Breaking the Static Assumption: A Dynamic-Aware LIO Framework via Spatio-Temporal Normal Analysis

Zhiqiang Chen, Cedric Le Gentil, Fuling Lin, *Graduate Student Member, IEEE*, Minghao Lu, *Member, IEEE*, Qiyuan Qiao, *Graduate Student Member, IEEE*, Bowen Xu, Yuhua Qi, and Peng Lu

***Abstract*—This paper addresses the challenge of Lidar-Inertial Odometry (LIO) in dynamic environments, where conventional methods often fail due to their static-world assumptions. Traditional LIO algorithms perform poorly when dynamic objects dominate the scenes, particularly in geometrically sparse environments. Current approaches to dynamic LIO face a fundamental challenge: accurate localization requires a reliable identification of static features, yet distinguishing dynamic objects necessitates precise pose estimation. Our solution breaks this circular dependency by integrating dynamic awareness directly into the point cloud registration process. We introduce a novel dynamic-aware iterative closest point algorithm that leverages spatio-temporal normal analysis, complemented by an efficient spatial consistency verification method to enhance static map construction. Experimental evaluations demonstrate significant performance improvements over state-of-the-art LIO systems in challenging dynamic environments with limited geometric structure.**

***Index Terms*—Localization, SLAM, mapping.**

I. Introduction

LIDAR-INERTIAL Odometry (LIO) has become essential for precise localization and mapping across robotics applications by integrating detailed 3D point clouds with high-frequency motion data. Despite its effectiveness, conventional LIO systems [1], [2], [3], [4] operate under a static environment assumption that rarely holds in practice. When moving objects like pedestrians and vehicles appear in the sensor field of view, they introduce significant errors in both pose estimation and map construction. These errors stem from registration algorithms such as Iterative Closest Point (ICP) [5] mistakenly using dynamic elements as fixed reference points. This problem becomes particularly acute in environments where dynamic objects predominate or where static geometric features are limited, leading to substantial localization failures as illustrated in Fig. 1.

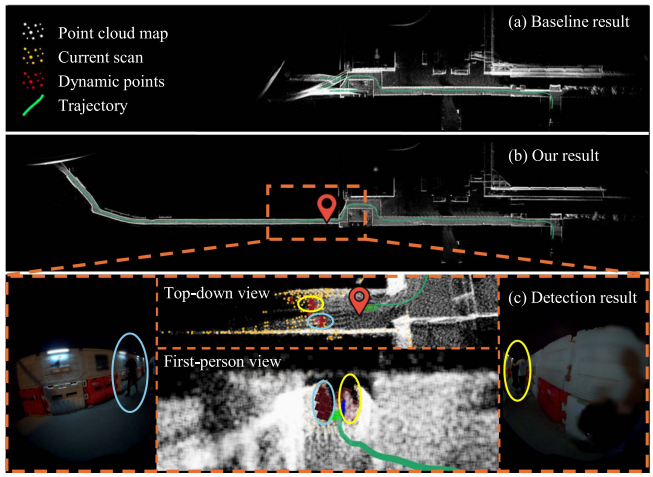

Fig. 1. Comparison of our method against FAST-LIO2 [3] for mapping and trajectory estimation. Colored ellipses highlight detected dynamic objects. Our approach reduces localization drift through effective dynamic object handling, especially in challenging geometric environments.

Existing approaches to this challenge primarily adopt a pre-processing strategy—removing dynamic objects before registration—rather than addressing the fundamental registration algorithm limitations. Geometric filtering methods [6], [7] rely on structural assumptions that restrict their applicability across diverse environments. Meanwhile, learning-based techniques [8] only detect predefined object categories rather than all dynamic points, while requiring substantial training data and showing limited generalization capabilities. These approaches create a circular dependency: accurate state estimation requires reliable static feature identification, yet effective dynamic object detection depends on precise pose information. Moreover, evaluation protocols for dynamic LIO systems have largely relied on autonomous driving datasets from geometrically-rich urban environments [9], [10]. The abundance of static features in these settings often minimizes dynamic object impact, understating the true challenges of localization in predominantly dynamic

Received 17 April 2025; accepted 12 October 2025. Date of publication 20 October 2025; date of current version 29 October 2025. This article was recommended for publication by Associate Editor M. Hanheide and Editor S. Behnke upon evaluation of the reviewers' comments. *(Corresponding author: Peng Lu.)*

Zhiqiang Chen, Fuling Lin, Minghao Lu, Qiyuan Qiao, Bowen Xu, and Peng Lu are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, SAR 999077, China (e-mail: lupeng@hku.hk).

Cedric Le Gentil is with the Institute for Aerospace Studies (UTIAS), University of Toronto , ON M3H 5T6, Canada.

Yuhua Qi is with the The School of Systems Science and Engineering, Sun Yatsen University, Guangzhou 510275, China.

The code and dataset are available at https://github.com/thisparticle/btsa.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3623436, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3623436

environments. This evaluation bias has led to an underappreciation of the dynamic localization problem's significance in practical applications.

In this paper, we address these critical limitations by proposing a novel framework that explicitly models and accounts for dynamic elements in the environment during point cloud registration. Our approach breaks the circular dependency between state estimation and dynamic object detection by integrating spatio-temporal normal analysis directly into the registration process, maintaining accurate trajectory estimation even in highly dynamic scenes with limited geometric structure. Our contributions include:

- A novel dynamic-aware ICP algorithm for robust point cloud registration in dynamic environments that addresses the circular dependency between localization and dynamic object detection
- The empirical demonstration of our method's effectiveness in truly challenging dynamic environments, including geometrically degraded scenes and scenarios dominated by dynamic objects.
- A computationally efficient spatial consistency verification approach that enhances static map construction.
- The open-source release of our code and a new dataset featuring challenging dynamic environments that current methods struggle to handle effectively.

II. Related Works

In the literature, the processes of ego-motion estimation and dynamic object detection are generally performed one after the other, and sometimes iteratively. In this section, we provide a brief overview of the different methods for Lidar-inertial state estimation and dynamic object detection.

A. Lidar-Inertial State Estimation

Lidar-inertial odometry combines lidar's precise geometric measurements with IMU's high-frequency motion data to enable robust state estimation. Early point-based approaches like LOAM [11] and its variants [12], [13] extract geometric features for real-time performance. Subsequent developments have improved computational efficiency through compact environmental representations including voxels [14] and surfels [15]. Modern registration techniques have advanced beyond simple distance metrics to incorporate probabilistic [4], semantic [16], and intensity-based [17] approaches for enhanced matching accuracy.

Despite these advances, conventional LIO systems struggle when operating in dynamic environments. While outlier rejection mechanisms and robust loss functions partially mitigate the impact of moving objects in moderately dynamic scenes, they become insufficient when a significant portion of the sensor data corresponds to dynamic elements. Recent efforts have attempted to address this limitation by integrating learning-based detection modules with odometry systems [8]. Although these approaches improve performance in dynamic scenes, they suffer from limited generalizability in highly dynamic environments. Alternative geometric detection methods [7], [18] show promise but remain constrained by specific assumptions about object geometry and require dense point cloud data, limiting their practical applicability. A critical limitation of existing approaches is their treatment of dynamic object detection as a separate processing step, creating a circular dependency between accurate pose estimation and reliable dynamic object identification. Some recent works [19], [20] attempt simultaneous odometry and dynamic object tracking by treating moving objects as landmarks. While these methods demonstrate improved accuracy in structured driving scenarios, they require smooth object trajectories and rely on deep learning detectors which limit their applicability to new environments.

B. Dynamic Object Detection

Dynamic object detection methodologies can be categorized into several distinct approaches. Offline removal methods [21], [22], [23] achieve impressive results by leveraging information from the complete accumulated map, but inherently sacrifice real-time operation capability.

Online approaches are primarily divided into learning-based and geometric-based paradigms. Learning-based methods employ neural network architectures to segment dynamic elements [24], [25], typically processing point cloud frames enhanced with temporal information. Despite their effectiveness, these approaches require extensive training datasets and often exhibit diminished performance when encountering novel object categories or sensor configurations.

Geometric-based techniques operate without labeled training data and include two primary subtypes. FreeSpace-based methods [26] maintain environmental representations through updated voxel occupancy states, while difference-based approaches [6], [27] identify dynamic elements by analyzing occlusion patterns between sequential scans. Although these methods avoid the data dependencies of learning-based approaches, they typically struggle with complex occlusion scenarios and angular incidence ambiguities. Recent spatio-temporal normal frameworks [28], [29] encode dynamic characteristics through surface geometry and motion relationships, but perform detection as post-processing after pose estimation, requiring future frames and focusing on static mapping rather than localization.

A fundamental challenge in this domain is that most existing methods presuppose accurate sensor pose information, leading to the aforementioned estimation/detection inter-dependence. Our work addresses this by integrating spatio-temporal normal analysis directly into the iterative registration loop, enabling simultaneous real-time refinement of pose estimates and dynamic point classification for the current frame.

III. Preliminaries

As this work leverages the concept of spatio-temporal normals for dynamic point detection, we derive here a rigorous definition of the spatio-temporal normals and their link to velocity beyond the original work in [28]. Let us denote a point cloud captured at time t^j as $\mathcal{P}^j = \{\mathbf{p}_i^j\}$, where each point $\mathbf{p}_i^j$ is expressed in the fixed reference frame $\mathcal{F}_W$. To incorporate temporal information, we augment each point with its acquisition time, creating a space-time representation $\tilde{\mathcal{P}}^j = \{(\mathbf{p}_i^j, t^j) | \mathbf{p}_i^j \in \mathbb{R}^3, t^j \in \mathbb{R}\}$. Throughout this paper, we use tilde notation $\tilde{()}$ to distinguish space-time elements from purely spatial ones.

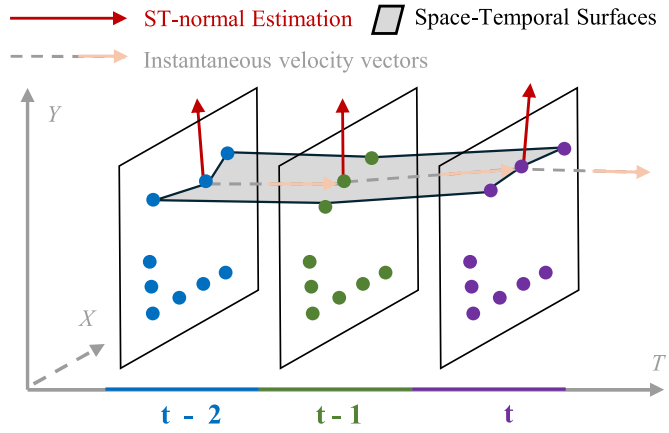

Fig. 2. Space-time velocity vectors in a 2D example. The instantaneous velocity vectors (in pink) are tangent to the space-time surfaces generated by motion, while the space-time normals (in red) are perpendicular to these surfaces. Each instantaneous velocity vector can be decomposed into a temporal component, determined by the frame rate, and a spatial component representing the underlying motion of points.

Algorithm 1: Dynamic-Aware Point Cloud Registration.

Input: Point cloud $\tilde{\mathcal{P}}$, Temporal sliding window map $\mathcal{M}_t$, Long-term voxel map $\mathcal{M}_v$, Initial pose $\hat{\mathbf{x}}^0$
Output: Optimized pose $\hat{\mathbf{x}}^\star$
$iter \leftarrow 0$, $converged \leftarrow false$
while $iter < max\_iter$ AND $\neg converged$ **do**
  Transform $\tilde{\mathcal{P}}$ to world coordinates using $\hat{\mathbf{x}}^{iter}$
  Compute spatio-temporal normal vectors $\tilde{\mathbf{n}}$ using $\mathcal{M}_t$
  Identify stable points $\tilde{\mathcal{P}}_s$ based on $\tilde{\mathbf{n}}$
  Find corresponding surfaces in $\mathcal{M}_v$ for each point in $\tilde{\mathcal{P}}_s$
  Optimize pose $\hat{\mathbf{x}}^{iter+1}$ using point-to-plane error metric
  $converged \leftarrow \|\hat{\mathbf{x}}^{iter+1} - \hat{\mathbf{x}}^{iter}\| < \epsilon$
  $iter \leftarrow iter + 1$
end while
$\hat{\mathbf{x}}^\star \leftarrow \hat{\mathbf{x}}^{iter}$
Update maps $\mathcal{M}_t$ and $\mathcal{M}_l$ with $\tilde{\mathcal{P}}$ and $\hat{\mathbf{x}}^\star$
return $\hat{\mathbf{x}}^\star$

1) Spatio-Temporal Surface and Its Normals: The spatio-temporal surface $\mathcal{S}$ is represented by the implicit function $g(x, y, z, t) = 0$ in 4D space. At any point $\tilde{\mathbf{p}}_i^j = (x_i^j, y_i^j, z_i^j, t^j)^\top$ on this surface, the normal vector equals the gradient $\nabla g = (a, b, c, d)^\top$, where components correspond to partial derivatives with respect to spatial coordinates and time. For a point moving along this surface, differentiating $g(x, y, z, t) = 0$ with respect to time yields $av_x + bv_y + cv_z + d = 0$, where (v_x, v_y, v_z) represents the point's velocity. Rearranging this equation, we obtain:

$$d = -(av_x + bv_y + cv_z). \tag{1}$$

This equation demonstrates that the temporal component d of the 4D normal vector quantifies the surface's temporal evolution, directly relating to the projection of instantaneous velocity onto the spatial gradient. As shown in Fig. 2, the velocity vector field aligning consecutive frames should be perpendicular to the surface normal field. This property enables us to identify dynamic points by comparing the temporal component d of the 4D normal with a predefined threshold.

2) Spatio-Temporal Normal Estimation: To estimate the four-dimensional surface normal $\tilde{\mathbf{n}} = (a, b, c, d)$ at point $\mathbf{p}_i^j$, we fit a tangent hyperplane to the local point cloud neighborhood $\mathcal{N}_i^j$. Concretely, $\tilde{\mathbf{n}}$ corresponds to the eigenvector associated with the smallest eigenvalue in the eigen decomposition of the covariance matrix

$$\mathrm{cov}_i^j = \frac{1}{\|\mathcal{N}_i^j\|} \sum_{\mathbf{p}_u^v \in \mathcal{N}_i^j} \left(\begin{bmatrix} \mathbf{p}_u^v \\ t_u^v \end{bmatrix} - \mathbf{m}_i^j \right) \left(\begin{bmatrix} \mathbf{p}_u^v \\ t_u^v \end{bmatrix} - \mathbf{m}_i^j \right)^\top, \tag{2}$$

where $\mathbf{m}_i^j = \frac{1}{\|\mathcal{N}_i^j\|} \sum_{\mathbf{p}_u^v \in \mathcal{N}_i^j} [\mathbf{p}_u^{v\top} \quad t_u^v]^\top$ is the neighborhood's centroid. Please note that accurate hyperplane fitting requires neighboring points to be uniformly distributed in both spatial and temporal dimensions within a sufficiently small neighborhood. While spatial uniformity is typically achievable, temporal uniformity presents challenges when processing live data, as it would require future data. As illustrated in Fig. 4(a), newly observed points can exhibit apparent non-zero velocities, potentially leading to false dynamic detections. However, such points are generally not reliable for the task of scan registration. In this work, we categorize both truly dynamic and unreliable points as "unstable points".

IV. Method

A. Overview

Let us consider an IMU and 3D LiDAR rigidly mounted. Inspired by [3], the proposed method estimates the system's pose $\mathbf{T}^j$ and velocity $\mathbf{v}^j$ at time t_j as well as the IMU biases using an Iterated Extended Kalman Filter (IEKF). We denote the system's overall state with $\mathbf{x}^j$. As illustrated in Fig. 3, the method consists of three main components: the input data pre-processing, our novel dynamic-aware point cloud registration algorithm, and the associated static map building. Based on IMU preintegration, the pre-processing step propagates the previous state estimate $\mathbf{x}^{j-1}$ to obtain a prior for $\mathbf{x}^j$ and corrects the motion distortion of the lidar point cloud. Before performing registration and static map building, both detailed in the next subsections, the point cloud is downsampled (voxel-based) to reduce the computational burden of the subsequent steps.

B. Dynamic-Aware Point Cloud Registration

Standard ICP registration consists of iteratively performing data association and geometric transformation estimation between two point clouds until convergence. Generally, ICP leverages a simple distance-based outlier rejection to omit erroneous associations. The proposed dynamic-aware point cloud registration extends the standard point-to-plane ICP to explicitly account for the presence of dynamic objects in the scene by extending the outlier rejection step based on spatio-temporal normal computation. The approach is summarized in Algorithm 1.

1) Dynamic-Aware ICP: To estimate the state $\mathbf{x}^j$ from the current scan $\tilde{\mathcal{P}}^j$, we fit local spatiotemporal planes to each point in $\tilde{\mathbf{p}}_i^j$ in $\tilde{\mathcal{P}}^j$ based on a temporal sliding window map $\mathcal{M}_t$ and the

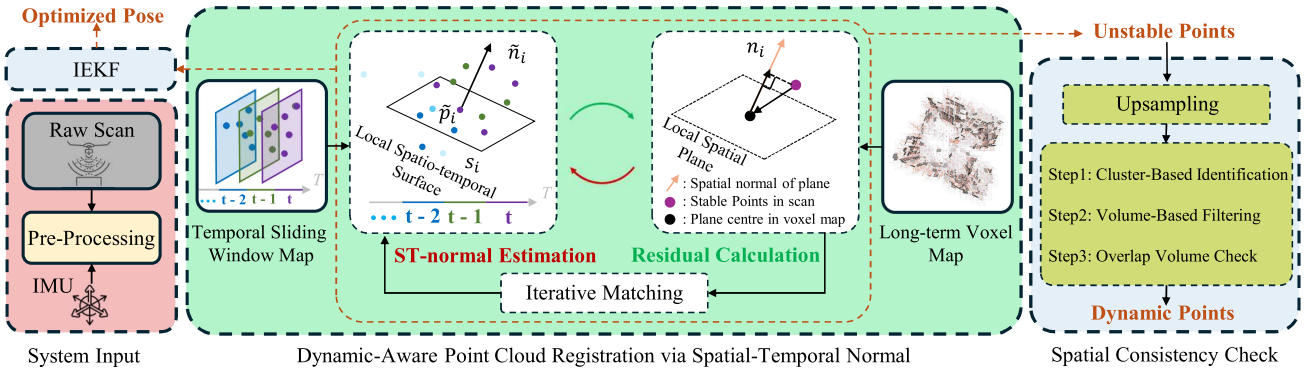

Fig. 3. An overview of the proposed framework for dynamic-aware point cloud registration and static map construction. the pre-processing step undistorts the point cloud, which is then input into the dynamic ICP loop with a temporal sliding window map. The optimized pose, eliminating dynamic elements, is derived using the proposed spatial-temporal estimation and selective update procedure. The static map is constructed by filtering out false positives from unstable points through a spatial consistency check.

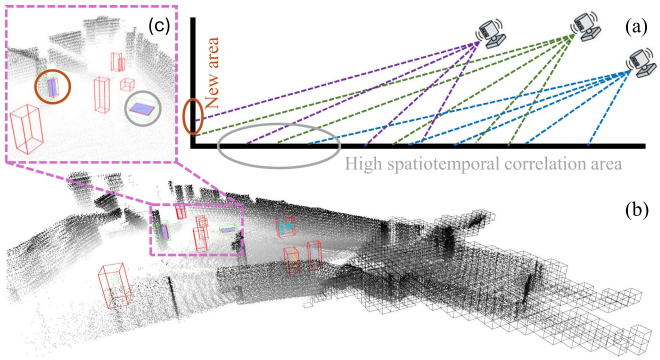

Fig. 4. Spatial consistency check. Figure (a) illustrates the causes of false positives in dynamic detection by using spatiotemporal normal vectors over a specified time window. Figure (b) provides an example demonstrating our approach to eliminating false positives based on the spatial consistency principle. Figure (c) is a zoomed-in view of Figure (b) that highlights two distinct cases of false positives.

initial guess of $\mathbf{x}^j$. The local temporal map $\mathcal{M}_t$ contains recent lidar data registered in a single reference frame. Each point in the current frame is classified as "stable" or "unstable" based on the temporal component d of the spatio-temporal normals (cf. (1)) using a predefined threshold d_{thr}. Unstable points are discarded as they correspond to a dynamic object or a newly observed area and are not reliable for pose estimation.

The stable points are then used for state estimation with respect to a global map $\mathcal{M}_v$ that only contains static elements of the environment. The state estimation corresponds to minimizing both IMU preintegration residuals and point-to-plane distances between $\mathcal{P}^j$ and $\mathcal{M}_v$:

$$\hat{\mathbf{x}}^j = \arg\min_{\mathbf{x}} \left(\|\mathbf{r}_{\text{IMU}}\|^2_{\Sigma} + \sum_i \|\mathbf{n}_i^T({}^W\mathbf{p}_i - \mathbf{q}_i)\|^2_{\mathbf{R}_i} \right), \quad (3)$$

where $\mathbf{r}_{\text{IMU}}$ denotes the IMU preintegration residuals with covariance Σ, $\mathbf{n}_i$ is the normal vector of the corresponding plane at target point $\mathbf{q}_i$ in $\mathcal{M}_v$, and $\mathbf{R}_j$ represents the LiDAR measurement covariance matrix. This process is repeated until the state estimate converges. Note that evaluating the point's dynamicity through spatio-temporal normal analysis is done at each iteration, thus jointly solving the problems of state estimation and dynamic object detection.

2) Dual-Map Architecture for Registration: As mentioned in the previous paragraph, the proposed dynamic-aware registration relies on a temporal sliding window map $\mathcal{M}_t$ and a long-term voxel map $\mathcal{M}_v$. This approach balances computational efficiency and accuracy: the local map $\mathcal{M}_t$ needs to be temporally dense for accurate spatio-temporal estimation, while the voxel-based long-term map $\mathcal{M}_v$ ensures global consistency of the pose estimate efficiently.

The temporal sliding window map ($\mathcal{M}_t$) retains approximately two seconds of recent LiDAR point cloud, providing the temporal context necessary for calculating spatio-temporal surface normal. We implement this using an iKdTree structure [30] paired with a double-ended queue. After pose estimation, each new point cloud frame is transformed to world coordinates and simultaneously added to both the queue and the k-d tree. Correspondingly, the oldest frame is removed from the queue and its points are deleted from the tree, ensuring the map remains temporally current. In parallel, we maintain a long-term voxel map ($\mathcal{M}_v$) that stores the spatial planes and their normal following the VoxelMap approach [14], which provides a more extensive spatial reference for accurate registration. This dual-map architecture effectively addresses the competing requirements of our system: temporal proximity for spatio-temporal normal calculation and spatial richness for robust localization.

3) Threshold Selection for Normal Vectors: The temporal component d in a spatio-temporal normal vector $\tilde{\mathbf{n}} = (a, b, c, d)$ indicates surface motion, with $d = 0$ for static points. We establish a threshold d_{thr} by examining the angle θ between $\tilde{\mathbf{n}}$ and its spatial projection $(a, b, c, 0)$. This angle represents the deviation into the temporal dimension caused by motion and is calculated as: $\cos\theta = \frac{a^2+b^2+c^2}{a^2+b^2+c^2+d^2} = 1 - \frac{d^2}{|\tilde{\mathbf{n}}|^2}$. This approach offers a physically interpretable threshold selection method. For example, setting $\theta_{thr} = 5.7^\circ$ yields $|d| \approx 0.1$, providing an intuitive parameter that directly relates to observable motion characteristics.

C. Static Map Building Via Spatial Consistency Check

When analyzing spatio-temporal normal vectors from the sliding time map, we identify unstable points representing both dynamic objects and false positives. Fig. 4(a) illustrates how these false positives emerge from two primary sources: newly observed regions appearing for the first time in the map, and

static areas where scan patterns change significantly between consecutive frames. In both cases, the calculation erroneously produces non-zero temporal components, misclassifying static points as dynamic. This misclassification stems from inherent LiDAR sensing limitations and data sparsity rather than actual movement in the environment.

To address false positives in our static mapping process, we implemented a spatial consistency detection approach based on a key observation: dynamic and static points exhibit distinct spatial distribution patterns—dynamic points cluster together while static points are typically surrounded by other static points. Our method first upsamples unstable points from the voxel-downsampled point cloud through nearest neighbor search to identify potentially unstable adjacent points. We then apply DBSCAN clustering [31] to effectively eliminate isolated false positives. Each resulting cluster is enclosed in a bounding box, as shown in Fig. 4(b). We discard oversized clusters as outliers and introduce a lightweight sliding voxel map, $\mathcal{M}_{scc}$, that maintains a short-term record of static areas near the sensor. By analyzing the volumetric overlap between candidate dynamic clusters and $\mathcal{M}_{scc}$, we can differentiate between genuine dynamic objects (minimal overlap) and false positives such as newly observed static regions (significant overlap), as illustrated in Fig. 4(c). This spatial consistency check substantially improves our static map's accuracy by reliably distinguishing true dynamic elements from detection artifacts.

TABLE I
DATASETS OF ALL SEQUENCES FOR EVALUATION

Scene Type	Abbreviation	Sequence Name	Path Length	Duration
GR	D_1	Promenade01	357m	278s
	D_2	Promenade02	267m	195s
	E_1	[10] - Dense_street_day_difficult_circle	2187m	803s
	E_2	[10] - Dense_street_day_medium_a	356m	120s
	E_3	[10] - Dense_street_day_medium_b	278m	125s
	U_1	[9] - UrbanNav-HK-Data20190428	1997m	487s
	U_2	[9] - UrbanNav-HK-Data20200314	1212m	300s
GD	G_1	[32] - Bridge01	3966m	382s
	G_2	[32] - Urban_tunnel03	6749m	335s
	D_3	Promenade03	276m	200s
DOD	D_4	MountainTopPark01	393m	289s
	D_5	MountainTopPark02	362m	270s
	D_6	MountainTopPark03	266m	226s

TABLE II
COMPARISON OF ABSOLUTE TRAJECTORY ERROR (RMSE, IN METERS) IN GEOMETRICALLY RICH DYNAMIC SCENES

Method	D_1	D_2	E_1	E_2	E_3	U_1	U_2
FAST-LIO2	2.15	✗	1.97	1.60	1.16	4.11	1.13
IG-LIO	1.98	✗	1.61	✗	1.18	3.91	0.92
Dynamic-LIO	-	-	4.80	1.06	1.27	4.18	1.91
TRLO	3.37	0.54	8.82	1.23	1.21	11.96	0.94
Ours	3.65	0.64	1.98	1.47	0.89	3.43	0.91

"-" indicates that the data for this sequence is not suitable for the algorithm to run. "✗" denotes that the algorithm diverges in this sequence, unable to obtain a meaningful trajectory. and represent the best and second-best results among all methods, respectively.

V. EXPERIMENTS

We evaluated our approach in challenging dynamic environments, with particular focus on scenarios containing numerous moving objects. Our experiments assessed odometry accuracy, static map construction capabilities across multiple LiDAR sensor types, and computational efficiency. All evaluations were conducted on a computer with an Intel i5-12490 CPU, 32 GB RAM, running Ubuntu 22.04.

A. Odometry

1) Baselines and Metrics: We benchmarked our method against four leading approaches: FAST-LIO2 [3] and iG-LIO [4], which represent state-of-the-art LiDAR-inertial odometry for static environments, and Dynamic-LIO [7] and TRLO [8], which address dynamic environments through different strategies. Dynamic-LIO uses geometric features to detect moving objects, while TRLO employs learning-based detection with multi-object tracking before performing localization on the filtered point cloud. For evaluation, we calculated the Root Mean Square Error (RMSE) between estimated and ground-truth trajectories after alignment using the evo package [33]. To account for algorithmic randomness, we report the mean RMSE from five independent runs per sequence.

2) Dataset: We evaluated our method using a diverse dataset comprising both public benchmarks and custom recordings, as detailed in Table I. The dataset includes sequences from ECMD [10] (E_x), UrbanNav [9] (U_x), GEODE [32] (G_x), and our custom recordings (D_x) with ground truth from RTK-GPS. The scenes of our custom dataset are illustrated in Fig. 5. These sequences were captured using various LiDAR sensors, including HDL-32E, VLP-16, and Livox Mid-360, representing both conventional spinning and solid-state scanning mechanisms. To facilitate systematic evaluation, we categorized the sequences into three types based on environmental complexity. Geometrically-Rich (GR) environments contain abundant structural features with moderate dynamic elements, where conventional methods typically perform adequately. Geometrically-Degenerate (GD) environments present fewer distinct features and occasionally underconstrained scenarios (such as featureless tunnels), while still containing moving objects. Dynamic-Object-Dominated (DOD) environments primarily consist of moving objects, presenting the greatest challenge for state estimation.

3) Geometrically Rich Benchmark: Table II shows localization performance across methods in GR environments. Most methods achieved comparable accuracy, with no clear advantage for dynamic-specific algorithms. Only in sequences D_1 and D_2, where moving objects significantly obstructed the sensor's view, did performance differences emerge.

These results highlight an important finding: in feature-rich environments, the structural elements provide sufficient constraints for accurate localization despite the presence of moving objects. Modern optimization techniques in LiDAR-inertial systems effectively treat points from dynamic objects as outliers when static features predominate. This explains why methods like iG-LIO perform well despite their static-world assumption. Consequently, evaluations limited to feature-rich environments may underestimate the benefits of dynamic object handling capabilities. A comprehensive assessment requires testing in more challenging conditions, particularly geometrically degenerate environments where dynamic objects have greater impact on localization performance, as demonstrated in the following section.

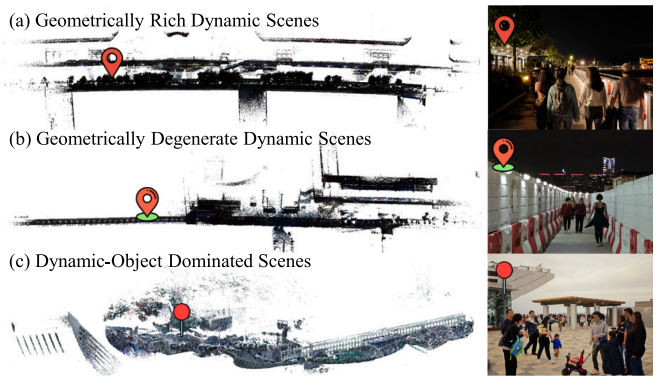

Fig. 5. Examples of 3D point cloud and scene image visualizations from self-collected dataset.

TABLE III
COMPARISON OF ABSOLUTE TRAJECTORY ERROR (RMSE, IN METERS) IN DYNAMIC-OBJECT DOMINATED SCENES

name	D_4	D_5	D_6
FAST-LIO	12.40	28.33	16.99
IG-LIO	0.74	✗	✗
Dynamic-LIO	-	-	-
TRLO	0.80	1.91	2.82
Ours	0.79	1.83	3.04

"-" indicates that the data for this sequence is not suitable for the algorithm to run. "✗" denotes that the algorithm diverges in this sequence, unable to obtain a meaningful trajectory. and represent the best and second-best results among all methods, respectively..

4) Geometrically Degenerate Benchmark: Fig. 6 (top row) presents results from GD environments. In these scenarios, moving objects significantly compromise methods that assume static scenes, as false point associations easily overwhelm the limited geometric constraints from surrounding structures. As shown in Fig. 6 (bottom row, D_3), FAST-LIO2's estimates can become immobilized while iG-LIO's trajectory gets distorted by dynamic objects.

While Dynamic-LIO and TRLO address dynamic environments, they separate object detection from point cloud registration. This creates an interdependence where accurate state estimation requires precise dynamic point segmentation and vice versa. Dynamic-LIO relies on ground segmentation and static map comparisons, but pose errors can corrupt this map, creating a cascading failure. TRLO uses frame-by-frame deep learning detection independent of registration, potentially causing inconsistent classifications across frames. Both approaches can experience trajectory drift when detection errors affect mapping.

Our approach resolves this interdependence by integrating dynamic point detection directly into registration. We filter moving objects during registration by modeling point velocity through spatiotemporal normal vectors calculated across all points within a defined time window. This avoids reliance on static map references and prevents error propagation. Our algorithm successfully estimates sensor trajectory across all sequences, consistently outperforming other methods. This demonstrates the clear advantage of our integrated approach to dynamic object handling.

TABLE IV
QUANTITATIVE COMPARISON OF DYNAMIC POINTS REMOVAL IN POINT CLOUD MAPS

Sensor \ Method		Beautymap	DUFOMap	MCDOD	Dynablox	OTD	Ours
Aeva	SA [%] ↑	69.34	91.93	61.60	**97.64**	80.74	91.50
	DA [%] ↑	79.98	79.33	94.71	64.78	43.22	**77.77**
	HA [%] ↑	74.12	84.57	74.21	75.81	54.26	**82.50**
Avia	SA [%] ↑	79.77	96.30	74.42	**95.21**	93.80	93.66
	DA [%] ↑	85.30	72.33	83.12	59.54	**76.66**	67.64
	HA [%] ↑	81.51	80.43	77.76	69.42	**83.20**	75.87
Ouster	SA [%] ↑	92.24	90.06	77.70	**97.02**	92.09	95.46
	DA [%] ↑	86.72	90.23	91.34	71.68	63.64	**72.07**
	HA [%] ↑	89.27	89.99	83.75	**82.09**	74.70	81.80
Velodyne	SA [%] ↑	76.62	85.18	46.18	**97.62**	84.56	74.15
	DA [%] ↑	84.73	13.12	85.40	31.87	45.73	**48.93**
	HA [%] ↑	80.10	21.58	59.36	47.33	**58.41**	57.79

and represent the best and second-best results among all methods, respectively. Bold and underlined text indicate the best and second-best results among online methods, respectively.

5) Dynamic-Object Dominated Benchmark: Table III shows odometry results for DOD sequences. Despite containing structural features, these environments present significant challenges as numerous moving elements frequently obscure static landmarks.

Methods lacking dynamic object handling capabilities (FAST-LIO and iG-LIO) demonstrated poorer trajectory accuracy throughout most sequences. While these approaches successfully maintained localization in the simpler D_4 scenario, they failed in the more challenging D_5 and D_6 environments. In contrast, our approach and TRLO, which explicitly account for moving elements, maintained reliable localization across all test cases. These results highlight the critical importance of robust dynamic object handling for accurate state estimation in highly dynamic settings.

B. Static Map Building

To evaluate our static map building capability in dynamic environments, we assessed our dynamic point filtering performance against several baseline methods. We compared with offline approaches (BeautyMap [22], DUFOMap [34]), a delayed incremental method [29], and online techniques (Dynablox [26], OTD [35]). For fair comparison, all methods used ground-truth poses. We conducted quantitative evaluation using the point-wise protocol from [36] on nine sequences from the Helimos dataset [37], which features highly dynamic scenes captured by four different LiDAR sensors (Avia, AEVA, VLP16, and Ouster128). Our evaluation metrics include Dynamic Accuracy (DA) and Static Accuracy (SA), measuring the recall of dynamic and static points respectively, and their harmonic mean (HA $= \frac{2\times SA\times DA}{SA+DA}$) as a balanced overall measure.

Table IV compares static map building performance across all test sequences. Our method achieves optimal online results with the Aeva sensor, nearly matching the best offline approaches. For other LiDAR types, we demonstrate the second-best online performance, showing consistent mapping capabilities across different sensors. Fig. 7 demonstrates how sensor characteristics influence performance. The effectiveness of our approach depends on point cloud density and distribution, which varies between sensor models. Sensors with sparse point distributions

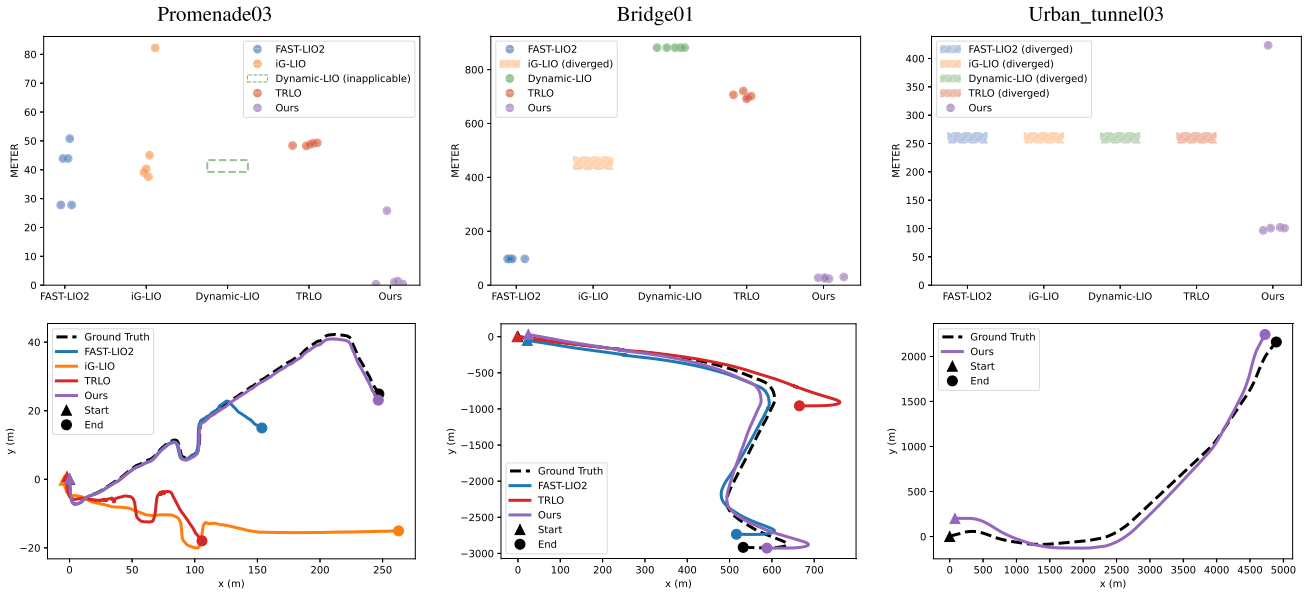

Fig. 6. Scatter plots of localization errors and corresponding trajectories in geometrically degenerate dynamic scenes. From left to right: results for sequences D_3 G_1, and G_2. The first row shows RMSE scatter plots from five experiments, with methods having an average RMSE greater than 1000 marked by dashed boxes with diagonal lines. The second row shows ground truth and algorithm trajectories, excluding those with RMSE over 800.

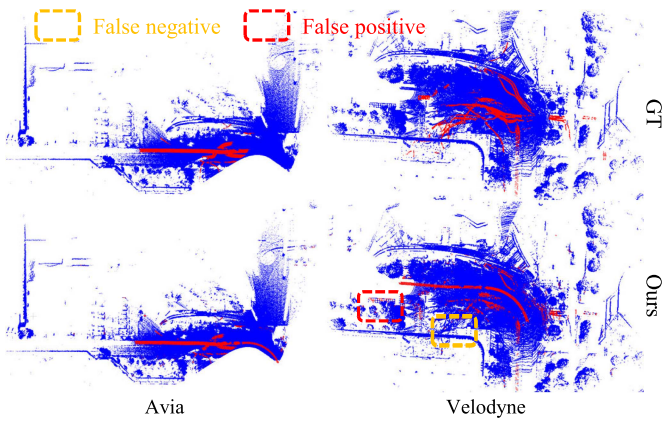

Fig. 7. Static map construction comparison across LiDAR sensors (Red: Dynamic, Blue: Static).

challenge the assumption of local uniformity needed for accurate normal vector estimation, increasing false positive rates. This explains the performance differences observed across sensor types in Table IV. While our spatial consistency check effectively reduces false positives and improves static accuracy compared to MCDOD [29], sparse dynamic points that fail to form clear clusters may remain undetected. This accounts for the lower dynamic accuracy with certain sensor configurations compared to MCDOD, highlighting a trade-off between static and dynamic point classification.

C. Ablation Experiments

To assess the contribution of each component in our unified framework for dynamic point handling and registration, we conducted an ablation study on three representative sequences. We compared the full system with two simplified variants:

TABLE V
ABLATION STUDY: RMSE COMPARISON OF DIFFERENT VARIANTS OF OUR METHOD ON REPRESENTATIVE SEQUENCES

	Ours w/o Dyn.	Ours w/ Sep. Det.	Ours w/ Full
GR(D_2)	0.79	**0.59**	0.64
GD(P_3)	35.84	0.60	**0.46**
DOD(D_4)	6.14	1.01	**0.79**

- *Baseline (No Dynamic Detection):* A simplified version that processes all points as static, equivalent to conventional LIO methods without dynamic handling.
- *Sequential Processing:* A version that first detects dynamic points using our spatiotemporal normal vector method based on initial pose estimates, then performs registration using only the filtered point cloud.

Table V shows that our full approach achieves the lowest RMSE in challenging GD and DOD scenarios. In the geometrically-rich sequence (D_2), the sequential detection variant performs slightly better, likely because abundant static features provide reliable initial pose estimates for detection. However, the substantial performance degradation when dynamic point detection is disabled demonstrates the critical importance of accounting for moving objects. Our unified approach delivers superior accuracy in geometrically degenerate and dynamic-dominated scenarios where initial pose estimates may be unreliable. These results confirm the effectiveness of integrating dynamic point handling directly within the registration pipeline, particularly for challenging environments.

D. Processing Time Analysis

We analyzed the computational efficiency of our approach using the *Promenade03* sequence. Our method achieved an average processing time of 49.69 ms per LiDAR scan, well below

the typical 100 ms scan interval, demonstrating real-time performance. The core dynamic-aware state estimation component, including all normal vector computations, required 37.38 ms, while our spatial consistency check for dynamic point filtering added only 6.08 ms of overhead. These results confirm that our approach maintains computational efficiency while handling dynamic environments effectively.

VI. Conclusion

This letter presents a dynamic-aware Lidar-Inertial Odometry framework that effectively addresses localization challenges in dynamic environments. By integrating spatio-temporal normal analysis into point cloud registration, our approach explicitly handles dynamic objects while resolving the circular dependency between state estimation and dynamic object detection that has constrained previous methods. Experimental results on both public datasets and our newly introduced dynamic environment dataset demonstrate superior performance across diverse scenarios, particularly in geometrically degraded scenes where conventional LIO systems struggle. Our spatial consistency verification method enhances map quality while maintaining computational efficiency, enabling real-time operation on standard hardware.

References

[1] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. 2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

[2] C. L. Gentil, T. Vidal-Calleja, and S. Huang, "IN2LAAMA: Inertial LiDAR localization autocalibration and mapping," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 275–290, Feb. 2021.

[3] 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, Aug. 2022.

[4] Z. Chen, Y. Xu, S. Yuan, and L. Xie, "iG-LIO: An incremental GICP-based tightly-coupled LiDAR-inertial odometry," *IEEE Robot. Automat. Lett.*, vol. 9, no. 2, pp. 1883–1890, Feb. 2024.

[5] F. Pomerleau et al., "A review of point cloud registration algorithms for mobile robotics," *Found. Trends Robot.*, vol. 4, no. 1, pp. 1–104, 2015.

[6] R. Wu, C. Pang, X. Wu, and Z. Fang, "Observation time difference: An online dynamic objects removal method for ground vehicles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 17997–18003.

[7] Z. Yuan, X. Wang, J. Wu, J. Cheng, and X. Yang, "LiDAR-inertial odometry in dynamic driving scenarios using label consistency detection," 2024, *arXiv:2407.03590*.

[8] Y. Jia, T. Wang, X. Chen, and S. Shao, "TRLO: An efficient LiDAR odometry with 3D dynamic object tracking and removal," *IEEE Trans. Instrum. Meas.*, vol. 74, pp. 1–10, 2025.

[9] L.-T. Hsu et al., "Urbannav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas," in *Proc. 34th Int. Tech. Meeting Satell. Division Inst. Navigation*, 2021, pp. 226–256.

[10] P. Chen et al., "ECMD: An event-centric multisensory driving dataset for SLAM," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 407–416, Jan. 2024.

[11] J. Zhang et al., "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst.*, 2014, vol. 2, no. 9., pp. 1–9.

[12] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[13] J. Lin and F. Zhang, "LOAM livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3126–3131.

[14] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online LiDAR odometry," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8518–8525, Jul. 2022.

[15] M. Ramezani et al., "Wildcat: Online continuous-time 3D LiDAR-inertial SLAM," 2022, *arXiv:2205.12595*.

[16] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient LiDAR-based semantic SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4530–4537.

[17] P. Pfreundschuh, H. Oleynikova, C. Cadena, R. Siegwart, and O. Andersson, "COIN-LIO: Complementary intensity-augmented LiDAR inertial odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 1730–1737.

[18] J. Lichtenfeld, K. Daun, and O. V. Stryk, "Efficient dynamic LiDAR odometry for mobile robots with structured point clouds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 10137–10144.

[19] Z. Zhu, J. Zhao, K. Huang, X. Tian, J. Lin, and C. Ye, "LIMOT: A tightly-coupled system for LiDAR-inertial odometry and multi-object tracking," *IEEE Robot. Automat. Lett.*, vol. 9, no. 7, pp. 6600–6607, Jul. 2024.

[20] Y.-K. Lin, W.-C. Lin, and C.-C. Wang, "Asynchronous state estimation of simultaneous ego-motion estimation and multiple object tracking for LiDAR-inertial odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 10616–10622.

[21] H. Lim, S. Hwang, and H. Myung, "ERASOR: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3D point cloud map building," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2272–2279, Apr. 2021.

[22] M. Jia, Q. Zhang, B. Yang, J. Wu, M. Liu, and P. Jensfelt, "Beautymap: Binary-encoded adaptable ground matrix for dynamic points removal in global maps," *IEEE Robot. Automat. Lett.*, vol. 9, no. 7, pp. 6256–6263, Jul. 2024.

[23] H. Lim et al., "ERASOR2: Instance-aware robust 3D mapping of the static world in dynamic scenes," in *Proc. Robot.: Sci. Syst.*, 2023.

[24] N. Wang, C. Shi, R. Guo, H. Lu, Z. Zheng, and X. Chen, "InsMOS: Instance-aware moving object segmentation in LiDAR data," in *Proc. 2023 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 7598–7605.

[25] B. Mersch, T. Guadagnino, X. Chen, I. Vizzo, J. Behley, and C. Stachniss, "Building volumetric beliefs for dynamic environments exploiting map-based moving object segmentation," *IEEE Robot. Automat. Lett.*, vol. 8, no. 8, pp. 5180–5187, Aug. 2023.

[26] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart, "Dynablox: Real-time detection of diverse dynamic objects in complex environments," *IEEE Robot. Automat. Lett.*, vol. 8, no. 10, pp. 6259–6266, Oct. 2023.

[27] H. Wu, Y. Li, W. Xu, F. Kong, and F. Zhang, "Moving event detection from LiDAR point streams," *Nature Commun.*, vol. 15, no. 1, 2024, Art. no. 345.

[28] R. Falque, C. Le Gentil, and F. Sukkar, "Dynamic object detection in range data using spatiotemporal normals," in *Proc. Australas. Conf. Robot. Automat.*, 2023.

[29] C. Le Gentil, R. Falque, and T. Vidal-Calleja, "Real-time truly-coupled LiDAR-inertial motion correction and spatiotemporal dynamic object detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 12565–12572.

[30] Y. Cai, W. Xu, and F. Zhang, "ikd-tree: An incremental KD tree for robotic applications," 2021, *arXiv:2102.10808*.

[31] M. Ester et al., "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 1996, vol. 96, no. 34, pp. 226–231.

[32] Z. Chen et al., "Heterogeneous LiDAR dataset for benchmarking robust localization in diverse degenerate scenarios," in *Proc. Int. J. Robot. Res.*, 2025.

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

[34] D. Duberg, Q. Zhang, M. Jia, and P. Jensfelt, "DUFOMap: Efficient dynamic awareness mapping," *IEEE Robot. Automat. Lett.*, vol. 9, no. 6, pp. 5038–5045, Jun. 2024.

[35] R. Wu, C. Pang, X. Wu, and Z. Fang, "Observation time difference: An online dynamic objects removal method for ground vehicles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 17997–18003. [Online]. Available: https://api.semanticscholar.org/CorpusID:270703199

[36] Q. Zhang, D. Duberg, R. Geng, M. Jia, L. Wang, and P. Jensfelt, "A dynamic points removal benchmark in point cloud maps," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst.*, 2023, pp. 608–614.

[37] H. Lim, S. Jang, B. Mersch, J. Behley, H. Myung, and C. Stachniss, "HeLiMOS: A dataset for moving object segmentation in 3D point clouds from heterogeneous LiDAR sensors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 14087–14094.