Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/XXXX.2022.1234567

TreeLoc++: Robust 6-DoF LiDAR Localization in Forests with a Compact Digital Forest Inventory

Minwoo Jung[1], Dongjae Lee[1], Nived Chebrolu[2], Haedam Oh[3], Maurice Fallon[3] and Ayoung Kim[1]

[1]Department of Mechanical Engineering, Seoul National University, Seoul, South Korea
[2]Department of Computer Science and Engineering, Indian Institute of Technology Bombay, India
[3]Oxford Robotics Institute, Department of Engineering Science, University of Oxford, Oxford, UK

Corresponding author: Ayoung Kim (email: ayoungk@snu.ac.kr).

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00461409); the Horizon Europe project DigiForest (101070405); and EPSRC Project Mobile Robotic Inspector (EP/Z531212/1).

ABSTRACT Reliable localization is essential for sustainable forest management, as it allows robots or sensor systems to revisit and monitor the status of individual trees over long periods. In modern forestry, this management is structured around Digital Forest Inventories (DFIs), which encode stems using compact geometric attributes rather than raw data. Despite their central role, DFIs have been overlooked in localization research, and most methods still rely on dense gigabyte-sized point clouds that are costly to store and maintain. To improve upon this, we propose TreeLoc++, a global localization framework that operates directly on DFIs as a discriminative representation, eliminating the need to use the raw point clouds. TreeLoc++ substantially strengthens the existing TreeLoc framework by reducing false matches in structurally ambiguous forests and improving the reliability of full 6-DoF pose estimation. Building upon TreeLoc, it augments coarse retrieval with a pairwise distance histogram that encodes local tree-layout context, subsequently refining candidates via tree diameter at breast height (DBH)-based filtering and yaw-consistent inlier selection to reduce mismatches. Furthermore, a constrained optimization leveraging tree geometry jointly estimates roll, pitch, and height, enhancing pose stability and enabling accurate localization without reliance on dense 3D point cloud data. Evaluations on 27 sequences recorded in forests across three datasets and four countries show that TreeLoc++ achieves precise localization with centimeter-level accuracy. We further demonstrate robustness to long-term change by localizing data recorded in 2025 against inventories built from 2023 data, spanning a two-year interval. The system represents 15 sessions spanning $7.98\,\mathrm{km}$ of trajectories using only $250\,\mathrm{KB}$ of map data and outperforms both hand-crafted and learning-based baselines that rely on dense point cloud maps. This demonstrates the scalability of TreeLoc++ for long-term deployment. TreeLoc++ is open-sourced at https://github.com/minwoo0611/TreeLoc-plusplus.

INDEX TERMS Environmental Monitoring, Global Localization, Place Recognition, Pose Estimation

I. Introduction

Reliable and accurate localization is essential in forestry and environmental monitoring, where systems operate in complex under-canopy environments [1–3]. Localization enables key long-term tasks such as repeated site access, consistent map maintenance under dynamic canopy conditions, and digital inventory updates for sustainable forest management [4–6]. For practical deployment, localization systems must be robust to challenges specific to forests such as structural repetition and seasonal variation, provide centimeter-level accuracy, and rely on input representations

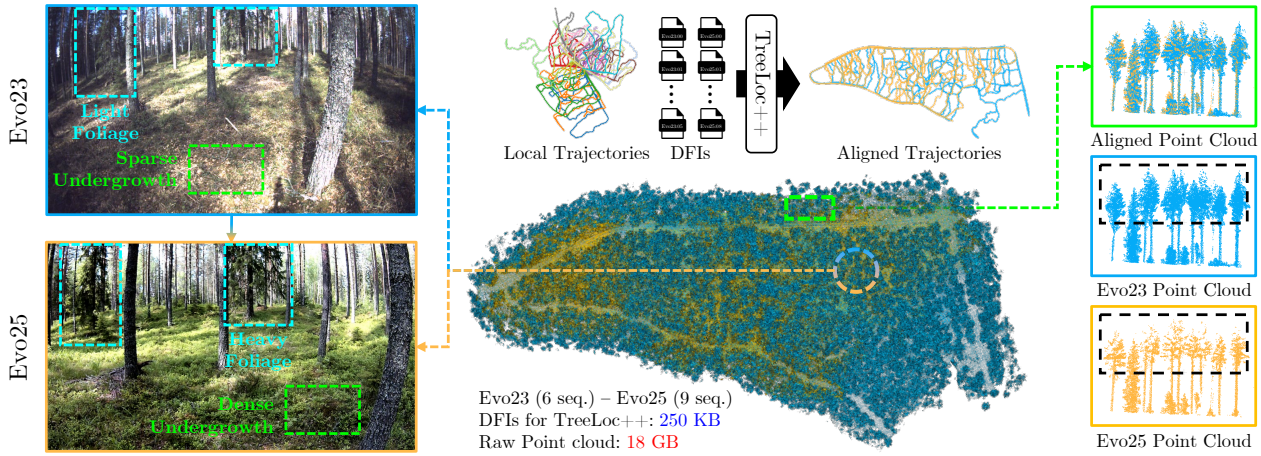


FIGURE 1. (Left) Images captured from the same location in the Evo forest two years apart, showing vegetation growth (light green) and denser foliage (cyan), which alter scene appearance and increase localization difficulty. (Right) TreeLoc++ achieves accurate multi-session alignment using only lightweight DFIs by leveraging tree attributes, avoiding raw point clouds that can vary across sessions due to different LiDAR sensors (black).

that minimize storage and computational overhead in long-term, resource-constrained missions [7, 8].

While satellite-based localization systems such as GNSS are widely used in outdoor environments, they struggle in forests due to signal degradation under dense canopies [9]. Even high-precision RTK services often fail to maintain consistent performance [10], resulting in meter-level errors, which are insufficient for tasks requiring tree-level correspondence such as growth estimation [11]. These limitations underscore the need for onboard, geometry-based alternatives that do not rely on external positioning infrastructure; LiDAR is a promising solution due to its ability to capture precise 3D geometric information [12, 13].

The standard LiDAR global localization pipeline typically involves place recognition for candidate retrieval [14–16] followed by pose estimation through local descriptor and feature matching [17–19] or scan registration [20–22]. Although effective in structured environments, they incur high memory usage due to dense point cloud storage, and their performance degrades in forests, where repetitive geometric layouts cause perceptual aliasing and seasonal changes further alter scene appearance and consistency [23]. This leads to more frequent retrieval errors and unstable pose estimates.

To mitigate reliance on dense point clouds and reduced discriminative power in forests, recent research has proposed compact, high-level representations as alternatives to raw point clouds [24, 25]. Segmentation-based methods reduce storage by clustering geometrically or semantically similar points, but their discriminative power remains limited in forests due to low semantic diversity, where only a few semantic categories recur [26]. Other methods such as NSM [27] and ESM [28] extract descriptors from individual tree instances, but still rely on point clouds and overlook inter-tree relationships, making them prone to perceptual aliasing in environments where tree geometries are indistinguishable. These limitations motivate the use of more compact and interpretable representations that are better suited to the structural characteristics of forest environments.

A more compact alternative would be to use a Digital Forest Inventory (DFI) directly for localization. A DFI represents each tree by attributes such as stem axis and diameter at breast height (DBH) without requiring any raw point clouds. TreeLoc [29] introduced a DFI-based framework that leverages inter-tree spatial relationships for matching. While efficient, it remains sensitive to structural aliasing, where locally similar tree layouts repeat across different parts of the forest, leading to incorrect matches and localization failures.

To overcome these limitations, we propose TreeLoc++, a forest-specific localization system that builds on TreeLoc to address three key limitations of the original pipeline: ambiguous coarse retrieval in repetitive layouts, false correspondences caused by triangle-hash collisions, and unstable full 6-DoF refinement. Compared with TreeLoc, TreeLoc++ strengthens the localization pipeline at three stages. First, it augments the Tree Distribution Histogram (TDH) descriptor with a complementary Pairwise Distance Histogram (PDH) that captures pairwise geometric context for more reliable candidate retrieval. Second, it refines tentative triangle matches through lightweight DBH filtering and yaw-consistent inlier voting, reducing attribute- and pose-inconsistent correspondences. Third, it improves geometric verification and pose estimation through an overlap score with a spatial proximity prior and a constrained optimization that jointly estimates roll, pitch, and height. In particular, the overlap score discourages associations between geometrically similar but spatially distant scenes, while the joint refinement yields more stable 6-DoF localization than estimating these components independently.

These additions improve both downstream metric localization and retrieval robustness while preserving the lightweight design of DFI-based localization. As shown in Fig. 1, TreeLoc++ offers a lightweight and scalable alternative to conventional point cloud-based localization systems. By us-

ing only a compact DFI as its prior map, it supports efficient multi-session alignment without retaining or revisiting raw scans. Its stem-based representation also enables accurate localization under seasonal change. This lightweight and stable localization pipeline supports continuous DFI updates across sessions, enabling long-term forest monitoring and management. Our contributions are summarized as follows:

- An advanced global localization framework for forests that resolves geometric ambiguity in tree-based configurations. Our framework also improves 6-DoF pose estimation by leveraging tree geometry and optimizing roll, pitch, and height simultaneously, enabling accurate localization in complex under-canopy conditions.
- A retrieval pipeline combining TDH and PDH, which jointly encodes individual tree attributes and inter-tree connectivity. This combination strengthens coarse retrieval, ensuring high recall and increasing the fraction of true positives among the top-ranked candidates.
- Robust outlier rejection through DBH-based filtering and yaw-consistent inlier voting. These methods reduce ambiguous matches caused by hash collisions, enabling more consistent candidate retrievals for precise place recognition and pose estimation, particularly in scenarios with repeated or symmetric tree layouts.
- Practical applicability across diverse localization scenarios such as inter-session localization, global map alignment, and continuous DFI updates. This is enabled by an overlap score that identifies true positives across sessions without environment-specific tuning, ensuring reliable performance in real-world deployments.
- Extensive benchmarking on three datasets with structural diversity and seasonal variations shows that TreeLoc++ consistently outperforms existing methods. Compared to the learning-based method LoGG3D-Net, it improves Recall@1 on `Evo` from 0.521 to 0.956 and reduces pose estimation error from decimeter-level to centimeter-level, while requiring only kilobytes of map storage with query times under $10\ \mathrm{ms}$.

II. Related Work

A. Digital Forest Inventories from Tree Segmentation

LiDAR plays a central role in forestry, supporting biodiversity monitoring, biomass estimation, and sustainable forest management [6, 30]. Both static terrestrial LiDAR systems and mobile LiDAR platforms provide high-resolution 3D data for detailed forest analysis [5]. A key application is individual tree segmentation [31–35], addressed through traditional clustering and deep learning methods such as PointNet++ [36] and RandLA-Net [37]. Public forest datasets [23, 26, 38, 39] have further accelerated research progress.

From segmented point clouds, tree attributes such as height, crown structure, stem position, orientation, and DBH can be extracted [35, 40]. These parameters form the basis of the DFI, a compact, point cloud-independent representation that supports applications in precision forestry and long-term monitoring. RealtimeTrees [35] introduced a learning-free pipeline for real-time DFI construction for mobile platforms, eliminating the need for post-processing. While DFI construction has become efficient, applying DFIs across multiple sessions introduces new challenges. Effective forest monitoring requires not only generating local inventories, but also localizing new observations against prior maps to detect revisits and to maintain correspondence between individual trees observed at different times. These capabilities are critical for enabling inventory updates, long-term monitoring, and consistent map integration, highlighting the need for robust global localization in forest environments [9, 41].

B. LiDAR-based Global Localization

Accurate forest localization is challenging due to GNSS degradation under dense canopies, making LiDAR-based global localization a viable alternative. This section reviews representative approaches based on hand-crafted descriptors and deep learning, and introduces inventory-based localization as a compact alternative. We highlight their limitations in forests to motivate our inventory-based formulation.

Hand-crafted Descriptors-based Localization: Conventional hand-crafted LiDAR place recognition methods are typically divided into global and local approaches, with global descriptors compressing a scan or submap into a compact representation for efficient nearest-neighbor retrieval [14–16, 42, 43]. Representative examples include Scan Context++ [14, 15], which encode scans on polar grids, RING++ [42, 44] with Radon transform-based yaw-translation decoupling, and SOLiD [16], formulated in azimuth-elevation space. While efficient in urban environments, these descriptors perform poorly in forests due to structural similarity and sensor variability, such as differences in angular resolution and mounting height, which lead to inconsistent projections and degraded robustness.

Local descriptors [45–47] aim to improve distinctiveness through salient geometric regions, but in forests, dense and structureless vegetation produces many indistinct keypoints, increasing computational cost and false positives. Projection-based methods such as MapClosure [19, 48] improve efficiency through ground-projected BEV matching, although their reliance on an observable ground plane often fails in natural environments. Triangle-based methods [17, 18, 49, 50] use hash-based indexing for efficient matching, but still degrade in forests due to unstable feature extraction from foliage and the lack of distinctive geometric structures.

Learning-based Localization: To address the limitations of hand-crafted descriptors, learning-based methods exploit deep networks for localization. Point-based methods [51, 52] such as TransLoc3D and LoGG3D-Net aggregate point-wise features into global descriptors, while sparse convolution methods [53, 54] such as MinkLoc3Dv2 improve scalability. Transformer-based forest localization models [55, 56], including HOTFormerLoc and ForestLPR, further capture long-range or vertical tree structure. In parallel, projection-

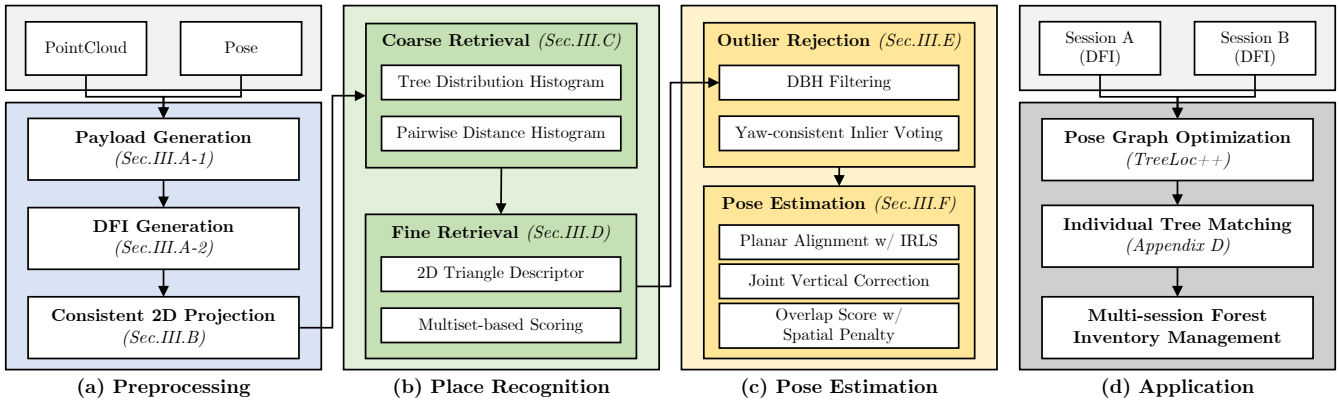


FIGURE 2. (a) TreeLoc++ extracts tree-level traits using RealtimeTrees [35] and projects them onto a consistent 2D plane. (b) From the projected DFI, two histogram descriptors and a 2D triangle descriptor are generated for place recognition. (c) After fine retrieval, outliers from incorrect triangle matches are rejected using DBH consistency and yaw estimation, followed by full 6-DoF pose estimation and verification with an overlap score augmented by a spatial penalty. (d) TreeLoc++ enables tree association across multiple sessions, supporting consistent matching and incremental trait updates.

based approaches such as BEVPlace++ [57] leverage 2D representations including BEV or range images [58, 59], and some methods further integrate verification or relative pose estimation into the localization pipeline [60–62]. However, in forest environments, these approaches remain challenged by domain shift and limited generalization across forest types.

Segment-based Localization: Segment-based methods represent scenes at the object level to improve robustness. Early approaches [25, 63] perform place recognition by extracting and matching 3D segments, later incorporating data-driven descriptors. Extending this segment-level perspective to natural environments, NSM [27] uses hand-crafted shape descriptors for segment matching, whereas ESM [28] replaces the descriptor with a learned deep representation. Both have shown promising results in forests, but segment-level descriptors can remain ambiguous for geometrically similar tree trunks, especially without broader inter-segment spatial relationships. Although semantic graph-based methods incorporate object-level relationships for structural reasoning [64, 65], their benefit in forests is limited by low semantic diversity, as only a few semantic classes repeatedly appear.

Inventory-based Localization: These limitations motivate the use of more compact and forest-specific representations for localization. To address these issues, recent work explores DFIs as compact, interpretable priors encoding tree attributes such as position, orientation, and DBH. Early inventory-based approaches rely on stem-level geometric matching. For example, Tremblay and Béland [66] perform marker-free registration via stem-triplet matching, but because the method relies mainly on inter-stem distances and DBH consistency, it remains vulnerable to structural ambiguity in repetitive forests. Liang et al. [67] also exploit stem centers and triangle relationships, but their method is limited to planar alignment and does not incorporate richer tree attributes. Their method further relies on brute-force triangle matching and repeatedly applies the transformation estimated from each triangle correspondence to all stem centers for verification, resulting in high computational cost.

TreeLoc [29] advances this direction by operating directly on DFIs, using TDH and 2D triangle descriptors in a coarse-to-fine matching pipeline. However, it remains vulnerable to repetitive stem layouts, where TDH-only coarse retrieval can miss pairwise layout cues and triangle-hash collisions can induce ambiguous correspondences. Compared with TreeLoc, TreeLoc++ strengthens the pipeline at three stages: it augments coarse retrieval with a complementary PDH, refines correspondences through lightweight DBH filtering and yaw-consistent inlier voting, and improves metric localization through overlap-based verification and joint refinement of roll, pitch, and height. These additions improve robustness in repetitive forests and long-term multi-session localization.

III. System of TreeLoc++

Fig. 2 shows the overall pipeline of TreeLoc++. The system begins with tree reconstruction from LiDAR scans and associated poses, generating a DFI. Based on this representation, localization is performed in three stages: (i) preprocessing, which projects tree locations onto a consistent 2D plane estimated from stem axes; (ii) place recognition, which retrieves candidate revisits through a coarse-to-fine matching strategy; and (iii) pose estimation, which performs geometric verification and refinement using robust filtering methods.

A. Digital Forest Inventory Construction

1) Tree Extraction

To ensure sufficient point density for tree segmentation and modeling, TreeLoc++ aggregates k consecutive LiDAR scans into a local submap called a payload, where each payload shares v scans with the previous one to maintain spatial continuity. Payloads are generated only when the system is moving, and their poses are obtained from a local trajectory provided by a LiDAR-based SLAM system.

Let $\mathcal{P}_u$ denote the payload indexed by u. For a given index t, we define a window of s payloads as $\mathcal{W}_t = \{t - \lfloor \frac{s-1}{2} \rfloor, \ldots, t + \lfloor \frac{s-1}{2} \rfloor\}$. These payloads are transformed into a common local frame and aggregated into a submap

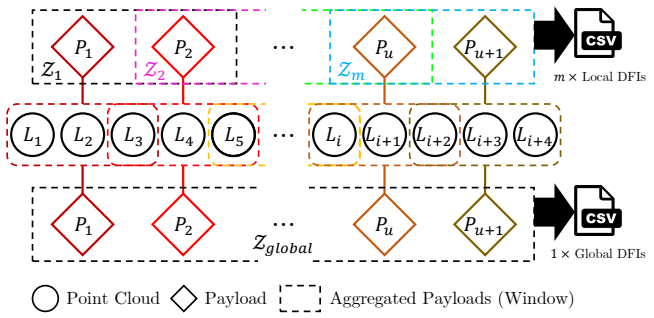


FIGURE 3. Payload-based forest inventory generation. (Top) Payloads are aggregated over temporal windows to form local inventories along the trajectory. (Bottom) All payloads are merged into a global inventory, which is queried at sampled poses to generate local inventories for localization.

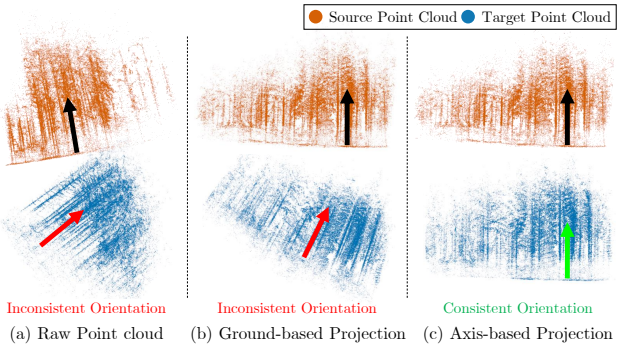


FIGURE 4. Comparison of alignment strategies for 2D projection. Arrows indicate the vertical direction of the ground plane after transformation. (a) Raw point clouds show inconsistent directions, with the target (red) misaligned with the source (black). (b) Ground-based alignment remains inconsistent across views. (c) Axis-based alignment using tree stems yields consistent directions, with the source and target aligned (green).

$\mathcal{Z}_t = \bigcup_{u \in \mathcal{W}_t} \mathbf{T}_{t \leftarrow u} \cdot \mathcal{P}_u$, where $\mathbf{T}_{t \leftarrow u}$ transforms each payload into the reference frame of $\mathcal{P}_t$. This submap-based approach, illustrated in Fig. 3, provides the dense geometric input required for subsequent tree modeling.

Tree instances are extracted from $\mathcal{Z}_t$ using RealtimeTrees [35]. We refer the reader to [29, 35] for the underlying extraction details. The extracted instances are then classified based on geometric completeness, where those satisfying minimum observation range and angular span requirements are regarded as successfully reconstructed trees, while the remainder are retained as candidate stems. The resulting collection of trees and their geometric attributes constitutes the DFI, which serves as the fundamental data structure for subsequent localization.

2) Digital Forest Inventory for Localization

TreeLoc++ operates in two inventory modes: local and global. As illustrated in the upper part of Fig. 3, the local mode aggregates payloads within a temporal window $\mathcal{W}_t$ to support incremental tasks such as SLAM loop closure or multi-robot localization. By sliding this temporal window along the trajectory, a total of m local inventories are generated. Conversely, the global mode merges all mission payloads into a single window $\mathcal{W}_{\text{global}}$, where trees are extracted once over the entire dataset. For localization, the global inventory is queried at selected poses, such as trajectory points or spatially sampled locations, to extract nearby trees within a fixed radius. Whether derived from local windows or queried from the global mode, the resulting inventory is transformed into the local coordinate frame of the corresponding pose to ensure consistent descriptor generation. This design enables the reuse of globally consistent tree data for large-scale, multi-session registration and long-term mapping applications.

For each reconstructed tree, we extract its 3D stem orientation $\mathbf{A}_j \in \mathrm{SO}(3)$ and DBH d_j. The stem position $\mathbf{p}'_j \in \mathbb{R}^3$ is defined by its horizontal stem center $\mathbf{c}'_j$ and base height $\mathbf{b}'_j$, determined by the terrain elevation at the stem center. An inventory at time t is represented as $\mathbf{M}_t = (\mathbf{T}_t, \mathcal{I}_t)$, where $\mathbf{T}_t$ is the system pose and $\mathcal{I}_t = \{(\mathbf{A}_j, \mathbf{p}'_j, d_j)\}_{j=1}^{n_t}$ is the set of n_t tree attributes. These attributes are computed from the aggregated payloads $\mathcal{Z}_t$ using RealtimeTrees.

Unlike TreeLoc, which relies only on the current local inventory, TreeLoc++ hierarchically augments sparse local inventories using preceding observations when the reconstructed-tree count falls below a required threshold. The augmentation integrates reconstructed trees from preceding inventories ($\mathcal{I}_{t-1}$ to $\mathcal{I}_{t-5}$), while filtering duplicates using the local trajectory. It compensates for limited local visibility and segmentation inconsistencies, and is omitted in global mode, where the comprehensive aggregation already provides sufficient coverage. If the tree count remains insufficient, candidate stems are incorporated based on their observation frequency to stabilize descriptor generation.

B. Axis-Based Alignment and 2D Projection

To generate compact descriptors from DFI, TreeLoc++ projects stem centers onto a common 2D plane derived from the forest inventory $\mathcal{I}_t$. This removes height ambiguity while preserving inter-tree geometry for place recognition, but roll and pitch variations across viewpoints can distort the local xy-plane and lead to inconsistent projections.

Ground-plane fitting is often used to correct such distortion but is unreliable in forests with uneven or partially observed terrain. As shown in Fig. 4, ground-based alignment may work in some views but fail in others due to inconsistent terrain estimation. In contrast, alignment using stem axes provides a consistent frame across varying viewpoints. We therefore align each frame using the stem axes of reconstructed trees. For each tree j, we extract a unit vector $\mathbf{a}_j$ from the third column of its orientation matrix $\mathbf{A}_j \in \mathrm{SO}(3)$. A rotation $\mathbf{R}_t^A \in \mathrm{SO}(3)$ is then estimated to align the axes $\{\mathbf{a}_j\}$ to a reference direction $\mathbf{v} \in \mathbb{S}^2$ by solving

$$\mathbf{R}_t^A = \underset{\mathbf{R} \in \mathrm{SO}(3)}{\arg\min} \sum_j \left(1 - \left|\mathbf{v}^\top \mathbf{R} \mathbf{a}_j\right|\right)^2. \tag{1}$$

We set $\mathbf{v} = \mathbf{e}_z = (0, 0, 1)^\top$ in practice, although other reference directions can also be used. As in TreeLoc,

this alignment corrects roll and pitch while leaving yaw unconstrained, thereby preserving stable horizontal inter-tree geometry. After estimating $\mathbf{R}_t^A$, each stem center $\mathbf{p}'_j$ is transformed to the aligned frame as $\mathbf{p}_j = \mathbf{R}_t^A \mathbf{p}'_j$.

To obtain 2D coordinates for descriptor generation, the aligned stem $\mathbf{p}_j$ are projected onto the plane orthogonal to $\mathbf{v}$ using an orthonormal basis $\{\mathbf{u}_1, \mathbf{u}_2\}$ satisfying $\mathbf{u}_i^\top \mathbf{v} = 0$, resulting in projected coordinates $\mathbf{c}_j = \left[\mathbf{u}_1^\top;\ \mathbf{u}_2^\top\right] \mathbf{p}_j \in \mathbb{R}^2$. In the special case where $\mathbf{v} = \mathbf{e}_z$, this simplifies to:

$$\mathbf{c}_j = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(\mathbf{p}_j - (\mathbf{e}_z^\top \mathbf{p}_j)\mathbf{e}_z\right), \tag{2}$$

yielding a roll- and pitch-invariant 2D representation.

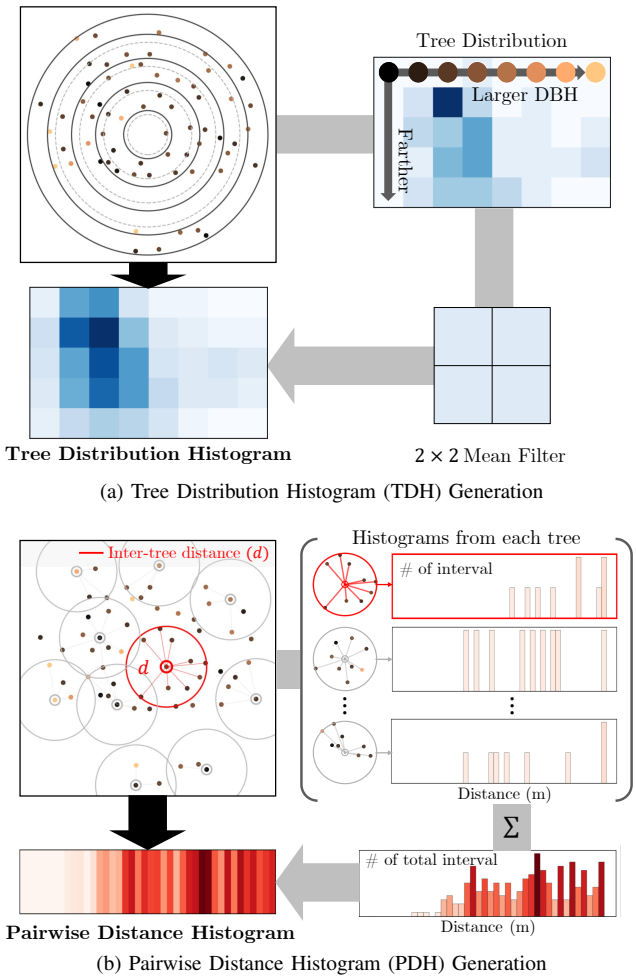
C. Coarse Retrieval via Tree-Based Histograms

To enable efficient and robust place recognition, we summarize the spatial structure of each inventory using two complementary histogram descriptors, TDH and PDH, computed from the reconstructed trees $\mathcal{I}_t$, as illustrated in Fig. 5.

Firstly, TDH encodes 2D spatial structure by assigning each tree to overlapping radial and DBH bins based on its stem center $\{\mathbf{c}_j\}$ and diameter $\{d_j\}$. Radial bins partition $[r_{\min}, r_{\max}]$ into n_r intervals with overlap w_r, while DBH bins partition $[d_{\min}, d_{\max}]$ into n_d intervals with overlap w_d. The resulting histogram $\mathbf{H}_t \in \mathbb{R}^{n_r \times n_d}$ is smoothed with a 2×2 uniform kernel and flattened into a 40-dimensional descriptor. Overlap parameters and smoothing improve robustness to noise and pose variation.

While TDH captures tree attributes via spatial binning, it lacks inter-tree relationships and is sensitive to the choice of reference center, varying with missing trees or slight shifts in their positions. To address this, we introduce PDH, a 1D histogram that encodes local structure by aggregating pairwise Euclidean distances between stem centers. Distances are binned over $[l_{\min}, l_{\max}]$ into n_{bins} intervals, yielding a 40-dimensional descriptor invariant to rotation and translation. Although structural perturbations may affect some distances, the aggregation distributes their impact across many bins, reducing sensitivity to local changes. Compared with TDH, PDH is generally more stable under slight translation deviations and missing-tree perturbations because it depends on pairwise distances rather than a reference-centered layout. However, since it does not explicitly encode the spatial distribution of trees around the local center, PDH is not always more discriminative than TDH as a standalone descriptor. We therefore use it to complement, rather than replace TDH.

For each query inventory at index t, both descriptors are compared against the database. Let $\mathbf{h}_t^{\text{TDH}}$ and $\mathbf{h}_t^{\text{PDH}}$ denote the query descriptors, and $\mathbf{h}_i^{\text{TDH}}$, $\mathbf{h}_i^{\text{PDH}}$ those of a candidate i. Descriptor distances are computed using the chi-square metric χ^2 as $d_i^{\text{TDH}} = \chi^2(\mathbf{h}_t^{\text{TDH}}, \mathbf{h}_i^{\text{TDH}})$ and $d_i^{\text{PDH}} = \chi^2(\mathbf{h}_t^{\text{PDH}}, \mathbf{h}_i^{\text{PDH}})$. Each distance is min-max normalized to balance the influence of TDH and PDH, with the result denoted as $\hat{d}$. The retrieval score is defined as $\text{score}_i = -(\hat{d}_i^{\text{TDH}} + \hat{d}_i^{\text{PDH}})$, such that lower descriptor distances yield higher scores. The top 100 candidates are retained for fine-grained retrieval.



(a) Tree Distribution Histogram (TDH) Generation

(b) Pairwise Distance Histogram (PDH) Generation

FIGURE 5. Generation of histogram descriptors. (a) TDH encodes tree distributions using overlapping radial and DBH bins. Each radial bin spans from a dashed circle to the second subsequent solid circle, illustrating the overlapping construction. (b) PDH aggregates inter-tree distance histograms computed for each tree into a 1D scene-level representation.

D. 2D Triangle Descriptor Using Tree Centers

To refine candidates from histogram-based coarse retrieval, we introduce a 2D triangle descriptor that is rotation- and translation-invariant, capturing local inter-tree geometry. Following geometric hashing principles [18], the descriptor is constructed from the aligned 2D stem centers.

For each center $\mathbf{c}_i$, we select its m nearest neighbors and form triangles by combining $\mathbf{c}_i$ with each unordered neighbor pair $(\mathbf{c}_j, \mathbf{c}_k)$. Each triangle is represented by its side lengths $(\ell_{ij}, \ell_{jk}, \ell_{ik})$, sorted in ascending order as (ℓ_1, ℓ_2, ℓ_3) for permutation invariance, together with its area A. All geometric quantities are quantized with resolution δ_ℓ as $\ell_i^{\text{q}} = \lfloor \ell_i / \delta_\ell \rfloor$ and $A^{\text{q}} = \lfloor A / \delta_\ell \rfloor$. The final triangle hash

is computed by:

$$h'_{ijk} = ((\ell_3^{\mathrm{q}} \cdot \rho + \ell_2^{\mathrm{q}}) \bmod U \cdot \rho + \ell_1^{\mathrm{q}}) \bmod U,$$
$$h_{ijk} = \left(h'_{ijk} \cdot \rho + A^{\mathrm{q}}\right) \bmod U. \tag{3}$$

where ρ is a large prime and U is the maximum hash range. These hashes index triangles for efficient matching.

Metadata $\mathbf{D}_{ijk} = \{\mathcal{A}_i,\ \mathcal{A}_j,\ \mathcal{A}_k,\ \mathbf{q}_{ijk},\ s\}$ is stored in each triangle, where $\mathcal{A}$ contains tree attributes, $\mathbf{q}_{ijk}$ is the triangle centroid, and s is the scene index. This metadata is directly used in the later TreeLoc++ stages for outlier rejection and geometric verification.

For each scene s, we store the multiset of triangle hashes $\mathcal{H}_s = \{h_{ijk}\}$. Let $\mathcal{K}_s$ denote the set of unique hash keys in $\mathcal{H}_s$. During fine retrieval, a query scene Q is compared with a candidate scene C by counting shared triangle hashes:

$$S(Q, C) = \sum_{h \in \mathcal{K}_Q \cap \mathcal{K}_C} \min\big(\mathrm{freq}_Q(h),\ \mathrm{freq}_C(h)\big), \tag{4}$$

where $\mathrm{freq}_Q(h)$ and $\mathrm{freq}_C(h)$ are the occurrences of hash h in $\mathcal{H}_Q$ and $\mathcal{H}_C$, respectively. Unlike simple set intersection used in TreeLoc, multiset-based scoring accounts for repeated geometric patterns, improving similarity discrimination. The top 10 candidates ranked by $S(Q, C)$ proceed to geometric verification.

E. Outlier Rejection via DBH Filtering and Yaw Voting

To extract an inlier set for geometric verification, we refine the triangle matches. Ambiguous hash collisions can produce false correspondences, which often manifest as attribute- or pose-inconsistent matches. We suppress these outliers through DBH filtering and yaw voting.

1) Outlier Rejection via DBH Filtering

When a common hash value appears multiple times, it is unclear which triangles should be matched, as shown in Fig. 6a. Let m and n be the numbers of triangles in the query and candidate scenes sharing a given hash. In such cases, at most $\min(m, n)$ valid correspondences can exist.

To resolve this ambiguity, we use the triangle DBH from metadata $\mathbf{D}$. Let $\mathbf{d}_Q^i = (d_{Q1}^i, d_{Q2}^i, d_{Q3}^i)$ and $\mathbf{d}_C^j = (d_{C1}^j, d_{C2}^j, d_{C3}^j)$ be the DBH vectors of the i-th and j-th triangles. Since DBH vectors follow the sorted side length order, we can directly compare them and define the matching cost as

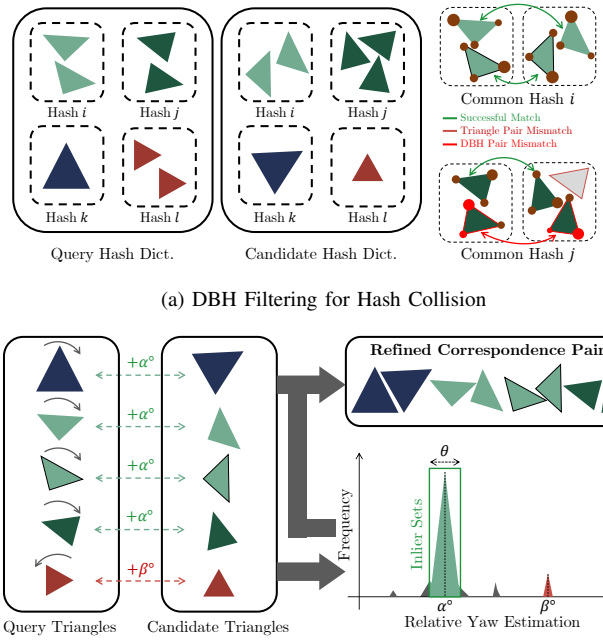
$$\Delta_{ij} = \sum_{v=1}^{3} \left| d_{Qv}^i - d_{Cv}^j \right|. \tag{5}$$

We select up to $\min(m, n)$ one-to-one correspondences with minimum total cost and reject pairs violating τ_{DBH}:

$$\max_{v \in \{1,2,3\}} \left| d_{Qv}^i - d_{Cv}^j \right| < \tau_{\mathrm{DBH}}. \tag{6}$$

2) Yaw-Consistent Inlier Voting

Even after DBH filtering, repetitive local structures may still include incorrect correspondences that induce erroneous pose estimates. To enforce global consistency, we analyze the relative yaw rotation implied by each remaining triangle match as shown in Fig. 6b.



(a) DBH Filtering for Hash Collision

(b) Yaw Voting for Outlier Suppression

FIGURE 6. Inlier set selection via DBH filtering and yaw voting. (a) Ambiguous triangle correspondences sharing the same hash are disambiguated using DBH consistency. (b) Yaw angles from remaining correspondences are aggregated, and only those consistent with the dominant yaw are retained.

For each matched triangle pair, a 2D rigid transformation is estimated using singular value decomposition (SVD)-based alignment, from which the relative yaw θ_k is extracted. We exclude matches that yield a reflection with $\det(\mathbf{R}_k) < 0$. The remaining angles are accumulated into a histogram to identify the dominant yaw θ^*. The final inlier set is defined as:

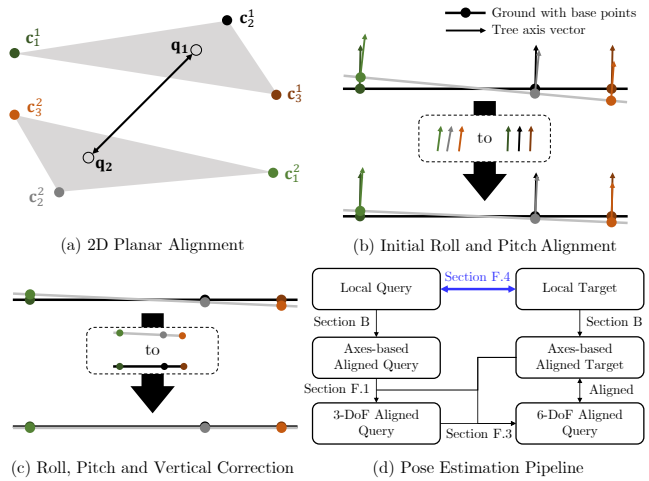
$$\mathcal{I} = \{k \mid |\theta_k - \theta^*| < \tau_{\mathrm{yaw}}\}, \tag{7}$$

where τ_{yaw} is a predefined angular threshold.

This two-step process, DBH-based disambiguation followed by yaw voting, yields correspondences consistent in both local attributes and global pose. The inlier set is then used for geometric verification and 6-DoF pose estimation.

F. Geometric Verification and 6-DoF Pose Estimation

We estimate the 6-DoF relative transformation between a query scene $\mathcal{I}_Q$ and a candidate scene $\mathcal{I}_C$. Following the shared geometric verification backbone of TreeLoc, TreeLoc++ strengthens the refinement with IRLS-based planar refinement, tree axis-based roll-pitch estimation, joint vertical correction, and a spatial penalty-aware overlap score. This process is applied to the top 10 candidates retrieved via triangle hash matching, and the best match is selected through geometric verification. An overview of the pipeline is shown in Fig. 7.



(a) 2D Planar Alignment (b) Initial Roll and Pitch Alignment

(c) Roll, Pitch and Vertical Correction (d) Pose Estimation Pipeline

FIGURE 7. Pose estimation pipeline. (a) Initial 2D planar alignment is estimated using triangle correspondences. (b) Roll and pitch are aligned using the axes of matched trees. (c) Vertical correction is performed by jointly refining height together with roll and pitch. (d) The final 6-DoF transformation is obtained, summarizing the overall pose estimation process.

1) Planar Alignment via Inlier Triangles and IRLS Refinement

Using the inlier triangle correspondences, we extract the matched centroids $\{(\mathbf{q}_u^Q, \mathbf{q}_u^C)\}_{u=1}^N$ from metadata $\mathbf{D}$ and estimate an initial planar transformation $\mathbf{T}_{C\leftarrow Q}^{\text{init}} \in \text{SE}(2)$, parameterized by $(\mathbf{R}_{\text{2D}}^{\text{init}}, \mathbf{t}_{\text{2D}}^{\text{init}})$, via least-squares alignment:

$$\mathbf{R}_{\text{2D}}^{\text{init}}, \mathbf{t}_{\text{2D}}^{\text{init}} = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{u=1}^{N} \left\| \mathbf{q}_u^C - (\mathbf{R}\mathbf{q}_u^Q + \mathbf{t}) \right\|^2. \quad (8)$$

TreeLoc++ then further refines this initial planar estimate using iteratively reweighted least squares (IRLS) to suppress residual outliers before subsequent 6-DoF refinement. IRLS is initialized with $(\mathbf{R}_{\text{2D}}^{\text{init}}, \mathbf{t}_{\text{2D}}^{\text{init}})$, and we refine the planar transformation over all vertex-level matches $(\mathbf{c}_i^Q, \mathbf{c}_i^C)$ in corresponding triangles:

$$\mathbf{R}_{\text{2D}}, \mathbf{t}_{\text{2D}} = \arg\min_{\mathbf{R},\mathbf{t}} \sum_i w_i \left\| \mathbf{c}_i^C - (\mathbf{R}\mathbf{c}_i^Q + \mathbf{t}) \right\|^2. \quad (9)$$

where weights w_i follow a Huber kernel based on residuals.

2) Tree-Level Correspondences and Roll-Pitch Estimation

Applying the planar alignment, each query $\mathbf{c}_i^Q$ is transformed as $\widetilde{\mathbf{c}}_i = \mathbf{R}_{\text{2D}}\mathbf{c}_i^Q + \mathbf{t}_{\text{2D}}$. Tree-level correspondences are then established by finding candidate matches $\mathbf{c}_j^C$ via nearest-neighbor search, subject to spatial and DBH thresholds:

$$\left\| \widetilde{\mathbf{c}}_i - \mathbf{c}_j^C \right\| < \tau_d, \quad |d_i^Q - d_j^C| < \tau_{\text{DBH}}. \quad (10)$$

Using the matched pairs $\mathcal{M} = \{(i,j)\}$, we solve (9) once to update $(\mathbf{R}_{\text{2D}}, \mathbf{t}_{\text{2D}})$. To support the subsequent vertical correction, TreeLoc++ next estimates residual roll and pitch from tree-axis correspondences. The same set $\mathcal{M}$ is used to estimate roll and pitch by aligning axis vectors $\mathbf{a}_i^Q$ and $\mathbf{a}_j^C$. With yaw fixed by $\mathbf{R}_z$ extracted from $\mathbf{R}_{\text{2D}}$, we estimate the residual roll-pitch rotation $\mathbf{R}_{\phi,\psi}$ via RANSAC over tree axis correspondences:

$$\mathbf{R}_{\phi,\psi} = \arg\min_{\mathbf{R}} \sum_{(i,j)\in\mathcal{M}} \left(1 - (\mathbf{a}_j^C)^\top \mathbf{R}\mathbf{R}_z\mathbf{a}_i^Q\right). \quad (11)$$

3) Vertical Correction and Final Scoring

TreeLoc++ further introduces a joint vertical correction of height, roll, and pitch using matched tree base heights. Thanks to the roll-pitch estimation using tree axis, the remaining roll-pitch error is small. This justifies the use of a small-angle approximation when modeling vertical misalignment. For a 3D point (x, y, z), the effect of roll (ϕ) and pitch (ψ) on the vertical coordinate is given by

$$z' = -x\cos\phi\sin\psi + y\sin\phi + z\cos\phi\cos\psi. \quad (12)$$

Yaw is omitted as it does not affect the vertical component. For small ϕ and ψ, this approximates to $z' \approx z - \psi x + \phi y$.

Using tree base heights, the vertical relationship between a matched tree pair $(i,j) \in \mathcal{M}$ is approximated as

$$b_j^C \approx \tilde{b}_i^Q, \quad \tilde{b}_i^Q = \hat{b}_i^Q + \Delta z - \Delta\psi\,\hat{x}_i^Q + \Delta\phi\,\hat{y}_i^Q, \quad (13)$$

where $(\Delta z, \Delta\phi, \Delta\psi)$ denote the global vertical offset and small residual roll-pitch corrections, and $(\hat{x}_i, \hat{y}_i)$ and $\hat{b}_i^Q$ denote the planar coordinates and base height of the query tree after applying the planar alignment $\mathbf{T}_{\text{2D}}$ and the roll-pitch correction $\mathbf{R}_{\phi,\psi}$. The parameters $(\Delta z, \Delta\phi, \Delta\psi)$ are estimated by minimizing the residual $r_{ij} = b_j^C - \tilde{b}_i^Q$ over all matched pairs $(i,j) \in \mathcal{M}$ using a RANSAC-based least-squares estimator for robustness. The resulting correction is expressed as a 3D transformation:

$$\mathbf{T}_{\text{6D}} = \begin{bmatrix} \mathbf{R}_{\Delta\phi,\Delta\psi}\mathbf{R}_{\phi,\psi} & [0,0,\Delta z]^\top \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{T}_{\text{3D}}, \quad (14)$$

where $\mathbf{T}_{\text{3D}}$ denotes the 2D alignment $\mathbf{T}_{\text{2D}}$ lifted to SE(3).

4) Final 6-DoF Transformation in Local Frames

All transformations so far are computed in the reference-aligned frame, where roll and pitch are aligned using tree axis-based 2D projection. To recover the full 6-DoF transformation between the original local frames, we undo this alignment using the corresponding rotations $\mathbf{R}_Q^A$ and $\mathbf{R}_C^A$.

Let $\mathbf{T}_Q^A = \text{diag}(\mathbf{R}_Q^A, 1)$ and $\mathbf{T}_C^A = \text{diag}(\mathbf{R}_C^A, 1)$. The final transformation from the query to the candidate frame is given by $\mathbf{T}_{C\leftarrow Q}^{\text{6D}} = (\mathbf{T}_C^A)^{-1}\,\mathbf{T}_{\text{6D}}\,\mathbf{T}_Q^A$. This yields a global 6-DoF relative pose purely from tree-level geometric information.

To assess alignment quality and select the best match, we compute an overlap score combining geometric consistency and spatial proximity:

$$\mathcal{O}(Q,C) = \frac{|\mathcal{M}_{Q,C}|}{|\mathcal{T}_Q| + |\mathcal{T}_C| - |\mathcal{M}_{Q,C}|} \cdot p(\|\mathbf{t}\|), \quad (15)$$

where $|\mathcal{M}_{Q,C}|$ is the number of matched trees, and $|\mathcal{T}_Q|$, $|\mathcal{T}_C|$ are the tree counts in the query and candidate scenes.

TABLE 1. Summary of datasets used for evaluation.

	Dataset / Sequence	Length	Environment Characteristics
Oxford Forest Place Recognition	`Evo:Single`	1.53 km	Tall mixed-species forest, moderate density
	`Stein am Rhein`	0.70 km	Sparse coniferous trees, flat terrain
	`Wytham`	0.70 km	Ground vegetation with cluttered trees, severe occlusion
	`Forest of Dean`	0.75 km	Widely spaced oak trees, minimal ground vegetation
	`Evo23:00-05`	4.69 km	Same forest as `Evo`, different traversals
Ours	`Evo25:00-08`	3.29 km	Same forest as `Evo`, vegetation growth, seasonal change
Wild-Places	`Venman01-04`	12.68 km	Moderate vegetation density, mixed terrain, relatively stable appearance
	`Karawatha01-04`	20.24 km	Dense vegetation, uneven terrain, pronounced appearance changes

The penalty term $p(\|\mathbf{t}\|) = \exp\left(-\|\mathbf{t}\|^2/\sigma_t^2\right)$ introduces a spatial prior that favors candidates with smaller estimated planar displacement $\mathbf{t}$ from $\mathbf{T}^{6D}_{C\leftarrow Q}$, where σ_t is a scaling parameter controlling the spatial constraint. This reflects the intuition that large translation shifts often lead to discrepancies in reconstruction quality; as the same tree is viewed from substantially different distances or angles, varying degrees of occlusion and sparsity degrade its geometric consistency. The candidate with the highest score $\mathcal{O}(Q, C)$ is selected, along with its estimated transformation $\mathbf{T}^{6D}_{C\leftarrow Q}$.

IV. Datasets and Evaluation Metrics

A. Dataset Summary

We evaluated TreeLoc++ on 27 sequences collected across diverse forest environments, including both complete plot mappings and point-to-point route traversals. Dataset characteristics are summarized in Table. 1 and Fig. 8.

We first utilized the Oxford Forest Place Recognition dataset [23], which contains ten sequences collected from small forest plots across three countries. All data were recorded using a backpack-mounted platform that incurs substantial roll and pitch motion while being carried during recording, presenting a further challenge for localization.

To assess long-term performance under pronounced appearance changes, we also introduce nine newer sequences, `Evo25:00-08`, collected two years later in the same forest as `Evo:Single`. This dataset captured environmental changes caused by vegetation growth and seasonal variation, facilitating the evaluation of long-term global localization. Ground truth trajectories for `Evo25` were obtained via multi-session alignment, with details provided in Appendix A. An overview of the data acquisition setup is shown in Fig. 9.

FIGURE 8. Example scenes for each dataset and sequence.

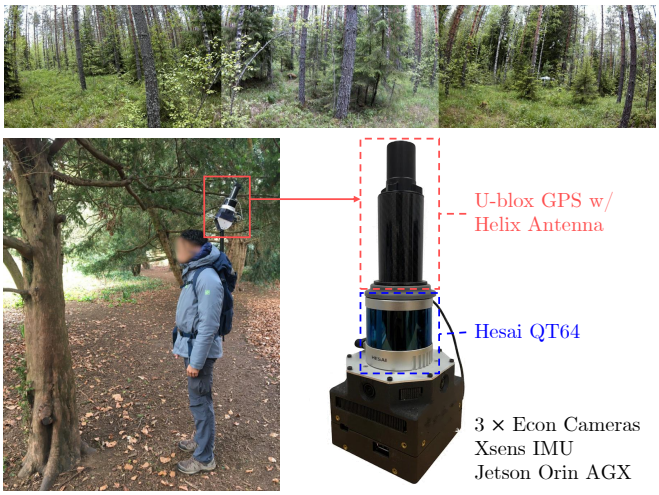


FIGURE 9. Overview of the `Evo25` and acquisition platform. (Top) Example scenes captured in `Evo25`. (Bottom) Backpack-based data acquisition system equipped with a LiDAR, IMU, GPS antenna, and three cameras.

To complement the plot inventory datasets, we used the Wild-Places dataset [39], which consists of eight sequences recorded in two large-scale forest parks in Australia: `Venman01-04` and `Karawatha01-04`. These sequences span substantially longer, forest access road trajectories with temporal gaps, enabling large-scale evaluation.

B. Baselines and Evaluation Protocol

To ensure fair evaluation, we standardized input preprocessing across all methods, including trajectory estimation and submap construction. We first estimated the poses used for payload aggregation and forest inventory generation. To reflect a realistic setup where localization operates alongside online state estimation, we used FAST-LIO2 [68] for `Evo:Single` and `Stein am Rhein`, and DLIO [69] for the remaining Oxford sequences. For Wild-Places, ground truth poses were used. Then, scans were aggregated into payloads and accumulated over time to form pose-centric submaps, from which forest inventories were generated. All methods used consistent spatial cropping centered at the

TABLE 2. (Exp A-1) Intra-session place recognition performance in Oxford Forest Place Recognition Dataset. Bold: Best, *Italic*: Second-best.

Dimension	Method	Evo:Single R@1	Evo:Single MR	Evo:Single MF1	Evo:Single AUC	Forest of Dean R@1	Forest of Dean MR	Forest of Dean MF1	Forest of Dean AUC
Learning-based (Wild-Places)	TransLoc3D	0.425	0.001	0.596	0.570	0.570	0.002	0.726	0.659
	LoGG3D-Net	0.521	0.082	0.685	0.722	*0.776*	0.182	0.874	0.776
	MinkLoc3Dv2	0.470	0.010	0.608	0.462	0.492	0.058	0.652	0.582
	BEVPlace++	0.872	0.351	0.932	0.973	0.712	0.212	0.837	0.858
	ForestLPR	0.465	0.001	0.635	0.611	0.439	0.058	0.610	0.567
	HOTFormerLoc	0.530	0.019	0.743	0.820	0.583	0.126	0.748	0.780
Hand-crafted (Untrained)	SC++	0.347	0.326	0.762	0.696	0.426	0.333	0.759	0.862
	SOLID	0.222	0.000	0.372	0.352	0.155	0.071	0.367	0.388
	RING++	0.567	0.126	0.223	0.125	0.674	0.194	0.781	0.718
	BTC	0.553	0.531	0.761	0.864	0.559	*0.772*	0.720	0.692
	MapClosure	0.359	0.064	0.528	0.359	0.166	0.014	0.285	0.154
	NSM	0.836	*0.762*	0.961	*0.992*	0.486	0.676	0.885	*0.912*
	TreeLoc	*0.890*	0.228	*0.955*	0.984	0.348	0.045	0.897	0.891
	TreeLoc++	**0.956**	**0.933**	**0.991**	**0.999**	**0.891**	**0.898**	**0.985**	**0.997**

· Recall@1 (R@1), Maximum Recall at 100% Precision (MR), Maximum F1 score (MF1), and Area Under the Precision–Recall Curve (AUC).

query pose, with regions of $30\ \mathrm{m}\times30\ \mathrm{m}$ for Oxford and $60\ \mathrm{m}\times60\ \mathrm{m}$ for Wild-Places to account for its larger scale.

In our intra-session evaluation, both queries and candidates used local inventory mode. For inter-session evaluation, we considered three configurations: local-local, local-global (with uniformly sampled global poses), and global-global (with inventories from a reference trajectory). For global baselines, the full point clouds were merged to form the global map and then the same cropping was applied during the matching. We compared TreeLoc++ to baselines for place recognition, metric localization, and pose estimation.

Place Recognition: We compared TreeLoc++ with learning-based and hand-crafted methods. Learning-based methods included point-based approaches (TransLoc3D [51], MinkLoc3Dv2 [53], LoGG3D-Net [52], HOTFormerLoc [55]) and the BEV-based methods such as BEVPlace++ [57] and ForestLPR [56]. TransLoc3D and BEVPlace++ were trained on Wild-Places, while the others used pretrained checkpoints. Hand-crafted baselines included global descriptor methods (Scan Context+ [15], SOLiD [16], RING++ [42]), local descriptor methods (BTC [18], MapClosure [48]), segment-based method (NSM [27]), and inventory-based method such as TreeLoc [29]. Because NSM localizes queries directly in the global map rather than via scan-to-scan retrieval, we used the nearest database scan to its estimated query pose as the retrieval result. A retrieval was considered correct if within $5\ \mathrm{m}$ of the ground truth; additional analyses under varying distance thresholds are provided in Appendix E. We report Recall@1 (R@1), Maximum Recall at 100% Precision (MR), Maximum F1 score (MF1), and Area Under the Precision-Recall Curve (AUC).

Metric Localization: We evaluated whether each method could recover an accurate pose estimation. LoGG3D-Net and MinkLoc3Dv2 were extended with their feature-based localization, while BEVPlace++ includes its own localization module. Hand-crafted baselines included TreeLoc, RING++, MapClosure, BTC and NSM. As BEVPlace++ and RING++ only estimate (x, y, yaw), we report both 2D and 3D results.

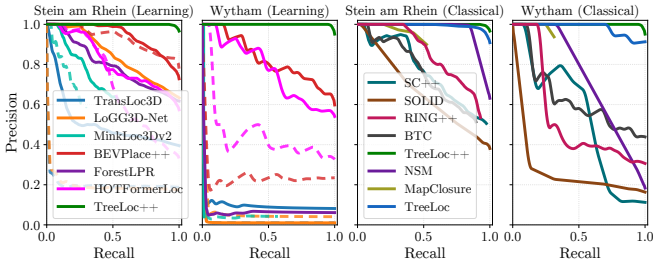


FIGURE 10. (Exp A-1) Precision–Recall curves comparing TreeLoc++ with learning-based (left) and hand-crafted (right) methods. Some learning-based methods outperform hand-crafted baselines but remain below TreeLoc++, while hand-crafted methods show a sharp precision drop at low recall. TreeLoc++ achieves a higher AUC than TreeLoc.

We report Recall@50cm (R@50): the fraction of queries whose estimated pose is within $50\ \mathrm{cm}$ and $5°$ of ground truth, regardless of retrieval correctness. The Success Rate (SR) adds the requirement that the underlying retrieval must also be a true positive. For these successful cases, we report the Average Translation Error (ATE) and Average Rotation Error (ARE). Further benchmark results about pose estimation with registration methods are provided in Appendix C.

V. Results

A. Intra-session Place Recognition

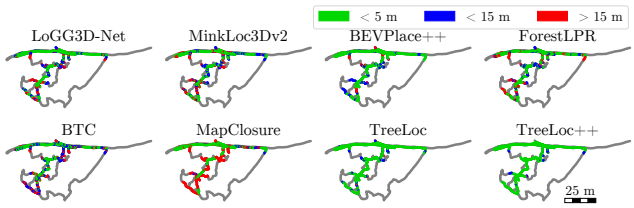
1) Oxford Forest: Small and Dense Off-road Forests

We begin our evaluation with intra-session cases. The first experiment focuses on intra-session place recognition using the Oxford Forest dataset, which consists of short, small-scale off-road sequences characterized by frequent revisits and substantial variations in viewpoint and occlusion.

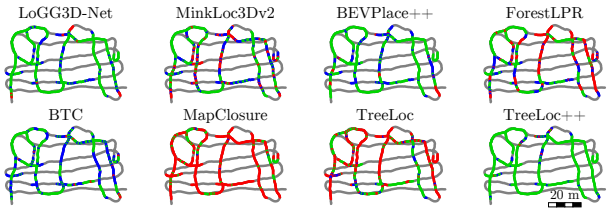
Evaluation protocol: We compared TreeLoc++ against hand-crafted and learning-based baselines on four sequences. To avoid trivial matches, we excluded the 50 most recent frames from the candidate database.

TreeLoc++: TreeLoc++ achieved the best performance across all metrics (Table. 2), maintaining high precision and the highest MR (Fig. 10). This superior MR is primarily driven by the translation penalty in the overlap score defined in (15). Under frequent revisits and occlusions, where candidates exhibit similar geometry, this penalty favors spatially closer matches and improves true positive retrieval. On `Forest of Dean`, TreeLoc++ preserved high R@1 despite repetitive geometry; its multiset-based scoring introduced in (4) preserves each tentative triangle match, consistently outperforming TreeLoc as shown in Fig. 11.

Hand-crafted baselines: Hand-crafted descriptors gave poor performance in these forest sequences. Global descriptor-based methods performed poorly, as compressing entire scans into compact global features limits discriminative power in vertically repetitive and structurally ambiguous scenes. Likewise, local descriptor-based methods suffered from viewpoint sensitivity and unstable keypoint selection. The segment-based NSM performed strongly on `Evo:Single` due to direct localization against the global map, but still lagged behind TreeLoc++ and degraded on the



(a) Intra-session Matching in Stein am Rhein



(b) Intra-session Matching in Forest of Dean

FIGURE 11. (Exp A-1) Matching results in two sequences. TreeLoc++ identifies the most true positives (green). In failure cases, it still retrieves spatially adjacent candidates (blue) rather than distant false positives (red).

other sequences because correspondences among geometrically similar tree trunks remain ambiguous and inter-segment spatial relationships are not explicitly modeled. Furthermore, as seen in the PR curves in Fig. 10, the baselines exhibited abrupt precision drops or early termination, indicating sensitivity to outliers and reduced matchability.

Learning-based baselines: Learning-based methods show improved performance over hand-crafted baselines as shown in Table. 2 but still suffer from low MR and limited accuracy. While BEVPlace++ is the most competitive, its performance drops sharply in cluttered scenes like Wytham (Fig. 10). As shown in Fig. 11, these methods yield fewer true positives and more near-matches (blue points) than TreeLoc++, failing to distinguish locations accurately. This is primarily due to a persistent domain shift of the training dataset, where inherent structural variations between forests prevent the extraction of robust, generalizable features. A detailed analysis of the impact of training data is provided in Appendix B.

While the evaluation targets a small-scale setting with frequent revisits, we next consider a more challenging scenario with larger spatial extent and stronger structural aliasing.

2) Wild-Places: Large-scale Forests with Larger Databases

To further validate intra-session place recognition in a more challenging forest environment, we extend our experiments to the Wild-Places dataset. Compared to Oxford, Wild-Places features longer trajectories, larger databases, and fewer revisits, increasing the overall complexity of the task. Moreover, the data are collected along forest access roads, making the dataset less suitable for tree reconstruction. The two sites present complementary characteristics: Venman exhibits dense, structurally repetitive scenes, whereas Karawatha contains open areas with sparse geometric features.

TABLE 3. (Exp A-2) Intra-session place recognition performance in Wild-Places.

Category / Method		Venman03			Venman04			Karawatha03		
		R@1	MR	MF1	R@1	MR	MF1	R@1	MR	MF1
Learning-based (Wild-Places)	TransLoc3D	0.632	0.003	0.775	0.801	0.003	0.890	0.693	0.001	0.819
	LoGG3D-Net	0.635	0.000	0.777	0.814	0.113	0.898	**0.867**	0.009	*0.929*
	MinkLoc3Dv2	*0.722*	0.049	*0.793*	0.853	0.037	0.921	0.814	0.114	0.894
	BEVPlace++	0.354	0.000	0.530	0.885	0.234	0.940	0.842	0.152	0.915
	ForestLPR	0.501	0.025	0.668	0.743	0.045	0.853	0.599	0.010	0.749
	HOTFormerLoc	0.621	0.018	0.766	*0.911*	0.037	*0.953*	0.849	0.069	0.921
Hand-crafted (Untrained)	SC++	0.014	0.000	0.222	0.262	0.050	0.494	0.368	0.059	0.583
	SOLID	0.030	0.000	0.060	0.204	0.013	0.453	0.139	0.008	0.267
	RING++	0.123	0.000	0.429	0.301	0.026	0.482	0.411	0.042	0.583
	BTC	0.158	0.030	0.364	0.168	0.061	0.579	0.236	0.022	0.563
	MapClosure	0.180	0.000	0.305	0.741	0.164	0.851	0.585	0.051	0.738
	NSM	0.016	*0.333*	0.500	0.524	*0.645*	0.899	0.503	*0.617*	0.880
	TreeLoc	0.624	0.057	0.792	0.662	0.016	0.809	0.479	0.017	0.738
	TreeLoc++	**0.940**	**0.978**	**0.990**	**0.924**	**0.980**	**0.993**	*0.864*	**0.897**	**0.982**

· Recall@1 (R@1), Maximum Recall at 100% Precision (MR), and Maximum F1 score (MF1)

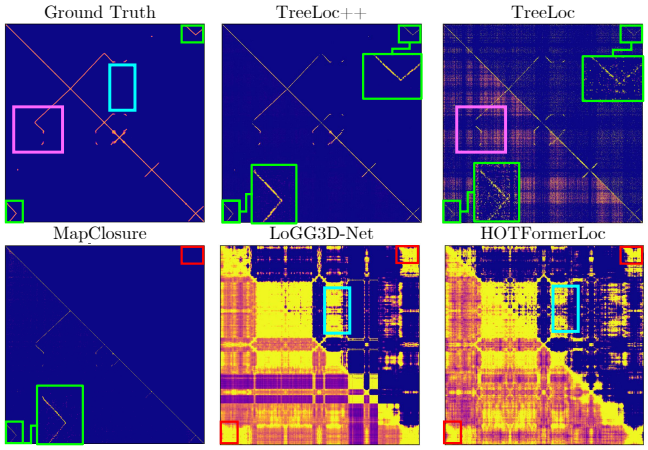
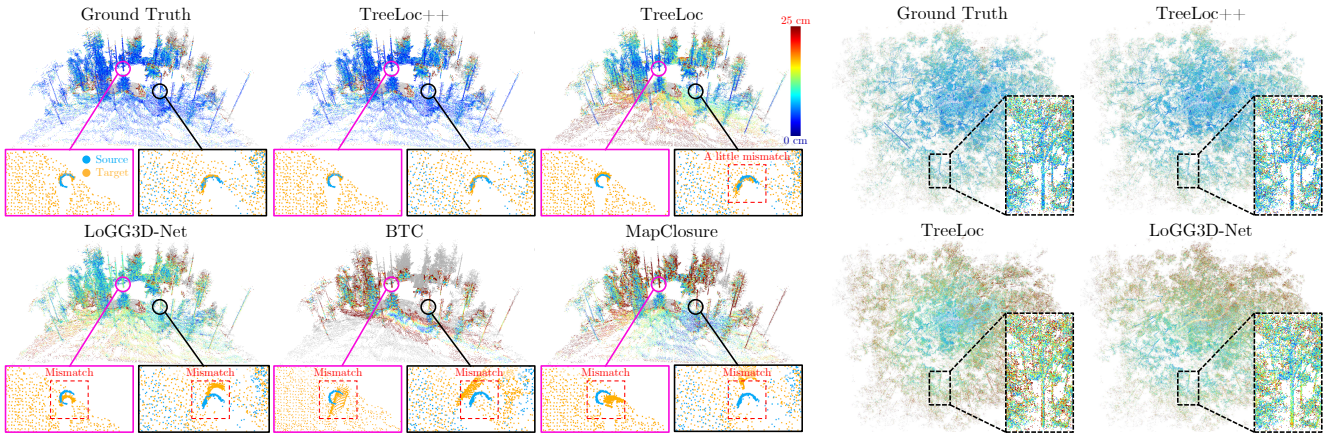


FIGURE 12. (Exp A-2) Feature similarity matrices for Karawatha04 (warmer colors indicate higher similarity). The bottom-left and top-right triangular regions show mid (20–80%) and high (60–80%) similarities, respectively. TreeLoc++ aligns well with ground truth, particularly in the light-green boxed regions; others show noisy (pink), low-contrast (red), or spuriously high similarities across unrelated regions (cyan).

Evaluation protocol: We utilized four sequences from Wild-Places: Venman03–04 and Karawatha03–04. These sequences were not used to train any of the learning-based baselines. We followed the same intra-session evaluation protocol as used for the Oxford dataset.

Results: As summarized in Table. 3, TreeLoc++ achieved the strongest overall performance on Wild-Places. Despite being less suitable for tree reconstruction, TreeLoc++ forms multiple triangles from common trees for reliable retrieval (Fig. 14), while outlier rejection modules further reduce structural ambiguities that often cause failures in TreeLoc. By contrast, most hand-crafted baselines degrade more on Wild-Places, as they struggle to distinguish structurally similar locations in large databases, where repetitive trunk geometry yields similar segment-level structures. Learning-based methods such as LoGG3D-Net and BEVPlace++ achieved relatively high recall on Karawatha03, but lagged behind TreeLoc++ on other metrics due to weaker discrimination.

Similarity analysis: The similarity matrices in Fig. 12 explain both the advantage of TreeLoc++ and the increased



(a) Registration Error in `Evo:Single` (b) Registration Error in `Karawatha04`

FIGURE 13. (Exp B) Registration performance on `Evo` and `Karawatha04`. (a) Point-to-map nearest-neighbor distance after alignment, where TreeLoc and TreeLoc++ achieve low errors and TreeLoc++ is closest to ground truth. (zoom highlights trunk alignment). (b) In `Karawatha04`, TreeLoc++ shows blue-dominant patterns consistent with ground truth, while the other methods exhibit red-dominant regions indicating larger residual errors.

TABLE 4. (Exp B) Metric-level localization performance on two datasets.

	Dimension / Method	Stein am Rhein				Karawatha04			
		R@50	SR	ATE	ARE	R@50	SR	ATE	ARE
2D	LoGG3D-Net	0.676	0.613	0.150	0.478	*0.900*	*0.860*	0.131	0.229
2D	MinkLoc3Dv2	0.037	0.037	0.310	1.878	0.088	0.088	0.283	1.686
2D	BEVPlace++	0.853	0.746	0.152	0.644	0.502	0.495	0.217	0.299
2D	RING++	0.432	0.410	0.206	1.247	0.453	0.401	0.266	0.976
2D	BTC	0.593	0.386	0.231	0.843	0.527	0.292	0.158	0.422
2D	MapClosure	0.526	0.479	0.142	0.635	0.638	0.601	0.147	0.489
2D	NSM	0.605	0.605	0.082	0.363	0.700	0.700	0.059	*0.170*
2D	TreeLoc	*0.976*	*0.906*	*0.047*	*0.197*	0.654	0.465	*0.055*	0.187
2D	TreeLoc++	**0.982**	**0.941**	**0.046**	**0.142**	**0.924**	**0.902**	**0.038**	**0.140**
3D	LoGG3D-Net	0.656	0.599	0.173	1.051	*0.891*	*0.860*	0.157	*0.510*
3D	MinkLoc3Dv2	0.018	0.018	0.269	2.682	0.046	0.046	0.327	2.540
3D	BTC	0.310	0.243	0.239	2.219	0.403	0.272	0.227	1.678
3D	MapClosure	0.526	0.479	0.147	*0.756*	0.635	0.597	0.148	0.563
3D	NSM	0.523	0.523	*0.120*	1.348	0.650	0.650	*0.105*	0.759
3D	TreeLoc	*0.937*	*0.878*	0.121	1.378	0.593	0.423	0.135	1.151
3D	TreeLoc++	**0.939**	**0.906**	**0.087**	**0.470**	**0.924**	**0.902**	**0.068**	**0.307**

· Recall@50 cm (R@50), Success Rate (SR), Average Translation Error (ATE) [m], and Average Rotation Error (ARE) [°].

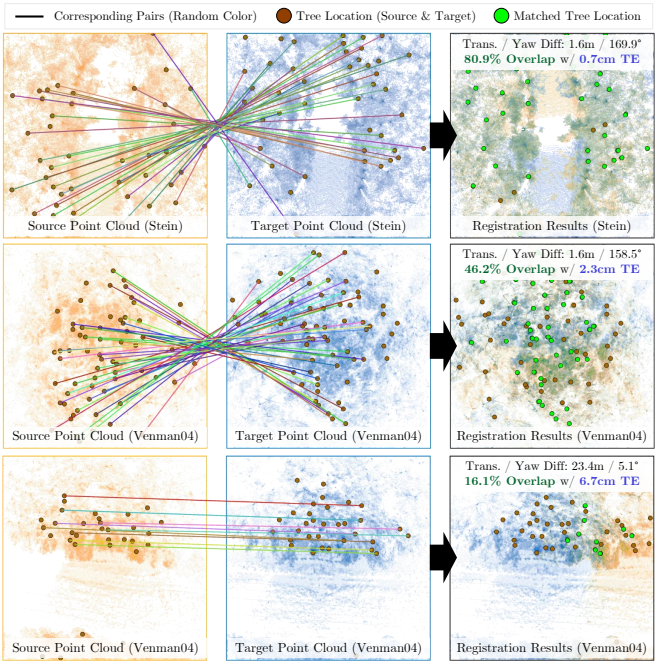


FIGURE 14. (Exp B) Each row visualizes tree correspondences after geometric verification. TreeLoc++ establishes matches not only in high-overlap cases (top), but also in moderate (middle) and low-overlap (bottom) scenarios, even beyond the positive threshold, despite complex tree arrangements. This accurate correspondence leads to centimeter-level errors.

difficulty of Wild-Places with larger databases. We visualize similarity matrices with 20-80% and 60-80% percentile ranges to highlight overall structure and high-confidence matches, respectively. With larger databases and fewer revisits, true positives become sparse and the separation between true positives and true negatives becomes harder to maintain, making retrieval more challenging. TreeLoc++ preserves this separation by closely following the ground truth spatial structure, enabling reliable retrieval. In contrast, TreeLoc shows lower contrast and more noise, increasing false positives, while MapClosure has limited dynamic range, reducing true positives and increasing false negatives. Other global descriptor baselines often assign high similarity to unrelated locations, degrading precision and increasing false positives.

In summary, TreeLoc++ achieves the strongest intra-session place recognition across diverse forests without training and remains robust across scales.

B. Intra-session Metric Localization

After completing the evaluation of intra-session place recognition on two forest datasets, we turn our attention to another key strength of this work: metric-level localization performance. In this section, we assess whether each method can recover the relative poses following place recognition.

Evaluation protocol: We evaluated metric localization on four sequences with high R@1 to ensure sufficient true-

TABLE 5. (Exp C) Inter-session place recognition performance in Wild-Places.

Category	Method	Venman01-04 (12 query-database pairs)								Karawatha01-04 (12 query-database pairs)							
		R@1		MR		MF1		AUC		R@1		MR		MF1		AUC	
		Mean	S	Mean	S	Mean	S	Mean	S	Mean	S	Mean	S	Mean	S	Mean	S
Learning-based (Wild-Places)	TransLoc3D	0.553	0.464	0.008	-0.048	0.707	0.678	0.604	0.511	0.711	1.316	0.013	-0.103	0.831	1.561	0.756	1.268
	LoGG3D-Net	0.660	**1.242**	0.003	-0.282	0.813	*1.382*	0.732	0.872	**0.897**	**1.848**	0.002	-0.278	*0.948*	**2.299**	0.921	*1.638*
	MinkLoc3Dv2	0.755	0.782	0.033	0.161	0.857	1.030	0.818	0.881	0.849	*1.607*	0.062	0.150	0.919	1.889	0.911	1.592
	BEVPlace++	0.735	0.701	0.091	-0.025	0.840	0.920	0.841	0.842	0.767	0.934	0.016	-0.193	0.868	1.179	0.767	0.822
	ForestLPR	0.718	0.695	0.023	0.006	0.842	1.006	0.839	1.010	0.790	1.136	0.072	0.024	0.885	1.366	0.885	1.205
	HOTFormerLoc	**0.899**	1.173	0.155	-0.057	*0.946*	1.448	*0.938*	*1.350*	*0.885*	1.341	0.088	0.026	0.941	1.628	*0.937*	1.471
Hand-crafted (Untrained)	SC++	0.245	0.429	0.056	0.129	0.425	0.631	0.396	0.430	0.370	0.632	0.117	0.146	0.560	0.817	0.564	0.704
	SOLID	0.150	0.199	0.014	-0.263	0.288	0.321	0.241	0.093	0.158	0.357	0.000	0.000	0.300	0.501	0.211	0.318
	RING++	0.322	0.448	0.057	-0.004	0.503	0.624	0.500	0.458	0.341	0.713	0.030	-0.143	0.509	0.848	0.402	0.439
	BTC	0.185	0.543	0.006	-0.337	0.331	0.594	0.216	0.338	0.068	0.169	0.005	-0.193	0.221	0.150	0.128	0.024
	MapClosure	0.478	0.338	0.054	-0.008	0.618	0.459	0.465	0.279	0.484	0.420	0.036	-0.121	0.641	0.571	0.479	0.347
	NSM	0.287	0.014	*0.444*	*0.221*	0.701	0.634	0.824	0.859	0.248	0.086	*0.544*	*0.545*	0.828	0.948	0.907	1.270
	TreeLoc	0.766	0.940	0.046	-0.019	0.890	1.202	0.853	0.954	0.486	0.938	0.003	0.785	0.858	1.747	0.838	1.577
	TreeLoc++	*0.896*	*1.207*	**0.970**	**1.469**	**0.993**	**2.303**	**0.998**	**3.281**	0.728	1.266	**0.849**	**1.008**	**0.974**	*2.149*	**0.995**	**2.801**

· Recall@1 (R@1), Maximum Recall at 100% Precision (MR), Maximum F1 score (MF1), Area Under the Precision–Recall Curve (AUC), and Stability Ratio (S).

positive retrievals for downstream pose estimation. Similar to SGV [60], we evaluate LoGG3D-Net and MinkLoc3Dv2 using feature-based correspondences and RANSAC registration to directly estimate the 6-DoF pose after retrieval.

TreeLoc++: As shown in Table. 4, TreeLoc++ achieved the best overall performance across both 2D and 3D metrics. While TreeLoc performed well in `Stein am Rhein`, its accuracy dropped in cluttered scenes such as `Karawatha04`. This degradation was more pronounced in 3D due to TreeLoc's independent estimation of roll, pitch, and height, which is prone to compounded errors. In contrast, TreeLoc++ jointly estimates these components, yielding more stable and accurate results. Moreover, its high R@50 score indicates successful localization even when the top-1 retrieval does not fall within the evaluation threshold.

Baselines: BEV-based methods suffered from viewpoint sensitivity and resolution constraints. BEVPlace++ and RING++ do not estimate roll and pitch, causing BEV projections to degrade under viewpoint changes. MapClosure attempts ground-plane compensation but fails under uneven terrain or occlusions (Fig. 4), and ORB-based descriptors on BEV images remain insufficiently distinctive. Similarly, local- and segment-based methods such as BTC and NSM are affected by partial observations and occlusions in forests. These factors destabilize BTC's local features and NSM's segment centroids, thereby limiting their localization accuracy even when NSM directly localizes against a global map.

For learning-based methods using 3D point features, performance depended strongly on feature resolution. MinkLoc3Dv2 uses heavily downsampled representations, limiting discriminative power. In the case of LoGG3D-Net, it retained more spatial detail but still lagged behind TreeLoc++. Learning-based methods also incurred substantial computational overhead, as they must compute pairwise distances between high-dimensional local features to establish feature-level correspondences for metric localization.

Alignment accuracy: To assess alignment accuracy, we evaluated registration on the same set of successful query-match pairs across methods, measuring accuracy via nearest neighbor correspondences within 50 cm. As shown in Fig. 13a, TreeLoc++ produced the most accurate registration, closely similar to the ground truth. TreeLoc and LoGG3D-Net exhibited moderate misalignments, which became more pronounced in cluttered scenes such as `Karawatha04` (Fig. 13b). BTC and MapClosure showed severe failures. TreeLoc++'s performance stems from robust tree-level correspondences in Fig. 14, which remain reliable under partial overlap and enable centimeter-level accuracy comparable to point-based registration methods, as detailed in Appendix C.

C. Inter-session Place Recognition

The previous three experiments focused exclusively on performance within intra-session settings. We now extend our evaluation to inter-session scenarios with larger temporal gaps, analyzing the robustness of the proposed method under more challenging long-term environmental changes. We aim to determine whether methods maintain consistent performance under seasonal and structural variation.

Evaluation protocol: TreeLoc++ was evaluated in an inter-session setting using the local inventory mode on `Venman01-04` and `Karawatha01-04`. Learning-based methods were trained on Wild-Places, following the official protocol, which restricts evaluation to predefined regions, as shown in Fig. 15. To quantitatively assess robustness across sessions, we introduce a Stability Ratio, $S = \log(\mu/(\sigma + \epsilon))$, where μ and σ denote the mean and standard deviation of the performance metric over the 12 inter-session pairs, and ϵ is a small constant for numerical stability. Unlike raw variance, this ratio favors methods that are both high-performing and consistent.

TreeLoc++: As reported in Table. 5, TreeLoc++ achieved the highest mean performance and Stability Ratio on most

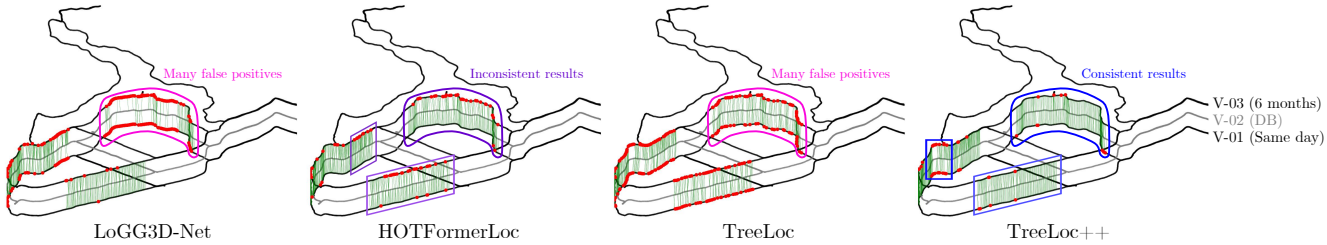
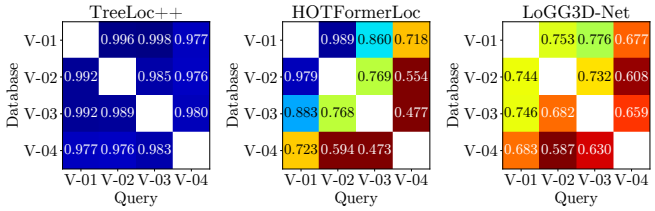
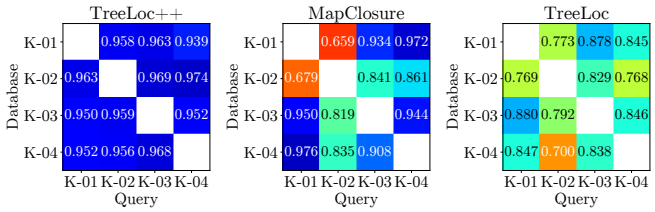


FIGURE 15. (Exp C) Inter-session place recognition on `Venman01-03`. Following the Wild-Places protocol, evaluation is conducted on unseen regions excluding the training area. LoGG3D-Net and TreeLoc produce numerous false positives (red dots) across sessions (pink). HOTFormerLoc performs well for same-day sequences (`Venman01-02`) but degrades on `Venman03` with a 6-month gap (purple). TreeLoc++ remains consistent across temporal shifts (blue) and retrieves more true positives, even though its false positive regions also appear consistently across sessions.



(a) Accuracy in `Venman`



(b) Accuracy in `Karawatha`

FIGURE 16. (Exp C) Accuracy matrix for inter-session place recognition. Each cell represents the fixed-threshold accuracy for one query-database session pair. Blue tones indicate higher accuracy, while red tones indicate lower accuracy. A desirable pattern is a uniformly blue matrix, which indicates that a method maintains consistently high performance across different session pairs without requiring pair-specific threshold tuning. TreeLoc++ shows the most reliable performance across all session pairs.

metrics. Even when learning-based models were trained on the same dataset, TreeLoc++ surpassed them in both mean performance and stability, ranking second only in R@1 on `Venman`. On `Karawatha`, recall decreased in open areas with fewer distinctive vertical structures; nevertheless, TreeLoc++ achieved Stability Ratio comparable to or higher than several learning-based methods.

Baselines: Most hand-crafted methods degraded in the inter-session setting, indicating limited robustness to temporal variation. While TreeLoc outperformed other hand-crafted baselines, it still lagged behind learning-based methods. NSM also degraded substantially under seasonal variation, as it matches individual segment descriptors without modeling inter-segment connectivity. This makes it more sensitive to appearance changes in similar trunk segments, so that even when the mean performance remains relatively high, the Stability Ratio becomes noticeably lower. Among learning-based approaches, LoGG3D-Net and HOTFormerLoc achieved the highest mean performance; however, LoGG3D-Net degraded on `Venman`, and HOTFormerLoc's low Stability Ratios suggest inconsistent performance across sessions. As shown in Fig. 15, LoGG3D-Net produced frequent mismatches, and HOTFormerLoc deteriorated with increasing temporal gaps. Furthermore, while these methods still retrieve true positives via minimum descriptor distance, accompanying false positives necessitate an appropriate threshold to filter out incorrect matches.

Fixed-threshold transfer: Consequently, we assessed inter-session robustness using fixed-threshold accuracy analysis in Fig. 16. To reflect realistic deployment where thresholds cannot be retuned for each session pair, we fixed a single operating threshold per method by maximizing F1 score on `01-02` pair, and applied it unchanged to all remaining pairs.

As shown in Fig. 16a and Fig. 16b, TreeLoc++ was the only method that consistently achieved high accuracy across all pairs. In contrast, baseline methods exhibited large fluctuations. HOTFormerLoc and LoGG3D-Net degraded substantially on `Venman04` due to pronounced environmental changes. Conversely, MapClosure and TreeLoc performed poorly on `Karawatha01-02` because low thresholds admitted many false positives, but improved in later sessions where larger appearance differences made discrimination easier. Overall, these results suggest that the similarity scores used by the baselines are brittle under fixed thresholds. In contrast, the proposed overlap score $\mathcal{O}(Q, C)$ maintains stable separation between true and false matches, demonstrating robust performance without session-specific tuning.

D. Multi-session Metric Localization with Global DFIs

This experiment assesses the feasibility of using multiple global forest inventories to localize incoming trajectories using local observations in a scalable and robust way.

Evaluation protocol: We evaluated relocalization against a pre-built global forest inventory that serves as a persistent reference map. Query frames were processed in local inventory mode, while the database consisted of a single global inventory. To cover the spatial extent of the reference map, we constructed database entries at $5\,\mathrm{m}$ intervals in the horizontal plane as shown in Fig. 17a. At each sampled location, a local inventory was formed by extracting nearby trees, and the corresponding TreeLoc++ descriptor was stored in the global descriptor database. Experiments were

TABLE 6. (Exp D) Global relocalization results, where loop closure and inter-session constraints are added incrementally from additional sequences.

		Metrics	LoGG3D-Net SR	LoGG3D-Net F1	BTC SR	BTC F1	MapClosure SR	MapClosure F1	TreeLoc $\mathcal{O} > 0.2$	**TreeLoc++** $\mathcal{O} > 0.2$
25:00	+ Loop	Size	*1.2 MB*		44MB		2.6 MB		**227.9 KB**	
		# Const.	16	**55**	5	14	1	1	33	*38*
		ATE	0.35	10.40	0.37	*0.31*	0.46	0.46	**0.31**	0.32
		ARE	1.53	45.61	1.63	**1.33**	1.96	1.96	*1.49*	1.53
	+ 25:02	Size	*3.7 MB*		90.8 MB		7.7 MB		**11.2 KB**	
		# Const.	0	29	0	0	0	*25*	2	**58**
		ATE	33.73	20.60	33.80	33.78	34.05	15.20	*10.21*	**0.57**
		ARE	41.58	63.96	42.20	41.52	42.48	24.36	*7.93*	**1.75**
	+ 25:03	Size	*2.0 MB*		20.2 MB		2.9 MB		**5.9 KB**	
		# Const.	0	*17*	0	1	0	15	7	**24**
		ATE	33.73	19.35	33.80	23.97	34.05	11.50	*2.74*	**0.60**
		ARE	41.58	53.14	42.20	26.64	42.48	37.06	*7.05*	**1.92**
	+ 25:05	Size	*3.9 MB*		74MB		7.3 MB		**14.9 KB**	
		# Const.	0	1	0	0	0	*6*	3	**25**
		ATE	33.73	18.66	33.80	23.97	34.05	25.93	*1.96*	**0.53**
		ARE	41.58	51.55	42.20	26.64	42.48	80.74	*3.49*	**1.91**
V-01	+ Loop	Size	*11.3 MB*		1.60 GB		21.7 MB		**3.2 MB**	
		# Const.	314	**1400**	61	0	84	336	273	*338*
		ATE	**6.74**	174.78	7.18	175.88	6.88	7.95	7.53	*6.76*
		ARE	*2.94*	71.39	3.06	66.90	2.96	3.52	3.36	**2.92**
	+ V-02	Size	*54.1 MB*		2.00 GB		130.5 MB		**68.7 KB**	
		# Const.	217	**1451**	6	46	0	19	69	*602*
		ATE	*1.72*	146.71	2.90	295.14	23.60	104.68	5.47	**1.23**
		ARE	**0.99**	52.05	1.56	81.21	5.00	75.77	2.25	*1.13*

· Validated success constraints (SR), F1 score-based constraints (F1), descriptor and inventory storage (Size), number of selected constraints (# Const.), Average Translation Error (ATE) [m], and Average Rotation Error (ARE) [°].

conducted on `Evo23:00-03`, `Evo25:00`, `02`, `03`, `05`, and `Venman01-02`. For each sequence, initial odometry was obtained using GeoTransformer [70]. We first generated intra-session constraints using local inventories, and then incrementally added inter-session constraints by matching queries to the global descriptor database.

Constraint selection differed by method. Baseline constraints were selected either by (i) ground truth-validated success (SR) or (ii) a confidence threshold chosen to maximize F1 from intra-session localization. For TreeLoc and TreeLoc++, we used a fixed threshold, $\mathcal{O}(Q, C) > 0.2$. To ensure a sufficient number of constraints under the SR-based setting, we relaxed the true-positive distance threshold to 10 m. All selected constraints were optimized using GTSAM [71]. Trajectory accuracy was evaluated with the Evo toolkit [72] using ATE and ARE. Except for intra-session loop evaluations, we report results without applying SE(3) alignment during Evo evaluation. We additionally report the number of constraints produced by each method, as well as the storage required for descriptors and forest inventories.

Relocalization accuracy: As reported in Table. 6, TreeLoc++ achieved among the lowest trajectory errors in nearly all settings by correcting raw odometry with accurate and reliable constraints. Even when baselines were restricted to SR-validated constraints, TreeLoc++ typically attained lower or comparable error by leveraging a larger set of accurate constraints. When baselines used F1-based thresholds, the number of selected constraints increased due to false positives, substantially degrading trajectory accuracy.

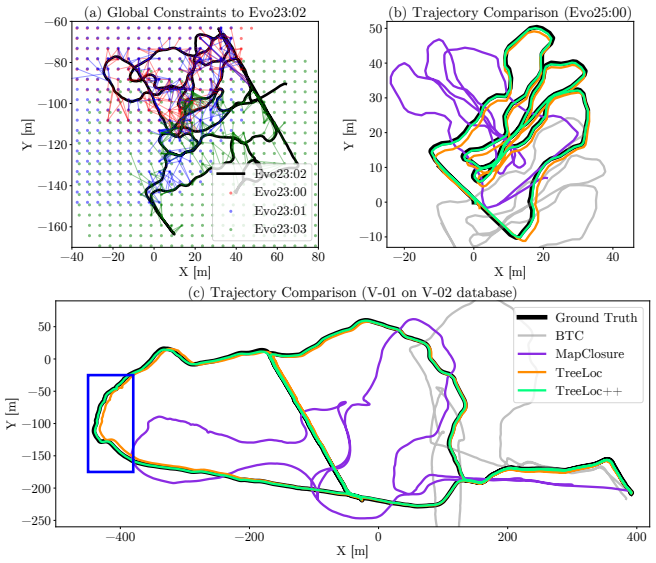


FIGURE 17. (Exp D) Relocalization against a global forest inventory. (a) A global descriptor database is built by querying the global inventory at uniformly sampled poses on a 5 m grid, efficiently covering the full map extent. (b-c) Using relocalization constraints to align trajectories to the global frame, TreeLoc++ produces trajectories closest to the ground truth, as highlighted in the blue box. In particular, the blue box in (c) highlights the close agreement between the aligned trajectory and the ground truth.

Overall, these results indicate that threshold-based constraint selection in existing methods is brittle in practical global relocalization, consistent with observations in earlier sections.

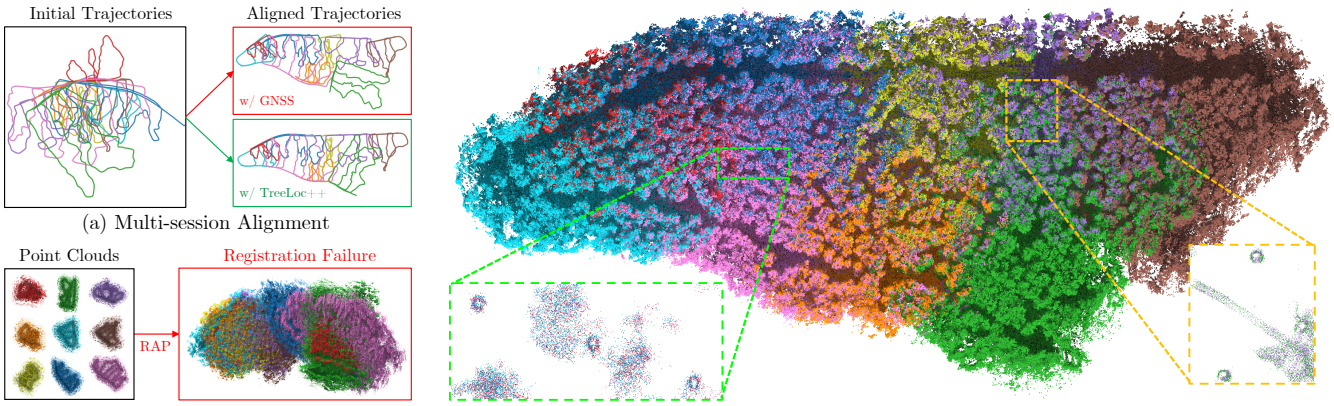
Fig. 17a illustrates `Evo23:02`, where TreeLoc++ aligned the trajectory to the global reference frame by incorporating constraints from three inter-session sequences. As shown in Fig. 17b-c, the optimized trajectories closely match the ground truth, benefiting from consistent and accurate constraints. On `Venman`, which contains 6,918 database entries, TreeLoc++ demonstrated its reliability by efficiently querying the large-scale global database and aligning all trajectories into a common world frame.

Storage and scalability: TreeLoc++ also offers clear storage advantages. Compared to other descriptor-based methods, it requires substantially less storage due to its compact representation. Although the `Loop` configuration stores a local inventory per pose, the total storage remains on the megabyte scale, and the global inventory is even smaller, as shown in Table. 6. These results show that TreeLoc++ supports accurate and scalable global trajectory estimation with minimal storage overhead.

E. Multi-session Registration with Global DFIs

In this section, we seek to address the following question: whether global forest inventories alone are sufficient for accurate map-to-map registration across sessions, in the absence of raw point clouds or GNSS.

Evaluation protocol: This section evaluates map-to-map registration using only global forest inventories, without relying on raw point clouds. For each session, a single global forest inventory served as the map representation. System



(a) Multi-session Alignment

(b) Multi-session Registration using RAP

(c) Combined Map after Multi-session Alignment (TreeLoc++)

FIGURE 18. (Exp E) Multi-session alignment results. (a) Each session trajectory is expressed in its own local frame, with all trajectories starting at the origin; GNSS-based alignment remains inconsistent across sessions, with clear misalignment in the green trajectory, whereas TreeLoc++ yields more accurate multi-session alignment. (b) Point cloud-based multi-session registration fails in repetitive forest environments due to ambiguous geometry. (c) Combined map generated using TreeLoc++ aligned poses, where the zoomed views in the orange and green boxes highlight the consistent alignment of tree trunks and fallen trees observed across multiple sessions, demonstrating accurate alignment.

TABLE 7. (Exp F-1) Invariance analysis across different orientation

Methods	Yaw (13 pairs)			Roll & Pitch (13 pairs)			Full Rot. (169 pairs)		
	Mean	σ	S	Mean	σ	S	Mean	σ	S
MapClosure	0.363	0.002	2.281	0.273	0.132	0.316	0.281	0.130	0.336
LoGG3D-Net	0.664	0.002	2.592	0.168	0.258	-0.186	0.166	0.252	-0.181
HOTFormerLoc	0.385	0.361	0.028	0.191	0.288	-0.179	0.109	0.196	-0.255
TreeLoc++	**0.951**	**0.001**	**3.024**	**0.952**	**0.001**	**3.075**	**0.952**	**0.001**	**3.200**

· Standard deviation (σ) and Stability Ratio (S).

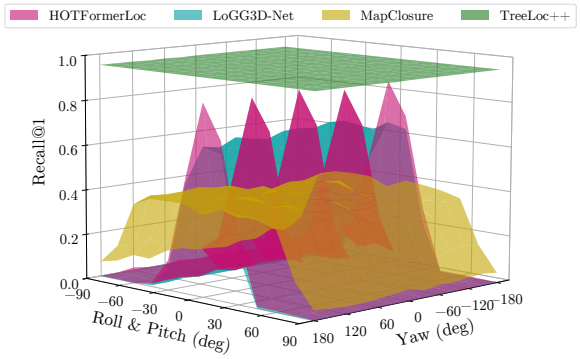


FIGURE 19. (Exp F-1) R@1 performance under synthetic orientation perturbations. TreeLoc++ maintains high recall across the full rotation range, whereas the baselines degrade under roll-pitch perturbations.

poses were sampled along each session, and at each sampled pose, nearby trees were extracted from the global inventory to form pose-centric inventories. TreeLoc++ descriptors were computed for these samples and matched across all session pairs to establish inter-session constraints, using a fixed overlap threshold of $\mathcal{O}(Q, C) > 0.2$; the rationale for this choice is discussed in Appendix F.2.

TreeLoc++: Fig. 18 illustrates the initial and aligned trajectories of `Evo25:00-08`, along with a combined point-cloud visualization obtained by transforming scans using the aligned poses. Thanks to TreeLoc++'s high localization accuracy, trunk structures align consistently across sessions, even though raw point clouds are not used for constraint generation or registration.

GNSS and point-cloud baselines: For comparison, we performed multi-session alignment using GNSS by estimating a rigid transform between each session's local trajectory and its GNSS-referenced trajectory. As shown for `Evo25:08` (green), this produced noticeable misalignments and inconsistent trajectories across sessions, indicating that GNSS provides only coarse alignment and lacks the precision needed for stem-level consistency in forests. We further compared against RAP [73], a point cloud-based registration baseline, as shown in Fig. 18b. In the unstructured and repetitive forest geometry, RAP failed to converge to a correct alignment, resulting in substantial registration errors.

Long-term scalability: Finally, Fig. 1 shows large-scale alignment across all 15 sessions from `Evo23:00-05` and `Evo25:00-08`. Despite a two-year gap and different LiDAR FoVs (104° in 2025 vs. 31° in 2023), TreeLoc++ aligned all sessions using only the global forest inventory, with a total map size of 250 KB. These results highlight the robustness of TreeLoc++ and demonstrate that lightweight, geometry-based inventories support scalable multi-session registration, enabling consistent inventory maintenance and long-term ecological monitoring as detailed in Appendix D.

F. Robustness and Efficiency Analysis

1) Viewpoint Robustness

Evaluation Protocol: We evaluated the rotation invariance of TreeLoc++ using `Venman02` as the database and `Venman03` as the query. To simulate viewpoint variation, we applied synthetic rotations to the database point clouds and DFIs. Yaw was varied from $-180°$ to $180°$ in $30°$ steps, and roll and pitch were jointly varied from $-90°$ to $90°$ in $15°$ steps. For each augmented database, we measured R@1 and quantified consistency via the Stability Ratio (S).

Results: As shown in Table. 7 and Fig. 19, all methods except TreeLoc++ failed under roll-pitch perturbations. HOTFormerLoc exhibited cyclic local peaks under yaw rotations

TABLE 8. (Exp F-2) Total runtime for all queries on large databases [s]

Method	Place Recognition	Pose Estimation	Total Time
BTC	382.29	10.49	392.78
MapClosure	5.60	4.46	10.06
TreeLoc++	**2.98**	**3.43**	**6.41**

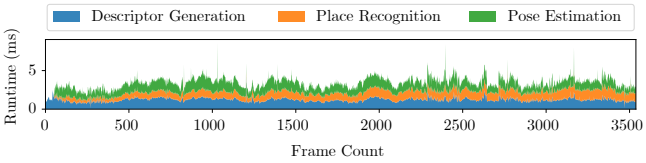


FIGURE 20. (Exp F-2) Runtime breakdown for intra-session localization in `K-03`. Colored bands indicate descriptor generation, place recognition, and pose estimation time for each query frame. TreeLoc++ maintains consistently low per-query latency throughout the sequence, with descriptor generation remaining below 1 ms.

at 90° intervals. LoGG3D-Net and MapClosure showed partial robustness to small tilts, but their accuracy deteriorated as perturbations increased. In contrast, TreeLoc++ maintained high R@1 across all rotations, enabled by yaw-invariant TDH/PDH and 2D triangles, and axis-based alignment that stabilizes projections under tilt. These results confirm TreeLoc++'s robustness to severe viewpoint variations.

2) Runtime Efficiency

Evaluation Protocol: We conducted two experiments to analyze runtime. First, we reported the total inter-session localization time, including place recognition and pose estimation (geometric verification). To compare localization efficiency across methods, we followed the large-scale setup from §D, using 6,918 database entries and 1,451 queries. Second, to analyze TreeLoc++ in detail, we used `Karawatha03` to measure the runtime of descriptor generation, place recognition, and pose estimation separately. For a fair comparison, we evaluated only methods with publicly available C++ implementations on an 11th Gen Intel Core i7-11700 @ 2.50 GHz and an NVIDIA GeForce RTX 3080 GPU.

Results: As shown in Table. 8, TreeLoc++ achieved the lowest runtime among all methods. BTC incurred high runtime due to costly triangle-wise comparisons during place recognition, while MapClosure was faster by leveraging lightweight ORB-based image matching. TreeLoc++ outperformed both by avoiding per-candidate geometric checks and limiting pose estimation to a small set of top-ranked matches. As shown in Fig. 20, TreeLoc++ also benefits from fast descriptor generation via DFI, requiring less than 1 ms per query. This makes on-demand descriptor computation nearly as efficient as preloading, avoiding additional I/O cost and storage. Moreover, TreeLoc++ maintains low runtime as the database grows, due to efficient filtering using TDH and PDH. Although histogram distance computations grow with the database size, the increase remains small thanks to the low-dimensional histogram. These design choices enable TreeLoc++ to support fast localization while conserving compute for downstream tasks like mapping and planning.

TABLE 9. (Exp G-1) Ablation studies on TDH and PDH

Config.		`Stein am Rhein`				`Karawatha03`			
TDH	PDH	# TP	FNR	R@1	R@50	# TP	FNR	R@1	R@50
✗	✗	-	-	0.939	0.935	-	-	0.820	0.845
✓	✗	15.069	0.024	0.933	**0.943**	3.852	0.063	0.834	0.863
✗	✓	16.701	0.022	0.939	0.933	3.771	0.076	0.817	0.845
✓	✓	**17.016**	**0.020**	**0.941**	0.939	**4.662**	**0.038**	**0.864**	**0.874**

· Number of true positives (# TP), False Negative Rate (FNR), Recall@1 (R@1) and Recall@50 cm (R@50).

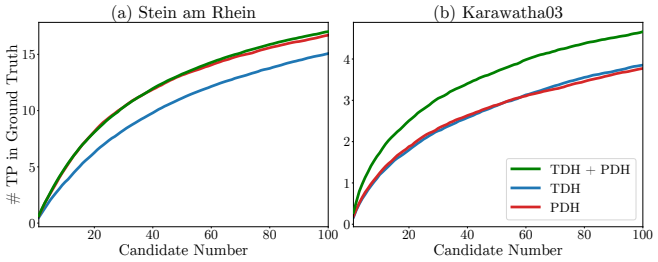


FIGURE 21. (Exp G-1) Number of true positives versus candidate count in coarse retrieval. In `Stein am Rhein`, TDH+PDH stays only slightly above PDH because PDH already retrieves most true positives in this compact scene. In `Karawatha03`, the gap becomes larger, showing that TDH and PDH provide complementary cues in a larger and more structurally ambiguous environment.

G. Ablation Studies

The contribution of each TreeLoc++ component was analyzed through an ablation study. Experiments were conducted on two contrasting environments: the small-scale `Stein am Rhein` and the large-scale `Karawatha03`. Complementary analyses of parameter sensitivity, robustness to DFI quality, and the statistical comparison between TreeLoc and TreeLoc++ are provided in Appendix F and G.

1) Ablation on Coarse Retrieval with TDH and PDH

Evaluation Protocol: We assessed the impact of TDH and PDH by incrementally enabling each component and measuring candidate retrieval performance. We report R@1 and R@50, the average number of true positives (# TP) in the retrieved candidate set, and the false negative rate (FNR), defined as the fraction of queries for which the ground truth match is absent from the retrieved candidates.

Results: As shown in Table. 9, `Stein am Rhein` exhibited only minor changes in recall, since its limited spatial extent already yields many true positives in coarse retrieval. Accordingly, PDH alone already recovers most true positives, and combining TDH and PDH provides only a marginal additional benefit, slightly increasing the number of true positives while reducing the FNR. In this compact setting, PDH also retrieves more true positives than TDH because pairwise-distance statistics are already stable thanks to the sufficient number of observed trees. On the larger `Karawatha03`, this combination substantially improved both recall and FNR. In this larger and more structurally ambiguous scene, PDH alone can be less distinctive because different locations may share similar pairwise-

TABLE 10. (Exp G-2) Ablation studies on refinements

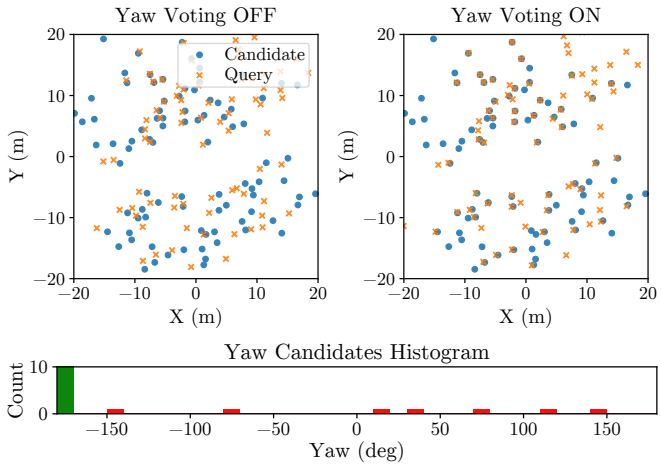
Refinement			Stein am Rhein				Karawatha03			
Pen.	Vot.	Filt.	R@1	AUC	R@50	SR	R@1	AUC	R@50	SR
✗	✗	✗	0.902	0.983	0.933	0.872	0.644	0.825	0.809	0.577
✓	✗	✗	0.927	**0.998**	0.935	0.894	0.803	0.995	0.808	0.730
✓	✓	✗	0.933	**0.998**	**0.939**	0.902	0.852	**0.998**	0.850	0.796
✓	✓	✓	**0.941**	**0.998**	**0.939**	**0.906**	**0.864**	**0.998**	**0.874**	**0.816**

· Spatial Penalty (Pen.), Yaw Voting (Vot.), DBH filtering (Filt.), Recall@1 (R@1), Area Under the Precision–Recall Curve (AUC), Recall@50 cm (R@50) and Success Rate (SR).

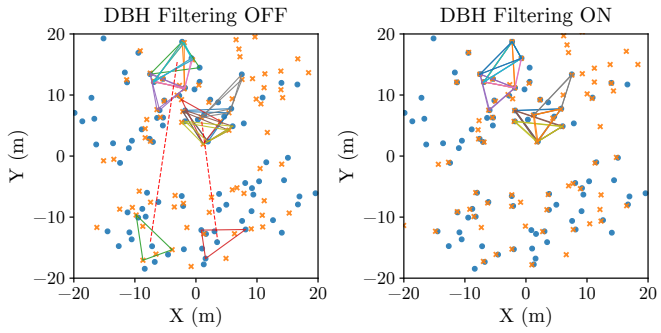
distance distributions, while TDH alone remains sensitive to reference-centered layout variations caused by missing trees or slight translation shifts. Their combination therefore provides complementary cues from pairwise geometric stability in PDH and location-centered distribution information in TDH, yielding more true positives, lower FNR, and better place recognition performance. Fig. 21 also reflects this difference: in Stein am Rhein, the TDH+PDH curve stays only slightly above PDH because PDH already retrieves most true positives, whereas in Karawatha03, the gap becomes larger because the combined descriptor better suppresses structurally ambiguous candidates. When using only triangle descriptors without histograms, geometrically similar but distant triangles frequently collide in the hash space, causing false scene matches. In contrast, combining histograms stabilizes retrieval by jointly encoding inter-tree relationships and location-centered distributions, providing complementary cues for robust candidate selection.

2) Ablation on Candidate Re-ranking and Outlier Rejection

Evaluation Protocol: We evaluated three refinement strategies: the translation penalty, yaw voting, and DBH filtering. Since these components influence both place recognition and metric localization, we report R@1, AUC, R@50, and SR.
Results: As shown in Table. 10, enabling the translation penalty improved R@1, AUC, and SR, while R@50 changed only marginally. By promoting spatially adjacent candidates, it increased R@1, and this led to more accurate overlap scoring, resulting in higher AUC. The number of correctly retrieved true positives also increased, which raised the SR. Yaw voting and DBH filtering further improved performance by suppressing outlier triangle correspondences, leading to more accurate scoring and correspondence selection. These gains were more pronounced on Karawatha03, where the large inventory generated a substantial number of triangles, making the matching stage more prone to outliers. Fig. 22a illustrates that yaw voting enables more reliable pose estimation on Karawatha03; despite the presence of outlier correspondences, it selects a consistent inlier set around the dominant yaw, improving query-candidate alignment. DBH filtering further improves performance when applied after yaw voting by removing residual outliers that yaw voting alone cannot suppress. As shown in Fig. 22b, these outliers otherwise lead to inaccurate pose estimation, whereas DBH filtering yields more consistent correspondences.



(a) Effect of Yaw Voting in Karawatha03



(b) Effect of DBH Filtering in Karawatha03

FIGURE 22. (Exp G-2) Ablation of yaw voting and DBH filtering. (a) Registration with yaw voting disabled and enabled, where a histogram built from triangle matches suppresses orientation outliers by selecting the dominant yaw, improving query-candidate alignment. (b) 10 triangle matches with DBH filtering disabled and enabled; consistent colors indicate corresponding pairs. Without DBH filtering, outlier pairs (red dashed) remain and degrade pose estimation, while DBH filtering removes them.

TABLE 11. (Exp G-3) Ablation studies on system components

System Components			Stein am Rhein				Karawatha03			
Reuse	IRLS	Corr.	R@1	R@50	ATE	ARE	R@1	R@50	ATE	ARE
✗	✗	✗	0.939	0.915	0.152	1.331	0.825	0.844	0.099	0.730
✓	✗	✗	0.939	0.921	0.154	1.356	0.852	0.852	0.101	0.737
✓	✓	✗	**0.941**	0.927	0.149	1.359	**0.864**	0.871	0.103	0.751
✓	✓	✓	**0.941**	**0.939**	**0.087**	**0.470**	**0.864**	**0.874**	**0.080**	**0.260**

· Preceding-inventory reuse (Reuse), Iterative Reweighted Least Squares (IRLS), Joint Vertical Correction (Corr.), Recall@1 (R@1), Recall@50 cm (R@50), Average Translation Error (ATE) [m], and Average Rotation Error (ARE) [°].

3) Ablation on TreeLoc++ Extensions over TreeLoc

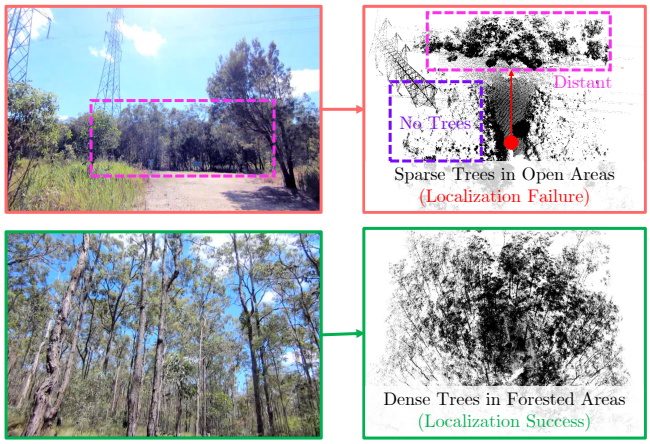
Evaluation Protocol: We evaluated three extensions in TreeLoc++ over TreeLoc: reusing preceding inventories to increase the number of trees, planar alignment via IRLS, and vertical correction for full 6-DoF pose refinement. Their impact was quantified using R@1, R@50, ATE, and ARE.
Results: As shown in Table. 11, reusing preceding windows had little effect on Stein am Rhein, where most frames already contain sufficient tree observations. In contrast, in open or sparse areas of Karawatha03, this reuse increased

the number of matched triangles, improving recall. IRLS refinement was particularly effective in occluded or sparsely observed scenes. When tree centers were inconsistently estimated, IRLS stabilized the 2D alignment around the initial estimate, producing more consistent correspondences and improved recall. While these two strategies mainly improved retrieval, their impact on ATE and ARE was limited because the pose refinement stage was unchanged. In contrast, vertical correction substantially reduced ATE and ARE by jointly optimizing roll, pitch, and height, enabling accurate 6-DoF localization without degrading recall.

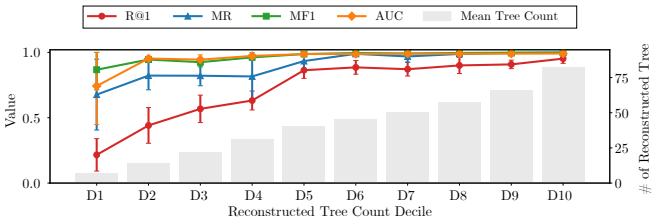
VI. Limitations of TreeLoc++

Failure Cases in Open Areas: TreeLoc++ performance varies substantially between the open areas and deep-forest regions. In the open areas of `Karawatha03`, tree observations tend to be sparse and distant, providing insufficient geometric constraints for robust localization as shown in Fig. 23a. This often leads to degraded downstream pose estimation. However, in deep forests, the density of nearby trunks allows the system to achieve highly successful localization through stable correspondences. As also shown in Fig. 23b, the average inter-session results across all pairs in `Karawatha` indicate that recall decreases when only a small number of trees are available, and gradually recovers as tree density increases. In contrast, F1 score and AUC degrade much less, suggesting that TreeLoc++ can still evaluate candidate matches precisely even in sparse scenes. While TreeLoc++ is environment-dependent, it naturally complements GNSS: open areas typically offer strong GNSS reception thanks to the lack of canopy obstruction, whereas dense forests provide richer geometric features for reliable localization when GNSS becomes unreliable. This suggests a practical deployment strategy that uses GNSS in open areas and switches to TreeLoc++ as tree density increases.

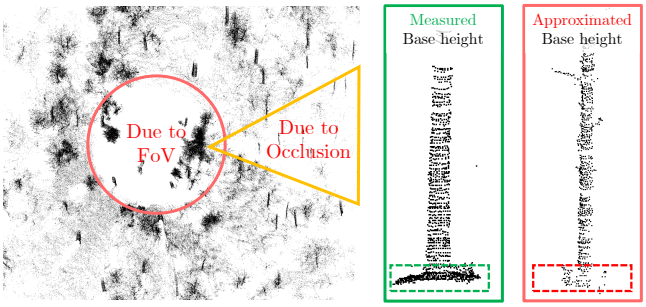
Inaccurate Base Height under Terrain Occlusion: TreeLoc++ relies on estimated base heights for full 6-DoF alignment, so base height errors can destabilize roll, pitch, and height refinement. As shown in Fig. 23c, this occurs when limited sensor FoV or severe occlusions prevent sufficient terrain observations around a tree. In such cases, the base height must be approximated from nearby data, which can deviate from the true value. This issue is more pronounced in local inventory mode due to limited payload coverage. In contrast, global inventory mode mitigates it by leveraging all explored payloads during tree reconstruction, providing broader multi-view coverage for base height estimation. Nevertheless, the practical impact is marginal: the error increase from 3-DoF to 6-DoF alignment remains at the centimeter level with stable rotation error. Furthermore, RANSAC-based filtering effectively rejects base height outliers, as supported by results in Table. 4 and Appendix C.



(a) Examples of Localization Failure and Success across Different Areas



(b) Sensitivity to Reconstructed Tree Count in `Karawatha`



(c) Inaccurate Base Height from FoV and Occlusion

FIGURE 23. Limitations of TreeLoc++. (a) Sparse and distant tree observations in open areas provide insufficient geometric constraints, leading to failure. In contrast, dense forests enable successful localization through abundant correspondences. (b) Inter-session place recognition performance on `Karawatha` versus reconstructed tree count, where D1-D10 denote deciles from the sparsest to the densest scans. Performance generally improves with increasing tree count, showing that sparse scans provide weaker geometric constraints and lower overlap. (c) Limited sensor FoV and occlusions prevent direct ground measurements, requiring an approximation of the base height that may introduce errors relative to the true base height.

VII. Conclusion

This paper presented TreeLoc++, a global localization framework for forest environments based on DFIs that eliminates the need to store raw point clouds. TreeLoc++ builds local and global inventories by reconstructing tree instances from aggregated payloads. For localization, it performs stem-axis-based alignment and 2D projection, and retrieves candidates via a coarse-to-fine matching strategy, starting with TDH and PDH, and 2D triangle descriptor for efficient retrieval. It then

rejects outliers through DBH filtering and yaw-consistent inlier voting, and refines the 6-DoF pose through geometric verification with constrained optimization of roll, pitch, and height. We evaluated TreeLoc++ on 27 sequences from three datasets, including multi-session data collected two years apart. The results demonstrate robust place recognition and centimeter-level localization accuracy, while maintaining a compact map representation of just 250 KB across 15 sessions covering $7.98\,\mathrm{km}$ trajectories. TreeLoc++ outperformed both hand-crafted and learning-based baselines in accuracy and achieved the lowest runtime. Ablation studies further confirmed that each proposed component contributes to the overall performance and robustness. Future work will focus on inventory construction. Although TreeLoc++ is accurate and efficient, the current pipeline relies on individual tree segmentation and forest inventory generation using RealtimeTrees, which requires aggregating point cloud payloads. With improvements to DFI construction in the future, we would envisage TreeLoc++ achieving further gains in localization performance.

REFERENCES

[1] H. Yin, X. Xu, S. Lu, X. Chen, R. Xiong, S. Shen, C. Stachniss, and Y. Wang, "A Survey on Global LiDAR Localization: Challenges, Advances and Open Problems," *Intl. J. of Comput. Vision*, vol. 132, no. 8, pp. 3139–3171, 2024.

[2] M. Miettinen, M. Ohman, A. Visala, and P. Forsman, "Simultaneous Localization and Mapping for Forest Harvesters," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2007, pp. 517–522.

[3] M. Mattamala, J. Frey, P. Libera, N. Chebrolu, G. Martius, C. Cadena, M. Hutter, and M. Fallon, "Wild visual navigation: fast traversability learning via pre-trained models and online self-supervision," *Autonomous Robots*, vol. 49, no. 3, p. 19, 2025.

[4] M. Mattamala, N. Chebrolu, J. Frey, L. Freißmuth, H. Oh, B. Casseau, M. Hutter, and M. Fallon, "Building Forest Inventories With Autonomous Legged Robots—System, Lessons, and Challenges Ahead," *IEEE Trans. of Field Robot.*, vol. 2, pp. 418–436, 2025.

[5] X. Liang, V. Kankare, J. Hyyppä, Y. Wang, A. Kukko, H. Haggrén, X. Yu, H. Kaartinen, A. Jaakkola, F. Guan *et al.*, "Terrestrial laser scanning in forest inventories," *ISPRS J. Photogramm. Remote Sensing*, vol. 115, pp. 63–77, 2016.

[6] M. Pierzchała, P. Giguère, and R. Astrup, "Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM," *Comput. Elec. Agric.*, vol. 145, pp. 217–225, 2018.

[7] S. W. Chen, G. V. Nardari, E. S. Lee, C. Qu, X. Liu, R. A. F. Romero, and V. Kumar, "SLOAM: Semantic Lidar Odometry and Mapping for Forest Inventory," *IEEE Robot. and Automat. Lett.*, vol. 5, no. 2, pp. 612–619, 2020.

[8] X. Liu, G. V. Nardari, F. Cladera, Y. Tao, A. Zhou, T. Donnelly, C. Qu, S. W. Chen, R. A. Romero, C. J. Taylor *et al.*, "Large-Scale Autonomous Flight With Real-Time Semantic SLAM Under Dense Forest Canopy," *IEEE Robot. and Automat. Lett.*, vol. 7, no. 2, pp. 5512–5519, 2022.

[9] R. Valbuena, F. Mauro, R. Rodriguez-Solano, and J. Manzanera, "Accuracy and precision of GPS receivers under forest canopies in a mountainous environment," *Span. J. Agric. Res.*, vol. 8, no. 4, pp. 1047–1057, 2010.

[10] I.-S. Lee and L. Ge, "The performance of RTK-GPS for surveying under challenging environmental conditions," *Earth Planets Space*, vol. 58, no. 5, pp. 515–522, 2006.

[11] M. Dassot, T. Constant, and M. Fournier, "The use of terrestrial LiDAR technology in forest science: application fields, benefits and challenges," *Ann. For. Sci.*, vol. 68, no. 5, pp. 959–974, 2011.

[12] H.-M. Cho, J.-H. Oh, J.-W. Park, Y.-S. Choi, J.-S. Lee, and S.-K. Han, "Application of Real-time Positioning Systems to a Forest Stand for Precision Forest Management," *Sens. Mater.*, vol. 34, no. 12, pp. 4651–4668, 2022.

[13] M. A. Wulder, C. W. Bater, N. C. Coops, T. Hilker, and J. C. White, "The role of LiDAR in sustainable forest management," *The forestry chronicle*, vol. 84, no. 6, pp. 807–826, 2008.

[14] G. Kim and A. Kim, "Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018, pp. 4802–4809.

[15] G. Kim, S. Choi, and A. Kim, "Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments," *IEEE Trans. Robot.*, vol. 38, no. 3, pp. 1856–1874, 2022.

[16] H. Kim, J. Choi, T. Sim, G. Kim, and Y. Cho, "Narrowing Your FOV With SOLiD: Spatially Organized and Lightweight Global Descriptor for FOV-Constrained LiDAR Place Recognition," *IEEE Robot. and Automat. Lett.*, vol. 9, no. 11, pp. 9645–9652, 2024.

[17] C. Yuan, J. Lin, Z. Zou, X. Hong, and F. Zhang, "STD: Stable Triangle Descriptor for 3D place recognition," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2023, pp. 1897–1903.

[18] C. Yuan, J. Lin, Z. Liu, H. Wei, X. Hong, and F. Zhang, "BTC: A Binary and Triangle Combined Descriptor for 3-D Place Recognition," *IEEE Trans. Robot.*, vol. 40, pp. 1580–1599, 2024.

[19] S. Gupta, T. Guadagnino, B. Mersch, I. Vizzo, and C. Stachniss, "Effectively Detecting Loop Closures using Point Cloud Density Maps," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2024, pp. 10 260–10 266.

[20] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 314–333, 2021.

[21] H. Lim, S. Yeon, S. Ryu, Y. Lee, Y. Kim, J. Yun, E. Jung, D. Lee, and H. Myung, "A Single Correspondence Is Enough: Robust Global Registration to Avoid Degeneracy in Urban Environments," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2022, pp. 8010–8017.

[22] H. Lim, D. Kim, G. Shin, J. Shi, I. Vizzo, H. Myung, J. Park, and L. Carlone, "KISS-Matcher: Fast and Robust Point Cloud Registration Revisited," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2025, pp. 11 104–11 111.

[23] H. Oh, N. Chebrolu, M. Mattamala, L. Freißmuth, and M. Fallon, "Evaluation and Deployment of LiDAR-based Place Recognition in Dense Forests," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2024, pp. 12 824–12 831.

[24] L. Li, X. Kong, X. Zhao, T. Huang, W. Li, F. Wen, H. Zhang, and Y. Liu, "SSC: Semantic Scan Context for Large-Scale Place Recognition," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2021, pp. 2092–2099.

[25] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based place recognition in 3D point clouds," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2017, pp. 5266–5272.

[26] K. Vidanapathirana, J. Knights, S. Hausler, M. Cox, M. Ramezani, J. Jooste, E. Griffiths, S. Mohamed, S. Sridharan, C. Fookes *et al.*, "WildScenes: A benchmark for 2D and 3D semantic segmentation in large-scale natural environments," *Intl. J. of Robot. Research*, vol. 44, no. 4, pp. 532–549, 2025.

[27] G. Tinchev, S. Nobili, and M. Fallon, "Seeing the Wood for the Trees: Reliable Localization in Urban and Natural Environments," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018, pp. 8239–8246.

[28] 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," *IEEE Robot. and Automat. Lett.*, vol. 4, no. 2, pp. 1327–1334, 2019.

[29] M. Jung, N. Chebrolu, L. C. de Lima, H. Oh, M. Fallon, and A. Kim, "TreeLoc: 6-DoF LiDAR Global Localization in Forests via Inter-Tree Geometric Matching," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2026.

[30] C. J. Gleason and J. Im, "Forest biomass estimation from airborne LiDAR data using machine learning approaches," *Remote Sens. Environ.*, vol. 125, pp. 80–91, 2012.

[31] W. Li, Q. Guo, M. K. Jakubowski, and M. Kelly, "A New Method for Segmenting Individual Trees from the Lidar Point Cloud," *Photogramm. Eng. Remote Sens.*, vol. 78, no. 1, pp. 75–84, 2012.

[32] L. Wallace, A. Lucieer, and C. S. Watson, "Evaluating Tree Detection and Segmentation Routines on Very High Resolution UAV LiDAR Data," *IEEE Trans. Geosci. and Remote Sensing*, vol. 52, no. 12, pp. 7619–7628, 2014.

[33] K. Itakura, S. Miyatani, and F. Hosoi, "Estimating Tree Structural Parameters via Automatic Tree Segmentation From LiDAR Point Cloud Data," *IEEE J. of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 555–564, 2022.

[34] M. V. Malladi, T. Guadagnino, L. Lobefaro, M. Mattamala, H. Griess, J. Schweier, N. Chebrolu, M. Fallon, J. Behley, and C. Stachniss, "Tree Instance Segmentation and Traits Estimation for Forestry Environments Exploiting LiDAR Data Collected by Mobile Robots," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2024, pp. 17 933–17 940.

[35] L. Freißmuth, M. Mattamala, N. Chebrolu, S. Schaefer, S. Leutenegger, and M. Fallon, "Online Tree Reconstruction and Forest Inventory on a Mobile Robotic System," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2024, pp. 11 765–11 772.

[36] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Sys.*, vol. 30, 2017.

[37] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, 2020, pp. 11 105–11 114.

[38] M. V. Malladi, N. Chebrolu, I. Scacchetti, L. Lobefaro, T. Guadagnino, B. Casseau, H. Oh, L. Freißmuth, M. Karppinen, J. Schweier *et al.*, "DigiForests: A Longitudinal LiDAR Dataset for Forestry Robotics," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2025, pp. 1459–1466.

[39] J. Knights, K. Vidanapathirana, M. Ramezani, S. Sridharan, C. Fookes, and P. Moghadam, "Wild-Places: A Large-Scale Dataset for Lidar Place Recognition in Unstructured Natural Environments," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2023, pp. 11 322–11 328.

[40] B. N. Bailey and M. H. Ochoa, "Semi-direct tree reconstruction using terrestrial LiDAR point cloud data," *Remote Sens. Environ.*, vol. 208, pp. 133–144, 2018.

[41] M. Hussein, K. Iagnemma, and M. Renner, "Global Localization of Autonomous Robots in Forest Environments," *Photogramm. Eng. Remote Sens.*, vol. 81, no. 11, pp. 839–846, 2015.

[42] X. Xu, S. C. Lu, J. Wu, H. Lu, Q. Zhu, Y. Liao, R. Xiong, and Y. Wang, "RING++: Roto-Translation Invariant Gram for Global Localization on a Sparse Scan Map," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4616–4635, 2023.

[43] L. He, X. Wang, and H. Zhang, "M2DP: A novel 3D point cloud descriptor and its application in loop closure detection," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2016, pp. 231–237.

[44] S. C. Lu, X. Xu, H. Yin, Z. Chen, R. Xiong, and Y. Wang, "One RING to Rule Them All: Radon Sinogram for Place Recognition, Orientation and Translation Estimation," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2022, pp. 2778–2785.

[45] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2009, pp. 3212–3217.

[46] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *Comput. Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

[47] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3D lidar datasets," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2013, pp. 2677–2684.

[48] S. Gupta, T. Guadagnino, B. Mersch, N. Trekel, M. V. Malladi, and C. Stachniss, "Efficiently Closing Loops in LiDAR-Based SLAM Using Point Cloud Density Maps," *arXiv preprint arXiv:2501.07399*, 2025.

[49] Y. Zhang, Y. Tian, G. Yang, Z. Li, M. Luo, E. Li, and F. Jing, "Intensity Triangle Descriptor Constructed From High-Resolution Spinning LiDAR Intensity Image for Loop Closure Detection," *IEEE Robot. and Automat. Lett.*, vol. 9, no. 10, pp. 8937–8944, 2024.

[50] Y. Park, G. Pak, and E. Kim, "RE-TRIP: Reflectivity Instance Augmented Triangle Descriptor for 3D Place Recognition," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2025, pp. 2226–2232.

[51] T.-X. Xu, Y.-C. Guo, Z. Li, G. Yu, Y.-K. Lai, and S.-H. Zhang, "TransLoc3D: point cloud based large-scale place recognition using adaptive receptive fields," *Commun. Inf. Syst.*, vol. 23, no. 1, pp. 57–83, 2023.

[52] K. Vidanapathirana, M. Ramezani, P. Moghadam, S. C. Sridharan, and C. Fookes, "LoGG3D-Net: Locally Guided Global Descriptor Learning for 3D Place Recognition," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2022, pp. 2215–2221.

[53] J. Komorowski, "Improving Point Cloud Based Place Recognition with Ranking-based Loss and Large Batch Training," in *Proc. Intl. Conf. Pattern Recog.*, 2022, pp. 3699–3705.

[54] M. Jung, S. Jung, H. Gil, and A. Kim, "HeLiOS: Heterogeneous LiDAR Place Recognition via Overlap-based Learning and Local Spherical Transformer," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2025, pp. 2204–2211.

[55] E. Griffiths, M. Haghighat, S. Denman, C. Fookes, and M. Ramezani, "HOTFormerLoc: Hierarchical Octree Transformer for Versatile Lidar Place Recognition Across Ground and Aerial Views," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, 2025, pp. 6648–6658.

[56] Y. Shen, T. Tuna, M. Hutter, C. Cadena, and N. Zheng, "ForestLPR: LiDAR Place Recognition in Forests Attentioning Multiple BEV Density Images," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, 2025, pp. 6659–6669.

[57] L. Luo, S.-Y. Cao, X. Li, J. Xu, R. Ai, Z. Yu, and X. Chen, "BEV-Place++: Fast, Robust, and Lightweight LiDAR Global Localization for Autonomous Ground Vehicles," *IEEE Trans. Robot.*, vol. 41, pp. 4479–4498, 2025.

[58] L. Luo, S.-Y. Cao, B. Han, H.-L. Shen, and J. Li, "BVMatch: Lidar-Based Place Recognition Using Bird's-Eye View Images," *IEEE Robot. and Automat. Lett.*, vol. 6, no. 3, pp. 6076–6083, 2021.

[59] M. Jung, L. F. T. Fu, M. Fallon, and A. Kim, "ImLPR: Image-based LiDAR Place Recognition using Vision Foundation Models," in *Proc. 9th Annual Conf. on Robot Learning*, vol. 305, 2025, pp. 3318–3340.

[60] K. Vidanapathirana, P. Moghadam, S. C. Sridharan, and C. Fookes, "Spectral Geometric Verification: Re-Ranking Point Cloud Retrieval for Metric Localization," *IEEE Robot. and Automat. Lett.*, vol. 8, no. 5, pp. 2494–2501, 2023.

[61] D. Cattaneo, M. Vaghi, and A. Valada, "LCDNet: Deep Loop Closure Detection and Point Cloud Registration for LiDAR SLAM," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2074–2093, 2022.

[62] J. Komorowski, M. Wysoczanska, and T. Trzcinski, "EgoNN: Egocentric Neural Network for Point Cloud Based 6DoF Relocalization at the City Scale," *IEEE Robot. and Automat. Lett.*, vol. 7, no. 2, pp. 722–729, 2022.

[63] R. Dube, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: Segment-based mapping and localization using data-driven descriptors," *Intl. J. of Robot. Research*, vol. 39, no. 2-3, pp. 339–355, 2020.

[64] G. Pramatarov, D. De Martini, M. Gadd, and P. Newman, "BoxGraph: Semantic Place Recognition and Pose Estimation from 3D LiDAR," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2022, pp. 7004–7011.

[65] W. Ma, H. Yin, P. J. Y. Wong, D. Wang, Y. Sun, and Z. Su, "TripletLoc: One-Shot Global Localization Using Semantic Triplet in Urban Environments," *IEEE Robot. and Automat. Lett.*, vol. 10, no. 2, pp. 1569–1576, 2025.

[66] J.-F. Tremblay and M. Béland, "Towards operational marker-free registration of terrestrial lidar data in forests," *ISPRS J. Photogramm. Remote Sensing*, vol. 146, pp. 430–435, 2018.

[67] Y. Liang, J. Liu, J. Lei, J. Muhojoki, A. Kukko, H. Kaartinen, J. Hyyppä, D. Xu, and W. Zhang, "Ground-to-air collaborative LiDAR global localization in forest environments," *Expert Syst. Appl.*, vol. 309, p. 131267, 2026.

[68] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast Direct LiDAR-Inertial Odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, 2022.

[69] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct LiDAR-Inertial Odometry: Lightweight LIO with Continuous-Time Motion Correction," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2023, pp. 3983–3989.

[70] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, S. Ilic, D. Hu, and K. Xu, "GeoTransformer: Fast and Robust Point Cloud Registration With Geometric Transformer," *IEEE Trans. Pattern Analysis and Machine Intell.*, vol. 45, no. 8, pp. 9806–9821, 2023.

[71] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction," Georgia Institute of Technology, Tech. Rep. GT-RIM-CP&R-2012-002, 2012.

[72] M. Grupp, "evo: Python package for the evaluation of odometry and SLAM," https://github.com/MichaelGrupp/evo, 2017.

[73] Y. Pan, T. Sun, L. Zhu, L. Nunes, I. Armeni, J. Behley, and C. Stachniss, "Register Any Point: Scaling 3D Point Cloud Registration by Flow Matching," *arXiv preprint arXiv:2512.01850*, 2025.

[74] D. Wisth, M. Camurri, and M. Fallon, "VILENS: Visual, Inertial, Lidar, and Leg Odometry for All-Terrain Legged Robots," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 309–326, 2023.

[75] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Intl. J. of Robot. Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[76] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *Intl. J. of Robot. Research*, vol. 36, no. 1, pp. 3–15, 2017.

[77] S. Ao, Q. Hu, H. Wang, K. Xu, and Y. Guo, "BUFFER: Balancing Accuracy, Efficiency, and Generalizability in Point Cloud Registration," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, 2023, pp. 1255–1264.

[78] M. Seo, H. Lim, K. Lee, L. Carlone, and J. Park, "BUFFER-X: Towards Zero-Shot Point Cloud Registration in Diverse Scenes," in *Proc. IEEE Intl. Conf. on Comput. Vision*, 2025, pp. 3851–3862.

Minwoo Jung (Student Member, IEEE) received the B.S. degree in civil and environmental engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2021, and the M.S. degree in mechanical engineering from Seoul National University (SNU), Seoul, South Korea, in 2023. He is currently pursuing the Ph.D. degree in mechanical engineering at SNU, Seoul, South Korea, under the supervision of Prof. Ayoung Kim. He was a visiting research student at the Oxford Robotics Institute (ORI), University of Oxford, Oxford, U.K. His research interests include LiDAR-based robot perception and robust multi-sensor localization.

Dongjae Lee (Student Member, IEEE) received the B.S. degree in mechanical engineering from Seoul National University (SNU), Seoul, South Korea, in 2023. He is currently pursuing the Ph.D. degree in mechanical engineering at SNU, Seoul, South Korea, under the supervision of Prof. Ayoung Kim. He was a visiting research student at the Oxford Robotics Institute (ORI), University of Oxford, Oxford, U.K. His research interests include dense 3-D reconstruction and embodied navigation.

Nived Chebrolu (Member, IEEE) received the Ph.D. degree in agricultural robotics from the University of Bonn, Bonn, Germany, in 2021. He subsequently held a postdoctoral research position at the Oxford Robotics Institute (ORI), University of Oxford, Oxford, U.K. He is currently an Assistant Professor with the Department of Computer Science and Engineering at the Indian Institute of Technology Bombay, Mumbai, India. His research focuses on developing autonomous and intelligent robotic systems for a broad range of real-world domains, including autonomous vehicles, service robots, precision agriculture, forestry, industrial maintenance, and disaster response.

Haedam Oh (Student Member, IEEE) received the M.Eng. degree in engineering science from the University of Oxford, Oxford, U.K., in 2024. He is currently pursuing the D.Phil. degree in engineering science in the Dynamic Robot Systems Group at the Oxford Robotics Institute (ORI), University of Oxford, Oxford, U.K., under the supervision of Prof. Maurice Fallon. His research interests include localization and mapping with LiDAR and vision.

Maurice Fallon (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from University College Dublin, Dublin, Ireland, in 2004, and the Ph.D. degree in acoustic source tracking from the University of Cambridge, Cambridge, U.K., in 2008. From 2008 to 2015, he was a Postdoc and a Research Scientist at MIT, Cambridge, MA, USA, working on marine robotics and the DARPA Robotics Challenge. He is currently a Professor at the University of Oxford, Oxford, U.K., and leads the Dynamic Robot Systems Group at the Oxford Robotics Institute, University of Oxford, Oxford, U.K. His research interests include probabilistic methods for localization, mapping, multisensor fusion, and robot navigation.

Ayoung Kim (Senior Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Seoul National University (SNU), Seoul, South Korea, in 2005 and 2007, respectively, and the M.S. degree in electrical engineering and the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2011 and 2012, respectively. She was an Associate Professor at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, from 2014 to 2021. She is currently a Professor at SNU, Seoul, South Korea. Her research interests include spatial representation, SLAM, and navigation.

Appendix

A. Ground Truth Generation for Evo25

To generate consistent ground truth trajectories for the nine sequences in `Evo25:00-08`, we first estimated local trajectories using VILENS [74] with LiDAR and IMU data. The sessions were then coarsely aligned based on common locations. To further refine this alignment, inter-session constraints were added at the point cloud level via robust pairwise frame matching using TEASER++ [20]. All TEASER++ matches were manually verified to ensure high alignment quality. Candidate frame pairs were selected via nearest neighbor matching across all possible session combinations. Finally, all local trajectories and inter-session constraints were jointly optimized using GTSAM to produce globally consistent multi-session trajectories.

To quantify the geometric reliability of this pseudo-ground truth, we evaluated cross-session registration consistency using point-to-point nearest-neighbor distances, following the same nearest-neighbor error analysis used in Fig. 13 and a point-cloud distance-based validation similar to [39]. Using representative mission point clouds from `Evo25:00`, `Evo25:02`, `Evo25:03`, and `Evo25:05`, which were also used in §D, we transformed them into the optimized global frame and measured nearest-neighbor distances between mission pairs. As shown in Fig. 24a-d, regions with substantial overlap mostly exhibit low error, whereas boundary regions with limited overlap show larger distances because sparse LiDAR observations provide fewer reliable cross-session correspondences.

Fig. 24e summarizes these trends using violin plots of the distance error distributions for representative mission pairs. Although the median error becomes slightly larger as overlap decreases, the medians remain around or below 0.2 m, and the highest-density portions of all distributions lie below 0.1 m. These results indicate that the optimized trajectories are sufficiently accurate to serve as pseudo-ground truth for evaluating TreeLoc++. The remaining uncertainty is mainly confined to low overlap boundary regions, while the evaluated trajectory segments are mostly drawn from revisited areas with substantial overlap. Therefore, its impact on the main evaluation is expected to be small.

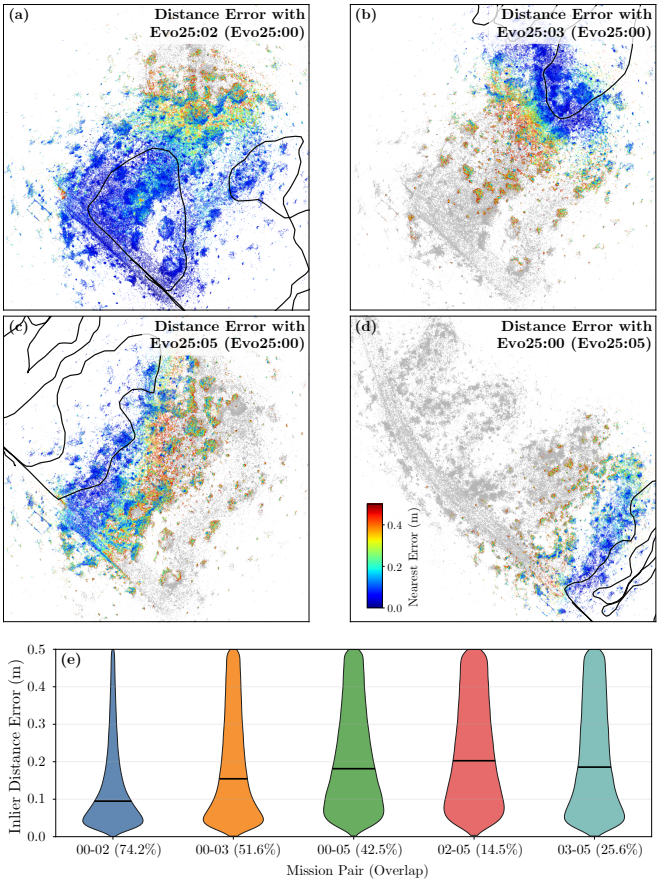


FIGURE 24. Registration consistency across `Evo25` missions. (a-c) Top-view error maps of `Evo25:00` with respect to `Evo25:02`, `Evo25:03`, and `Evo25:05`. (d) Top-view error map of `Evo25:05` with respect to `Evo25:00`. Points whose nearest neighbor lies within 0.5 m are treated as inliers and color-coded by error (blue: low, red: high), while invalid correspondences are shown in gray. Black lines indicate the optimized trajectory of the reference mission. Overlapping regions mostly show low error, whereas boundary regions with limited overlap show larger distances. (e) Violin plots of inlier error distributions for representative mission pairs, with overlap ratios in parentheses and the median shown by a black bar. The densest portion of the distribution remains below 0.1 m.

B. Learning-based Methods with Training Datasets

We analyzed the impact of training data on learning-based baselines. Each learning-based method was evaluated under three training regimes: (1) urban, (2) in-domain forest, and (3) out-of-domain forest.

Urban vs. Forest: We analyzed the impact of training data on learning-based baselines by using urban datasets (KITTI [75] and Oxford RobotCar [76]) for training, and evaluating the models on the Oxford Forest Place Recognition Dataset. BEVPlace++ and LoGG3D-Net were trained on KITTI, while the other models used Oxford RobotCar. All models employed pretrained checkpoints, except for TransLoc3D. As shown in Table. 12, models trained on urban datasets exhibited limited generalization to forest environments. This trend is also reflected in Fig. 25a, where urban-trained models underperformed in terms of PR curves, with performance degrading across all sequences. In contrast, models trained on Wild-Places achieved moderate performance, particularly in structured scenes such as `Stein am Rhein`. While training on Wild-Places improved overall performance, their performance still remained limited across the evaluation sequences. This suggests two key insights: first, that models trained on urban datasets struggle to generalize to forest environments, even though such datasets are large-scale and readily available, which limits their practical applicability; and second, that a domain gap persists even within forest environments, degrading performance across evaluation sequences. We further investigate this forest domain gap in the Wild-Places experiments.

TABLE 12. Comparative analysis of intra-session place recognition in Oxford Forest using urban-trained versus forest-trained models

Dimension	Method	Stein am Rhein				Wytham			
		R@1	MR	MF1	AUC	R@1	MR	MF1	AUC
Learning-based (Urban)	TransLoc3D	0.187	0.000	0.316	0.198	0.010	0.000	0.020	0.011
	LoGG3D-Net	0.175	0.002	0.298	0.197	0.041	0.000	0.078	0.044
	MinkLoc3Dv2	0.426	0.037	0.398	0.233	0.041	0.000	0.078	0.025
	BEVPlace++	0.727	0.375	0.865	0.941	0.235	0.000	0.383	0.220
	HOTFormerLoc	0.330	0.136	0.598	0.703	0.327	0.063	0.535	0.414
Learning-based (Forest)	TransLoc3D	0.395	0.004	0.566	0.507	0.082	0.000	0.151	0.091
	LoGG3D-Net	0.633	0.257	0.776	0.843	0.010	0.012	0.020	0.012
	MinkLoc3Dv2	0.538	0.086	0.577	0.429	0.031	0.010	0.020	0.006
	BEVPlace++	*0.776*	*0.263*	*0.900*	*0.929*	*0.592*	*0.172*	*0.795*	*0.806*
	HOTFormerLoc	0.570	0.107	0.768	0.811	0.541	0.113	0.718	0.756
Ours	**TreeLoc++**	**0.941**	**0.937**	**0.986**	**0.998**	**0.796**	**0.962**	**0.994**	**0.993**

· Recall@1 (R@1), Maximum Recall at 100% Precision (MR), Maximum F1 score (MF1), and Area Under the Precision–Recall Curve (AUC).

TABLE 13. Comparative analysis of intra-session place recognition in Wild-Places using in-domain-trained versus out-of-domain forest-trained models

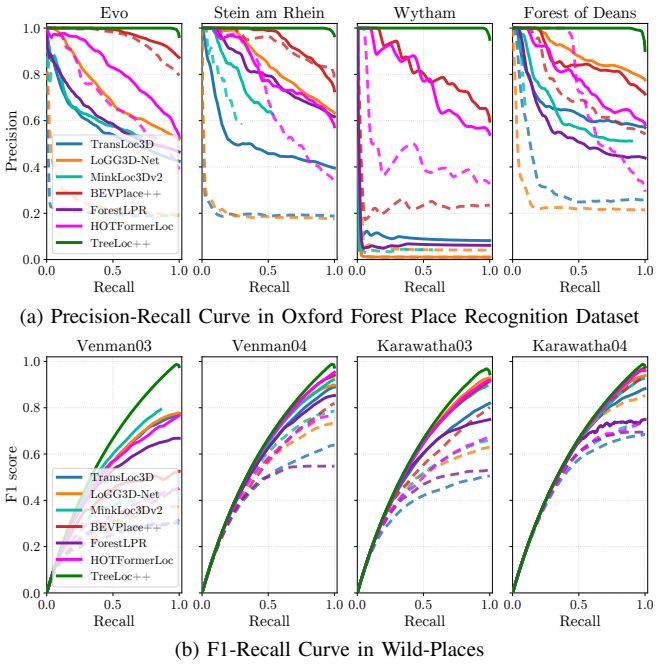
Dimension	Method	Venman04				Karawatha04			
		R@1	MR	MF1	AUC	R@1	MR	MF1	AUC
Learning-based (Oxford Forest)	TransLoc3D	0.469	0.000	0.638	0.496	0.520	0.000	0.684	0.618
	LoGG3D-Net	0.584	0.042	0.737	0.745	0.745	0.332	0.854	0.921
	MinkLoc3Dv2	0.647	0.029	0.785	0.774	0.593	0.134	0.744	0.776
	BEVPlace++	0.694	0.015	0.819	0.666	0.860	0.088	0.929	0.951
	ForestLPR	0.188	0.000	0.317	0.181	0.377	0.037	0.548	0.554
	HOTFormerLoc	0.613	0.013	0.765	0.717	0.587	0.031	0.749	0.742
Learning-based (Wild-Places)	TransLoc3D	0.801	0.003	0.890	0.868	0.790	0.003	0.883	0.928
	LoGG3D-Net	0.814	0.113	0.898	0.934	0.885	0.012	0.939	0.945
	MinkLoc3Dv2	0.853	0.037	0.921	0.913	0.867	0.360	0.927	0.947
	BEVPlace++	0.885	*0.234*	0.940	*0.964*	**0.945**	*0.617*	*0.975*	*0.987*
	ForestLPR	0.743	0.045	0.853	0.839	0.599	0.271	0.749	0.865
	HOTFormerLoc	*0.911*	0.037	*0.953*	0.955	*0.924*	0.359	0.966	0.978
Ours	**TreeLoc++**	**0.924**	**0.980**	**0.993**	**0.998**	0.903	**0.940**	**0.995**	**0.998**

· Recall@1 (R@1), Maximum Recall at 100% Precision (MR), Maximum F1 score (MF1), and Area Under the Precision–Recall Curve (AUC).

In-domain vs. out-of-domain forest: We examined the effect of training data from different forest domains by comparing models trained on Wild-Places and the Oxford Forest Place Recognition Dataset, evaluated on Wild-Places sequences. As shown in Table. 13, methods trained on Oxford Forest showed a clear drop in performance, comparable to that of urban-trained models, despite both datasets being forest environments. Similarly, in Fig. 25b, methods trained on Oxford Forest consistently yielded lower F1 scores, indicating decreased retrieval quality. These results suggest that, even within the same broader domain, forest environments can differ substantially, making it difficult for learning-based methods to generalize. This highlights the potential of well-designed hand-crafted methods to provide more robust performance across diverse forest conditions. In particular, TreeLoc++, which achieves learning-comparable performance without training, emerges as a strong alternative to learning-based approaches in forest environments.

C. Pose Estimation Benchmark: Registration Methods

Evaluation protocol: We evaluated TreeLoc++ as a registration method against learning-based baselines (GeoTransformer [70], BUFFER [77], BUFFER-X [78]) and hand-crafted baselines (Quatro [21], TEASER++ [20], KISS-Matcher [22]). To evaluate performance across spatial separations, we grouped query-candidate pairs into translation ranges of $1-4$ m, $4-7$ m, and $7-10$ m. For each range, we selected two candidates per query from TreeLoc++ retrieval outputs (one above and one below the midpoint) based on overlap ratio, without guaranteeing true positives. To avoid bias toward nearby candidates, we disabled the translation penalty $p(\|\mathbf{t}\|)$ during candidate selection. Classical baselines were evaluated with two voxel resolutions, small (S) and large (L), as well as a tree-only setting (T) using stem centers and base heights. Voxel sizes were $0.5/1.0$ m for `Evo:Single` and $1.0/2.0$ m for `Venman03`. We also tested whether TreeLoc++ tree-level features (stem centers and base heights) improve existing registration pipelines, and compared runtime to assess efficiency. Learning-based baselines were evaluated on the $1-4$ m range using official configurations and pretrained checkpoints.



(a) Precision-Recall Curve in Oxford Forest Place Recognition Dataset

(b) F1-Recall Curve in Wild-Places

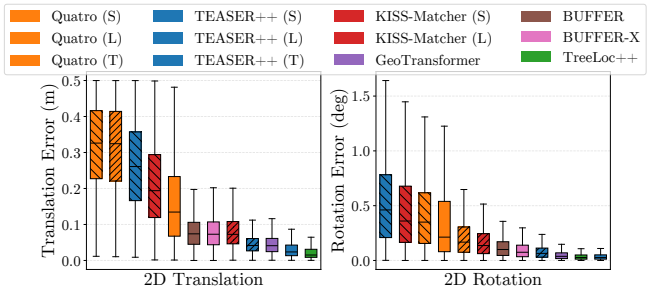
FIGURE 25. Precision–Recall and F1–Recall curves showing the impact of training domains. (Top) Urban-trained models (dashed) perform worse on Oxford Forest, compared to models trained on forest datasets (solid), indicating limited generalization to natural environments. (Bottom) Even across forest datasets, models trained on Oxford Forest (dashed) perform worse on Wild-Places, compared to those trained directly on Wild-Places (solid), highlighting challenges in cross-forest generalization.

TreeLoc++: As summarized in Table. 14, TreeLoc++ achieved highly competitive 2D pose accuracy across all distance ranges. In the $1-4$ m range, it outperformed both classical and learning-based baselines in translation and rotation error, as visualized in Fig. 26a. In 3D, accuracy degraded with distance for all methods; nevertheless, TreeLoc++ maintained centimeter-level errors comparable to dense point cloud-based approaches. Fig. 26b shows in-

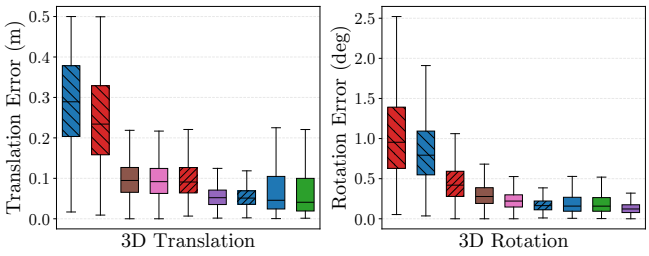
TABLE 14. Pose estimation performance comparison with registration methods.

	Dimension / Method	Evo:Single 1–4 m ($N = 4670$) SR	ATE	ARE	Evo:Single 4–7 m ($N = 4668$) SR	ATE	ARE	Evo:Single 7–10 m ($N = 4669$) SR	ATE	ARE	Venman03 1–4 m ($N = 4189$) SR	ATE	ARE	Venman03 4–7 m ($N = 4481$) SR	ATE	ARE	Venman03 7–10 m ($N = 4097$) SR	ATE	ARE
2D	Quatro (S)	0.948	0.108	0.269	0.856	0.161	0.446	0.794	0.198	0.623	0.547	0.311	0.239	0.551	0.309	0.259	0.458	0.291	0.359
	Quatro (L)	0.818	0.208	0.727	0.479	0.265	1.077	0.242	0.290	1.250	0.381	0.315	0.448	0.322	0.316	0.625	0.261	0.312	0.809
	Quatro (T)	0.881	0.164	0.199	0.745	0.199	0.314	0.698	0.212	*0.365*	*0.848*	0.163	0.408	0.836	0.170	0.437	0.706	0.183	0.521
	TEASER++ (S)	**1.000**	0.059	*0.197*	*0.997*	0.094	0.318	0.979	0.133	0.463	**1.000**	0.046	0.081	**1.000**	0.061	0.110	**1.000**	0.083	0.157
	TEASER++ (L)	0.848	0.223	0.844	0.506	0.278	1.263	0.247	0.296	1.306	0.815	0.262	0.562	0.620	0.290	0.694	0.433	0.299	0.862
	TEASER++ (T)	**1.000**	*0.038*	**0.039**	**1.000**	*0.047*	*0.056*	**0.999**	*0.052*	**0.068**	**1.000**	*0.036*	**0.039**	*0.999*	*0.055*	**0.064**	*0.991*	*0.074*	**0.090**
	KISS-Matcher (S)	*0.997*	0.079	0.306	0.970	0.128	0.531	0.861	0.172	0.731	**1.000**	0.082	0.179	*0.999*	0.114	0.268	0.974	0.158	0.391
	KISS-Matcher (L)	0.794	0.197	0.815	0.400	0.257	1.135	0.168	0.272	1.208	0.839	0.214	0.516	0.628	0.261	0.724	0.357	0.282	0.921
	TreeLoc++	**1.000**	**0.017**	**0.039**	**1.000**	**0.025**	**0.054**	*0.998*	**0.031**	**0.068**	**1.000**	**0.027**	*0.042*	0.991	**0.046**	*0.071*	0.967	**0.067**	*0.105*
3D	TEASER++ (S)	**1.000**	*0.065*	**0.197**	**0.996**	0.103	0.509	**0.975**	0.146	0.740	**1.000**	**0.055**	**0.176**	**1.000**	**0.073**	**0.241**	**1.000**	**0.098**	**0.338**
	TEASER++ (L)	0.833	0.236	1.277	0.468	0.298	1.936	0.199	0.322	2.297	0.778	0.290	0.871	0.555	0.324	1.126	0.357	0.340	1.446
	TEASER++ (T)	0.990	0.074	0.263	*0.961*	*0.100*	**0.417**	*0.934*	*0.116*	**0.526**	*0.976*	0.084	*0.208*	0.920	0.112	*0.352*	0.845	*0.131*	*0.521*
	KISS-Matcher (S)	*0.995*	0.090	0.718	0.958	0.148	1.092	0.815	0.198	1.405	**1.000**	0.100	0.467	*0.997*	0.137	0.661	*0.962*	0.188	0.879
	KISS-Matcher (L)	0.749	0.225	1.721	0.315	0.290	2.237	0.114	0.310	2.389	0.788	0.246	1.082	0.530	0.295	1.413	0.267	0.320	1.706
	TreeLoc++	0.990	**0.061**	*0.261*	0.960	**0.090**	*0.418*	0.930	**0.112**	*0.531*	*0.976*	*0.080*	0.211	0.910	*0.110*	0.360	0.824	0.143	0.544

· S and L use small and large voxel sizes with FPFH-based correspondences, while T uses tree centers and base heights with initial all-to-all correspondences.
· Success Rate (SR), Average Translation Error (ATE) [m], and Average Rotation Error (ARE) [°].



(a) 2D Pose Estimation Error in Venman03



(b) 3D Pose Estimation Error in Venman03

FIGURE 26. Box plots of pose errors on the Venman03 for the 1–4 m interval. Boxes indicate the IQR, with the median marked inside. TreeLoc++ consistently exhibits low median error and tight error distribution.

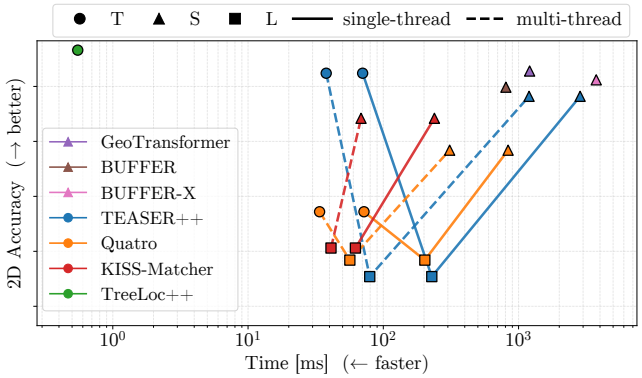


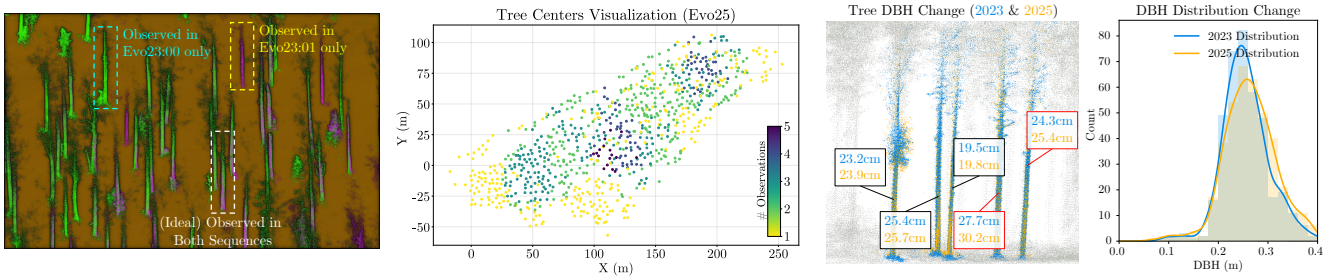
FIGURE 27. 2D pose estimation error with time consumption on the Evo:Single for the 1–4 m translation interval

creased interquartile range (IQR), but the medians remained low, supporting the effectiveness of TreeLoc++ for metric localization.

Tree-based vs. point-based registration: To evaluate the value of tree-level features, we compared TreeLoc++ with TEASER++ (T). They achieved comparable accuracy on sparse inputs, indicating that our matching strategy offers robustness similar to that of maximum-clique optimization. In contrast, classical point-based baselines were more sensitive: Quatro performed poorly due to restrictive orientation assumptions, TEASER++ required fine voxel resolutions to achieve high success, and KISS-Matcher offered an efficiency-accuracy trade-off.

These results further highlight a trade-off between sparse and dense representations. Specifically, dense point cloud-based methods (S) gain an advantage in 3D stability, as evidenced by their stable performance in Venman03, by utilizing ground points to provide comprehensive 6-DoF constraints. By comparison, tree-based methods can become unstable when tree centers and base points are degraded by limited visibility or long-range occlusion. In practice, TreeLoc++ mitigates this by using the translation penalty $p(\|\mathbf{t}\|)$ during retrieval to favor closer candidates, where tree-level features are more distinctive, thereby improving robustness for metric localization.

Runtime: TreeLoc++ also provides a clear computational advantage by combining compact tree-level representations with retrieval-informed correspondences. As shown in Fig. 27, it completes full pose estimation in under 1 ms on a single CPU thread by reusing correspondences and avoiding expensive clique solving. In contrast, classical and learning-based baselines typically require over 100 ms, and can



(a) Session-wise Tree Reconstructions (b) Tree Distribution and Observations in Evo25 (c) DBH Changes between Evo23 and Evo25

FIGURE 28. (a) Trees reconstructed by RealtimeTrees in Evo23:00–01. Different trajectories lead to session-specific observations, motivating inventory management for partially covered areas. (b) Tree-center distributions in Evo25 after TreeLoc++ alignment; brighter colors indicate lower observation rates, highlighting regions requiring additional exploration. (c) Tree point clouds and corresponding DBH measurements from 2023 and 2025 (left). Continuous ecological monitoring is enabled by tracking tree correspondences over time. The DBH distribution for trees in the 20–80% growth percentile range (right) shows a slight shift to the right.

exceed 1 s at small voxel sizes. This makes them impractical for real-time deployment despite their accuracy, whereas TreeLoc++ incurs minimal latency while maintaining high precision.

In summary, TreeLoc++ achieves accurate and efficient 2D localization, while delivering 3D localization performance competitive with point cloud-based registration methods at a substantially lower computational cost. Its comparable performance to TEASER++ (T) indicates that tree inventory features, when paired with precise matching, are sufficient for robust forest localization.

D. Multi-Session Digital Forest Inventory Maintenance

Problem setting: We demonstrate a practical DFI maintenance pipeline by updating tree traits across multiple sessions (Fig. 2) and illustrating how the updated inventory can be used in long-term forest monitoring. In practical forest deployments, robots cannot exhaustively explore all regions and instead revisit forests along a limited set of traversed paths, yielding uneven observation coverage over time. Consequently, reconstructions can be session-dependent: some trees are successfully reconstructed in one session but missed in another due to viewpoint, occlusion, or partial coverage, as illustrated in Fig. 28a. This motivates multi-session updates, where observations are fused across sessions to recover missing trees and refine tree-level attributes, maintaining a consistent inventory for the regions of interest.

Multi-session fusion: Building on the multi-session registration results, we establish tree correspondences across sessions by transforming inventories into a common global frame and associating them via spatial proximity and stem traits. Instead of re-running the reconstruction on aggregated clusters, we consolidate the per-session tree traits by adopting the attributes from the session with the minimum DBH for each unique tree. This approach is designed to mitigate the overestimation of DBH typically caused by alignment errors, such as odometry drift or sensor noise. This unified inventory ensures more robust tree-level attributes for consistent downstream analyses.

Observation-aware map consistency: The updated multi-session DFI integrates observations across sessions into a consistent stem map, enabled by TreeLoc++'s robust global alignment and tree association. Fig. 28b shows the spatial distribution of stem centers and the number of sessions in which each tree was reconstructed. Trees in frequently revisited areas appear across multiple sessions, whereas those near the map boundaries are reconstructed in fewer sessions. This spatial pattern qualitatively indicates successful consolidation of observations in well-covered regions, while also revealing under-observed areas that may require further exploration.

Temporal analysis using tree-level attributes: Fig. 28c illustrates the potential for long-term ecological monitoring by comparing tree-level attributes across sessions. Using TreeLoc++, we identify consistent tree correspondences between Evo23 and Evo25, enabling reliable tracking of individual trees over time. However, detecting DBH changes over a two-year interval is challenging, as actual biological growth is limited. Furthermore, LiDAR sensor noise, such as the 3 cm accuracy of the Hesai QT64 used in Evo25, introduces measurement variability. Consequently, some trees in Fig. 28c (left, red boxes) show DBH variations exceeding the typical growth rate of $2-4$ mm per year. Nevertheless, the DBH distribution for the 20-80% percentile range (Fig. 28c, right) exhibits a slight rightward shift consistent with long-term growth. This morphological stability allows TreeLoc++ to leverage trees as reliable long-term landmarks, supporting robust global localization and consistent ecological monitoring across sessions.

E. Ablation Study on Positive Threshold Selection

Evaluation Protocol: We assessed how localization performance varied with different true positive thresholds by evaluating MF1, R@50, and ATE. We considered three thresholds: 3 m (as used in Wild-Places), 10 m (from the Oxford Forest Place Recognition dataset), and 5 m as a middle ground adopted in this manuscript. As in previous experiments, LoGG3D-Net and MinkLoc3Dv2 employed

TABLE 15. Localization performance under varying true-positive distance thresholds. Dataset-level N denotes the total number of queries, while threshold-level N indicates queries with at least one true-positive candidate within the given distance.

	Dimension / Method	Evo:Single (Queries: $N = 2334$)									Karawatha04 (Queries: $N = 1547$)								
		3 m ($N_{TP} = 1138$)			5 m ($N_{TP} = 1361$)			10 m ($N_{TP} = 1715$)			3 m ($N_{TP} = 328$)			5 m ($N_{TP} = 328$)			10 m($N_{TP} = 342$)		
		MF1	R@50	ATE	MF1	R@50	ATE	MF1	R@50	ATE	MF1	R@50	ATE	MF1	R@50	ATE	MF1	R@50	ATE
2D	LoGG3D-Net	0.596	0.726	0.125	0.685	0.671	0.128	0.741	0.566	0.146	0.863	*0.892*	0.125	0.939	*0.900*	0.131	0.940	*0.878*	0.119
2D	MinkLoc3Dv2	0.537	0.040	0.324	0.640	0.035	0.310	0.702	0.026	0.297	0.813	0.087	0.327	0.928	0.088	0.283	0.979	0.073	0.305
2D	BEVPlace++	0.889	0.990	0.122	0.933	0.979	0.134	0.939	0.920	0.151	*0.892*	0.511	0.211	*0.975*	0.502	0.217	**0.997**	0.499	0.225
2D	RING++	0.268	0.538	0.217	0.223	0.555	0.222	0.182	0.467	0.225	0.839	0.461	0.266	0.834	0.453	0.266	0.841	0.434	0.272
2D	BTC	0.725	0.951	0.142	0.761	0.929	0.145	0.843	0.860	0.167	0.506	0.531	0.163	0.621	0.527	0.158	0.817	0.529	0.156
2D	MapClosure	0.475	0.940	0.145	0.528	0.935	0.167	0.555	0.923	0.196	0.821	0.643	0.145	0.875	0.638	0.147	0.965	0.636	0.160
2D	TreeLoc	*0.904*	**1.000**	**0.022**	*0.955*	**0.998**	**0.024**	*0.981*	**0.993**	**0.028**	0.690	0.656	*0.049*	0.816	0.654	*0.055*	0.922	0.644	*0.055*
2D	**TreeLoc++**	**0.986**	*0.999*	*0.025*	**0.991**	*0.995*	*0.026*	**0.990**	*0.992*	*0.031*	**0.971**	**0.926**	**0.037**	**0.992**	**0.924**	**0.038**	*0.995*	**0.909**	**0.039**
3D	LoGG3D-Net	0.596	0.704	0.144	0.685	0.645	0.151	0.741	0.539	0.171	*0.863*	*0.889*	0.152	*0.939*	*0.891*	0.157	0.940	*0.872*	0.148
3D	MinkLoc3Dv2	0.537	0.030	0.338	0.640	0.026	0.334	0.702	0.019	0.307	0.813	0.028	0.302	0.928	0.046	0.327	*0.979*	0.018	0.298
3D	BTC	0.725	0.684	0.177	0.761	0.632	0.185	0.843	0.546	0.206	0.506	0.408	0.227	0.621	0.403	0.227	0.817	0.400	0.237
3D	MapClosure	0.475	0.916	0.154	0.528	0.897	0.177	0.555	0.881	0.207	0.821	0.640	0.145	0.875	0.635	0.148	0.965	0.632	0.160
3D	TreeLoc	*0.904*	*0.976*	*0.089*	*0.955*	*0.967*	*0.095*	*0.981*	*0.939*	*0.107*	0.690	0.594	*0.128*	0.816	0.593	*0.135*	0.922	0.583	*0.144*
3D	**TreeLoc++**	**0.986**	**0.992**	**0.070**	**0.991**	**0.982**	**0.073**	**0.990**	**0.955**	**0.080**	**0.971**	**0.919**	**0.061**	**0.992**	**0.924**	**0.068**	**0.995**	**0.909**	**0.074**

· Maximum F1 score (MF1), Recall@50 cm (R@50), Average Translation Error (ATE) [m].

feature-based matching and RANSAC-based registration, without re-ranking.

Performance Trends: As shown in Table. 15, most methods exhibited increased MF1 as the threshold grew, while R@50 and ATE tended to degrade. This behavior was expected, as larger thresholds admitted more distant candidates as true positives, increasing recall and thus MF1. However, this also reduced geometric overlap between matched pairs, leading to less reliable correspondences and increased translation error. This trend was consistent across both 2D and 3D localization.

Stability across Thresholds: TreeLoc++ maintained strong performance across all thresholds and was the only method that consistently achieved centimeter-level ATE. In contrast, most baseline methods showed large variations in MF1 and R@50 as the threshold changed, indicating a strong dependence on the threshold and reduced robustness to the definition of true positives. This variation indicated that the similarity scoring and correspondence selection mechanisms did not reliably capture real-world spatial relationships, leading to exaggerated sensitivity to threshold values and reduced stability in localization performance.

Conclusion: These findings highlighted that TreeLoc++ delivered consistent and robust localization performance across a wide range of true positive thresholds, in contrast to baseline methods that were highly sensitive to threshold selection. This stability underscored its suitability for real-world deployment, where fixed thresholds may not always be well defined.

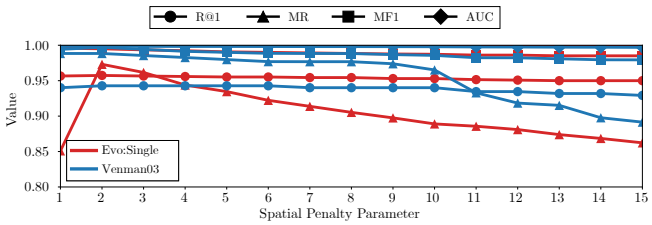


FIGURE 29. Sensitivity of TreeLoc++ to the spatial penalty parameter σ_t on `Evo:Single` and `Venman03`. The plots report Recall@1 (R@1), Maximum Recall at 100% precision (MR), Maximum F1 score (MF1), and Area Under the Precision–Recall Curve (AUC). Performance is generally stable across a wide range of σ_t, showing that the method is not highly sensitive to this parameter except for MR.

F. Ablation Study on Parameter Sensitivity

1) Effect of the Spatial Penalty Parameter

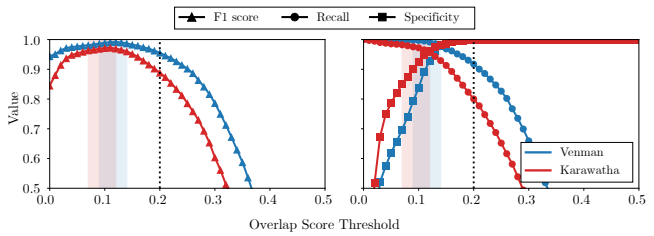
Evaluation Protocol: We evaluated the sensitivity of TreeLoc++ to the spatial penalty parameter σ_t, which controls the translation penalty in the overlap score $\mathcal{O}(Q, C)$. Following the same intra-session setup used in §1, we varied σ_t from 1 to 15 and reported R@1, MR, MF1, and AUC on `Evo:Single` and `Venman03`.

Performance Trends: As shown in Fig. 29, TreeLoc++ remained stable over a broad range of σ_t. Among the reported metrics, MR was the most sensitive, because a larger σ_t relaxed the translation penalty and retained more geometrically distant candidates. Nevertheless, on both datasets the reduction in MR stayed within about 0.1 up to $\sigma_t = 15$, while the other metrics varied only slightly, indicating limited practical sensitivity.

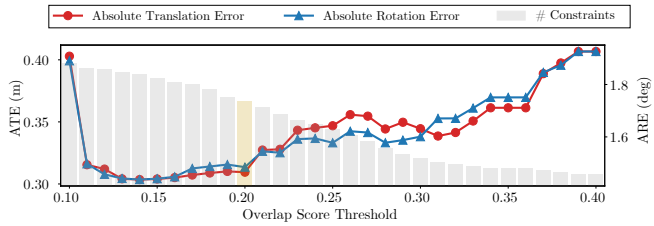
Parameter Choice: In practice, σ_t can be selected according to the spatial range over which the estimated pose is intended to remain reliable. In our experiments, we set $\sigma_t = 5$, consistent with the positive distance criterion used for place recognition. Although $\sigma_t = 5$ was not always the best-performing value, it provided a stable and reliable operating point across both datasets.

2) Effect of the Overlap Score Threshold

Evaluation Protocol: We analyzed the sensitivity of TreeLoc++ to the fixed overlap score threshold used for match acceptance. The threshold was first estimated on Wild-Places and then applied uniformly to all datasets. For inter-session



(a) Inter-session place recognition under varying overlap score thresholds.



(b) Multi-session localization under varying overlap score thresholds.

FIGURE 30. (a) Inter-session place recognition performance on `Venman` and `Karawatha` under varying overlap score thresholds. The shaded regions indicate the min-max ranges of pairwise F1-optimal thresholds, and the dotted line marks the fixed threshold of 0.2, chosen as a conservative operating point favoring high specificity. (b) Effect of the overlap score threshold on multi-session localization. The lines show ATE and ARE, and the bars show the number of accepted constraints. The threshold of 0.2 is highlighted. Over the range of 0.1-0.4, the localization result changed only slightly, with about 10 cm variation in ATE and 0.5° variation in ARE, indicating low practical sensitivity.

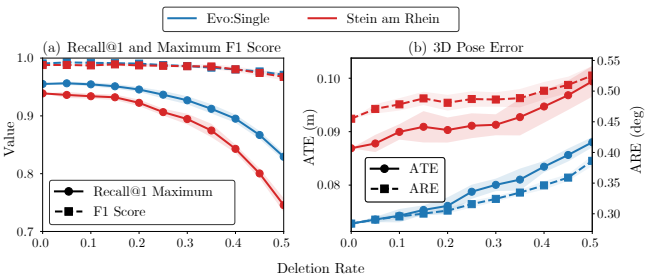


FIGURE 31. Sensitivity to missing tree detections in intra-session localization. The plots show the mean and standard deviation of Recall@1 (R@1), Maximum F1 score (MF1), and 3D pose errors under random deletion of input trees, averaged over 10 trials, on `Evo:Single` and `Stein am Rhein`. Although Recall@1 decreased as more trees were removed, MF1 and 3D pose errors remained comparatively stable, indicating that TreeLoc++ was relatively robust to missing tree detections.

place recognition, we evaluated all 24 directed pairs from Venman and Karawatha, and reported the mean F1 score across pairs, together with recall and specificity over all query results, where specificity is defined as $\mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$. For multi-session localization, we additionally reported ATE, ARE, and the number of accepted constraints.

Threshold Trade-off and Selection: As shown in Fig. 30a, the F1-optimal thresholds were generally concentrated around 0.1, consistent with the trend observed in §C. Thus, the fixed threshold of 0.2 was not chosen because it maximized place recognition performance on every dataset, but because it provided a conservative operating point for practical localization, where suppressing false positives is as important as achieving high recall. Correspondingly, recall at 0.2 was lower than near the F1-optimal region, whereas specificity was substantially higher.

Localization Robustness: This choice was further supported by the multi-session localization results in Fig. 30b. We used `Evo25:00` for this evaluation under the same setup as in §D. Although the F1-optimal region was generally near 0.1, it could produce relatively larger trajectory errors. By contrast, the selected threshold of 0.2 yielded consistently low ATE and ARE, even if it was not the absolute optimum for either metric. Moreover, sensitivity remained limited in practice: over the range of 0.1-0.4, the variation was only about 10 cm in ATE and 0.5° in ARE. These results indicate that the overlap threshold was not highly sensitive and that $\mathcal{O} > 0.2$ was a stable and practical choice.

3) Robustness to Missing Detections and DFI Noise

Evaluation Protocol: To better separate localization performance from upstream DFI quality, we conducted two controlled robustness tests. For intra-session localization, we randomly deleted 0-50% of the input trees in the DFI used for descriptor generation on `Evo:Single` and `Stein am Rhein`, and reported R@1, MF1, ATE, and ARE over 10 trials. For inter-session place recognition, we perturbed only the query-side DFI by adding Gaussian noise to tree centers, DBH, or both, with standard deviations of 1, 3, and 5 cm, and evaluated all directed pairs in Venman and Karawatha. We again reported R@1 and MF1 over 10 trials.

Effect of Missing Tree Detections: As shown in Fig. 31, increasing tree deletion progressively reduced R@1 on both `Evo:Single` and `Stein am Rhein`, whereas the MF1 degraded more slowly. Importantly, the degradation remained moderate, with the reduction in R@1 staying within about 0.1 even when about 40% of the input trees were removed. The 3D localization errors also increased only gradually, and even at 50% deletion, ATE rose by only about 1–2 cm and ARE by about 0.1°. These results suggest that missing detections mainly weakened retrieval, while having a smaller effect on candidate discrimination and downstream pose estimation once a valid match was found.

Effect of Upstream DFI Quality: As shown in Fig. 32, Recall@1 was substantially more sensitive to tree-center perturbation than to DBH perturbation, whereas the MF1 remained comparatively stable, similar to the trend observed under missing detections. Even under DFI perturbations, the retrieval metrics remained within 0.1-0.2 of the nominal case, indicating limited degradation. This suggests that degraded geometry primarily affected retrieval ranking, while TreeLoc++ could still reject outliers and identify correct correspondences once informative candidates were retrieved. The joint perturbation largely followed the center-noise trend, indicating that accurate tree positions were more critical than accurate DBH estimates for robust inter-session matching.

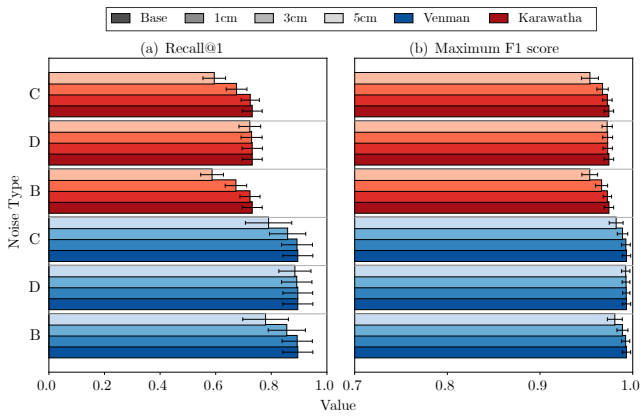


FIGURE 32. Sensitivity to upstream DFI perturbations in inter-session place recognition. The plots show the mean and standard deviation of Recall@1 (R@1) and Maximum F1 score (MF1) under random query-side perturbations of tree centers (C), DBH (D), and both (B), averaged over 10 trials. While R@1 was more sensitive to center perturbation than to DBH perturbation, MF1 remained relatively stable, indicating that TreeLoc++ could still reject outliers and identify correct matches once informative candidates were retrieved.

Conclusion: These controlled experiments show that TreeLoc++ is more sensitive to missing detections and tree-position errors than to DBH noise. However, the degradation was gradual rather than abrupt, and the method remained comparatively stable in MF1 and localization accuracy under moderate perturbations. Overall, these results suggest that, although upstream DFI quality contributes to performance variation, the gains of TreeLoc++ are not solely attributable to a specific tree reconstruction pipeline such as RealtimeTrees.

G. TreeLoc vs. TreeLoc++: Statistical Evaluation

Evaluation Protocol: To assess whether the gains of TreeLoc++ over TreeLoc were statistically meaningful, we compared the two methods on four representative intra-session sequences: `Evo:Single`, `Stein am Rhein`, `Venman03`, and `Karawatha03`. These sequences span both the smaller Oxford Forest setting and the more challenging Wild-Places setting. We summarized the comparison using three complementary views: binary performance, discordant-query counts, and performance improvements, covering R@1, R@50 in 3D, SR in 2D and 3D, and MF1, AUC, ATE, and ARE. Exact McNemar tests were used for the query-level binary metrics underlying both the binary and discordant comparisons, and Wilcoxon signed-rank tests were used for ATE and ARE on the common-success subset. For all reported metrics, confidence intervals were estimated using paired bootstrap resampling over queries with 2500 samples; for MF1 and AUC, significance was also assessed using the same paired bootstrap procedure.

Binary and Discordant Comparisons: As shown in Fig. 33a and Fig. 33b, TreeLoc++ consistently outperformed TreeLoc in both binary and discordant-query comparisons. The gains were smaller on the Oxford Forest sequences,

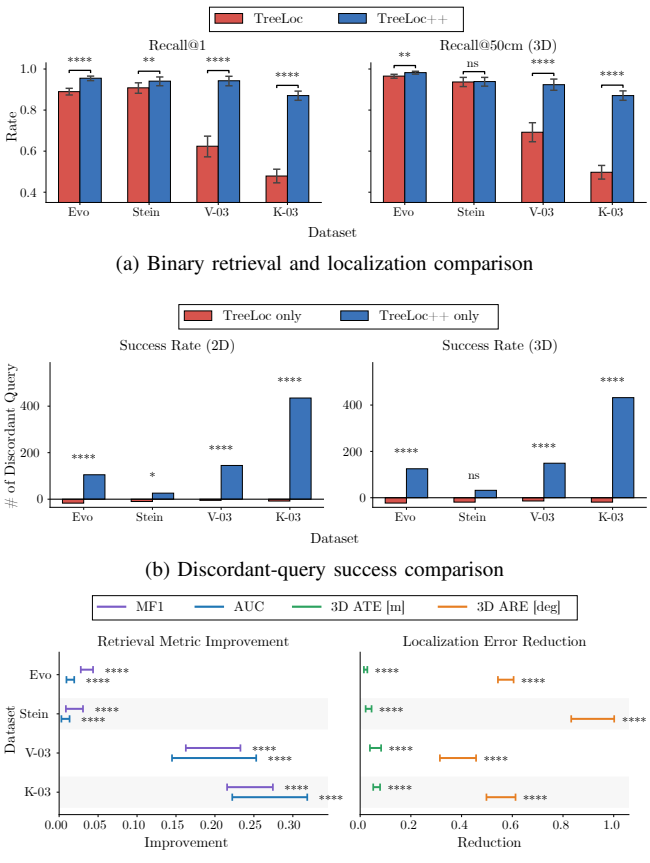


FIGURE 33. TreeLoc vs. TreeLoc++ statistical comparison on four intra-session sequences. Error bars indicate 95% bootstrap confidence intervals. In (a) and (b), asterisks denote significance levels from exact McNemar tests; in (c), they denote paired bootstrap significance for MF1 and AUC, and Wilcoxon signed-rank significance for ATE and ARE. (a) Binary comparison of Recall@1 and Recall@50cm (3D). (b) Discordant-query counts for Success Rate (2D and 3D), showing queries localized only by TreeLoc++ or only by TreeLoc. (c) Comparison of retrieval and localization gains, including improvements in MF1 and AUC and reductions in ATE and ARE.

especially `Stein am Rhein`, where TreeLoc already performed strongly and the stricter 3D criteria were statistically similar. By contrast, on the more challenging Wild-Places sequences, `Venman03` and `Karawatha03`, TreeLoc++ achieved substantially larger and statistically significant improvements across all binary and discordant comparisons. This indicates that the advantage of TreeLoc++ was not driven by a few isolated cases, but by a systematic increase in the set of queries that could be localized successfully.

Magnitude of Performance Improvements: As shown in Fig. 33c, the gains of TreeLoc++ extended beyond binary success rates. TreeLoc++ improved both MF1 and AUC on all four sequences, with the largest gains again appearing on `Venman03` and `Karawatha03`. It also consistently reduced downstream localization errors. Averaged across the four sequences, TreeLoc++ reduced 3D ATE by about 31.8% and 3D ARE by about 61.4%. The corresponding confidence

intervals indicate that the improvements were consistent in both retrieval quality and downstream pose accuracy.

Conclusion: Overall, these statistical comparisons support the conclusion that the improved performance of TreeLoc++ over TreeLoc was systematic rather than incidental. The improvements appeared not only in query-level recall and success rates, but also in retrieval quality and final pose accuracy, with the largest gains observed in the more cluttered and structurally ambiguous forest scenes.