LEADER: Learning Reliable Local-to-Global Correspondences for LiDAR Relocalization

Supplementary Material

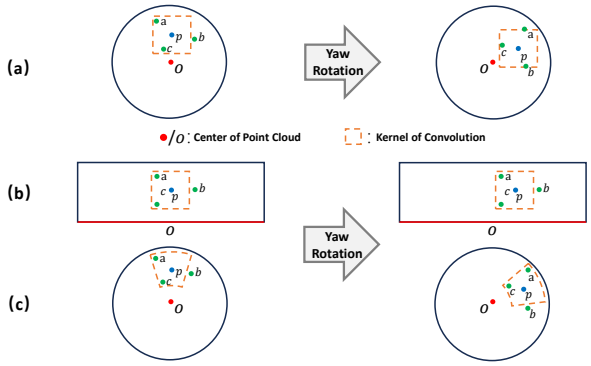


Figure 10. Illustration of cylindrical projection and convolution correspondence. **(a)** Original point cloud $\mathcal{P}_1$ in Cartesian coordinates. **(b)** Projected point cloud $\mathcal{P}_2$ in cylindrical coordinates after Spatial Transformation. **(c)** The equivalent convolution operation in $\mathcal{P}_1$ corresponding to the rectangular convolution kernel applied on $\mathcal{P}_2$, which forms a sector-shaped receptive field.

6. Supplement for Method

6.1. The planar rectification in Spatial Transformation

Given the raw point cloud $\mathcal{P}_{\text{raw}} = \{\mathbf{p}_i = (x_i, y_i, z_i)\}_{i=1}^{N}$ and the corresponding transformation matrix $\mathbf{T}$, we first perform the ground plane rectification using Patchwork++ [3]:

1. Estimate ground plane equation $ax + by + cz + d = 0$ via RANSAC[11]:
2. Compute rotation matrix $\mathbf{R}_{\text{plane}}$ aligning ground normal $\mathbf{n} = (a, b, c)$ with z-axis:

$$\theta = \arccos\left(\frac{\mathbf{n}\cdot\mathbf{e}_z}{\|\mathbf{n}\|}\right), \quad \mathbf{v} = \frac{\mathbf{n}\times\mathbf{e}_z}{\|\mathbf{n}\times\mathbf{e}_z\|}, \tag{13}$$
$$\mathbf{R}_{\text{plane}} = \exp(\theta\mathbf{v}_\times) \quad \text{(Rodrigues' formula)}.$$

3. We denote the transformation matrix for the planar rectification as $\mathbf{T}_{\text{plane}}$:

$$\mathbf{T}_{\text{plane}} = \begin{bmatrix} \mathbf{R}_{\text{plane}} & \mathbf{t}_{\text{plane}} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \mathbf{t}_{\text{plane}} = \frac{-d}{\|\mathbf{n}\|^2}\mathbf{n}. \tag{14}$$

4. We apply the following rectification to correct the point cloud to horizontal:

$$\mathbf{p}_i' = \mathbf{R}_{\text{plane}} \cdot \mathbf{p}_i + \mathbf{t}_{\text{plane}}. \tag{15}$$

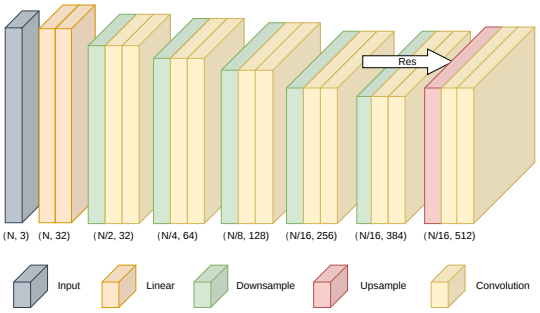


Figure 11. Overview of the proposed encoder architecture.

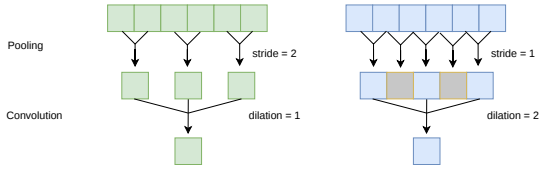


Figure 12. Illustration of the dilated cyclic sparse convolution used in stage 5 to enlarge the receptive field without reducing spatial resolution (right).

6.2. Detailed Encoder Architecture

As illustrated in Figure 11, our encoder begins with a 3-dimensional input, which is first projected to 16 dimensions using a residual fully connected layer, and then further projected to 32 dimensions. The architecture then consists of six stages: stages 1–5 perform downsampling followed by two convolutional layers, and stage 6 performs upsampling, fusion, and two convolutional layers.

For stages 1–4, downsampling is implemented with a cyclic sparse convolution using stride=2, kernel=2, and dilation=1, whose output is concatenated with the result of max pooling. This is followed by two cyclic sparse convolution layers with stride=1, kernel=3, and dilation=1. After the fourth downsampling stage, the point cloud has been downsampled by a factor of 16, resulting in relatively sparse points. In stage 5, to further enlarge the receptive field without reducing the number of points, we set stride=1 and increase dilation to 2 in the cyclic sparse convolution, as illustrated in Figure 12.

Stage 6 performs upsampling using a transposed cyclic sparse convolution with stride=1, kernel=2, and dilation=1. The upsampled features are then concatenated with the output from stage 4 and projected to 512 dimensions. Finally,

Table 6. Localization results for the NCLT dataset at different yaw angles. All values are given as (m, °).

Yaw Angle	LEADER	LightLoc	DiffLoc
Original	0.31, 1.81	1.70, 2.95	1.19, 2.31
Yaw +180°	0.31, 1.81	21.70, 21.57	5.58, 36.36
Random Yaw	0.31, 1.81	30.51, 22.46	4.57, 76.50

two cyclic sparse convolution layers with stride=1, kernel=3, and dilation=2 are applied to produce the final enriched features.

6.3. Regressor hyperparameters

The design of our regression network is inspired by the FFN layers in the Transformer architecture and the commonly used pooling layers in the field of computer vision, and the hyperparameters, including the number of heads and layers, are set based on empirical values.

7. RPGE Module Analysis

7.1. Projection Description for Yaw Robustness

In the Spatial Transformation module, we convert the Cartesian coordinates to the Cylindrical coordinates. In Eq. (1), the z-axis remains unchanged during projection; thus, we omit the z-axis (equivalent to a top-down view) and voxelization to illustrate the principle of this projection. The visualization is shown in Fig. 10. Let the original point cloud be $\mathcal{P}_1$ and the projected point cloud be $\mathcal{P}_2$. The kernel of sparse convolution is rectangular. When applying convolution on the projected point cloud $\mathcal{P}_2$, the rectangular kernel corresponds to a sector in the original point cloud $\mathcal{P}_1$. When a yaw rotation occurs in $\mathcal{P}_1$, the point cloud rotates clockwise or counterclockwise. In contrast, in $\mathcal{P}_2$, this rotation is transformed into a translation. Convolution inherently possesses translation equivariance. Consider the changes before and after rotation at a point p. In $\mathcal{P}_1$, both the region covered by the convolutional kernel at point p and the corresponding points at different kernel positions change, leading to a lack of rotation robustness. However, in $\mathcal{P}_2$, the region and relative relationships covered by the convolutional kernel at point p remain consistent before and after rotation, thereby endowing the method with yaw robustness.

Furthermore, after projection, the originally continuous yaw angles in $\mathcal{P}_1$ become discontinuous at the two ends of $\mathcal{P}_2$. To address this, we employ circular sparse convolution by pre-padding both sides according to the kernel size. After convolution, the padded regions are removed, as Eq. (2). This ensures full-range yaw robustness.

7.2. Robustness Analysis of Spatial Transformation

To validate the effectiveness of the Spatial Transformation module, we conduct comprehensive comparisons including our method and two state-of-the-art methods (LightLoc [6] and DiffLoc [5]) under varying yaw rotations on the NCLT dataset [2]. As shown in Table 6, the proposed Spatial Transformation demonstrates remarkable robustness to yaw variations:

- **Compared methods** exhibit comparable sensitivity to yaw angle variations: LightLoc achieves a performance of 21.70 m / 21.57° at yaw+180° and 30.51 m / 22.46° under random yaw, whereas DiffLoc attains 5.58 m / 36.36° and 4.57 m / 76.50° under the same respective conditions.
- **Our method** maintains consistent performance across all yaw angles, achieving stable errors of 0.31 m/1.81° regardless of the rotation magnitude.

These results conclusively demonstrate that our Spatial Transformation module is essential for achieving yaw-invariant localization, effectively mitigating performance degradation under rotational variations commonly encountered in real-world autonomous driving scenarios.

Table 7. Performance comparison of coordinate systems

Method	Position Error	Orientation Error
Cylindrical projection	0.31 m	1.81°
Spherical projection	0.35 m (↑ 12.9%)	1.97°(↑ 8.8%)

7.3. Coordinate System Ablation Study

Table 7 compares our Sptial Transformation (Cylindrical projection) with Spherical projection:

The Spherical Projection exhibits higher errors than our Spatial Transformation approach, with a +12.9% increase in position error (0.35 vs. 0.31 m) and a +8.8% increase in orientation error (1.97° vs. 1.81°).

8. TRR Module Analysis

8.1. TRR Loss Principle

We hereby elucidate the principle of our proposed Truncated Relative Reliability (TRR) loss. The Euclidean loss serves as the fundamental point-wise loss, as defined in Eq. (5) of the main text.

We perform two normalization operations on the reliability scores, given by Eqs. (6) and (7) in the main text, where the constant $K_s = \frac{\ln 10}{\pi}$.

The core components of TRR loss are defined in Eqs. (8) and (9) of the main text.

In Eqs. (6) and (7), we employ $\arctan$ to constrain values within $[-\frac{\pi}{2}, \frac{\pi}{2}]$, and scale them by K_s. The rationale for choosing $K_s = \frac{\ln 10}{\pi}$ is that it expands the output range of u_i to $[-\frac{\ln 10}{2}, \frac{\ln 10}{2}]$. After exponentiation in Eq. (8), this range becomes $[10^{-\frac{1}{2}}, 10^{\frac{1}{2}}]$, where the maximum value is 10 times the minimum. Therefore, the constant K_s constrains the ratio between maximum and minimum weights

in Eq. (8) to approximately 10. This prevents the weights from degenerating into the mean Euclidean loss when differences are too small, while also avoiding scenarios where the model focuses excessively on high-quality points, neglecting moderately effective points during training.

Furthermore, we employ two components for reliability normalization: u_i^{scale} and u_i^{cut}. In Eq. (8), w_i is normalized using both u_i^{scale} and u_i^{cut}.

When $u_i \in [-10\pi, 10\pi]$, $u_i^{\text{scale}} = u_i^{\text{cut}}$, and the weight calculation w_i becomes equivalent to softmax. Considering the total loss formulation $\mathcal{L}_{\text{TRR}} = \sum_i w_i \mathcal{L}_{\text{raw},i}$, we analyze the contribution of a specific point m. The total loss can be decomposed as:

$$\mathcal{L}_{\text{TRR}} = w_m \cdot \mathcal{L}_{\text{raw},m} + \sum_{j \neq m} w_j \cdot \mathcal{L}_{\text{raw},j} \tag{16}$$

where $\sum_{j \neq m} w_j = 1 - w_m$ due to the softmax-like normalization. For analytical clarity, we consider the average behavior of other points by defining $\bar{\mathcal{L}}_{\text{raw}}^{m-} = \frac{\sum_{j \neq m} w_j \mathcal{L}_{\text{raw},j}}{\sum_{j \neq m} w_j}$, allowing us to express the loss as:

$$\mathcal{L}_{\text{TRR}} \approx w_m \cdot \mathcal{L}_{\text{raw},m} + (1 - w_m) \cdot \bar{\mathcal{L}}_{\text{raw}}^{m-} \tag{17}$$

This approximation highlights the trade-off between point m and other points. When point m has high feature quality and consequently better regression quality ($\mathcal{L}_{\text{raw},m} < \bar{\mathcal{L}}_{\text{raw}}^{m-}$), the model can reduce the total loss by increasing u_m, which increases w_m and decreases the relative contribution of other points. Conversely, when point m has low quality, decreasing u_m reduces the total loss. This compels the model to prioritize learning high-quality points, while the K_s constant ensures all points receive adequate training by constraining the weight range.

When u_i exceeds $[-10\pi, 10\pi]$, we analyze the case where $u_i > 10\pi$. Consider point n with weight w_n and Euclidean loss $\mathcal{L}_{\text{raw},n}$, while other points have losses $\mathcal{L}_{\text{raw},i}$ and weights w_i. The total loss becomes:

$$\mathcal{L}_{\text{TRR}} = w_n \cdot \mathcal{L}_{\text{raw},n} + \sum_{i \neq n} w_i \cdot \mathcal{L}_{\text{raw},i} \tag{18}$$

When $u_n > 10\pi$, u_n^{scale} continues to increase while u_n^{cut} remains constant, resulting in $u_n^{\text{scale}} > u_n^{\text{cut}}$. In Eq. (8), the numerator increases while the denominator remains unchanged, causing w_n to increase. Since the denominator remains fixed, other weights w_i stay constant, leading to $\sum w_i > 1$ and consequently an increased total loss. To minimize loss, the model avoids u exceeding 10π. The case for $u_n < -10\pi$ follows similar reasoning.

The dual design of u_i^{scale} and u_i^{cut} ensures gradients exist even when u_i exceeds $[-10\pi, 10\pi]$. This avoids the gradient vanishing problem that would occur with only u_i^{scale} due to floating-point precision limitations, while also preventing the complete absence of gradients that would result from using only u_i^{cut} beyond the clamping range.

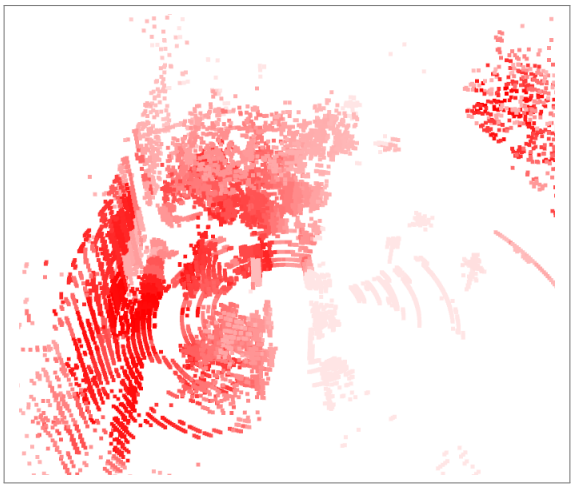


Figure 13. Visualization of Reliability scores. Points are color-coded by reliability (dark red: high, light/white: low).

8.2. Reliability Visualization

We visualize the reliability scores u, as shown in Fig. 13. The point cloud is color-coded by reliability, where darker red indicates higher reliability and lighter/white colors indicate lower reliability.

As observed, points on buildings and at the base of dense vegetation exhibit higher reliability, while ground points, sparse shrubs, and vegetation tops (leaves) show lower reliability. This demonstrates the model's ability to effectively distinguish between different structural elements in the environment.

Notably, we observe that ground points adjacent to buildings or dense vegetation also maintain relatively high reliability. This can be attributed to the receptive field of sparse convolution - the stacked convolutional layers in the encoder ensure that each point represents features from its local neighborhood. Thus, even ground points incorporate contextual information from nearby structures, enhancing their reliability.

8.3. Quantitative Analysis by Reliability Groups

To further validate the effectiveness of reliability scores, we partition points into quartiles based on their reliability rankings and evaluate pose estimation performance using each subset independently. The results are presented in Table 8.

The results clearly demonstrate that higher reliability correlates with better pose estimation accuracy. The highest reliability group achieves position and orientation errors of 0.32 m and 1.96° and respectively, while the lowest reliability group shows significantly higher errors of 2.39 m and 3.63°. This represents a 86.7% reduction in position error and a 46.0% reduction in orientation error for the highest

Table 8. Mean position error (m) and mean orientation error (°) across reliability quartiles.

Reliability Quartile	Error (m/°)
0-25% (Highest)	0.32, 1.96
25-50%	0.34, 1.97
50-75%	0.67, 2.33
75-100% (Lowest)	2.39, 3.63

Table 9. Performance improvement by integrating TRR loss into SGLoc

Metric	Position Error (m)	Orientation Error (°)
SGLoc (Original)	1.83	3.54
SGLoc + TRR	1.76 (↓ 3.83%)	2.61 (↓ 26.27%)

reliability group compared to the lowest reliability group.

8.4. TRR Loss Generalizability Analysis

To demonstrate the portability of our proposed TRR loss, we integrate it into SGLoc [4]. As shown in Table 9, quantitative improvements are observed as follows:

- **Position error:** Reduced from 1.83 m to 1.76 m (3.83% reduction)
- **Orientation error:** Reduced from 3.54° to 2.61° (26.27% reduction)

These results confirm that TRR loss is not merely specialized for our architecture, but serves as a generalizable strategy for other localization architecture.

8.5. Comparison with Alternative Loss Functions

To validate the effectiveness of our proposed TRR loss, we compare it against two alternative loss functions: the standard Euclidean distance mean loss and a modified version of the Matching loss from RSKDD-Net. Given ground truth coordinates $\mathbf{c}_i^{\text{gt}}$, predicted coordinates $\hat{\mathbf{c}}_i$, and reliability scores u_i, we evaluate the following loss functions:

- **Mean Euclidean loss**: This baseline loss function is defined as the mean of the per-point losses $\mathcal{L}_{\text{raw},i}$ over the entire point cloud:

$$\mathcal{L}_{\text{raw}} = \frac{1}{N}\sum_{i}^{N}\mathcal{L}_{\text{raw},i} \tag{19}$$

where N is the total number of points.

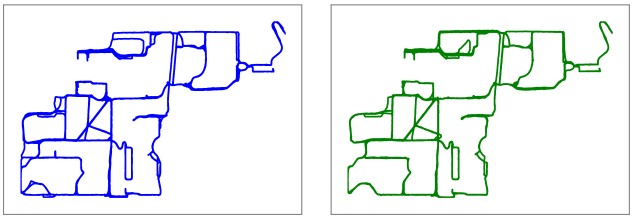
- **Modified Matching loss**: We adapt the RSKDD-Net [7] Matching loss to our framework. The formulation is:

$$w_i = \max\{\sigma^{\max} - \sigma_i, 0.01\}, \quad \tilde{w}_i = \frac{w_i}{\sum_j w_j} \tag{20}$$

$$\mathcal{L}_{\text{matching}} = \sum_i \tilde{w}_i \|\mathbf{c}_i^{\text{gt}} - \hat{\mathbf{c}}_i\|_2 \tag{21}$$

Table 10. Performance comparison of different loss functions on NCLT dataset

Method	Error (m, °)
Mean Euclidean loss	0.59, 2.12
Matching loss	0.43, 1.97
TRR loss (Ours)	**0.31, 1.81**



(a) training set (b) test set

Figure 14. Trajectory visualization of the NCLT dataset.

We set the hyperparameter $\sigma^{\max} = 1$.

- **TRR loss**: Our proposed loss as defined in Eq. (8) and Eq. (9) of the main text.

The quantitative comparison on the NCLT dataset is presented in Tab. 10. All methods maintain identical inference time (48 ms), ensuring fair comparison.

Our TRR loss demonstrates significant performance improvements over alternative approaches. Compared to the mean Euclidean loss, TRR achieves 47.4% and 14.6% reduction in position and orientation errors respectively. When compared to Matching loss, TRR reduces errors by 27.9% and 8.1%.

9. Evaluation Metrics

Let $\mathbf{T}_{\text{final},i}$ (Eq. (12)) denote the estimated pose transformation matrix of the i-th frame and $\mathbf{T}_{\text{gt},i}$ represent its ground-truth pose transformation matrix. The evaluation metrics are defined as:

Mean Position Error: For each frame i, let $\mathbf{t}_{\text{final},i} \in \mathbb{R}^3$ and $\mathbf{t}_{\text{gt},i} \in \mathbb{R}^3$ be the translation vectors extracted from $\mathbf{T}_{\text{final},i}$ and $\mathbf{T}_{\text{gt},i}$, respectively. The Mean Position Error is computed as:

$$\text{MPE} = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{t}_{\text{final},i} - \mathbf{t}_{\text{gt},i}\|_2 \tag{22}$$

where N is the total number of frames.

Mean Orientation Error: For each frame i, let $\mathbf{R}_{\text{final},i}$ and $\mathbf{R}_{\text{gt},i}$ be the rotation matrices from $\mathbf{T}_{\text{final},i}$ and $\mathbf{T}_{\text{gt},i}$, respectively. The Mean Orientation Error error (in degrees)

Table 11. Comparisons of different methods on the NCLT dataset.

Metric	LEADER	LEADER (xy)	RING/RING++
MPE (m)	0.31	0.28	16.34
MOE (°)	1.81	1.03	10.45
MedPE (m)	0.24	0.21	0.27
Recall@1	—	—	75.07%
Success@5m	99.72%	99.72%	92.79%

is calculated using:

$$\text{MOE} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{180}{\pi}\cdot\arccos\left(\frac{\text{tr}(\mathbf{R}_{\text{rel},i})-1}{2}\right)\right) \quad (23)$$

where $\mathbf{R}_{\text{rel},i} = (\mathbf{R}_{\text{final},i})^\top \mathbf{R}_{\text{gt},i}$ and $\text{tr}(\cdot)$ denotes the matrix trace. The error is bounded to $[0°, 180°]$.

10. Additional experiment

We compare our method against the retrieval-based approaches RING [8] and RING++ [10] on the NCLT dataset. The same mapping trajectories (with keyframes selected at 10 m intervals) and test trajectories as used in the main paper are adopted for these baselines, while the training and test trajectories for our method follow the same configuration as described in the Experiment section of the main paper. **All frames in the test set are used for evaluation**.

To enable a direct comparison, we additionally report the performance of our method in the xy-plane. Specifically, we compute the position error as the Euclidean distance error in xy coordinates, and the orientation error as the yaw angle difference. This is because the baseline methods directly provide localization results in the xy-plane. Table 11 summarizes the quantitative results. In the table, MPE, MOE, and MedPE denote Mean Position Error, Mean Orientation Error, and Median Position Error, respectively. For retrieval methods, Recall@1 indicates retrieval success within 5 m, while Success@5m represents final localization success after registration. For our method, Success@5m directly measures localization accuracy within 5m. For each metric, we report the best result achieved by either RING or RING++.

Based on the results in Table 11, our method achieves an MPE of 0.28 m and an MOE of 1.03° in the xy-plane, substantially outperforming RING/RING++, which yield 16.34 m and 10.45°, respectively. We attribute this large margin in part to the high failure rate of the retrieval-based baselines, which significantly inflates their average errors. Specifically, RING/RING++ attain a Recall@1 of only 75.07%, and after registration, their Success@5m reaches 92.79%, still leaving a failure rate of 7.21%. In contrast, our method achieves a Success@5m of 99.72%, with a failure rate of only 0.28%, which is less than 3.9% of that of the RING/RING++. Moreover, in terms of MedPE, which is largely unaffected by failed frames, our method achieves 0.21 m, still clearly outperforming the 0.27 m of RING/RING++.

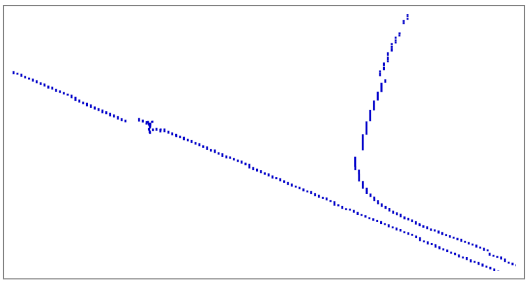


Figure 15. Ground truth trajectory visualization of Quality-enhanced Oxford dataset.

11. Dataset

11.1. Dataset Selection

We justify our dataset selection by addressing the absence of KITTI [2], a common LiDAR SLAM benchmark. Our task involves relocalization within a pre-built map, which requires significant overlap between training and test trajectories in **seen** scenes. However, most KITTI sequences lack such overlap, making it suboptimal for this purpose. Consequently, we use the NCLT and Oxford datasets, where training and test data are collected in the same environment with substantial trajectory overlap. Fig. 14 visualizes this overlap in the NCLT dataset, demonstrating the high spatial coincidence essential for our evaluation.

11.2. Analysis on Quality-enhanced Oxford Dataset

While our method achieves a 74.9% reduction in mean position error on the NCLT dataset compared to state-of-the-art methods, the improvement on the Quality-enhanced Oxford dataset [9] is 24.9%. Although still significantly superior, this performance gap warrants further analysis.

We attribute this discrepancy primarily to limitations in the ground truth of the Quality-enhanced Oxford dataset. Despite the calibration improvements introduced by SGLoc, the corrected trajectory exhibits noticeable jagged artifacts and discontinuities due to its floating-point precision limitations, as visualized in Fig. 15. These irregularities in the ground truth trajectory inevitably constrain further accuracy improvements, as they introduce inherent uncertainties during both training and test phases.

12. Visualization

A more comprehensive comparison between Quality-enhanced Oxford and NCLT is presented in Tabs. 12 and 13. As can be seen, LEADER has almost no catastrophic errors in its estimates, proving the robustness of its localization.

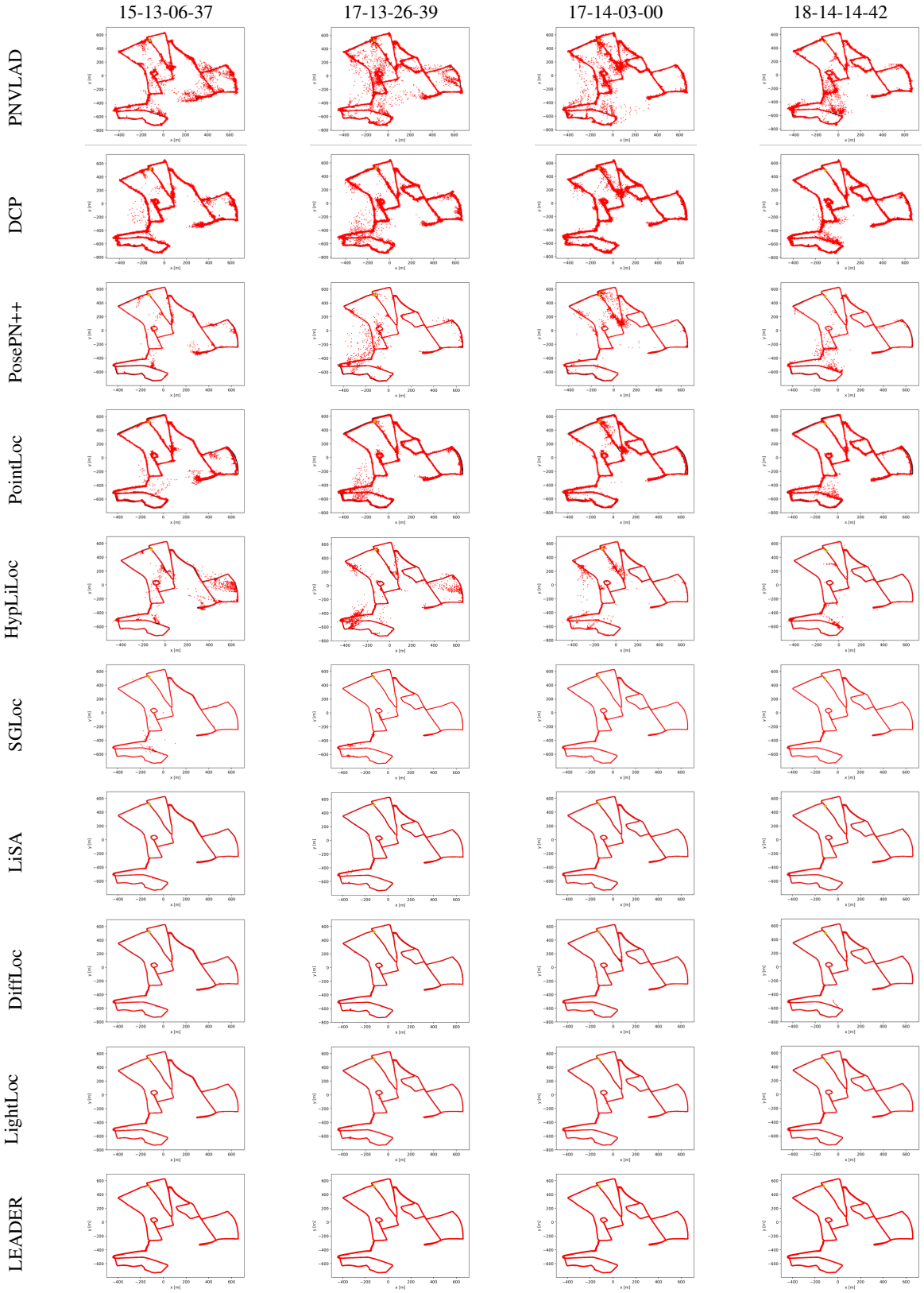


Table 12. Visualization on Quality-enhanced Oxford Dataset, the star indicates the starting point.

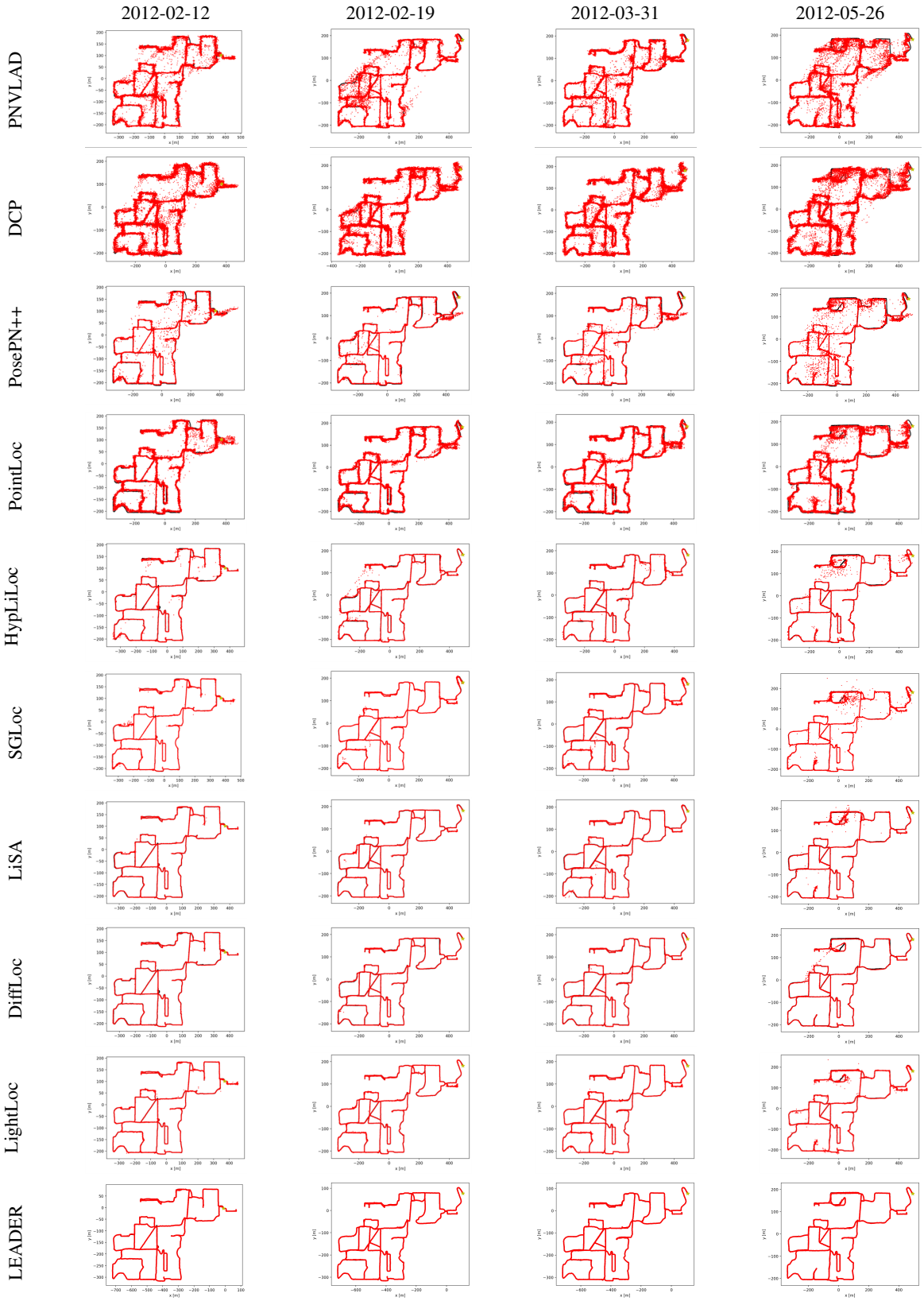


Table 13. Visualzation on NCLT Dataset, the star indicates the starting point.

References

[1] Nicholas Carlevaris-Bianco, Arash K. Ushani, and Ryan M. Eustice. University of Michigan North Campus long-term vision and lidar dataset. 35(9):1023–1035, 2015. 5

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 5

[3] Seungjae Lee, Hyungtae Lim, and Hyun Myung. Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3D point cloud. pages 13276–13283, 2022. 3, 1

[4] Wen Li, Shangshu Yu, Cheng Wang, Guosheng Hu, Siqi Shen, and Chenglu Wen. Sgloc: Scene geometry encoding for outdoor lidar localization. In *CVPR*, pages 9286–9295, 2023. 1, 2, 5, 6, 4

[5] Wen Li, Yuyang Yang, Shangshu Yu, Guosheng Hu, Chenglu Wen, Ming Cheng, and Cheng Wang. Diffloc: Diffusion model for outdoor lidar localization. In *CVPR*, pages 15045–15054, 2024. 1, 2, 5, 6

[6] Wen Li, Chen Liu, Shangshu Yu, Dunqiang Liu, Yin Zhou, Siqi Shen, Chenglu Wen, and Cheng Wang. Lightloc: Learning outdoor lidar localization at light speed. In *CVPR*, pages 6680–6689, 2025. 1, 2, 5, 6

[7] Fan Lu, Guang Chen, Yinlong Liu, Zhongnan Qu, and Alois Knoll. Rskdd-net: Random sample-based keypoint detector and descriptor. In *NeurIPS*, 2020. 4

[8] Sha Lu, Xuecheng Xu, Huan Yin, Zexi Chen, Rong Xiong, and Yue Wang. One ring to rule them all: Radon sinogram for place recognition, orientation and translation estimation. pages 2778–2785. IEEE, 2022. 5

[9] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. 36(1): 3–15, 2017. 5

[10] Xuecheng Xu, Sha Lu, Jun Wu, Haojian Lu, Qiuguo Zhu, Yiyi Liao, Rong Xiong, and Yue Wang. Ring++: Roto-translation-invariant gram for global localization on a sparse scan map. *IEEE TRO*, 2023. 1, 5

[11] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. 2018. 1