] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[12] K. Granström and T. B. Schön, "Learning to close the loop from 3D point clouds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2089–2095.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class AdaBoost," *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[15] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[16] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1–4.

[17] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[18] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D," *IEEE Robotics & Automation Magazine*, vol. 22, no. 4, pp. 110–124, 2015.

[19] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.

[20] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent Point Feature Histograms for 3D Point Clouds," *Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, 2008.

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[22] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, p. 629, 1987.

[23] S. F. G. Ehlers, M. Stuede, K. Nuelle, and T. Ortmaier, "Map Management Approach for SLAM in Large-Scale Indoor and Outdoor Areas," in *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 9652–9658.

[24] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System," in *Intelligent Autonomous Systems 13*, 2016, vol. 302, pp. 335–348.

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

[26] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, pp. 401–416, 2017.