Evaluation and Deployment of LiDAR-based Place Recognition in Dense Forests

Haedam Oh
St Edmund Hall
University of Oxford

A thesis submitted for the degree of
MEng 4th Year Project

2024

Abstract

Many LiDAR place recognition systems have been developed and typically focused on self driving scenarios in urban environments. Their performance in natural environments such as forests and woodlands have been studied less closely. In this thesis, we analyze the capabilities of four different LiDAR place recognition systems, both handcrafted and learning-based methods, using LiDAR data collected with a handheld device and a legged robot within dense forest environments. In particular, we focused on evaluating localization performance where there is significant translational and orientation difference between corresponding LiDAR scan pairs. This property is particularly important for forest survey systems where the sensor or robot does not follow a defined road or path and revisiting previously visited paths would be impractical. Extending our analysis we then incorporated the best performing approach, Logg3dNet, into a full 6-DoF pose estimation system—introducing several verification layers to achieve precise and reliable registration. We demonstrated the performance of our methods in three operational modes: online SLAM, offline multi-mission SLAM map merging, and relocalization into a prior map. Then we evaluated these modes using data captured in forests from three different countries. The overall place recognition system could acheive 80 % of correct loop closures candidates with baseline distances up to 5 m, and 60 % up to 10 m.

The work has been submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2024. A pre-print of the paper[1] and video[2] are available online.

[1] https://arxiv.org/abs/2403.14326
[2] https://drive.google.com/file/d/1EjFU06T_RmUtu6fig0JufpV0g8TkXDfD/view?usp=drive_link

Contents

1 Introduction

Motivation

Place recognition is an essential capability used to correct for the effect of pose estimation drifts suffered by a Simultaneous Localization and Mapping (SLAM) system. It can also be used to re-localize in previously mapped areas. LiDAR-based place recognition systems have demonstrated robustness by effectively capturing scene structure while being insusceptible to visual changes or minor structural changes, making them suitable for long-term navigation. Previous methods have primarily focused on the problem of autonomous driving in urban environments (which is characterized by long linear sequences), with their efficacy in natural settings much less tested. There are emerging applications of robots and autonomous systems in natural environments, including agriculture, environmental monitoring, conservation, and search and rescue, highlight the significance of extending research efforts beyond urban landscapes.

Challenges in natural environments

Despite the advancements of LiDAR-based place recognition in urban areas, its application in unstructured natural environments like forests with their irregular structures such as trees and bushes presents several notable challenges. Firstly, natural environments are irregular structures like trees, which not only lack fine structure but also undergo seasonal changes, affecting their shape and density over time. This complicates the

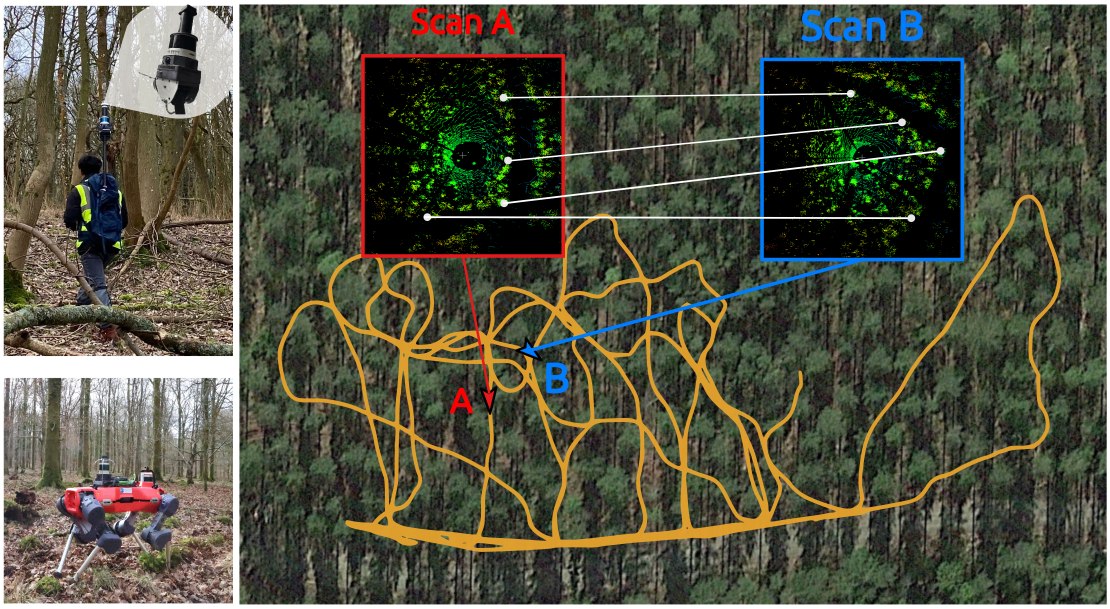

Figure 1.1: Place recognition in dense forest environments using a backpack-mounted LiDAR and legged robot equipped with LiDAR. Right: An example of loop closure detection within a deeply forested area, with offset distance of 9.2 meters and orientation of 56 degrees between the matched scans. White lines indicate some corresponding locations in the two point clouds (but no actual correspondences are used).

creation of consistent geometric representations crucial for accurate place recognition. Secondly, a limited field-of-view and occlusions of a LiDAR sensor result in a loss of information. In large-scale natural environments, LiDAR sensors often struggle to capture the complete vertical extent of the environments. This issue is particularly common within areas populated by mature plantations of tall trees. Undulating terrains and dense trees cause occlusions causing the scans produced by LiDAR sensor to be incomplete. Lastly, trajectories which do not follow defined roads or pathways when exploring dense forests pose another challenge: a place recognition capability of recognizing the sensors' location when not precisely revisiting a known location but passing merely 'near by' location.

Objectives

Recently, the Wild-Places dataset [15] has been made available. In it a large-scale forest dataset was captured by using hand-held spinning LiDAR, which captured seasonal changes within the forest. The dataset was used to evaluate state-of-the-art learning-based and handcrafted place recognition models in forest environments focusing on localization task. The dataset was limited to the access roads of an Australian national

park. The main outcome was the demonstration that learning-based descriptors such as Logg3dNet [3] showed better performance compared to handcrafted methods such as ScanContext [13]. While the findings suggested that LiDAR-based methods could provide a solution for place-based localization in forest environments, it was not conclusive if the results transferred to dense forest areas, especially those lacking strong structural cues like access roads. We are also interested in going further and incorporating the pipeline in precise 6DoF pose estimation and testing it in various operational modes for forestry or biodiversity monitoring applications.

Contributions

In this work, we present an evaluation of place recognition models to dense forest environments. We employ different LiDAR sensors with sequences recorded with backpack-mounted devices and on a quadruped mobile robot to perform a rigorous analysis of existing learning-based and handcrafted LiDAR place recognition systems. Further, we present a LiDAR-based SLAM system to assess the proposed loop-closure pairs and subsequently validate them through fine-registration in both online and offline modes. Finally, we demonstrate the approach being used for a purely relocalization task, wherein the system continuously localizes its position within a prior point cloud map made up of individual LiDAR scans.

In summary, the contributions of this thesis are:

- Evaluation of the performance of four LiDAR place recognition models, both handcrafted and learning-based, in dense forest environments.
- Analysis of loop-closure performance as a function of the relative distance and orientation between candidate loop pairs.
- Integration of the best performing method in online/single-session and offline/multi-session SLAM modes, as well as in relocalization tasks for forest inspection in a prior environment map.
- Release of our dense forest datasets with ground truth, collected with a backpack-mounted LiDAR in Finland, Switzerland and the UK.[1]

[1] Our datasets will be published at https://ori.ox.ac.uk/labs/drs/datasets-drs/

2
Related Work

In this section, we first review different LiDAR-based place recognition methods specifically focused on the scenario of place recognition using a single LiDAR scan. Methods involve feature extraction from raw point clouds to retrieve the most similar place and registration of this live point clouds scan to retrieved point clouds to estimate relative transformation. In particular, the second step is important since relative 6-DoF transform between the matches is crucial for integrating loop-closure into a SLAM system. There are various LiDAR-based place recognition approaches exist to extract descriptors: handcrafted models that can extract geometric features and summary statistics [13, 31], while learning-based approaches that employ Convolutional Neural Networks (CNNs) to compute high-level descriptors capable of distinguishing between different places [3, 17].

Handcrafted Descriptors: Among handcrafted approaches, ScanContext[13, 12] stands out as a widely adopted technique that generates a descriptor by encoding point clouds in a polar coordinate representation. It captures height information within defined angle and range sectors, then integrates them into a 2D descriptor. ScanContext has also been enhanced by incorporating additional information such as intensity [28] and semantics [18] to create more informative descriptors. Another type of handcrafted descriptor, STD [31], encodes boundaries of planes as vertices and connects them to create multiple triangles. STD operates without requiring a 360-degree scan, making it compatible with LiDAR systems with only a 90-degree field-of-view (e.g. Livox Aria).

Both ScanContext and STD can estimate a relative transformation between corresponding scans by matching their descriptors when detecting loop closure candidates in SLAM or relocalization tasks. However, these existing methods have been primarily tested in urban scenarios. In this work we specifically aim to study their performance in unstructured, natural environments such as forests.

Learning-based Descriptors: Alternatively, recent learning-based models, such as Logg3dNet [3], MinkLoc3D [16], and EgoNN [17] employ discretized representations and contrastive learning schemes to compute point-based local descriptors. This process is followed by generating a global descriptor of aggregated local features using methods like GeM [22], P2O [2], and NetVLAD [6].

In particular, we are interested in Logg3dNet [3] which we primarily tested and integrated into our system. As shown in Fig. 2.1, it uses a sparse convolutional U-network which performs sparse point-voxel convolution on a raw point cloud to embed local features. Local consistency loss (e.g triplet loss) tries to minimize the feature distance between positive pairs (in overlapping locations, typically less than 30 m apart) while maximizing the feature distance between negative pairs (in non-overlapping locations) in a contrastive manner. These local features are aggregated using second-order pooling followed by differentiable Eigen-value power normalization of SVD to form a global descriptor. A quadruplet loss is used for global descriptors Finally, combined local consistency loss and global scene-level loss are used to train the network in end-to-end fashion.

Similarly, EgoNN [17] employs a deep CNN architecture to extract local descriptors and key points through regression, subsequently aggregating them using GeM to form a global descriptor. Both EgoNN and Logg3dNet facilitate relative transformation by matching local keypoints with RANSAC, which in turn a final finer resolution registration using ICP. Finally, place recognition models which use Transformers [32, 30, 34] exist. They are known for their ability to capture long-term dependencies, but have high computational costs and often focus on the global-level place recognition problem not local-level pose estimation.

Other types of descriptors: As an alternative to whole scan descriptors, some methods utilize segments to capture the important elements, effectively compressing

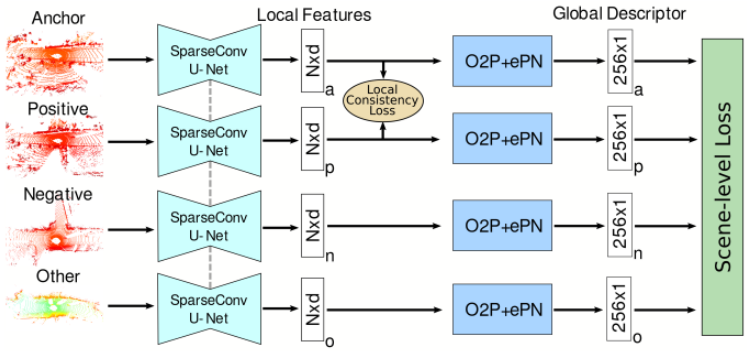

Figure 2.1: Logg3dNet architecture. Positive, negative, and anchor point cloud pairs are fed into a SparseConv U-Net with local consistency loss to enforce similarity in local feature embeddings. Local features are aggregated using second-order pooling (O2P) followed by Eigenvalue Power Normalization (ePN) to obtain a global descriptor with global scene-level loss.

the information in the entire point cloud map into a more compact representation. SegMatch [5] by Dubé et al. computes local segments and extracts geometric features as a descriptor. In their follow-up work, SegMap [4], these features were learned via a CNN, showing improved overall performance. Tinchev et al. [25, 24] applied the segment-based approach to natural environments, showing promising results. However, these methods are vulnerable when these segments cannot be reliably detected due to occlusions in dense forest environments, as well as long-term changes in the environment.

Benchmark datasets: Several LiDAR point cloud datasets are available for benchmarking place recognition models in urban scenarios [19, 7, 14]. However, there are few datasets for natural environments [26, 15]. In particular, the Wild-Places dataset [15] is the most notable because it is tailored to large-scale place recognition in forests. This dataset provides point clouds and ground truth poses collected in a forest national park in Australia using hand-held spinning LiDAR at various times of the year.

In this thesis, we assess the performance of four different place recognition models including both handcrafted (ScanContext and STD) and learning-based (Logg3dNet and EgoNN) models on our dense forest datasets. The datasets contain three different forest environments across different countries, Wytham (UK) which were collected by myself, Evo (Finland) and Stein am Rhein (Switzerland) collected by other member of Oxford Robotics Institute. These datasets were collected using a backpack LiDAR mapping device deep inside the forest, in contrast to previous methods that often concentrated on access roads.

3 Methods

3.1 Overview

In this section, we present an overview of the key components that make up our system as well as the different operational modes for evaluating the complete system's capabilities. Below summarizes the contents of this section, including the key components of our system and the different operational modes for evaluating its capabilities.

- **LiDAR-based SLAM system:** The LiDAR-based SLAM system is our base system which comprises VILENS LiDAR-inertial odometry [29] for state estimation and the pose graph SLAM [21] for achieving accurate localization and mapping. However, this system accumulates drift over time.
- **Place recognition server:** The place recognition server enables robust detection and verification of loop closures. This allows the SLAM system to correct the accumulated drift, improving the overall accuracy of the map (main focus of this project).
- **Evaluation of three different operational modes:** We introduce three different operational modes: online SLAM, offline multi-mission SLAM map merging, and relocalization modes. This will demonstrate capabilities of our system in various scenarios and highlight the importance of place recognition for achieving consistent and accurate maps.

3.2 LiDAR SLAM system

The LiDAR-based SLAM system is the base system that provides the robot's pose estimate and the 3D map of the environment. The system consists of two main components: state estimation module (LiDAR-inertial odometry) and pose graph SLAM system. The state estimation module provides the robot's incremental pose estimate based on the local sensor measurements around the mapping device. The pose graph SLAM system is used to optimize the robot's pose estimate globally correcting the accumulated drift from the state estimation module. We will provide a overview of the odometry system and the pose graph SLAM system in the following paragraphs.

LiDAR-intertial odometry For the state estimation, we use a LiDAR-intertial factor graph-based odometry system, VILENS [29] to provide a continuous odometry estimate at high frequency. VILENS uses four sensor modalities IMU, LiDAR, camera, and (leg kinematics) being tightly coupled in a factor graph as shown in Fig. 3.1 (Left). But in this project, we only used IMU and LiDAR sensors, which are sufficient for tracking the pose when running inside forest environments. The state estimation problem can be formulated as maximum a posteriori (MAP) problem for estimating all states $\mathbf{X}_k$, by optimizing over all states $\mathbf{X}_k$ given all measurements $\mathbf{Z}_k$ up to time k as follows:

$$\begin{aligned}\mathbf{X}_k &= \arg\max_{\mathbf{X}_k} p(\mathbf{X}_k|\mathbf{Z}_k) \propto p(\mathbf{X}_0)p(\mathbf{Z}_k|\mathbf{X}_k) \\ &= \arg\min_{\mathbf{X}_k} -\log p(\mathbf{X}_0) - \log p(\mathbf{Z}_k|\mathbf{X}_k)\end{aligned} \tag{3.1}$$

Assuming each measurement is conditionally independent with the effect of Gaussian noise, the optimization problem can be formulated as least squares problem:

$$\mathbf{X}_k = \arg\min_{\mathbf{X}_k} \|\mathbf{r}_0\|^2 + \sum_{i \in K_k} \left(\|\mathbf{r}^{\mathrm{I}}_{ij}\|^2_{\Sigma_{\mathbf{r}^{\mathrm{I}}_{ij}}} + \|\mathbf{r}^{\mathrm{L}}_{i}\|^2_{\Sigma_{\mathbf{r}^{\mathrm{L}}_{i}}} \right) \tag{3.2}$$

where $\mathbf{r}_0$, $\mathbf{r}^{\mathrm{I}}_{ij}$, and $\mathbf{r}^{\mathrm{L}}_{i}$ are residual error associated with prior, IMU preintegration factor, and LiDAR factor at key frame i respectively. IMU measurements are preintegrated to the pose and velocity between two consecutive nodes of the graph to provide position, velocity, and orientation. The LiDAR registration factor is used to compute the relative difference transformation between two LiDAR scans from estimated poses and ICP using

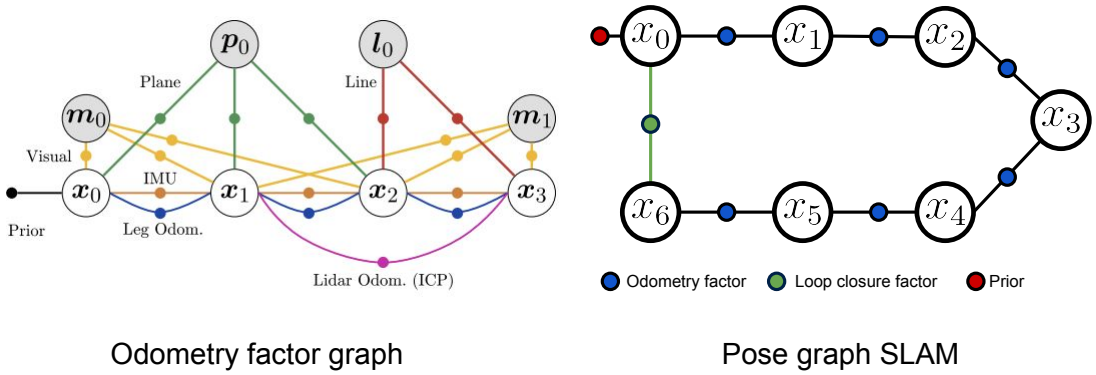

Figure 3.1: Left: VILENS odometry system as a factor graph representation. The factors are: prior (black), visual (yellow), lidar planes (green), lidar lines (red), preintegrated IMU (orange), preintegrated velocity (from leg kinematics, blue), and lidar odometry from ICP registration (magenta). State nodes are white, while landmarks are grey. In forest environments, only LiDAR odometry and IMU factors are used without visual factors. Right: Pose graph SLAM. The system optimizes the poses after successful loop closure verification. The pose graph is constructed from odometry factors and loop closure factors.

Iterative Closest Point (ICP). This optimization problem is solved using the iSAM2 algorithm [11], which will be explained in the next section.

This LiDAR-inertial odometry system is used to provide a locally consistent estimate of the robot's pose. However, the system can suffer from accumulated drift over time. An example of the drift accumulation is shown in Fig. 3.2. After a long traverse in the forest and return to the starting point, we can see that the odometry path diverges from the starting point (yellow line). The pose graph SLAM system (green line) is used to correct this drift by finding a suitable loop closure constraint (red line).

Pose graph SLAM We use a pose graph SLAM system to optimize the robot's pose estimate. The pose graph is constructed from odometry factors and loop closure factors as shown in Fig. 3.1 (Right). Each node contains a LiDAR scan and corresponding pose, and the edge is formed between consecutive nodes from incremental odometry estimates. Each loop closure adds another constraint between two nodes. The odometry factors are provided by VILENS, while the loop closure factors are obtained using the place recognition server developed during this project. The pose graph is optimized using the iSAM2 [11] algorithm as well.

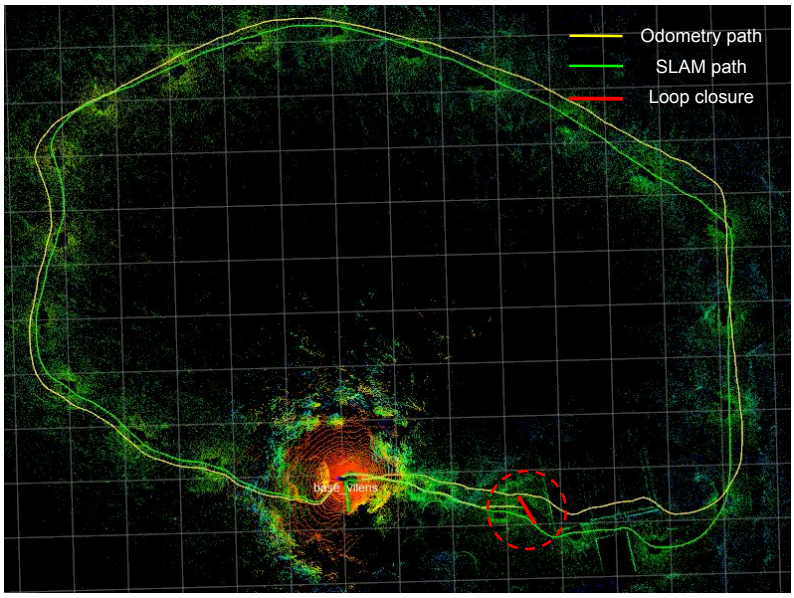

Figure 3.2: Illustrative visualization of both odometry and SLAM paths after a long traverse in Wytham Woods. The odometry path is shown in yellow line and the optimized SLAM path is shown in green. The SLAM path successfully minimized the odometry drift after a loop closure (red line) is found when the mapping device returned to the starting point.

Pose graph optimization iSAM [11](Incremental Smoothing and Mapping) is used to solve the least square minimization problem in pose graph SLAM when a new loop constraint is added. The pose graph optimization is formulated as non-linear least square problem:

$$F(\mathbf{x}) = \min_{\mathbf{x}} \sum_{(i,j)\in E} \|\mathbf{r}_{ij}\|^2_{\Sigma_{\mathbf{r}_{ij}}} + \sum_{(k,l)\in L} \|\mathbf{r}_{kl}\|^2_{\Sigma_{\mathbf{r}_{kl}}} \tag{3.3}$$

where $\mathbf{x}$ represent all poses and $\mathbf{r}_{ij}$ is the odometry residual error and $\mathbf{r}_{kl}$ is the loop closure residual error. Since this is non-linear least square problem, it should be solved by linearising the residual error using the first order Taylor approximation around the current guess $\mathbf{x}_0$ as follows:

$$\mathbf{r}_{ij}(\mathbf{x}_0 + \Delta\mathbf{x}) \approx \mathbf{r}_{ij}(\mathbf{x}_0) + \mathbf{J}_{ij}\Delta\mathbf{x} \tag{3.4}$$

Substituting back to the original $F(\mathbf{x})$,

$$\begin{aligned} F(\mathbf{x}_0 + \Delta\mathbf{x}) = \sum_{i,j\in E,L} \Big(& \mathbf{r}_{ij}(\mathbf{x}_0)^T \Sigma_{\mathbf{r}_{ij}} \mathbf{r}_{ij}(\mathbf{x}_0) \\ & + 2(\Delta\mathbf{x})^T \mathbf{J}_{ij}^T \Sigma_{\mathbf{r}_{ij}} \mathbf{r}_{ij}(\mathbf{x}_0) + (\Delta\mathbf{x})^T \mathbf{J}_{ij}^T \Sigma_{\mathbf{r}_{ij}} \mathbf{J}_{ij} \Delta\mathbf{x} \Big) \end{aligned} \tag{3.5}$$

$$= c + 2\mathbf{b}^T\Delta\mathbf{x} + \Delta\mathbf{x}^T H \Delta\mathbf{x}. \tag{3.6}$$

Then setting $\frac{\partial F}{\partial \Delta \mathbf{x}} = 0$ gives:

$$H\Delta\mathbf{x} = -\mathbf{b} \tag{3.7}$$

Then iteratively solve $\mathbf{x}* = \mathbf{x_0} + \Delta\mathbf{x}$. Note that directly inverting the sparse information matrix H is computationally expensive due to its size and infeasible to re-calculate every time when a new constraint is added. Instead, iSAM performs QR decomposition on the H matrix, resulting in a sparse upper triangular matrix. This allows efficient incremental addition of a new constraint and reordering so that this update is computational efficient and real-time capable.

In summary, the pose graph SLAM system is an essential component to maintain globally consistent poses and a map. When a loop closure is found, the pose graph SLAM optimizes the trajectory to minimize the effect of drift. As illustrated in Fig. 3.2, the odometry path (yellow line) tends to accumulate drift over time, while the optimized SLAM path (green line), after incorporating a loop closure constraint (red line), is able to correct this drift. However, integrating an incorrect loop closure into the pose graph can lead to a failure in the entire SLAM system. Therefore, it is crucial to have a reliable place recognition system as a front-end component, responsible for providing verified loop closure candidates. In the following section, we will discuss *place recognition server* which is responsible for providing reliable loop closure candidates to the pose graph SLAM system.

3.3 Place Recognition & Verification Server

The focus of the research in this report is on the place recognition server. Our place recognition pipeline (Fig. 3.3) consists of three steps: loop candidate proposals, coarse registration, and a final fine-registration. At each step, we perform appropriate checks to filter out incorrect loop closures.

The main inputs are the pose graph, with corresponding LiDAR scans attached to each pose, as well as the single query scan. The query scan is provided by different sources depending on the task we are solving. For example, in the relocalization task it will be a live scan directly from the LiDAR sensor. Further details are provided in the corresponding sections.

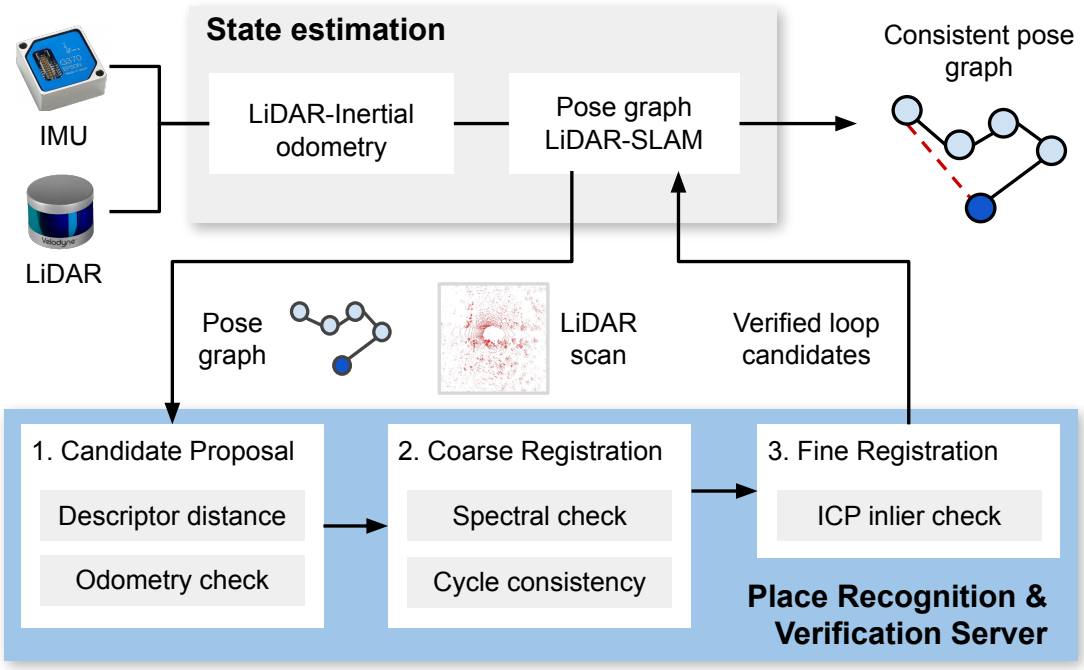

Figure 3.3: Our place recognition pipeline. Grey components: base state estimation modules. Blue components: place recognition server (the focus of the research in this project). VILENS provides a continuous odometry estimate at 10 Hz. Pose graph SLAM is used to optimize poses after successful loop closure verification. The place recognition module consists of three steps: Loop candidate proposal, coarse registration, and fine registration. We verify loop candidates at the global descriptor-level, local feature-level consistency, and finally fine registration level. A loop candidate is integrated in the pose graph only if it passes these three stages.

Step 1: Loop candidate proposals

Initial loop closure candidates are obtained by comparing global descriptors extracted from the pose graph scans as well as the query scan. In this thesis, we evaluate four state-of-the-art methods for descriptor extraction: the learning-based Logg3dNet [3] and EgoNN [17], as well as the handcrafted ScanContext [13] and STD [31]. Fig. 3.4 shows their descriptors examples computed on Wild-Place data.

Given the reference pose graph and the query scan, we compute a database of descriptors using all the scans in the pose graph, given by the matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$, where N is the number of poses in the pose graph and M the descriptor dimension. Additionally, we compute the descriptor for the query scan, denoted by $\mathbf{d}_q \in \mathbb{R}^{M \times 1}$. To obtain candidates, we compute the pairwise descriptor distances of the scan to the database using the cosine similarity:

$$\mathbf{S} = \mathbf{D} \cdot \mathbf{d}_q \in \mathbb{R}^{N \times 1} \tag{3.8}$$

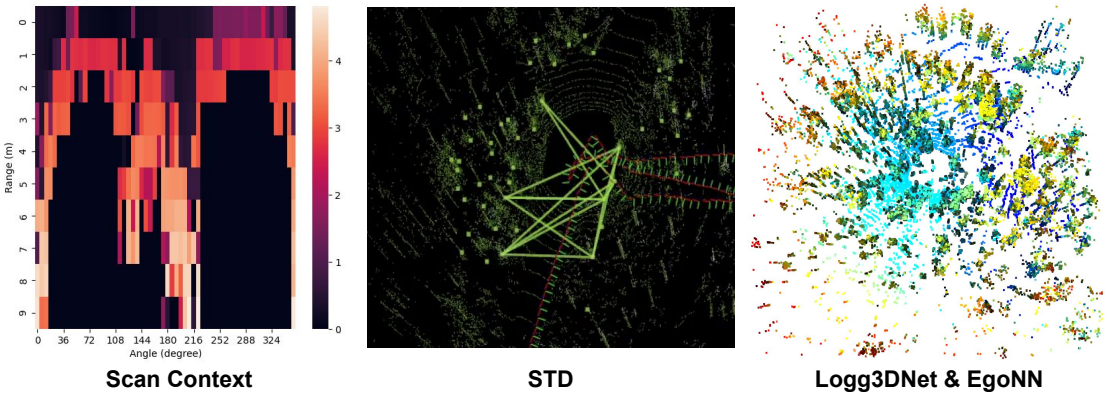

Figure 3.4: Different types of place recognition descriptors on Wild-Place [15] data. Left: ScanContext, Middle: STD, Right: EgoNN and Logg3dNet(Right). ScanContext encodes maximum height at each sector defined by radius and angle. This information is then transformed into a 2D descriptor where the horizontal axis represents the angle and the vertical axis represents the radius. STD computes corners of planes (in our case the trunks of trees) and connects them to form triangle cliques which can be quickly matched. EgoNN and Logg3dNet both compute learning-based pointwise descriptors in voxelized point clouds coloured by similar point features.

The vector of descriptor distances $\mathbf{S}$ is sorted by increasing distance, and only the top-k candidates are selecting using a distance threshold τ_s, which is set by the F_1-max score from testing data. If a spatial prior is available, for example from LiDAR-inertial odometry, we also perform an additional spatial check discarding all the candidates that are more than a (conservative) threshold distance of 20 m away from the query scan. The output is a set of candidate nodes $\{n_c\}$.

Step 2: Coarse Registration

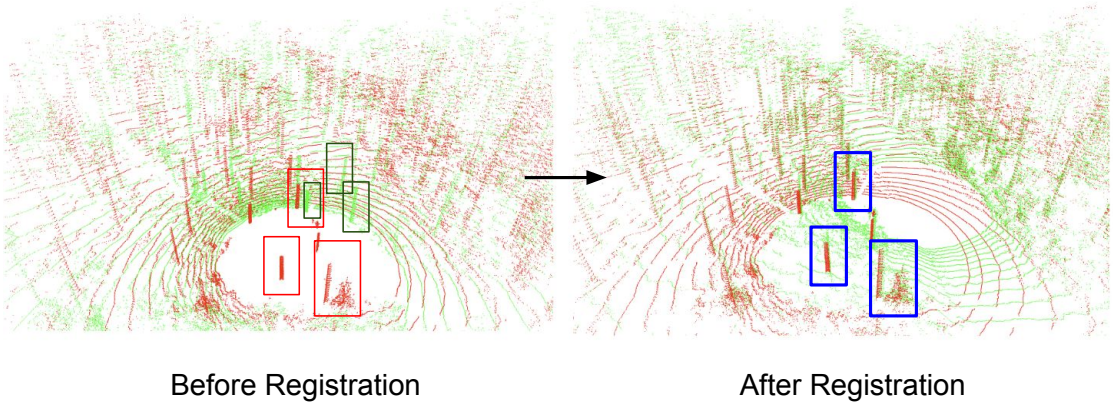

Figure 3.5: Registration example of two point clouds 7.3 m and 111° apart. Left image is before registration. Right image shows after RANSAC-registration. After RANSAC-registration, the trunks of trees scans (Blue boxes) are clearly matched with each other .

Next, we estimate the relative 6DoF transformation that expresses the pose associated to the query scan w.r.t each candidate node, which we denote $\Delta\mathbf{T}$. For the handcrafted methods (ScanContext and STD), the relative transformation is directly an output of descriptor computation. For the learning-based approaches, we use the point-wise feature vectors output in the forward pass of Logg3dNet and EgoNN for feature matching, which is used in a RANSAC-based pose estimation scheme [9] to estimate the relative transformation. We used *Open3D*'s RANSAC implementation [33].

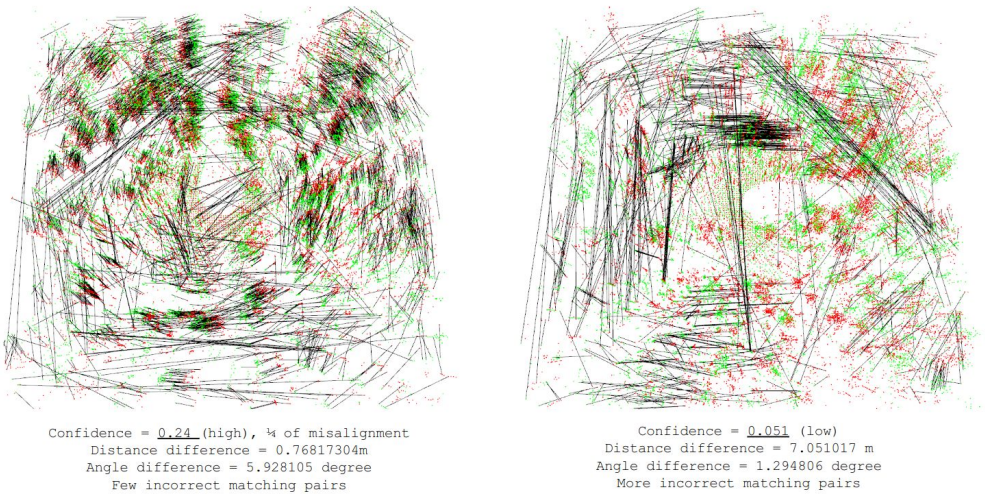

Figure 3.6: RANSAC matches obtained from Logg3dNet features for two examples: one with a small baseline distance and another with a large baseline distance. Left image shows short baseline distance 0.8 m and 6° viewpoint difference and right image shows large baseline distance 7 m and 1° viewpoint difference. The red points represent keypoints in the query point cloud, while green points depict candidate keypoints in the retrieved point cloud. Each keypoint possesses its unique feature vector, matched against the most similar feature in the retrieved point clouds (indicated by black lines). In the left image, with a short baseline distance and minimal viewpoint difference, RANSAC matching consistently discovers matching pairs along the horizontal axis. However, in the right image, where the baseline distance is significantly larger, some outliers are depicted as a cluster of vertical lines in the figure.

RANSAC Registration Fig. 3.6 shows illustrative examples of RANSAC feature matchings. We noticed that as the distance between two point clouds increases, there are more incorrect matches (outliers) due to small amount of point-size overlap between them. This also suggests that RANSAC only measures if two point clouds can be registered but is unable to determine if a loop candidate is actually a false positive or a correct loop candidate but with far baseline distance. Because of this we then introduced two more verification layers, Spectral Geometric Verification (SGV) and pairwise cycle consistency check.

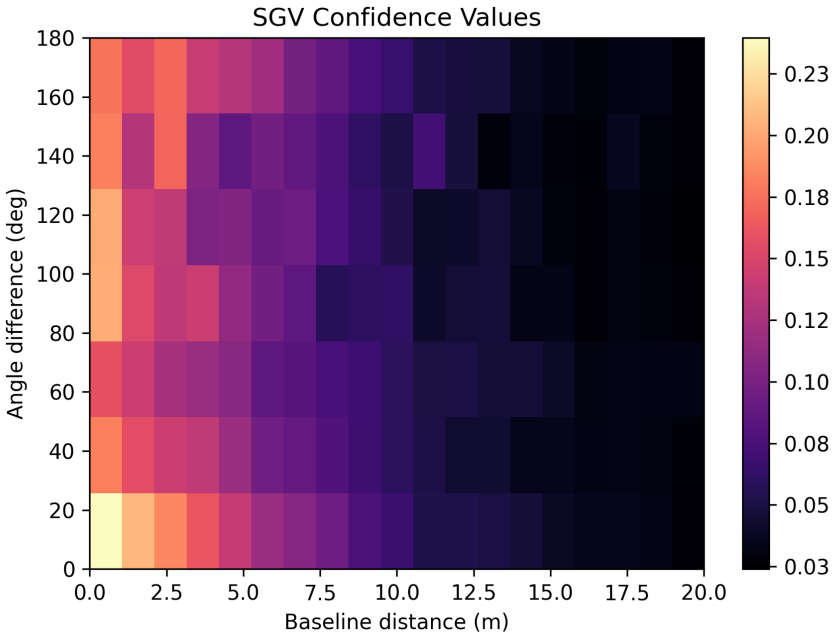

Figure 3.7: The SGV scores s are computed against baseline distances and angles of loop closures obtained (White bins are empty data points). A smaller SGV score corresponds to larger baseline distances, suggesting a reduced proportion of correct correspondences. For example, $s = 0.1$ approximately indicates 10 m baseline distance, and $s = 0.05$ indicates 20 m. Furthermore, it was observed that the SGV score remains constant across different angles, indicating a consistent registration process irrespective of viewpoint changes (viewpoint invariant), and useful indication of baseline distance.

Spectral Geometric Verification (SGV) We additionally verify the inlier matches using the *Spectral Geometric Verification*(SGV) [27] method, which provides an additional measure of the quality of the feature matches. Essentially, SGV checks pairwise cycle consistency of local feature correspondences between two point clouds. Given the correspondences between two point clouds, $\mathcal{C} = \{c_1, c_2 \dots c_n\}$ we can construct a graph $\mathbf{M} \in \mathbb{R}^{n \times n}$ where $M_{i,j}$ represents similarity score s between c_i and c_j. If both c_i and c_j are correct inliers they should be pairwise consistent in their coordinate frames with high similarity score s, but if one of correspondences is an outlier they will not be consistent with low similarity score s. For example, let's consider two correspondences $c_1 = \{\mathbf{p}_i^A, \mathbf{p}_l^B\}$, $c_2 = \{\mathbf{p}_j^A, \mathbf{p}_k^B\}$ where $\mathbf{p} \in \mathbb{R}^3$ between two point clouds A and B. The euclidean distance $\|\mathbf{p}_i^A - \mathbf{p}_j^A\|$ and $\|\mathbf{p}_k^B - \mathbf{p}_l^B\|$ should be consistent each other if they are both correct correspondences. Thus, the quality of the feature correspondences can

be found by minimizing the error d between these two distances:

$$d_{1,2} = \|\|\mathbf{p}_i^A - \mathbf{p}_j^A\| - \|\mathbf{p}_k^B - \mathbf{p}_l^B\|\| \tag{3.9}$$

$$M_{1,2} = max(0, 1 - \left(\frac{d_{i,j}}{d_{\text{thres}}}\right)^2) \quad \text{where} \quad 0 \leq M_{1,2} \leq 1 \tag{3.10}$$

d_{thres} typically set to 0.1 m to 0.2 m to consider the error in the registration process. If $d_{i,j}$ is larger than d_{thres}, we consider two correspondences are inconsistent and $M_{i,j}$ assigned to be zero. We then compute the total SGV score s considering all correspondences $\mathcal{C} = \{c_1, c_2 \dots c_n\}$ as follows:

$$s = \sum_{i,j} M_{i,j} \tag{3.11}$$

By finding maximum inliers $\mathbf{v}^*$, we can find the maximum SGV score s^*:

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \mathbf{v}^T \mathbf{M} \mathbf{v} \quad , \quad s^* = \mathbf{v}^{*T} \mathbf{M} \mathbf{v}^* \tag{3.12}$$

where $\mathbf{v}$ is the eigenvector of the largest eigenvalue of $\mathbf{M}$. The SGV score s is then used to verify the quality of the feature matches.

The authors of Logg3dNet proposed using the SGV check to re-ranking top-k candidates obtained from descriptor distance matching in Sec. 3.3. Instead we used SGV to verify registration quality against baseline distance. Fig. 3.7 shows the SGV score computed against the distance between two point clouds. We used SGV score as an indication of baseline distance of loop candidates. For example, we set $s = 0.1$ as a threshold for 10 m baseline distance, and reject the loop candidates if SGV score is smaller than 0.1.

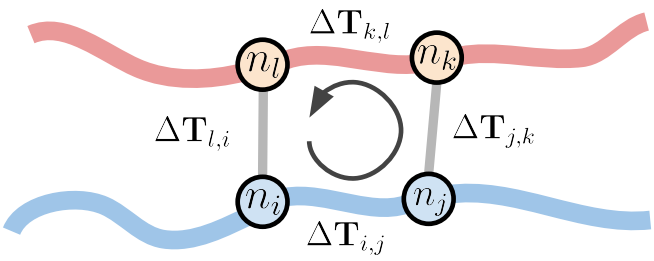

Figure 3.8: Our proposed pairwise cycle consistency check is general and applies to the online and offline multi-mission SLAM case, as well as relocalization tasks. We only need the relative transformation estimates (from the odometry system) and loop candidates (from the place recognition server) between four nodes n_i, n_j, n_k, n_l to verify the validity of a loop in the pose graph. Please refer to Sec. 3.3 for technical details.

Pairwise Loop Closures Cycle Consistency Check

Lastly, we carry out a *loop closure cycle consistency* verification (Fig. 3.8), which checks whether the relative transformations between pairs of nodes in pose graph are mutually consistent with one another. Given four pose graph nodes n_i, n_j, n_k, n_l, we test how close the following equivalence holds:

$$\Delta\mathbf{T}_{i,j}\,\Delta\mathbf{T}_{j,k}\,\Delta\mathbf{T}_{k,l}\,\Delta\mathbf{T}_{l,i} \approx \mathbf{I}_{4\times4} \tag{3.13}$$

If this difference is more than a threshold of 10 cm or 1° we reject the candidate.

This is shown in Fig. 3.8, the interpretation of these transformations change depending on operation modes: online SLAM (Sec. 3.4.1), offline multi-mission SLAM (Sec. 3.4.2), or pure relocalization (Sec. 3.4.3). However, the main idea remains the same: to verify if two successive loop candidates are consistent across different frames (Alg.1 shows some pseudocodes for computing pairwise cycle consistency check).

Step 3: Fine Registration

When given an initial registration from RANSAC in *Step 2*, we employ the Iterative Closest Point (ICP) algorithm [8] for fine registration of the proposed candidates. We use the *libpointmatcher* implementation [20], which also provides information on the quality of the registration, such as the proportion inlier points and the residual error of each point to access the alignment. We use the proportion of inliers (usually 3k-5k points out of 20k downsampled points clouds) and the residual error of 20 cm as a final verification step to reject loop candidates. The verified candidates are then used for SLAM or relocalization applications, which are detailed in the following sections.

3.4 Evaluation Modes

The objective of this project is to test our complete pipeline within dense forest environments in three different operational tasks:

- *Task A: Online SLAM*: the proposed loop candidates contributing to a globally-consistent pose graph mapping system in an incremental manner.

Algorithm 1: Pairwise Cycle Consistency Check: $\Delta\mathbf{T}_{t-1}$, $\Delta\mathbf{T}_t$

Input : $\Delta\mathbf{T}_t$ current loop closure
Output : *success* or $fail$ of the cycle consistency check

Initialize: n_i, n_j, n_k, n_l nodes

Function *Pairwise Cycle Consistency Check*

```
/* Skip the first loop closure */
```
if $\Delta\mathbf{T}_{t-1}$ *is None* **then**
  $\Delta\mathbf{T}_{t-1} \leftarrow \Delta\mathbf{T}_t$
  return $fail$
else
  $\Delta\mathbf{T}_{i,j} \leftarrow$ **Read** $Edge(i,j)$
  $\Delta\mathbf{T}_{j,k} \leftarrow$ **Read** $CurrentLoopClosure(\Delta\mathbf{T}_t)$
  $\Delta\mathbf{T}_{l,i} \leftarrow$ **Read** $PreviousLoopClosure(\Delta\mathbf{T}_{t-1})$
  $\Delta\mathbf{T}_{k,l} \leftarrow$ **Compute** $Edge(j,k)$
  $\mathbf{Err}(\Delta\mathbf{R}|\Delta\mathbf{t}) \leftarrow \|\mathbf{I} - \Delta\mathbf{T}_{i,j}\,\Delta\mathbf{T}_{j,k}\,\Delta\mathbf{T}_{k,l}\,\Delta\mathbf{T}_{l,i}\| = [\Delta\mathbf{R}|\Delta\mathbf{t}]$
  `/* Check if loop closure is consistent */`
  if $\boldsymbol{Err}(\Delta\mathbf{R}|\Delta\mathbf{t}) < threshold$ **then**
    Set $fails \leftarrow 0$
    Set $\Delta\mathbf{T}_{l,i} \leftarrow \Delta\mathbf{T}_{t-1}$
    return *success*
  else
    `/* Too many fails, reset */`
    if $fails > max\_fails$ **then**
      Set $fails \leftarrow 0$
      Set $\Delta\mathbf{T}_{t-1} \leftarrow \Delta\mathbf{T}_t$
    else
      $fails \leftarrow +1$
    end
    return $fail$
  end
end

- *Task B: Offline multi-mission SLAM*: loop candidates used to link different overlapping missions collected at different times and merged into a single merged map.
- *Task C: Relocalization*: place recognition with a prior map. This capability can enable autonomy within the map such as longer term monitoring or harvesting.

3.4.1 Task A: Online Single-mission SLAM

The first task we consider is LiDAR-based online SLAM. Our implementation defines it as an incremental pose graph estimation problem (see Fig. 3.9,(a)). Consider consecutive loop closures at nodes n_i, n_{i+1} and n_j, n_{j+1}. Edges are provided by relative estimates from our LiDAR-inertial odometry system (odometry factors, denoted by $\Delta\mathbf{T}_{i,i+1}, \Delta\mathbf{T}_{j,j+1}$),

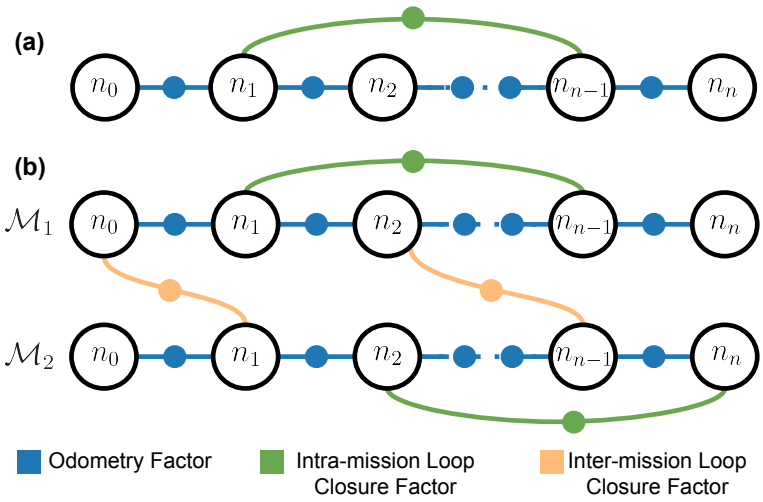

Figure 3.9: Pose graph formulation used for (a) online, and (b) offline multi-mission SLAM optimization. Each node n_i has a 6DOF pose $\mathbf{x}_i$, which corresponds to the main variables estimated on each case.

and verified loop closure candidates from our place recognition server (loop closure factors, $\Delta\mathbf{T}_{i+1,j}, \Delta\mathbf{T}_{i,j+1}$).

For the cycle consistency verification described in Sec. 3.3, we consider the relative transformation change between consecutive loop closure candidates w.r.t the pose graph poses and the odometry change (Fig. 3.8 i, j, k, l, replaced by i, $i+1$, j, $j+1$). Again, a cycle consistency needs to be satisfied:

$$\Delta\mathbf{T}_{i,i+1}\,\Delta\mathbf{T}_{i+1,j}\,\Delta\mathbf{T}_{j,j+1}\,\Delta\mathbf{T}_{i,j+1}^{-1} \approx \mathbf{I}_{4\times4} \tag{3.14}$$

3.4.2 Task B: Offline Multi-Mission SLAM

Offline multi-mission SLAM addresses the challenge of merging multiple pose graph SLAM missions $\mathcal{M}_{1,\dots,n}$, collected over time, with partly overlapping areas. The goal is to find inter-mission loop candidates to construct a unified map in a common reference frame.

Unlike the scenario of on-road navigation, where similar routes (hence locations) are revisited, we considered off-road scenarios where the missions are collected in dense forests, where it is often unfeasible to retrace the same paths on each sequence. To avoid inefficiently retracing our steps, we wish to identify loop candidates when passing no closer than about 10 m. This provides the flexibility when merging two partly overlapping

missions. Each mission $\mathcal{M}_i$ is defined by a pose graph with odometry factors and intra-mission loop closures, obtained during each independent online SLAM run (*Task A*). We aim to provide additional *inter-mission* loop candidates that bridge nodes across the missions, as shown in Fig. 3.9 (b). In this case, potential *inter-mission* loop candidates are obtained through successive one-on-one matching of each mission's nodes against one of other missions, and by integrating them into a unified pose graph.

For the loop proposal step, we execute the same procedures described in Sec. 3.3, but with a tighter descriptor distance threshold τ_s to provide less false positive candidates to the multi-mission cycle consistency check. The cycle consistency step considers pairs of nodes within the same mission, namely $n_i, n_j \in \mathcal{M}_1$ and $n_k, n_l \in \mathcal{M}_2$. The intra-mission relative transformation are then $\Delta\mathbf{T}_{i,j}, \Delta\mathbf{T}_{k,l}$, while the inter-mission relative transformations between loop candidates are given by $\Delta\mathbf{T}_{i,k}, \Delta\mathbf{T}_{j,l}$:

$$\Delta\mathbf{T}_{i,j}\,\Delta\mathbf{T}_{j,l}\,\Delta\mathbf{T}_{k,l}^{-1}\,\Delta\mathbf{T}_{i,k}^{-1} \approx \mathbf{I}_{4\times4} \tag{3.15}$$

3.4.3 Task C: Relocalization

Lastly, we consider the case in which a prior map of the forest is available, e.g. from online SLAM. Our place recognition server is used as a relocalization module, by using the loop candidate proposals to produce initial pose estimates and then executing coarse-to-fine registration. This enables real-time localization of the LiDAR sensor's base $\mathtt{B}$, with the prior map's coordinate frame $\mathtt{M}$, denoted by $\mathbf{T}_{\mathtt{MB}}$.

The pairwise cycle consistency is defined between the current and the last successful loop closure candidate: Given the last successful loop closure $\mathbf{T}_{\mathtt{MB}}(t-1)$ and the current loop closure candidate estimate $\mathbf{T}_{\mathtt{MB}}(t)$, we compared them to the odometry estimates at the same timestamps $\mathbf{T}_{\mathtt{OB}}(t-1)$ and $\mathbf{T}_{\mathtt{OB}}(t)$, where $\mathtt{O}$ indicates the fixed odometry frame. The cycle consistency check is then defined as:

$$\underbrace{\mathbf{T}_{\mathtt{MB}}(t)^{-1}\,\mathbf{T}_{\mathtt{MB}}(t-1)}_{\Delta\mathbf{T}\text{ in }\mathtt{M}\text{ frame}}\;\underbrace{\mathbf{T}_{\mathtt{OB}}(t-1)^{-1}\,\mathbf{T}_{\mathtt{OB}}(t)}_{\Delta\mathbf{T}\text{ in }\mathtt{O}\text{ frame}} \approx \mathbf{I}_{4\times4} \tag{3.16}$$

Finally, ICP is used to fine-localize the LiDAR sensor against the corresponding individual map scan (instead of the full map point cloud).

4
Systems

4.1 Introduction

In this section, we will discuss the hardware and software implementation of our complete system setup. First, we will begin with hardware setup encompassing the sensors and platforms utilized for data collection. Next, we will show the details of implementations of our complete system in three different operating modes: online SLAM, offline multi-mission SLAM map merging, and relocalization within ROS framework. In all those three modes, the place recognition server module provides loop closure information. VILENS odometry is used as the base state estimation module in online SLAM and relocalization, and VILENS SLAM module is used in online SLAM mode managing pose graph.

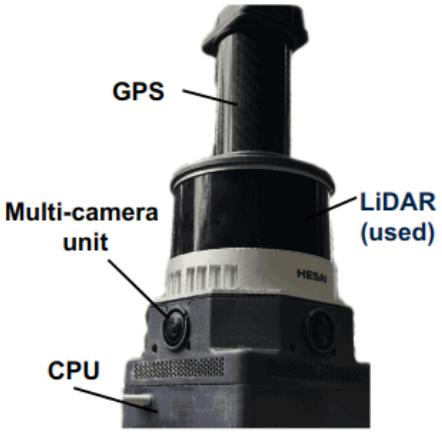

Figure 4.1: Image of *Frontier* device comprising a LiDAR, IMU, three cameras, GPS and CPU board.

Sensor Type	Name	Characteristics
LiDAR	Hesai Pandar XT-32 (*Frontier-15*)	• 10 Hz • 360° × 31° FoV • 0.18° × 1° Res. • 0.5–120 m Range
	Hesai Pandar QT-64 (*Frontier-19*)	• 10 Hz • 360° × 104° FoV • 0.6° × 1.45° Res. • 0.1–60 m Range
Cameras	Sevensense, Alphasense	• 30 Hz • 126° × 92.4° FoV • RGGB Bayer Fisheye • 1440×1080 pixels
IMU	Bosch BMI085	• 400 Hz • 6-axis
GNSS	U-blox	• CNR:30-50 dB

Table 4.1: Sensor specifications of *Frontier* multi-sensor unit device.

4.2 Hardware Setup

We use a multi-sensor unit called *Frontier*, which comprises a LiDAR, IMU, three cameras and onboard CPU computing (additionally GPS) shown in Fig. 4.1. We used two different *Frontier* device, *Frontier-15* and *Frontier-19* which contains Hesai XT-32 and Hesai QT-64 LiDAR respectively and the details of sensor specifications are summarized in Tab. 4.1. For the different *Frontier* devices, we modify configuration settings on both VILENS odometry and VILENS SLAM (details in [29, 21]). For data collection, we mainly used human and legged robot platforms as shown in Fig. 4.2. We typically mapped the forest area in zigzag patterns to ensure coverage in both forward and backward.

Figure 4.2: Platform setup for forest mapping. Backpack (shown in left) with *Frontier* mounted for a human mapping scenario, and a legged robot (shown in right) for forest surveying and monitoring applications.

4.3 Implementations

4.3.1 Robotic Operating System (ROS) Framework

The Robot Operating System (ROS) is a flexible framework for developing software for robotics applications as shown in Fig. 4.3. One of the key concepts in ROS is the publisher-subscriber model, where nodes (independent processes) communicate by publishing messages on topics and subscribing to messages on those topics. Publishers broadcast data, such as sensor readings or robot state information, while subscribers receive and process these data. The nodes and published topics then estalbish a network that represents the data sharing; this can be visualized with ROS tools such as *rqt graph* as shown in Fig. 4.3. Additionally, ROS facilitates communication between nodes through service calls and servers. Services allow nodes to request specific tasks or information from one another, where the client sends a request message and the server responds accordingly, enabling more complex interactions beyond simple message passing shown in service call box in Fig. 4.3. This modular and distributed architecture of ROS makes it well-suited for building modular and scalable robotic systems.

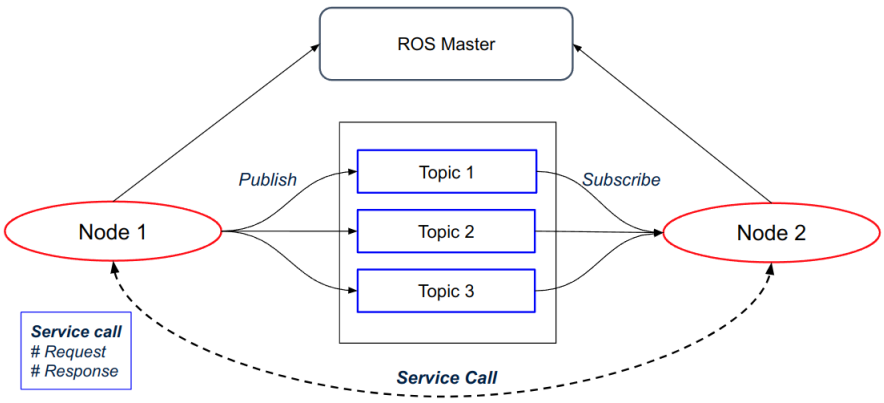

Figure 4.3: ROS Framework. Nodes are shown in red elipses, published topics are shown in blue boxes. Additionally, we also define service call, a client node requests a specific service message and a server node responses a service message based on request. Edges are connections between nodes.

4.3.2 Online SLAM

This is a basic setup for single mission SLAM. Previosuly, place recognition model *Scan Context*, was directly integrated inside VILENS SLAM. But now, place recognition server

acts like a separate module and is shown in Fig. 4.4. The place recognition server subscribes to LiDAR scan and pose graph topics from VILENS SLAM. The place recognition server then publishes service messages to VILENS SLAM node. Service message contains the loop candidate information including candidate id and relative transformation. The overall online SLAM nodes and topics used are shown in Fig. 4.4.

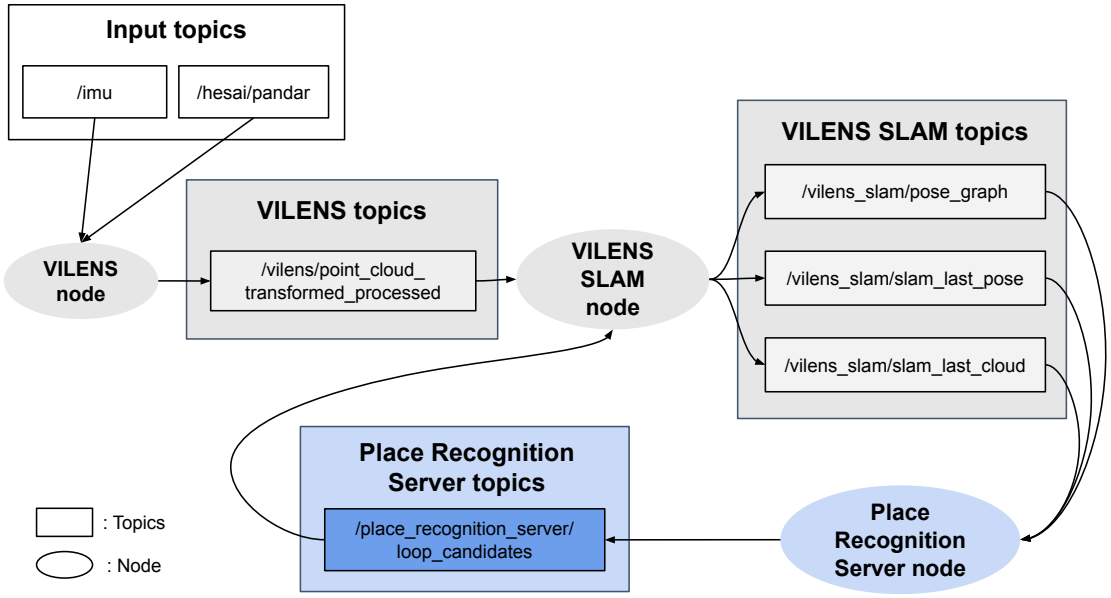

Figure 4.4: Online SLAM Implementation. Loop closure candidates information is transmitted to VILENS SLAM node from the place recognition server node. This occurs whenever new LiDAR point clouds comes in to the place recognition server node from VILENS SLAM node.

4.3.3 Offline Multi-mission SLAM Map Merging

Offline Multi-mission SLAM maps merging runs in post-processing. Loop closure requests and detections are transmitted between VILENS SLAM Offline node and the place recognition server node via service calls. From VILENS SLAM Offline node individual point clouds, cloud ids of a mission and pose graph can be used as input to place recognition server node. The place recognition server node builds the descriptor database incrementally mission by mission. Then, when it finds loop closures it makes service calls back to VILENS SLAM Offline node. The overall offline multi-mission SLAM nodes and topics used are shown in Fig. 4.5.

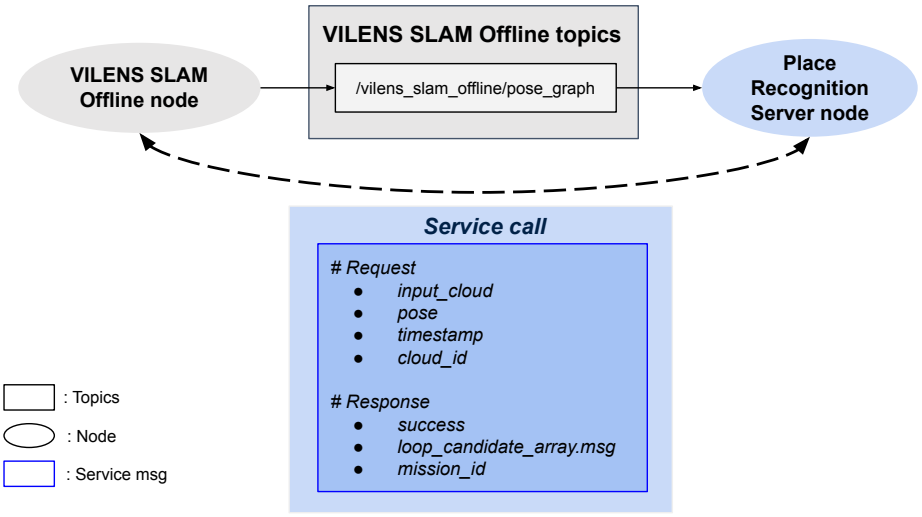

Figure 4.5: Offline multi-mission SLAM map merging implementation. For each mission, every input point clouds are passed to the place recognition server node to find loop closures. The place recognition server node then makes a service call back to VILENS SLAM Offline node.

4.3.4 Relocalization

The relocalization application showcases the use of the place recognition server when a complete prior map is available. As shown in Fig. 4.6, the VILENS SLAM node is replaced by the place recognition client node, which stores the prior map's individual point clouds, pose graph, and descriptors. Similar to the online SLAM mode, the place recognition server node processes the incoming sensor data and extracts descriptors to search for loop closure candidates against the prior map's descriptors. However, unlike the Online SLAM mode where the VILENS SLAM node performed pose graph optimization, the place recognition server does not perform this optimization. Instead, it simply sends the detected loop closure messages to the place recognition client node. The place recognition client node then uses these loop closures to relocalize and visualize the current robot's pose within the prior map's coordinate frame, without modifying the prior map itself.

In order to visualize the pose of the robot continuously, we need to complete the transformation tree between *map* and *odom* frames shown in Fig. 4.7. Transformation tree is a hierarchical structure that manages coordinate frame transformations in ROS. This follows two steps: when loop closures are verified from place recognition server, we can compute transformation between *map* and *base*. Then from VILENS odometry, we access

transformation between *base* and *odom*. We then compute the transformation between *map* and *base* then publish as transformation message. Now this verified transformation is synchronized with VILENS odometry to visualize the pose of the robot. Whenever new loop closures are verified, we change current transformation between *map* and *odom* and republish as transformation message. With localization now expressed with respect to the prior map we can demonstrate this utility by creating a virtual visualization of the forest from the perspective of a camera mounted with the LiDAR sensor.

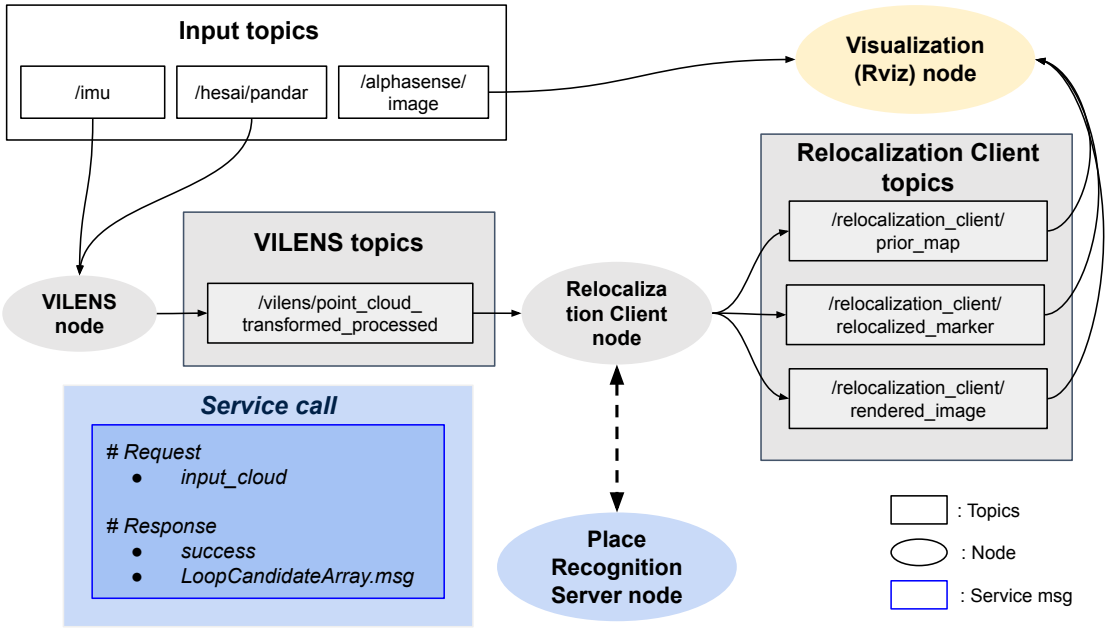

Figure 4.6: Relocalization implementation. Similar to Online SLAM mode, place recognition server node provides a loop closures message to the place recognition client node. The place recognition client node then continuously publishes transformation to relocalize the current robot's pose in map frame. Finally, robot pose, camera images and virtual tree models can then be drawn in visualization node.

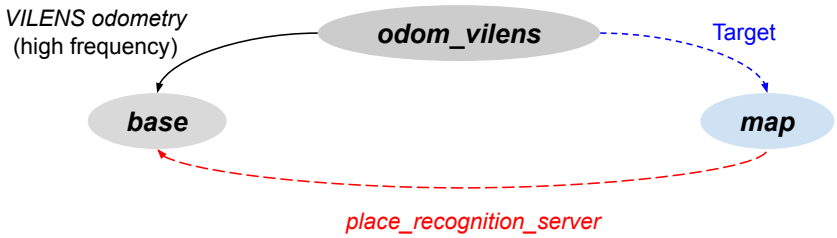

Figure 4.7: Transformation tree in relocalization mode. Transformation between *map* and *odom* frames should be connected by two steps. First, getting transformation between *map* and *base* from the place recognition server then use most recent transformation between *base* and *odom* from VILENS odometry to compute target transformation between *map* and *odom*.

5
Experiments

In this section, we rigorously evaluate our place recognition pipeline to assess its performance in dense forest environments. Our evaluation includes four distinct test sites featuring varying forest compositions: Evo (Finland) is characterized by coniferous trees while Stein-Am-Rhein (Switzerland), Wytham Woods (UK) and Forest of Dean (UK) contain both broad-leaf and coniferous tree species. We evaluate all three operational modes of our system: Online SLAM, Offline Multi-Mission SLAM, and Relocalization. The experiments conducted are as follows:

I. Evaluation of four different place recognition methods at the descriptor-level, tested across multiple forest environments with different LiDAR setups. (Sec. 5.1)

II. Performance assessment during both online and offline SLAM operations within dense forest settings. (Sec. 5.2 & Sec. 5.3).

III. Analysis of successful loop closures, based on the baseline distance and orientation differences between the two candidate scans. (Sec. 5.2)

IV. Demonstration of the application of relocalization in a previously mapped forest environment, showcasing its utility in an inspection task performed by a quadruped robot. (Sec. 5.4)

Figure 5.1: Illustrative examples of four forests used in the experiments. From top left to bottom right: Evo (Finland, May), Stein am Rhein (Switzerland, Sep), Wytham Woods (UK, Feb), and Forest of Dean (UK, March). The datasets were collected using backpack-carried LiDAR systems across different seasons and forest types.

5.1 Place Recognition Descriptors

In this experiment, we evaluated the descriptors of four different place recognition models (Logg3dNet, EgoNN, ScanContext, STD) focusing on their ability to accurately capture loop-candidates in forest environments. Logg3dNet and EgoNN models are learning-based methods, which were pre-trained on the Wild-Places dataset.

Datasets

We collected the dataset with two different types of LiDAR sensors mounted on a backpack: Hesai XT32 and Hesai QT64 as discussed in previous Sec. 4.2. Note that Wild-Place[15] dataset was collected using an inclined VLP-16 LiDAR mounted on a continuously spinning motor. This enabled the capture of the canopy of the forest, which is difficult for our backpack LiDAR setup. In the following we summarize the characteristic of the different forests:

Table 5.1: Summary statistics of evaluation datasets. Abbreviations used are as follows: Avg (Average),Desc (Descriptor evaluation), Online (Online SLAM), Offline (Offline multimission), Reloc. (Relocalization), Indiv. (Individual), Indiv.pc (Individual point clouds), tree density is roughly estimated by counting the average number of distinct trees within a 7 m radius from the center of the sensor then scaled up to per hectare.

Dataset	**Eval Modes**	**Missions name**	**Avg. Mapping area(ha)**	**Avg. Duration (min)**	**Avg. #Indiv. pc**	**Avg. #points per indiv.pc**	**Avg. #trees per hectare**
Stein am Rhein	Desc.	exp03	0.27 ha	13 min	363	120k	220
Wild-Place	Desc.	K-04	Perimeter 3.17km	48 min	5805	300k	270
Evo	Desc. Online. Offline.	exp16 exp18	0.74 ha	24 min	969	110k	500
Wytham Woods	Desc. Online. Offline.	C D E F	1.2 ha	22 min	707	55k	900
Forest of Deans	Online. Offline. Reloc.	exp01 exp02 exp03	0.45 ha	17 min	649	50k	130

- **Stein am Rhein:** Stein am Rhein dataset was collected in a forest in Switzerland in October 2023. The dataset was collected using the Hesai XT32 LiDAR sensor. It displays mixed species and bushes with a low tree density.

- **Wild-Place:** Wild-Place dataset is open-source forest dataset which was collected by the Australian research team in forests in Australia over different seasons. The dataset was collected using a VLP-16 LiDAR sensor mounted on a spinning motor. The trajectories were along the open access roads and the forest canopy was captured. The dataset is characterized by a low tree density.

- **Evo:** Evo dataset was collected in forests in Finland. The dataset was collected in May 2023 using the Hesai XT32 LiDAR sensor. The forest is characterized by tall coniferous trees with medium tree density.

- **Wytham Woods:** Wytham Woods dataset was collected in a densely wooded area in the UK in February 2024. The dataset was collected using the Hesai QT64

LiDAR sensor. It is characterized by a high tree density and with complex terrains featuring hills and valleys.

- **Forest of Dean:** Forest of Dean dataset was collected in a forest in the UK in March 2024. The dataset was collected using the Hesai QT64 LiDAR sensor. The dataset displays a sparser plantation with a low tree density.

Precision-Recall Curves

Precision-recall curves (See Fig. 5.2) show how accurately (precision) and frequently (recall) each model detects correct loop candidates within a reasonable distance threshold (here set to 10 m) at various descriptor thresholds τ_s in four different forests. Among positive candidates (whose descriptor distance $< \tau_s$), we classify them into true positive (TP) and false positive (FP) depending on whether the candidate is actually located on their ground truth location within the loop-closure distance threshold, 10 meters. Similarly, among negative candidates, we classify them into true negative (TN) and false negative (FN) depending on their true location.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN} \qquad \text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.1}$$

From the precision-recall curves (Fig. 5.2), it is evident that Logg3dNet consistently outperforms the other models across the four different forests. Particularly, on the Evo and Stein am Rhein datasets, where a longer range with narrow field-of-view Hesai XT32 LiDAR was employed, Logg3dNet showed the best performance both in terms of precision and recall, with a more gradual fall in precision and recall. In contrast, ScanContext has much lower precision. This is because a limited vertical field-of-view of LiDAR sensor would produce inconsistent height values for descriptor bins depending on different viewpoints and descriptors not being translational invariant would lead to incorrect detection of the candidate. In more challenging scenarios such as Wytham Woods, handcrafted models show a notable decline in performance. Meanwhile, Logg3dNet remains robust, successfully retrieving a substantial portion of correct loop-candidates, achieving a 70% precision at a 50% recall rate.

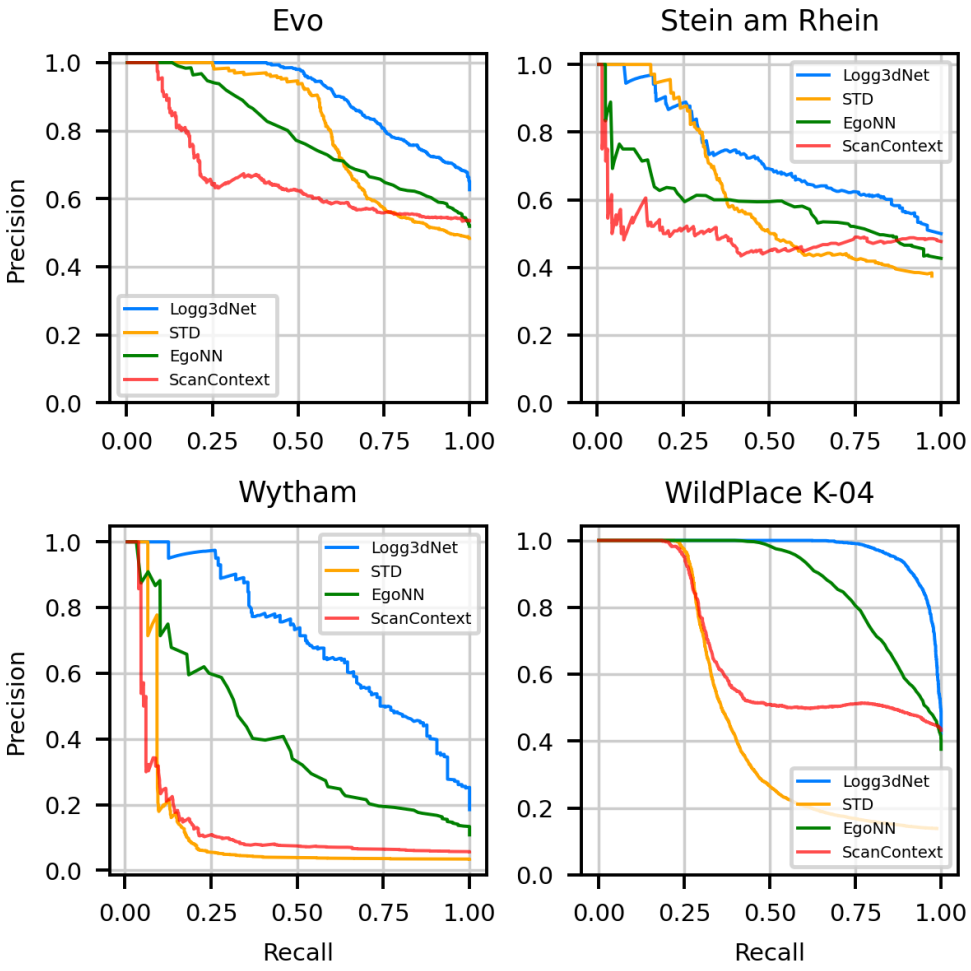

Figure 5.2: Precision-recall curves on four different forest datasets. Evo (Finland, May), Stein am Rhein (Switzerland, Oct), Wytham woods (UK, Feb), Wild-Places [15] (Australia). Evo and Stein am Rhein datasets were collected by Hesai XT32, and Wytham woods dataset was collected by Hesai QT64. Datasets were collected by backpack-carried LiDAR within dense forests. Only top-1 candidate within 10 m of the ground truth position is regarded as a true positive candidate.

Similarity matrix (heatmap)

To further analyze the distinctiveness of each descriptor, we measured the descriptor distances between all query and database descriptors. This is shown in Fig. 5.3 as a similarity matrix, which provides a visual representation of the discriminative potential of each descriptor. Consistent with the precision-recall curves, Logg3dNet descriptors exhibited higher similarity with the ground truth matrix as observed in the highlighted areas, indicating a high true-positive rate and low false-positive rate, respectively. This implies that Logg3dNet descriptors can effectively detect corresponding loop-candidates

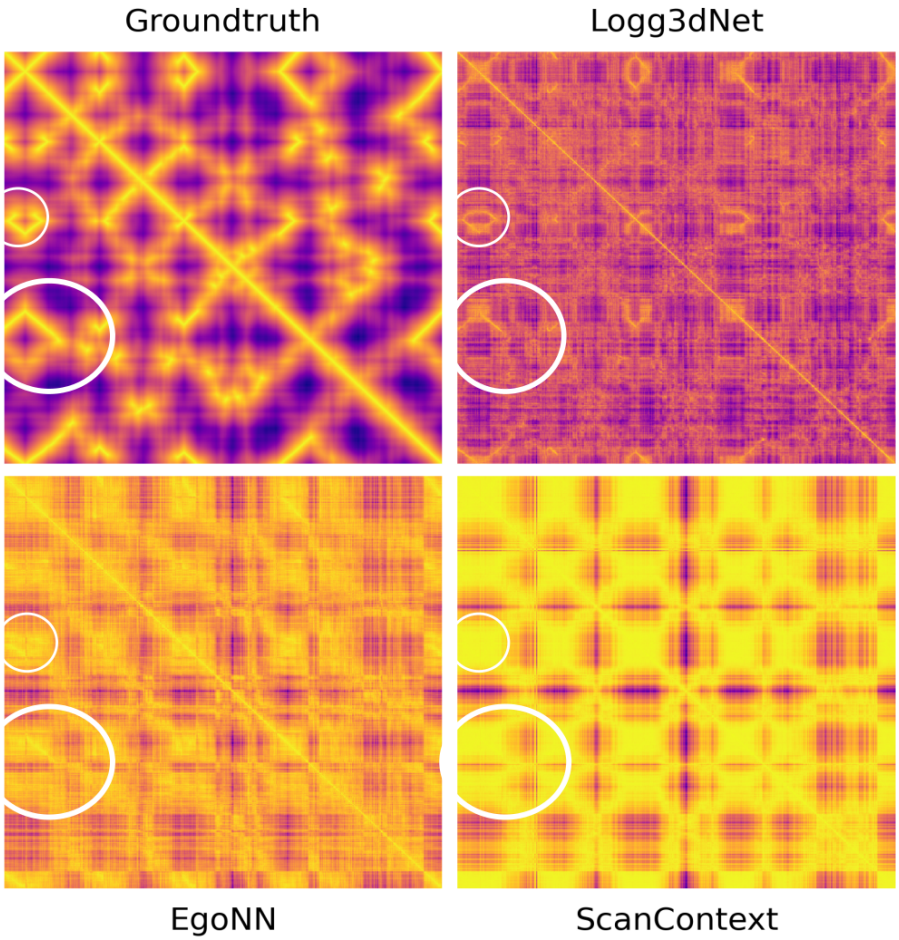

Figure 5.3: The top right image shows the ground truth distance matrix, where each element represents the Euclidean distance between two poses. Yellow hues depict poses less than 10 m apart, while farther poses are shown in purple shades. The remaining similarity matrices (heatmaps) depict the cosine distances between descriptors for the Evo dataset. Yellow hues indicate a high degree of similarity between descriptors of different scans, whereas purple shades represent low similarity. Patterns that more closely resemble the ground truth distance matrix (top-left) indicate better descriptor performance. The Logg3dNet descriptors exhibit patterns most similar to the ground truth, whereas the ScanContext descriptors are the least discriminative among these models. We use the value of τ_s that corresponds to the maximum F_1 score in the evaluation.

during revisits, whereas EgoNN and ScanContext tend to be less discriminative, often returning numerous false-positive candidates. Based on this evidence, we chose Logg3dNet as main the place recognition method for the rest of the experiments.

5.2 Online Single Mission SLAM

In this experiment, we tested our complete pipeline (described in Sec. 3.3) and investigate its online place recognition capability, wherein loop closures from the place recognition

module are integrated into VILNES SLAM system. First, illustrative results of loop closures obtained from different forest datasets are shown and discussed. Then, we further analyzed the loop closure statistics by distance and viewpoint differences. Lastly, we compared place recognition performance with handcrafted model, *ScanContext*.

Fig. 5.5 presents an illustrative example of online SLAM running on the Evo dataset sequence. In this mode, we can see loop closures (red lines) inside the dense forest between previous scans and the current scan. We can achieve these loop closures within the dense forest environment between previous scans and the current scan without having to revisit or retrace the previously traversed paths.

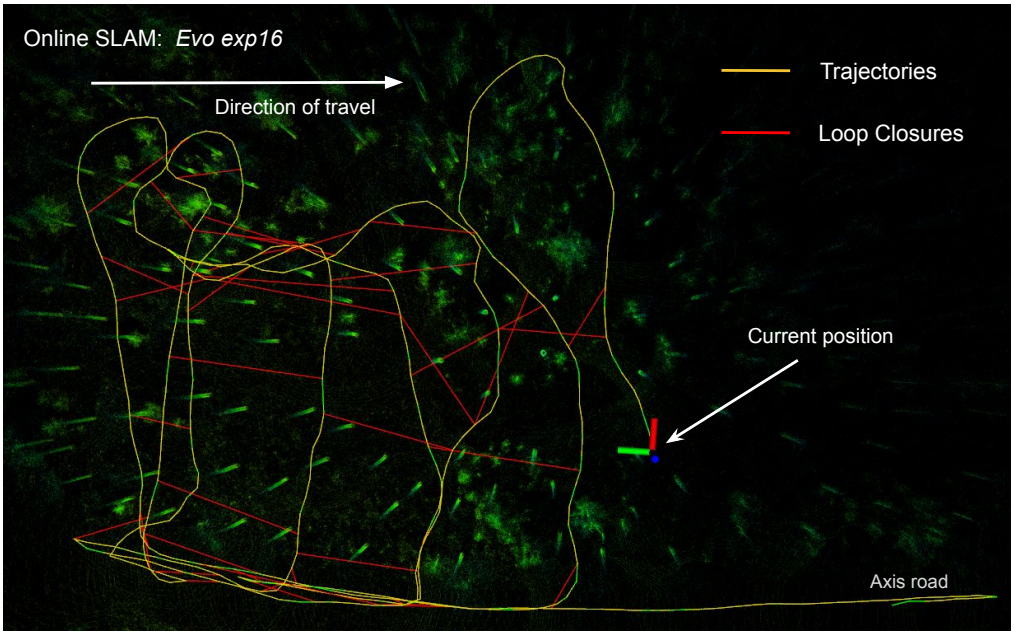

Figure 5.4: Illustration of online SLAM mode running with the *place recognition server* on the Evo exp16 sequence. Red lines are loop closures successfully captured by the place recognition module while mapping the forest (yellow trajectories).

Online Place Recognition Results

In this section we analyzed how the loop candidates are obtained and verified at each stage of the pipeline. To summarize typical performance, we analyzed each of three steps of the pipeline: candidate proposal, coarse registration, and fine registration.

- **Step 1: Loop candidate proposal (Blue)** Many loop closure candidates are initially proposed (shown with blue lines in Fig. 5.5 and Fig. 5.6) under a descriptor

matching threshold τ_s of F_1-max score. Next, we simply use a fixed threshold of 20 m and reject loop closures beyond this conservative estimate using the odometry information, assuming that the accumulated drift over such short traverses is relatively small compared to the 20 m threshold. It is important to note that for much longer traverses where significant drift is expected, using the marginal covariance of the pose graph to reject statistically improbable loop closure candidates would be a more appropriate. Additionally, the database D is incrementally built as the sensor moves through the environment, and the most recent 30 seconds of data are excluded to prevent loop closures with immediately recent measurements.

- **Step 2: Coarse Registration (Orange)** A subset of the initial loop closure candidates from Step 1 are identified using RANSAC feature matching (highlighted in orange lines in Fig. 5.5 and Fig. 5.6). These candidates are then verified using SGV and pairwise consistency checks.

- **Step 3: Fine Registration (Red)** The loop closure candidates that pass the consistency and ICP (Iterative Closest Point) steps are integrated into the SLAM framework. The final loop closures (shown in red lines in Fig. 5.5 and Fig. 5.6) are the ICP verified loop candidates from Step 2. At this stage, we also reject any additional loop closures from similar positions to avoid over-constraining the pose graph (pose graph constraint density filter).

Evo (Fig. 5.5) We tested on Evo dataset as shown Fig. 5.5. Evo16 sequence was densely scanned where each zigzag turn is about 10-15m apart, whereas Evo12 was mapped sparsely with about 20-30m apart. For Evo16, we can observe frequent loop candidates up to 20 m between each zigzag turn, whereas for Evo12, fewer final loop closures were found than for Evo16. Initial candidates with an inter-pose distance above 20 m are strongly rejected in the earlier stages of odometry estimate check and verification layers. In both experiments, we observe that the system can reliably detect loop closures inside the dense forest and correct the SLAM system's drift.

Wytham Woods (Fig. 5.6) We further tested on a more challenging dataset taken in Wytham Woods, which is a densely wooded area with uneven terrain, including

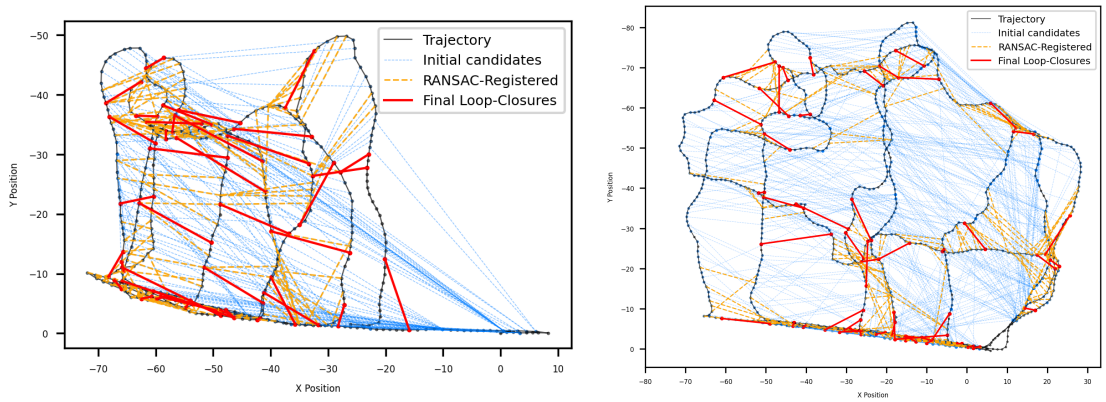

Figure 5.5: Online SLAM results on Evo dataset. Evo16 (Left) and Evo12 (Right). Initial candidates (Blue) proposed by descriptor distance and odometry check. Yellow lines are RANSAC-registered candidates which have successfully passed SGV and pairwise consistency checks. Red lines are final loop closures after ICP verification and pose graph constraints density filter in pose graph.

hills. In Fig. 5.6, we demonstrate that a sufficiently large numbers of correct RANSAC-registered loop candidates, but ICP could not fine-register them. For example, regarding Mission D and C shown in Fig. 5.6, our system correctly detected loop candidates and RANSAC registered them (Yellow lines) at each turn. However, the ICP rejected almost all candidates. This was due to very small number of overlapping point clouds between two scans (usually 2k-3k out of 20k points, ∼10 % overlaps). Unfortunately, lowering ICP thresholds to less than ∼ 3k points to accept these candidates introduced false positives, resulting in a failure of system. In this dataset we see that our system suffers from a high rate of outliers due to occlusions and the large pose-to-pose translation at the far distance making ICP registration challenging. In summary, our system can correct drift in both sequences but is unable to find loop closures at the start.

Loop Closure Statistics

Alongside visual illustrations of loop closures, we conducted a comprehensive analysis of loop closure statistics based on distance and viewpoint angles, shown Fig. 5.7. Our findings show that the system can successfully identify loop closure pairs when there are considerable pose-to-pose baseline distances (10–20 m).

Baseline distance We observed that despite the large baseline, a significant portion of initial candidates can be registered using RANSAC-based matching, indicating that

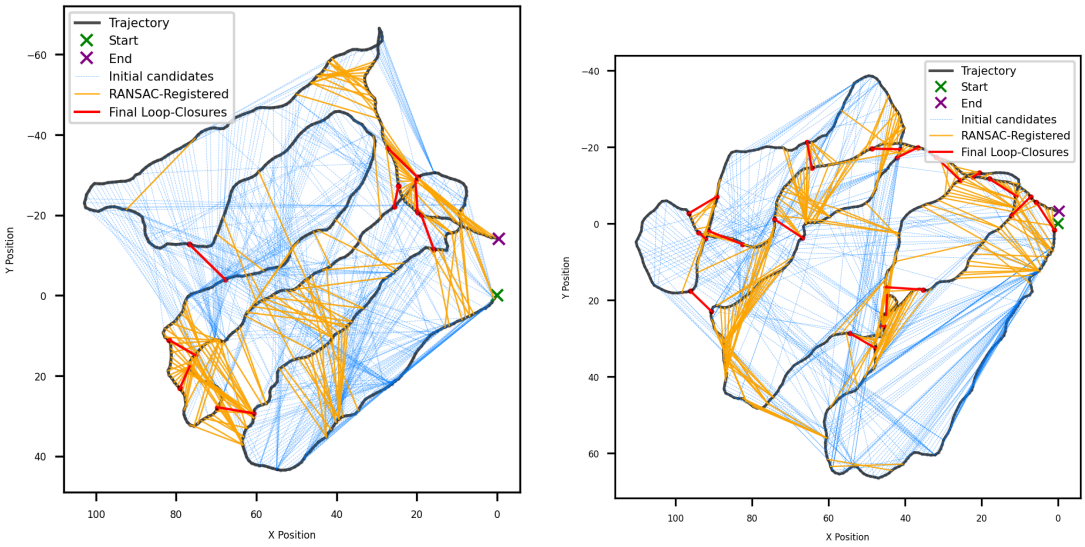

Figure 5.6: Online SLAM results from Wytham Woods, two sequences Mission D(Left) and Mission C(Right) are shown. Light blue lines are initial candidates proposed by descriptor distance and odometry check. Yellow lines are successful RANSAC-registered candidates which passed the SGV and pairwise consistency checks. Red lines are final accepted loop closures after ICP verification and checking constraints density in pose graph. In both missions, ICP struggled to verify loop candidates from correctly RANSAC-registered due to the slow degree of overlap in the point clouds.

the correspondences are accurate. However, the proportion of candidates verified by ICP decreases as the distance between scans increases. Specifically, when scans are 10 m apart, ∼60 % of RANSAC-registered candidates are successfully verified by ICP, and when 15 m apart, only ∼40 % remains verified. This decrease is due to the diminishing overlap ratio between corresponding scans with increasing distance, making the accurate convergence of ICP challenging.

Viewpoints difference Similarly, regarding viewpoint orientation differences, we observed that a significant proportion of loop candidates, up to a 90° difference, were verified during the RANSAC-registration stage, constituting approximately 80–50 % of the initial candidates. Subsequently, during ICP inliers checks, approximately 80–60 % of those RANSAC-registered candidates were finally verified.

However, we observed a degradation in proportion of candidates beyond 90° to 180° at both the RANSAC-registration and ICP checks. The proportion of RANSAC-registered candidates decreased to below ∼30 % of the initial candidates, likely due to the rotational invariance limitations of the Logg3dNet descriptors. The proportion of

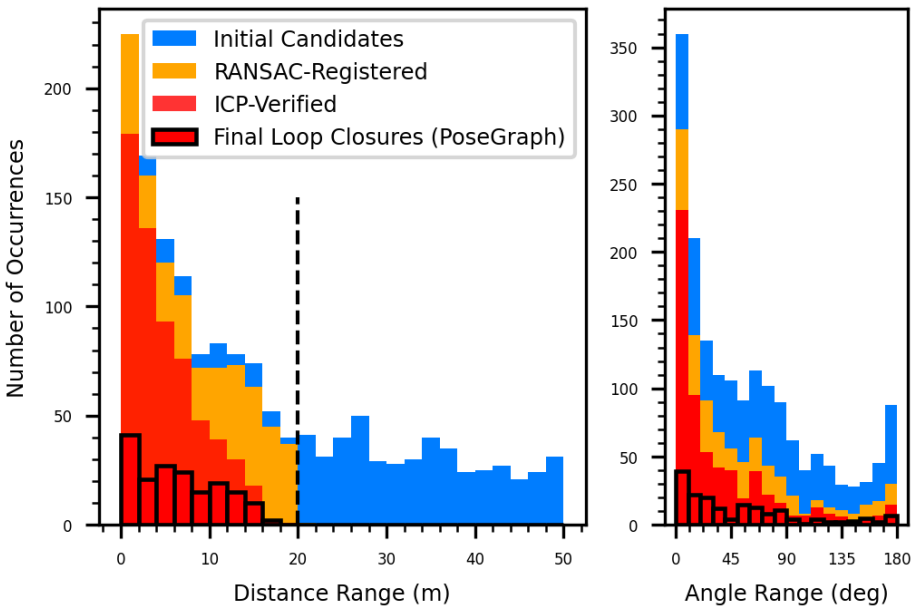

Figure 5.7: Loop closures distribution by distance and angle at various stages of the pipeline on Evo dataset. Initial candidates based on descriptor distance are shown in blue. Candidates beyond 20 m are rejected using odometry information. Candidates within 20 m undergo RANSAC pre-registration with additional verification steps of SGV[27] and pairwise checks (Yellow). Then these candidates are refined using ICP fine-registration (Red), and final loop closures in the pose graph after checking constraints density in pose graph (red with black outlined). Both of these categories count as success.

ICP-verified candidates from RANSAC-registered candidates also declined to 40–10 %, which we assume is due to the small number of overlapping inlier points in the occluded point clouds captured from different viewpoints and locations. Nonetheless, despite this degradation, the number of final loop closures integrated into the SLAM system proved to be sufficient for correcting the drift.

Model Comparison: ScanContext

So far we have shown capability of Logg3dNet reliably finding loop closures inside the forest. We now show how ScanContext performs (again integrated within our SLAM system) to provide a comparison to Logg3dNet-based system. Fig. 5.8 shows ScanContext's loop closure results on the Evo dataset. The figure can be directly compared with Fig. 5.5 (Left image) to see the difference in loop closure performance between Logg3dNet and ScanContext. From Fig. 5.8, we can observe that ScanContext could not detect any loop closures inside the dense forest producing a very high number of false positives

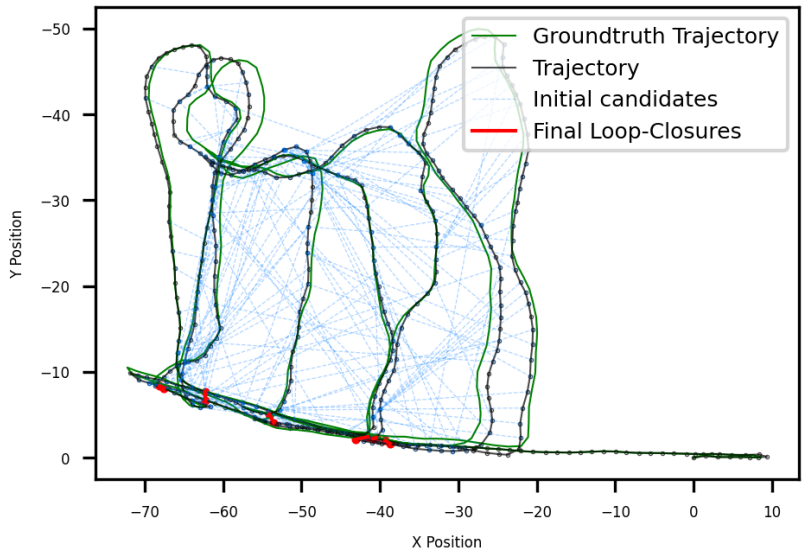

Figure 5.8: Loop closures with ScanContext on Evo exp16 dataset. ScanContext could not detect any loop closures inside the dense forest but only in the open access road, whereas Logg3dNet successfully found loop closures inside the forest (as shown in previous Fig. 5.5). Additionally, the ScanContext loops were only found within 3 m (same location), whereas Logg3dNet detected loop closures up to 15 m. Because of this the Scancontext-enabled SLAM system accumulates drift over time.

(shown in blue lines), and could only achieve successful loop closures in the open access roads(red lines at the bottom of the plot). Among those loop closures along open access roads, ScanContext detected only within 3 m (same location). Finally, ScanContext accumulated drift over time while Logg3dNet corrected this drift. This demonstrates the very limited capability of ScanContext compared to Logg3dnet, and this experiment validates the robustness of Logg3dNet compared to ScanContext (handcrafted method) in dense forest environments. It was evident ScanContext shows a drop in performance in 5.1, and now we evaluate if ScanContext can detect any loop candidates and find 6DoF transformation reliably inside the forest.

5.3 Offline Multi-Mission SLAM

In this experiment, we showcased the ability of our approach to obtain loop closures between different mapping missions (e.g taken on different days) and to merge those missions into a common map. Below shows the results of merging different sequences within three different datasets: Wytham Woods, Evo, and the Forest of Dean. Each

individual mission covers approximately one hectare, with merged map areas ranging from three to five hectares.

Evo (Fig. 5.9) Evo forest primarily comprises tall, mature coniferous trees. We mapped a total of 6 missions in the main site and successfully merged them. In particular, we tested the robustness of our system by placing XT32 LiDAR at a 45° inclination aimed at more densely capturing the forest canopy. Despite the asymmetry in point clouds introduced by this inclination change, which primarily captured points in the forward direction, our approach successfully identified loop closures and achieved multi-mission map merging.

Wytham Woods (Fig. 5.10) Wytham Woods are the most challenging due to having the highest tree density, as well as the foliage, and vegetation present. Four multi-missions were captured with wide field-of-view Hesai QT64 LiDAR. Despite the challenges of the test site, our system successfully identified loop closures between different missions and merged them as shown in Fig. 5.10. During individual mission processing, slight odometry drift was noted in the SLAM system both at the beginning and end of each mission. However, during the multi-mission map merging process, the system successfully corrected this drift occurring in overlapping areas.

Forest of Dean (Fig. 5.11) Forest of Dean is the least dense forest and a total of 3 missions collected there were merged easily. We observed that there were more frequent *intermission* loop closures in the Forest of Dean compared to Wytham in overlapping areas. This is because more widely spread tree locations and minimal foliage produced much fewer occlusions and cleaner point clouds with higher number of overlapping points between two intermission scans. This favorable environment of Forest of Dean makes multi-mission map merging process easier when registering point clouds during the RANSAC-registration and ICP steps of loop candidates.

Merged map quality (Fig. 5.12) The quality of the merged maps were examined by checking point clouds of trees. We observed distinct tree trunks has no apparent signs of drift. The circular shapes of tree trunks were clearly seen in the point clouds, indicating that the map scanned in different positions were registered correctly.

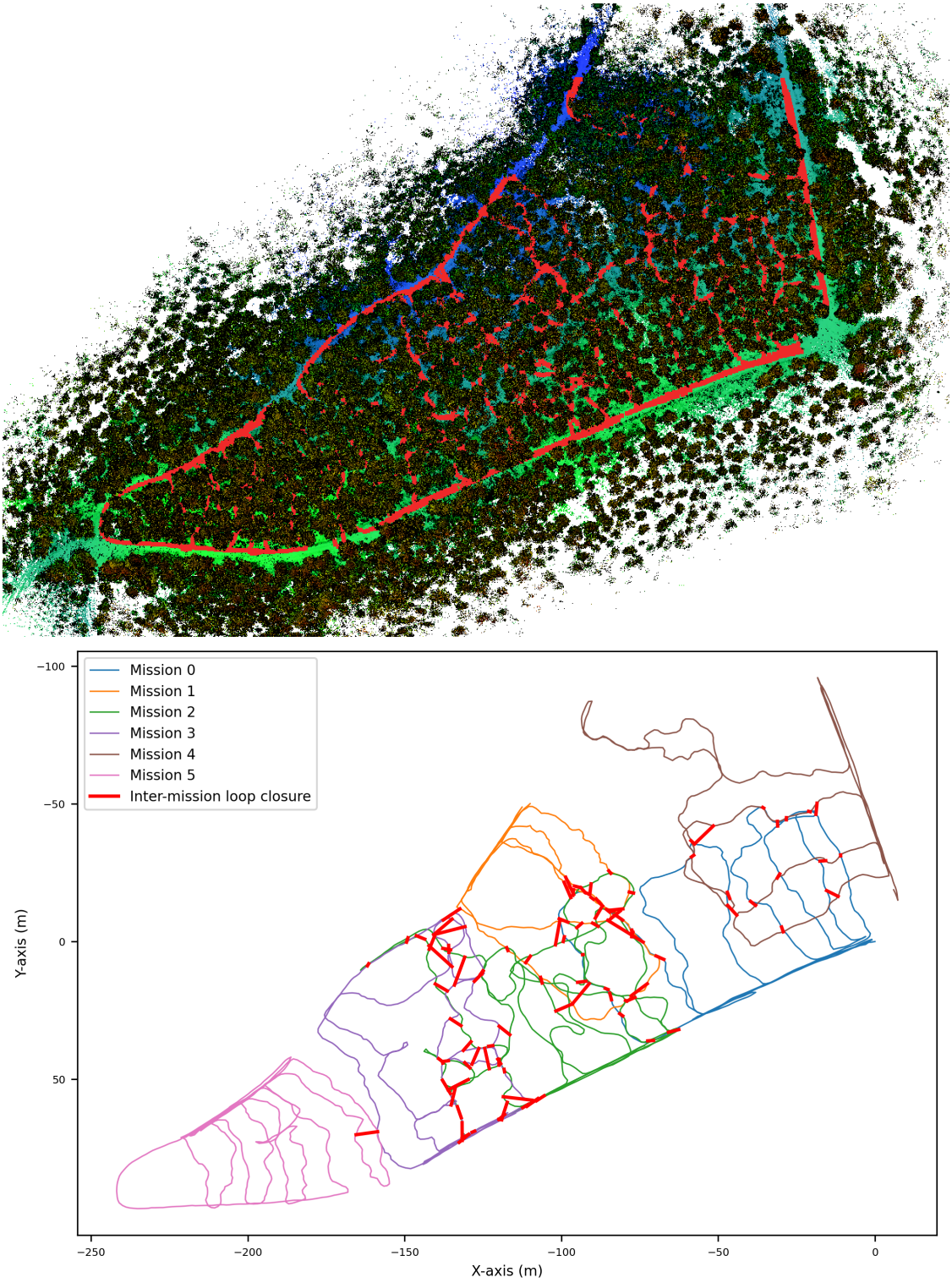

Figure 5.9: Offline multi-mission SLAM (Evo). Merged point clouds (top) show successful loop closures between different missions. Inter-mission loop closures(bottom) identified when viewpoints are closely aligned. This is due to the 45° inclination of the LiDAR used in the main test site of Evo.

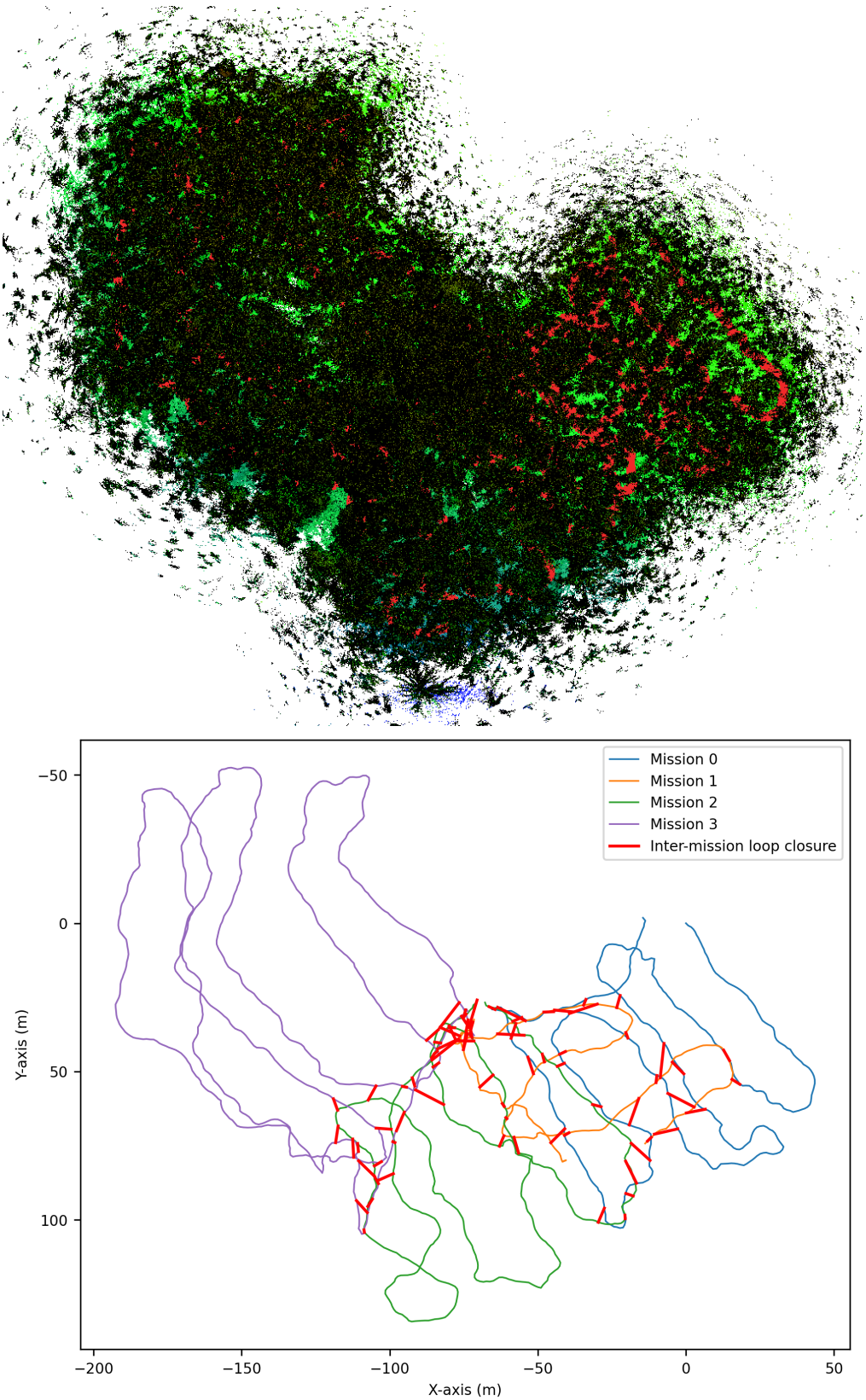

Figure 5.10: Offline multi-mission SLAM (Wytham Woods). Merged point clouds map(top) shows successful loop closures (bottom) between the different missions in a densely wooded area with high tree density.

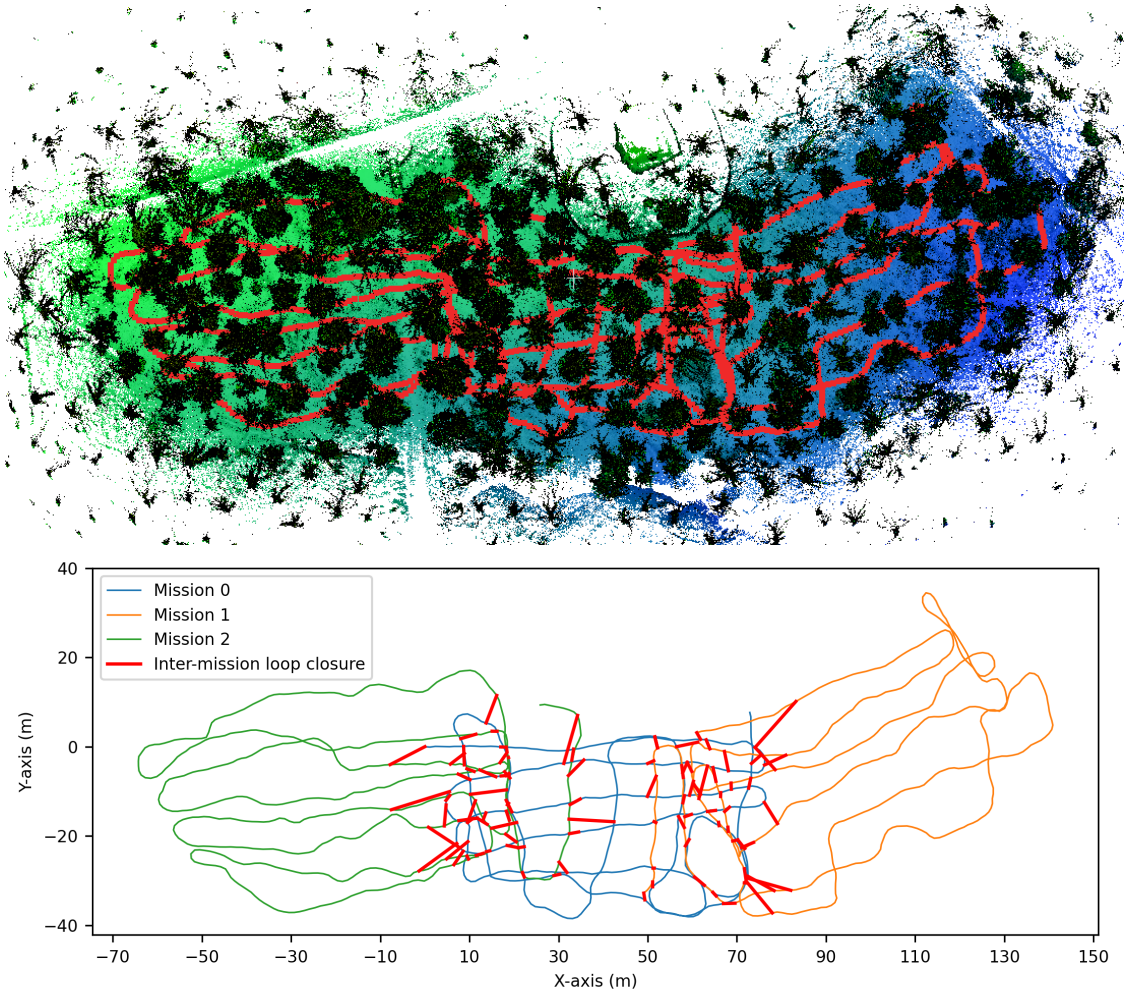

Figure 5.11: Offline multi-mission SLAM(Forest of Dean). The merged map (top) shows successful loop closures (bottom) between the different missions. This plantation is much sparser than the others.

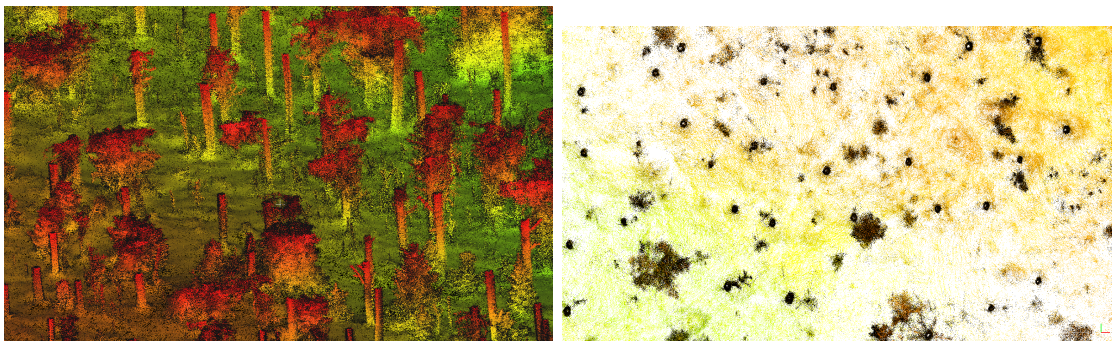

Figure 5.12: Merged map quality. Trunks of trees from a bird's-eye view (right image). We observed the circular shapes of tree trunks clearly seen in the point clouds, indicating the map scanned in different viewpoints were registered correctly.

Overall, our experiments showed a potential capability for efficient large-scale mapping without any specific condition as to the route take. In particular, we did not need to start at the open access roads nor follow exactly the same paths to achieve loop closures between inter-missions.

5.4 Relocalization

This experiment showcases a demonstration of relocalization in a prior map. In this scenario, our system demonstrates the ability to continuously track the position of the mapping device within a prior SLAM map once an initial loop closure is established. Unlike the multi-mission SLAM use case, the prior map is now fixed as ground truth. To reiterate - the prior map (in green tree point clouds in Fig. 5.13) is made up of individual scans with a descriptor rather than a single giant point cloud. We set the pairwise consistency threshold to ∼10 cm to enable consistent relocalization only within centimeter level of error.

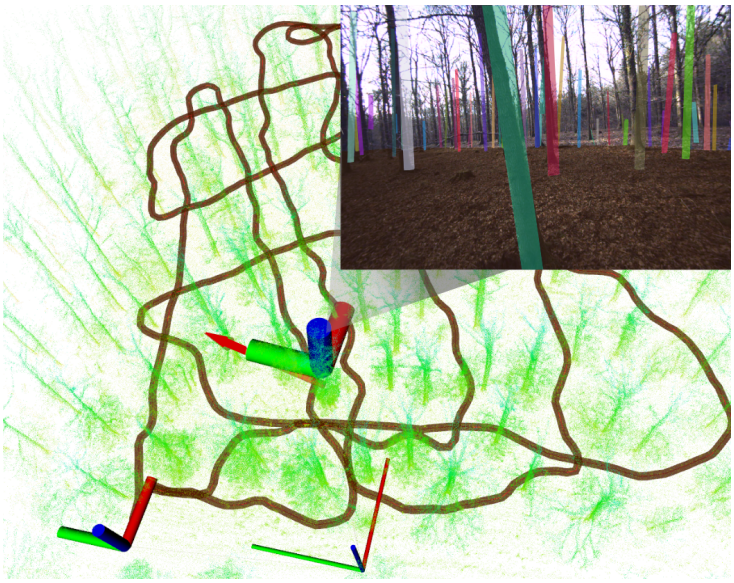

Figure 5.13: Demonstration of relocalization capability. The LiDAR sensor (illustrated by large marker) is relocalized in a prior map (dark brown). We render a virtual view of the forest digital map synchronized with images from our camera (up right image). The virtual tree models were generated by fitting cylindrical shapes to individual trees in the point cloud data, as described in [10]. Red arrow shows a successful localization at that pose.

Forest Surveying Demonstration Fig. 5.14 present illustrative examples of this demonstration, where the quadruped robot is localized with respect to a prior SLAM map generated using a backpack LiDAR system. Drift accumulated by the robot was

corrected by finding loop candidates in the prior map of individual scans as shown in Fig. 5.14. The left image shows the incorrect position of virtual tree model due to incorrect state estimate in odometry system. After relocalization, the virtual tree model is rendered again at the correct relocalized position.

Overall, this capability enables the real-time rendering of a virtual forest map overlaid with associated data, such as Diameter at Breast Height (DBH) and species information, onto the camera images. Such feedback is highly beneficial for tasks such as taking forest inventories by foresters or enabling autonomous harvesting.

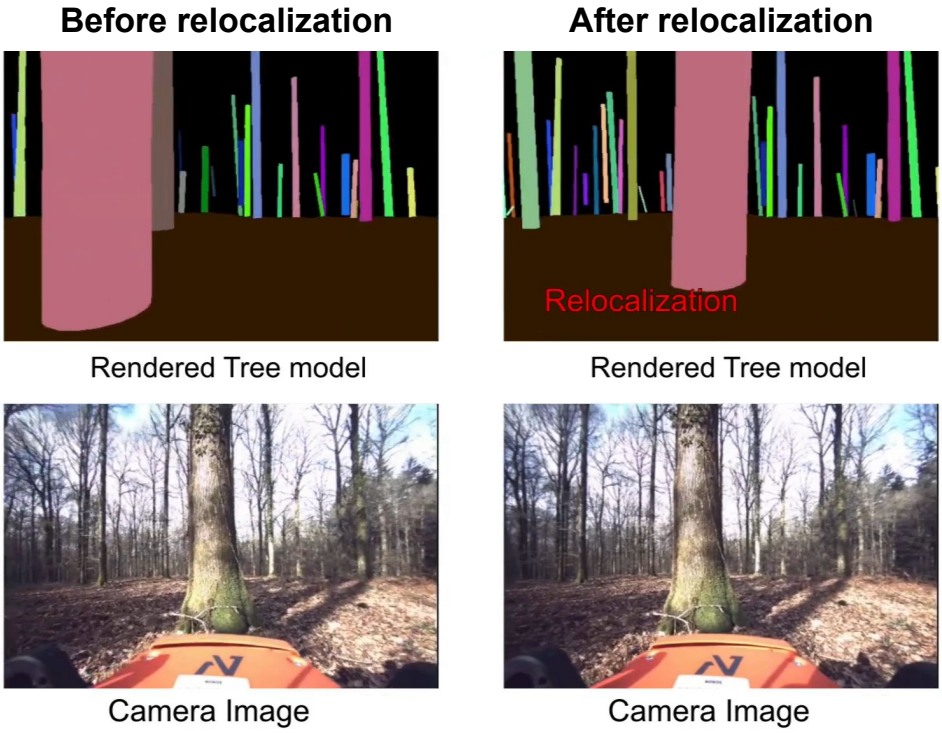

Figure 5.14: Demonstration of relocalization of a quadruped robot before and after. Left image shows incorrect position of virtual tree model due to drift accumulated in odometry. Right image shows correct position of virtual tree model after relocalization.

Relocalization Failure In the relocalization scenario, relocalizing into an incorrect candidate leads to a sudden jump in current position (shown in left image in Fig. 5.15). This is easily detected by checking pairwise cycle consistency between map-base $\Delta\mathbf{T}_{\mathtt{MB}}$ and odometry-base frame $\Delta\mathbf{T}_{\mathtt{OB}}$, and effectively prevent the relocalization failure by 50 %. Fig. 5.15 shows an example of incorrect localization with an incorrect loop candidate (Left). Right image shows pairwise cycle consistency verification filtering

out incorrect loop candidate.

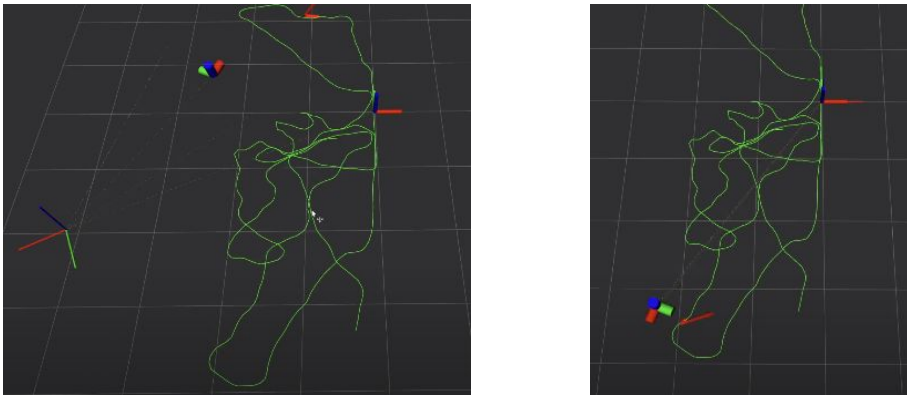

Figure 5.15: Left: relocalization failed due to incorrect loop candidate. Right: after filtering out incorrect loop candidate by pairwise cycle consistency checking.

5.5 Parameter analysis & Computational Analysis

In this section, we will discuss our final verification step of the ICP inlier-based check in more detail. We will also provide a brief computational analysis of the system, including runtime and memory consumption.

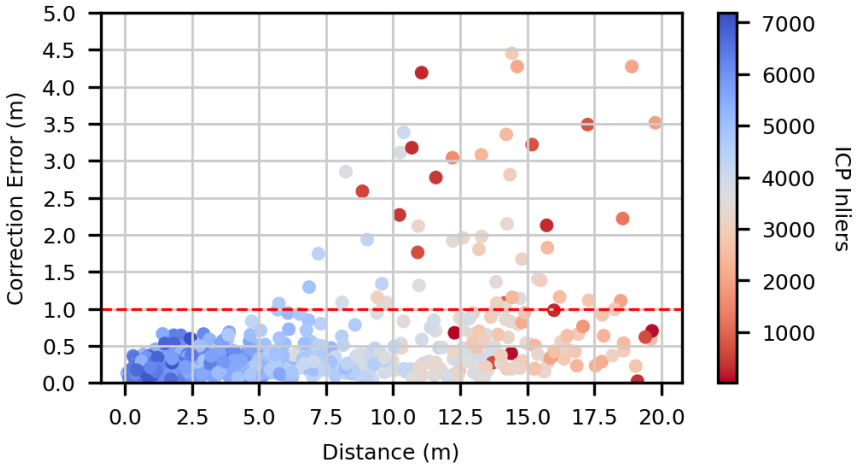

Figure 5.16: Analysis of final ICP registration check. X-axis shows the distribution of loop-candidates by distance after ICP. Y-axis shows ICP correction error w.r.t. the coarse-registration from RANSAC. Color indicates the number of ICP inliers, 30 iterations and RMSE= 0.01m, where clouds are cropped 50 by 50m, then downsampled to 20k points.

Study of ICP inlier-based check Our final experiment investigates the effect of the ICP inlier-based check on loop closure integration within the final pose graph optimization.

Any incorrect loop closures should never be introduced into the optimized pose graph. We will consider this in the context of the online SLAM scenario.

Fig. 5.16 illustrates the corrections (on top of the initial transformation prior) as estimated by the ICP registration at various distances. Additionally, the figure color-codes the points based on the number of inlier points obtained during the registration process, with blue indicating a large number of inliers and red indicating a smaller number. Our observation suggests that loop closures occurring more than 10 m apart, which also propose a substantial transformation correction often have fewer inlier ICP points and are thus less reliable. Based on this analysis, we need to establish an inlier threshold to ensure that corrections are limited to ICP corrections of less than 1 meter (shown in dotted red line). We will further discuss how to set this threshold in the following section.

Setting ICP inlier thresholds The ICP inlier threshold is the minimum number of points required for fine registration loop candidates. Only if the number of inliers turned out to be higher than a threshold will we accept the loop closure candidate. The ICP threshold should be fine-tuned to $\sim$ 3k-5k points depending on the LiDAR setup and forest environments. There is trade-off between the maximum baseline distance and the robustness of the loop-closure as shown in Fig. 5.17. For an example, in the Evo dataset, we set the threshold to 3k points, whereas in the Wytham Woods, we set the higher threshold to 4k-5k points.

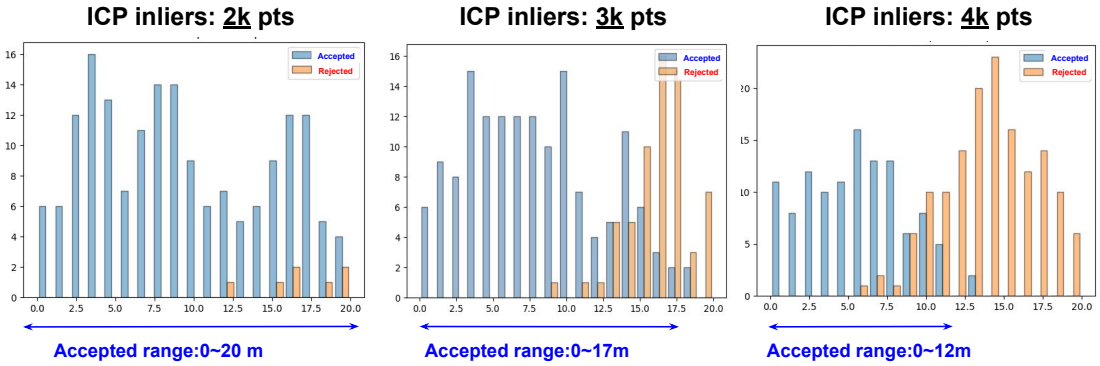

Figure 5.17: Loop closure distribution by distance at various ICP inlier thresholds. Blue lines are accepted loop candidates, and red lines are rejected candidates. A stricter (higher) ICP inlier threshold will provide more robust loop closures, but the baseline distance range will be shorter, whereas lower ICP inliers will allow loop closures with a larger baseline distance, but there is an increased chance of getting false positive loop candidates.

Runtime Analysis Currently the system requires a GPU for descriptor extraction. The system runs at 0.73 Hz on a laptop with an Intel i7-10750H CPU and an NVIDIA RTX A3000 GPU. The processing time for the top-1 loop closure candidate detection is shown in Tab. 5.2. The descriptor extraction and RANSAC featuring matching are the most time-consuming process. While these results are specific to the laptop configuration, a rate close to 1 Hz is sufficient for real-time applications.

Process	**Preprocessing**	**Descriptor**	**SGV**	**RANSAC**	**Total**
Avg. Time (s)	0.015	0.09	0.001	1.1	1.3 /0.73 Hz

Table 5.2: Avg. processing times for various tasks. The most time-consuming process is RANSAC-based feature matching. The system runs at 0.73 Hz.

Memory Consumption The global descriptor has a dimensionality of (1,1028), requiring 8 bytes of memory, while the local descriptor, with dimensions (N, 256) (where N is the number of keypoints), consumes approximately 2KB of memory each. GPU memory usage, primarily dominated by local descriptors, totals around 2GB for mapping a typical single mission comprising about 1000 scans.

Voxel Size During preprocessing, voxelizing input point clouds is essential for utilizing *MinkowskiEngine [1], SparseTorch [23] with CUDA*. The voxel size significantly impacts model performance. Since both Logg3dNet and EgoNN are trained on Wild-Place dataset, which is different from our dataset in terms of density and range of LiDAR point clouds, we empirically identified the optimal voxel size between 0.1–1.0 m through direct experimentation on our dataset. Tab. 5.3 shows that optimal voxel sizes are 0.1 m and 0.6 m for Logg3dNet on Evo dataset. Since the difference is not significant, after considering we decided to use voxel size of 0.6 m considering computational efficiency.

Testing	Voxel size (cm)					
Datasets	10	20	40	60	80	100
Evo-12	**0.89**	0.79	0.77	**0.86**	0.79	0.78
Evo-16	**0.74**	0.66	0.66	**0.66**	0.64	0.63
Evo-18	0.71	0.75	**0.75**	**0.76**	0.73	0.71

Table 5.3: $F1_{max}$ score of Logg3dNet with different voxel sizes on Evo dataset.

6
Conclusion

In this thesis, we conducted extensive testing of LiDAR place recognition systems in dense forest environments. We presented a place recognition and verification system that leverages three stages of loop-candidate verification: at the global descriptor-level, local feature-level, and fine point cloud level. These place recognition modules were seamlessly integrated into a pose graph SLAM system and evaluated across three distinct scenarios: online SLAM, offline multi-mission SLAM, and relocalization. Our experiments provide further insights on the performance of currently available LiDAR-based place recognition methods in dense forests. We also studied three different applications which could take advantage of this 6-DoF localization, opening future applications for forest inventory, inspection, and autonomous tasks.

In future work, we plan to use this system to function as a standalone *place recognition server*, allowing for easier integration of new LiDAR place recognition methods and to make it available for use by other members of the DRS group. We also plan to deploy the place recognition server in real-time on a Frontier device with an NVidia Jetson GPU board in future field trials.

Bibliography

[1] Christopher Choy et al. “4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks”. In: *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* 2019 (page 47).

[2] Kavisha Vidanapathirana et al. “Locus: Lidar-based place recognition using spatiotemporal higher-order pooling”. In: *IEEE Int. Conf. Robot. Autom.* 2021 (page 5).

[3] Kavisha Vidanapathirana et al. “LoGG3D-Net: Locally guided global descriptor learning for 3D place recognition”. In: *IEEE Int. Conf. Robot. Autom.* 2022 (pages 3–5, 12).

[4] R. Dube et al. “SegMap: 3D Segment Mapping using Data-Driven Descriptors”. In: *Proc. of Robotics: Science and Systems (RSS)* (2018) (page 6).

[5] R. Dube et al. “SegMatch: Segment Based Place Recognition in 3D Point Clouds”. In: *IEEE Int. Conf. Robot. Autom.* 2017 (page 6).

[6] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. “NetVLAD: CNN Architecture for Weakly Supervised Place Recognition”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2018) (page 5).

[7] J. Behley and M. Garbade et al. “SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences”. In: *IEEE Int. Conf. Comput. Vis.* 2019 (page 6).

[8] Paul J. Besl and Neil D. McKay. “A Method for Registration of 3-D Shapes”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* (1992) (page 17).

[9] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* (1981) (page 14).

[10] Leonard Freißmuth, Matias Mattamala, Nived Chebrolu, Simon Schaefer, Stefan Leutenegger, and Maurice Fallon. *Online Tree Reconstruction and Forest Inventory on a Mobile Robotic System.* 2024 (page 43).

[11] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J. Leonard, and Frank Dellaert. “iSAM2: Incremental smoothing and mapping using the Bayes tree”. In: *Intl. J. of Robot. Res.* 31.2 (2012), pp. 216–235 (pages 9, 10).

[12] Giseop Kim, Sunwook Choi, and Ayoung Kim. “Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments”. In: *IEEE Trans. Robot.* (2021) (page 4).

[13] Giseop Kim and Ayoung Kim. “Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map”. In: *IEEE/RSJ Int. Conf. Intell. Robots Syst.* 2018 (pages 3, 4, 12).

[14] Giseop Kim, Yeong Sang Park, Younghun Cho, Jinyong Jeong, and Ayoung Kim. “MulRan: Multimodal Range Dataset for Urban Place Recognition”. In: *IEEE Int. Conf. Robot. Autom.* 2020 (page 6).

[15] Joshua Knights and Kavisha Vidanapathirana et al. “Wild-Places: A Large-Scale Dataset for Lidar Place Recognition in Unstructured Natural Environments”. In: *IEEE Int. Conf. Robot. Autom.* 2023 (pages 2, 6, 13, 28, 31).

[16] Jacek Komorowski. “MinkLoc3D: Point Cloud Based Large-Scale Place Recognition”. In: *IEEE Winter Conf. Appl. Comput. Vis.* 2021 (page 5).

[17] Jacek Komorowski, Monika Wysoczanska, and Tomasz Trzcinski. "EgoNN: Egocentric Neural Network for Point Cloud Based 6DoF Relocalization at the City Scale". In: *IEEE Robot. Autom. Lett.* (2022) (pages 4, 5, 12).

[18] Lin Li and Xin Kong et al. "SSC: Semantic Scan Context for Large-Scale Place Recognition". In: *IEEE/RSJ Int. Conf. Intell. Robots Syst.* 2021 (page 4).

[19] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. "1 year, 1000 km: The Oxford RobotCar Dataset". In: *Int. J. Rob. Res.* (2017) (page 6).

[20] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. "Comparing ICP Variants on Real-World Data Sets". In: *Auton. Robot.* (2013) (page 17).

[21] Alexander Proudman, Milad Ramezani, Sundara Tejaswi Digumarti, Nived Chebrolu, and Maurice Fallon. "Towards real-time forest inventory using handheld LiDAR". In: *Robot. Auton. Syst.* (2022) (pages 7, 22).

[22] Filip Radenović, Giorgos Tolias, and Ondřej Chum. "Fine-Tuning CNN Image Retrieval with No Human Annotation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2019) (page 5).

[23] Haotian Tang and Yang et al. "TorchSparse++: Efficient Training and Inference Framework for Sparse Convolution on GPUs". In: *Proc. of the ACM Intl. Conf. on Mobile Computing and Networking (MobiCom)*. 2023 (page 47).

[24] G. Tinchev, A. Penate-Sanchez, and M. Fallon. "Learning to see the wood for the trees: Deep laser localization in urban and natural environments on a CPU". In: *IEEE Robot. Autom. Lett.* (2019) (page 6).

[25] Georgi Tinchev, Simona Nobili, and Maurice Fallon. "Seeing the Wood for the Trees: Reliable Localization in Urban and Natural Environments". In: *IEEE/RSJ Int. Conf. Intell. Robots Syst.* 2018 (page 6).

[26] Samuel Triest and Matthew Sivaprakasam et al. "Tartandrive: A large-scale dataset for learning off-road dynamics models". In: *IEEE Int. Conf. Robot. Autom.* 2022 (page 6).

[27] Kavisha Vidanapathirana and Peyman Moghadam et al. "Spectral Geometric Verification: Re-Ranking Point Cloud Retrieval for Metric Localization". In: *IEEE Robot. Autom. Lett.* (2023) (pages 15, 37).

[28] Han Wang, Chen Wang, and Lihua Xie. "Intensity Scan Context: Coding Intensity and Geometry Relations for Loop Closure Detection". In: *IEEE Int. Conf. Robot. Autom.* 2020 (page 4).

[29] David Wisth, Marco Camurri, and Maurice Fallon. "VILENS: Visual, Inertial, Lidar, and Leg Odometry for All-Terrain Legged Robots". In: *IEEE Trans. Robot.* (2023) (pages 7, 8, 22).

[30] Yan Xia and Yusheng Xu et al. "SOE-Net: A Self-Attention and Orientation Encoding Network for Point Cloud Based Place Recognition". In: *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* 2021 (page 5).

[31] Chongjian Yuan and Lin et al. "STD: Stable Triangle Descriptor for 3D place recognition". In: *IEEE Int. Conf. Robot. Autom.* 2023 (pages 4, 12).

[32] Wenxiao Zhang and Chunxia Xiao. "PCAN: 3D attention map learning using contextual information for point cloud based retrieval". In: *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* 2019 (page 5).

[33] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018) (page 14).

[34] Zhicheng Zhou and Cheng Zhao et al. "NDT-Transformer: Large-Scale 3D Point Cloud Localisation using the Normal Distribution Transform Representation". In: *IEEE Int. Conf. Robot. Autom.* 2021 (page 5).