



ARTICLE

DGMSE: A Real-Time Dynamic Object Removal Framework Based on Detection-Guided Markov State Estimation

Jiahua Kou¹, Chengbo Guo^{1,*}, Weiyue Xing¹, Zheng Yang¹, Jiaxuan Cao¹, Shufa Sun¹ and Yanling Guo²

¹School of Civil Engineering and Transportation, Northeast Forestry University, Harbin, China

²School of Mechatronics Engineering, Northeast Forestry University, Harbin, China

*Corresponding Author: Chengbo Guo. Email: chengbo.guo@nefu.edu.cn

Received: 02 March 2026; Accepted: 26 May 2026; Published: 23 July 2026

ABSTRACT: Dynamic objects in LiDAR SLAM often introduce ghosting artifacts that degrade map quality. While offline methods can successfully clean these maps, they lack real-time capabilities. Conversely, online methods often suffer from state oscillation (where moving objects are misclassified as static when they temporarily stop) and incomplete point cloud removal. To address these challenges, we propose DGMSE, a real-time framework for removing dynamic point clouds in complex urban environments. Our approach consists of three sequential steps. First, the PointPillars 3D detection network quickly isolates potential dynamic objects, significantly reducing computational overhead. Second, to mitigate state oscillation, we introduce a five-state Markov chain to continuously estimate the probabilistic motion state of objects, which is not limited to simple moving-or-static classifications. This temporal memory prevents misclassification during sudden stops or motion transitions. Finally, instead of simply deleting all points inside a detection box, a density-adaptive clustering approach evaluates the geometric and motion characteristics of points, precisely segmenting dynamic objects from the static background without damaging permanent structures. Evaluated on the SemanticKITTI dataset, DGMSE achieves an F1-score of 0.95–0.99, achieving performance competitive with offline methods while satisfying real-time processing requirements.

KEYWORDS: LiDAR SLAM; dynamic object removal; deep learning; state estimation

1 Introduction

Accurate 3D point cloud maps are fundamental for autonomous driving, robotic navigation, and mobile mapping applications [1]. These maps provide the geometric foundation for reliable localization and path planning. Furthermore, they serve as critical data sources for High-Definition (HD) map construction and 3D scene understanding. Generally, Simultaneous Localization and Mapping (SLAM) technologies construct these maps by accumulating LiDAR point cloud data frame-by-frame [2,3].

However, a critical challenge arises in real-world deployment. Urban environments frequently violate the static world assumption used by most mapping algorithms. Dynamic entities, such as moving vehicles and pedestrians, introduce significant noise into the mapping process. Although advanced LiDAR-inertial odometry methods (e.g., RF-LIO [4], Faster-LIO [5], T-LOAM [6]) have improved pose estimation robustness, map construction in dynamic environments remains problematic.

The fundamental mechanism underlying this phenomenon is a conflict between the sequential scanning nature of LiDAR sensors and the strict static-world assumption inherent in traditional SLAM algorithms.

When dynamic entities (e.g., vehicles) move through the scene, the rotating LiDAR beams capture their surfaces at different times and spatial locations. Since standard SLAM front-ends assume all observations represent stationary structures, they erroneously register these temporally displaced point clouds into a single global coordinate frame, treating motion as if it were geometric structure.

As illustrated in Fig. 1a, this forced registration stretches a single moving vehicle into spurious geometric structures—the so-called “ghost traces” (shown in red). The negative impacts of these artifacts manifest in three critical dimensions. First, they degrade localization accuracy by introducing false geometric features that interfere with scan-matching and loop closure detection. Second, they compromise map fidelity by creating false obstacles that do not correspond to any real-world structure. Third, they threaten navigation safety, as path planning algorithms may treat these fake obstacles as real dangers, leading to unnecessary path changes or navigation failures. In contrast, Fig. 1b demonstrates that effective dynamic object removal produces a clean, artifact-free static map suitable for reliable navigation. Therefore, identifying and removing dynamic objects is a prerequisite for robust SLAM in real-world environments.

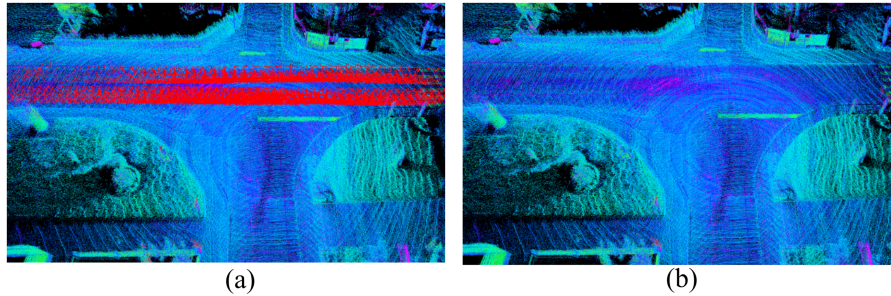


Figure 1: Dynamic object removal results comparison on the SemanticKITTI sequence 05. (a) Original point cloud map. (b) Static map after DGMSE processing.

To address this challenge, existing dynamic object removal approaches generally fall into two groups: offline and online processing [7,8]. Offline methods first construct a complete map. Then, they perform batch post-processing to remove dynamic traces. While these methods achieve high mapping accuracy through global consistency analysis, they are computationally expensive. Consequently, they cannot support real-time robotic applications. Conversely, online approaches use local temporal information from consecutive frames to remove dynamic points in real-time. However, existing online methods often suffer from state oscillation and incomplete removal. Objects may oscillate between dynamic and static classifications due to inadequate temporal modeling. For instance, when a vehicle waits at a traffic light or a pedestrian momentarily pauses, their instantaneous velocity drops to zero. Conventional methods lacking temporal memory often misclassify temporarily stationary entities as ‘static’. Consequently, this error registers their geometry into the map as erroneous artifacts when they move. Furthermore, relying on fixed threshold strategies and simple geometric constraints results in incomplete removal of dynamic points, particularly at object boundaries and in spatially sparse regions.

To overcome these limitations, we propose DGMSE (Detection-Guided Markov State Estimation). This is a real-time dynamic object removal framework designed for robust LiDAR SLAM. Unlike traditional methods relying on instantaneous geometric checks, our approach integrates 3D object detection with a probabilistic temporal model. First, DGMSE employs PointPillars for 3D object detection. This focuses processing resources on potential dynamic regions. We then introduce a five-state Markov chain model to estimate motion states. It utilizes observation-driven probability adjustments to ensure spatio-temporal consistency and mitigate state oscillations. Finally, a detection-guided multi-strategy approach combining

density-adaptive DBSCAN clustering and spherical region analysis refines the segmentation. This precisely removes dynamic points while preserving static structural features.

The main contributions of this work are summarized as follows:

1. We develop a systematic five-state Markov chain framework. It uniquely integrates velocity and detection confidence through observation-driven probabilistic adjustment. This design effectively mitigates state oscillations in dynamic SLAM scenarios.
2. We propose a detection-guided multi-strategy removal approach. It combines spatial constraints, density-adaptive clustering, and spherical region analysis. This combination addresses incomplete removal issues while achieving precise dynamic object removal and maintaining real-time performance.
3. Extensive experiments on the SemanticKITTI dataset demonstrate the effectiveness of DGMSE. It achieves high removal accuracy (F1-score: 0.95–0.99), competitive with offline methods, while satisfying real-time processing requirements. This highlights its practical value for both autonomous navigation reliability and high-fidelity mapping.

2 Related Work

Accurately identifying and removing dynamic points in point cloud data is essential for robust SLAM in dynamic environments. Various methods have been proposed in recent years to address this challenge [9], which can be broadly categorized into online and offline methods.

2.1 Online Methods

Online methods detect and remove dynamic points in real-time during the SLAM process, making them suitable for deployment in autonomous systems. However, two fundamental challenges persist across many online methods. First, without sufficient temporal memory, objects that temporarily stop moving are misclassified as static and erroneously registered into the map, causing the so-called state oscillation problem. Second, existing methods often lack precise spatial filtering within detection regions, leading to incomplete removal at object boundaries and in sparse point cloud areas. These two challenges are the core problems that DGMSE is specifically designed to address.

Geometry-driven approaches identify dynamic regions by analyzing spatial inconsistencies between consecutive frames, without relying on semantic priors. Early LiDAR SLAM systems such as LOAM [2], RF-LIO [4], and T-LOAM [6] treat dynamic points as outliers and suppress them through robust fitting or truncated least-squares optimization during scan-matching. More dedicated removal frameworks have since emerged: Yoon et al. [10] query freespace via motion-compensated ray casting to classify dynamic points online; DynamicFilter [11] detects dynamic points through two-stage visibility comparison against a local submap; OTD [12] judges voxel dynamics by comparing observation time differences between non-ground voxels and the ground beneath them; DO-Removal [13] uses ground fitting as a reference plane and applies region growing from geometric seed points for front-end real-time separation; Park et al. [14] employ nonparametric background modeling to distinguish dynamic foreground from static background in highly dynamic urban environments; Wu et al. [15] detect moving events directly from LiDAR point streams using spatiotemporal analysis; Huang et al. [16] propose dynamic object-aware LiDAR odometry with joint weightings estimation for urban areas; Pagad et al. [17] develop a robust method for removing dynamic objects from point clouds using spatial-temporal consistency checks; and Sun et al. [18] propose efficient spatial-temporal information fusion for LiDAR-based 3D moving object segmentation. Despite their diversity, these methods share a critical limitation: they typically rely on instantaneous spatial or temporal cues and carry limited long-term memory of past states. When a dynamic object temporarily stops, as in

urban stop-and-go traffic, the geometric signal vanishes, causing the object to be classified as static and integrated into the map as a ghost artifact.

To address the ambiguity inherent in purely geometric analysis, semantic-aware methods incorporate class-level priors to filter potentially movable objects. SuMa++ [19] fuses surfel-based LiDAR mapping with pixel-wise semantic labels to suppress dynamic classes during map construction. RangeNet++ [20] achieves real-time LiDAR point cloud semantic segmentation via range-image CNNs, enabling downstream dynamic filtering. SANet [21] combines semantic constraints with scan-matching to reject dynamic elements in LiDAR SLAM. Wang et al. [22] further push semantic segmentation onto edge devices, demonstrating real-time inference for unmanned systems. Integrated approaches [23] combine point-cloud segmentation with 3D scan-matching for mobile robot mapping. While effective at recognizing movable categories, these methods are constrained by the closed-world assumption of semantic models and incur significant computational overhead. More critically, they typically lack a probabilistic temporal dimension and precise spatial filtering: a semantically dynamic object that is currently stationary will still be removed, while boundary points outside the semantic mask are frequently missed.

Building upon both geometric and semantic foundations, detection-driven methods leverage 3D object detectors to obtain instance-level bounding boxes, enabling more precise spatial constraints. DL-SLOT [24] and TRLO [25] tightly couple object tracking with LiDAR-inertial odometry, filtering feature points on tracked dynamic objects before scan-matching. LIMOT [26] extends this approach with trajectory-based dynamic feature filtering within a sliding-window factor graph, jointly optimizing ego-vehicle and object poses. DOR-LINS [27] integrates ground pseudo-occupancy with beam tracing to verify dynamic candidates within detection regions. Detection-based dynamic SLAM systems [28–30] have also explored combining detection with geometric or probabilistic cues for precise removal. Despite these advances, two critical gaps remain. First, these methods predominantly rely on hard velocity thresholds or binary confidence scores to determine motion states, struggling to model the uncertainty inherent in transitional motions such as deceleration and stop-and-go scenarios. Second, after obtaining bounding boxes, most methods either delete all points within the box or apply fixed-radius clustering, which is often too coarse for precise boundary removal: static background points inside the box may be incorrectly deleted, while dynamic points near the box boundary in sparse regions might be missed. These two limitations are precisely what DGMSE addresses: temporal uncertainty is modeled through a five-state Markov chain, while density-adaptive spatial filtering is achieved through detection-guided clustering with spherical region analysis.

2.2 Offline Methods

In contrast to online approaches, offline methods process complete point cloud sequences through postprocessing operations, achieving superior accuracy through global consistency analysis but requiring complete trajectory information.

Ray tracing-based methods calculate occupancy probability by analyzing whether voxels are penetrated by LiDAR beams. OctoMap [31] provides an efficient probabilistic 3D mapping framework based on octrees. C-blox [32] extends this with a scalable TSDF-based dense mapping approach. Vizzo et al. [33] apply Poisson surface reconstruction to generate smooth continuous surfaces, while Kühner and Kümmele [34] propose large-scale volumetric scene reconstruction through efficient spatial indexing. Schauer and Nüchter [35] traverse voxel occupancy grids to identify dynamic objects. While achieving high accuracy, these methods face substantial computational overhead due to exhaustive voxel checking.

To reduce computational cost, visibility-based methods determine dynamic traces by checking occlusion relationships. Pomerleau et al. [36] use normal and incidence angles for long-term 3D map maintenance. Removert [8] adopts a remove-then-revert strategy with multi-resolution range images. However, when the

incidence angle is large, distant ground points may be mistakenly identified as dynamic points. Xiao et al. [37] conduct change detection between scans based on occupancy grids.

Additionally, statistical occupancy methods exploit temporal patterns across complete sequences. ERASOR [7] implements egocentric pseudo-occupancy ratios based on the assumption that dynamic objects are mostly in contact with the ground. ERASOR2 [38] extends this with instance segmentation but requires 200 ms/frame, unsuitable for real-time processing. BeautyMap [39] employs binary-encoded ground matrix for efficient storage. Du et al. [40] propose convex hull triangle mesh-based static mapping with multi-level voting strategies. Arora et al. [41] generate static maps by exploiting ground segmentation, while DUFOMap [42] employs global voxel traversal optimization.

Despite achieving superior accuracy, offline methods face three fundamental limitations: substantial computational resources and processing time, inability to prevent initial integration errors during mapping (creating a chicken-and-egg problem), and lack of real-time capability for autonomous systems. These limitations motivate the development of online methods that can detect and remove dynamic objects in real-time during the SLAM process.

2.3 Summary

Existing dynamic object removal methods struggle to balance accuracy and real-time performance. Online methods achieve real-time processing but often face two fundamental challenges. First, geometry-driven and semantic-aware methods rely on instantaneous cues without sufficient temporal memory, making them susceptible to state oscillation when objects temporarily stop moving. Second, most existing online methods lack precise spatial filtering within detection regions, leading to incomplete removal at object boundaries. Detection-driven approaches like DL-SLOT [24], TRLO [25], and LIMOT [26] represent promising directions by integrating object detection with SLAM optimization, sharing similar motivations with our work. However, they still rely on fixed velocity thresholds for determining dynamic and static states, often lacking systematic uncertainty modeling and adaptive parameter adjustment mechanisms. Offline methods provide superior accuracy through global consistency analysis but face substantial computational overhead, inability to prevent initial integration errors during mapping, and lack of real-time capability for autonomous deployment.

To address these limitations, this work proposes DGMSE, which combines Markov chain temporal modeling with detection-guided adaptive clustering to achieve robust real-time dynamic object removal while maintaining high accuracy. Specifically, DGMSE addresses state oscillation through a five-state Markov chain with observation-driven probability adjustment, and tackles incomplete boundary removal through detection-guided density-adaptive clustering with spherical region analysis.

3 Methods

3.1 DGMSE System Overview

As illustrated in Fig. 2, DGMSE employs a modular architecture consisting of three core components that sequentially process raw LiDAR point clouds into filtered static maps. The system addresses two fundamental challenges in dynamic object removal: state oscillation through temporal Markov modeling and incomplete removal through detection-guided adaptive clustering with spherical region analysis.

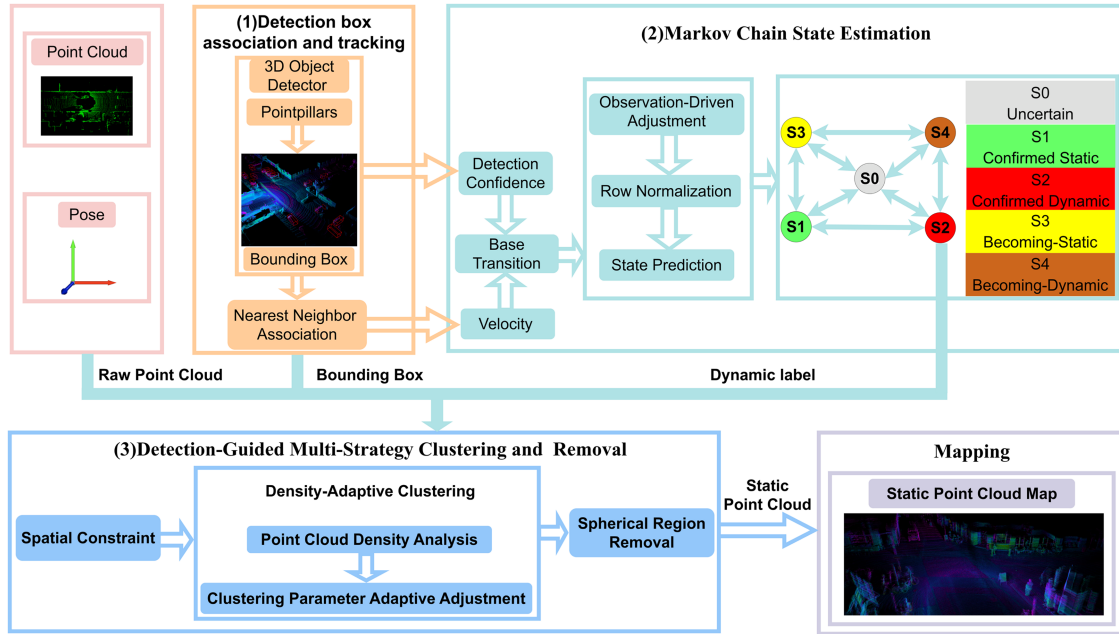


Figure 2: System architecture of DGMSE. The DGMSE framework comprises three core modules: (1) Detection and tracking with nearest neighbor association and coordinate transformation, (2) Markov chain state estimation with observation-driven adjustment, and (3) Detection-guided multi-strategy clustering and removal with three progressive stages, integrated with FAST-LIO2 for mapping.

The data flow proceeds through three sequential stages, with each module transforming its inputs to produce specific outputs for the next stage. In the first module for 3D object detection and tracking, the system takes raw point cloud P'_t as input and employs TensorRT-accelerated PointPillars [43] to perform real-time 3D object detection and generate bounding boxes $\{B_{t,m}\}$ with position, size, and confidence information. Nearest neighbor association maintains temporal consistency across frames, while detection results are transformed from LiDAR to world coordinates using SLAM poses T_t . This module outputs tracked bounding boxes $\{B_{t,m}\}$ with world-frame positions $\{p_{t,m}^w\}$ and velocities $\{v_{t,m}\}$.

The second module for Markov chain state estimation receives bounding boxes $\{B_{t,m}\}$, positions $\{p_{t,m}^w\}$, and velocities $\{v_{t,m}\}$ from the first module. This module employs a five-state probabilistic framework (S_0 : Uncertain, S_1 : Static, S_2 : Dynamic, S_3 : Becoming Static, S_4 : Becoming Dynamic) to estimate object motion states by combining velocity and confidence observations. The state estimation process consists of three sequential steps: observation-driven probability adjustment that adapts transition matrices based on current observations, row normalization that ensures valid probability distributions, and state prediction through maximum likelihood selection. This design provides temporal memory to suppress state oscillation. The output consists of motion state labels $\{S_{t,m}\}$ classifying each object as dynamic or static.

The third module for detection-guided multi-strategy removal takes as input the raw point cloud P'_t , bounding boxes $\{B_{t,m}\}$, and state labels $\{S_{t,m}\}$ from the previous modules. This module applies three progressive strategies for precise dynamic point removal. Spatial constraints extract candidates within detection bounding boxes, density-adaptive DBSCAN clustering segments objects with quality validation, and spherical region analysis removes dynamic points through multi-criterion consistency evaluation. The final output is a filtered static point cloud P_t for the SLAM backend.

The filtered static point clouds are processed by FAST-LIO2 [44] for precise mapping in dynamic environments. This modular design enables seamless integration with existing SLAM systems while maintaining real-time performance. Given consecutive raw LiDAR scans P'_t and P'_{t-1} at time t , the problem formulation addresses two key objectives: first, to identify and remove dynamic points to obtain clean static point clouds P_t and P_{t-1} , and second, to enable accurate state estimation and mapping by providing filtered static points to the SLAM backend. This formulation addresses the fundamental challenge of dynamic object removal while preserving valuable static information for robust localization.

Each component is detailed in the following subsections.

3.2 Detection Box Association and Tracking

3.2.1 Notation

We define the following notation used throughout the methodology section. Let $\mathcal{W}$, $\mathcal{L}$, and $\mathcal{B}$ denote the world, LiDAR, and robot body coordinate frames, respectively, where $\mathcal{L}$ and $\mathcal{B}$ are assumed to coincide. At time t , P'_t represents the raw point cloud, P_t represents the static point cloud after dynamic removal, and $\mathbf{T}_t = [\mathbf{R}_t | \mathbf{t}_t] \in SE(3)$ represents the robot pose with rotation $\mathbf{R}_t \in SO(3)$ and translation $\mathbf{t}_t \in \mathbb{R}^3$.

For detected objects, $B_{t,m}$ denotes the bounding box for the m -th object, where $m \in \{1, 2, \dots, M_t\}$ represents the index of detected objects at time t . The set $\mathcal{K}_t$ represents all tracked objects at time t .

3.2.2 Coordinate Transformation

PointPillars generates detection results in LiDAR frame $\mathcal{L}$, which are then transformed to world frame $\mathcal{W}$ using SLAM poses $\mathbf{T}_t = [\mathbf{R}_t | \mathbf{t}_t] \in SE(3)$ to facilitate temporal analysis across frames. The object center position in world coordinates is:

$$\mathbf{p}_{t,m}^w = \mathbf{R}_t \mathbf{p}_{t,m}^L + \mathbf{t}_t \quad (1)$$

where $\mathbf{p}_{t,m}^L$ denotes the center of the m -th bounding box in LiDAR coordinates.

3.2.3 Temporal Data Association

Objects are tracked across frames using nearest neighbor matching. For a detection at time t and tracked objects from time $t-1$, the association is determined by:

$$m^* = \arg \min_{k \in \mathcal{K}_{t-1}} \|\mathbf{p}_{t,m}^w - \mathbf{p}_{t-1,k}^w\|_2 \quad (2)$$

subject to the gating constraint $\|\mathbf{p}_{t,m}^w - \mathbf{p}_{t-1,k}^w\|_2 < d_{\text{th}}$. Category-specific distance thresholds are defined as:

$$d_{\text{th}} = \begin{cases} 3.0 \text{ m}, & \text{if } c = \text{vehicle} \\ 1.5 \text{ m}, & \text{if } c = \text{pedestrian} \\ 2.0 \text{ m}, & \text{if } c = \text{cyclist} \end{cases} \quad (3)$$

Upon successful association, the object velocity is estimated through first-order finite difference approximation:

$$\mathbf{v}_{t,m} = \frac{\mathbf{p}_{t,m}^w - \mathbf{p}_{t-1,m^*}^w}{\Delta t} \quad (4)$$

where Δt is the time interval between consecutive frames, and m^* denotes the matched track index from the previous frame. When a detection cannot be associated with any existing track, a new trajectory is initialized with state S_0 (Uncertain) and zero initial velocity.

3.3 Markov Chain State Estimation Model

3.3.1 Theoretical Foundation

The Markov chain module is designed around a specific failure mode of existing online methods: state oscillation. When a moving pedestrian temporarily stops (e.g., to check traffic before crossing a street), binary dynamic/static classifiers immediately switch the object's label to static, causing its points to be retained in the map. When the pedestrian resumes walking, the label reverts to dynamic. This oscillation produces intermittent ghost traces that neither accumulate nor disappear cleanly. The root cause is that binary classifiers have no memory of past states and treat each frame independently.

To suppress this behaviour, the five-state design introduces two transitional states (S_3 : Becoming Static, S_4 : Becoming Dynamic) that act as temporal buffers. An object must accumulate consistent evidence across multiple frames before transitioning to a confirmed state (S_1 or S_2), preventing single-frame velocity fluctuations from causing immediate label changes. The observation-driven adjustment further modulates transition probabilities using both velocity magnitude and detection confidence, so that a low-confidence detection weakens the evidence for any confirmed state rather than being treated as a hard observation.

To address the state oscillation problem in traditional binary classification approaches, we model each tracked object using a five-state Markov chain framework. For each object m , observations are derived from detection and tracking:

$$\mathbf{O}_{m,t} = \{v_t, c_t\} \quad (5)$$

where v_t is the velocity magnitude in world coordinates (derived from Eq. 4) and c_t is the detection confidence score.

The discrete state space captures typical motion transition patterns observed in dynamic environments:

$$\mathcal{S} = \{S_0, S_1, S_2, S_3, S_4\} \quad (6)$$

where S_0 represents Uncertain, S_1 represents Static, S_2 represents Dynamic, S_3 represents Becoming Static, and S_4 represents Becoming Dynamic. We maintain a velocity history window $\mathcal{W}_t$ of length $n = 3$ to compute mean velocity $\bar{v}_t$ and average confidence $\bar{c}_t$. The coefficient of variation is calculated as:

$$CV_t = \frac{\sigma_{v,t}}{\bar{v}_t + \epsilon} \quad (7)$$

where a small constant $\epsilon = 0.1$ is added for numerical stability. The velocity trend is defined as:

$$\Delta v_t = \bar{v}_t - \bar{v}_{t-1} \quad (8)$$

An initial observed state $\tilde{S}_t$ is determined through a heuristic mapping function based on current observations:

$$\tilde{S}_t = \begin{cases} S_0, & \text{if } |\mathcal{W}_t| < 3 \text{ or } \bar{c}_t < 0.5 \text{ or } CV_t \geq 0.3 \\ S_1, & \text{if } \bar{v}_t < v_{\text{static}}^{(c)} \text{ and } CV_t < 0.3 \\ S_2, & \text{if } \bar{v}_t > v_{\text{dynamic}}^{(c)} \text{ and } CV_t < 0.3 \\ S_3, & \text{if } v_{\text{static}}^{(c)} \leq \bar{v}_t \leq v_{\text{dynamic}}^{(c)} \text{ and } \Delta v_t < 0 \\ S_4, & \text{if } v_{\text{static}}^{(c)} \leq \bar{v}_t \leq v_{\text{dynamic}}^{(c)} \text{ and } \Delta v_t \geq 0 \end{cases} \quad (9)$$

where c represents the object category. Note that $\tilde{S}_t$ is primarily used for initializing the state of newly tracked objects when they first appear ($t = 0$ for that track). For existing tracks with sufficient history, the state is updated through the Markov chain transition process described in the following subsections, which directly utilizes the raw observations v_t and c_t to adjust transition probabilities while maintaining temporal smoothness. Transitional states S_3 and S_4 serve as temporal buffers to prevent abrupt switching between dynamic and static classifications. Category-specific velocity thresholds are defined in [Table 1](#).

Table 1: Category-specific parameter configurations.

Type	v_{static} (m/s)	v_{dynamic} (m/s)	v_{th} (m/s)	α_c	c_{th}
Vehicle	0.5	2.0	1.25	0.3	0.8
Pedestrian	0.2	0.8	0.50	0.15	0.8
Cyclist	0.3	1.5	0.90	0.25	0.8

3.3.2 Probability Transition Matrix

The five-state space $\mathcal{S} = \{S_0, S_1, S_2, S_3, S_4\}$ is designed to form a minimal complete cover of the observable motion regimes of dynamic objects in urban driving. In urban traffic, a tracked object can be in one of the following distinguishable conditions: (1) detection is unreliable or history is insufficient to make a judgment (S_0 : Uncertain); (2) the object has been consistently stationary over multiple frames (S_1 : Static); (3) the object has been consistently moving over multiple frames (S_2 : Dynamic); (4) the object is decelerating from motion toward a stop (S_3 : Becoming Static); (5) the object is accelerating from a stop toward motion (S_4 : Becoming Dynamic). These five conditions are mutually exclusive and collectively exhaustive for the purpose of dynamic point removal: any additional states would represent finer subdivisions within these regimes without changing the removal decision, while reducing the number of states below five collapses the critical distinction between confirmed and transitional motion, which is precisely the cause of state oscillation in binary classifiers. The state space design is therefore both necessary and sufficient for the task.

The base transition matrix $P^{(0)}$ is not manually specified but estimated from data through a statistical learning procedure. Each element $P_{ij}^{(0)} = P(S_{t+1} = S_j \mid S_t = S_i)$ is estimated as the empirical transition frequency:

$$P_{ij}^{(0)} = \frac{\sum_k \mathbf{1}[S_t^{(k)} = S_i, S_{t+1}^{(k)} = S_j]}{\sum_k \mathbf{1}[S_t^{(k)} = S_i]} \quad (10)$$

where the summation is over all consecutive frame pairs $(t, t + 1)$ across all tracked objects k in the training sequences. Ground-truth state labels $S_t^{(k)}$ are derived from SemanticKITTI annotations mapped to the five-state space using the velocity thresholds in [Table 1](#). This procedure yields a data-driven maximum likelihood estimate of the transition probabilities under the first-order Markov assumption, rather than a hand-crafted specification.

To ensure unbiased evaluation and avoid data leakage, $P^{(0)}$ and density thresholds (Section 3.4.2) are estimated from KITTI Raw sequences. Note that SemanticKITTI is derived from a subset of KITTI Raw; the sequences used for parameter estimation are therefore strictly excluded from all sequences used for evaluation and validation in this study, including SemanticKITTI sequences 00–10 and all other KITTI Raw sequences referenced elsewhere in this paper. The matrix captures general motion patterns (e.g., moving $\rightarrow$ stopping, stopping $\rightarrow$ moving) across diverse scenarios (city streets, residential areas, highways), ensuring generalization rather than overfitting to the test distribution.

By applying the empirical estimation in Eq. (10), we obtain the following 5×5 base transition matrix $P^{(0)}$:

$$P^{(0)} = \begin{bmatrix} 0.10 & 0.40 & 0.30 & 0.10 & 0.10 \\ 0.05 & 0.80 & 0.02 & 0.10 & 0.03 \\ 0.02 & 0.03 & 0.80 & 0.05 & 0.10 \\ 0.10 & 0.60 & 0.05 & 0.20 & 0.05 \\ 0.05 & 0.05 & 0.60 & 0.10 & 0.20 \end{bmatrix} \quad (11)$$

High diagonal values (0.80) in confirmed states reflect motion persistence, while transitional states (S_3 , S_4) exhibit higher mobility (0.60) toward confirmed states as designed.

3.3.3 Observation-Driven Probability Adjustment

The observed state $\tilde{S}_t$ (Eq. (9)) provides an initial estimate based on current measurements. However, to achieve temporal consistency and suppress state oscillation, we integrate this observation with the previous state S_t through a Markov chain transition process. The transition probabilities are dynamically adjusted based on velocity and confidence observations from $\mathbf{O}_{m,t}$. The adjusted transition probability is:

$$P_{ij}^{\text{adj}} = P_{ij}^{(0)} \cdot \beta_{ij}(v_t, c_t) \quad (12)$$

where the bounded adjustment factor is defined as:

$$\beta_{ij}(v_t, c_t) = \max(0.1, \min(3.0, 1 + \omega_v \phi_v(i, j, v_t) + \omega_c \phi_c(i, j, c_t))) \quad (13)$$

Bounds [0.1, 3.0] ensure numerical stability while allowing sufficient adjustment range. Empirically validated weights $\omega_v = 0.6$ and $\omega_c = 0.4$ balance velocity and confidence contributions. The velocity adjustment component is:

$$\phi_v(S_t, j, v_t) = \begin{cases} +0.5, & \text{if } v_t > 0.8v_{\text{th}} \text{ and } j \in \{S_2, S_4\} \\ +0.5, & \text{if } v_t < 0.2v_{\text{th}} \text{ and } j \in \{S_1, S_3\} \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

The velocity adjustment mechanism in Eq. (14) implements a threshold-based strategy to enhance state transition accuracy. When $v_t > 0.8v_{\text{th}}$, the adjustment factor increases probabilities toward dynamic states (S_2, S_4) by +0.5, accelerating response to moving objects. When $v_t < 0.2v_{\text{th}}$, it similarly increases probabilities toward static states (S_1, S_3). In the intermediate range ($0.2v_{\text{th}} \leq v_t \leq 0.8v_{\text{th}}$), no adjustment is applied ($\phi_v = 0$), preventing premature state switching during ambiguous transitional phases.

The velocity threshold v_{th} serves as the critical discriminator between static and dynamic motion regimes. It is defined as the category-specific midpoint between static and dynamic boundaries:

$$v_{th} = \frac{v_{static}^{(c)} + v_{dynamic}^{(c)}}{2} \quad (15)$$

The confidence adjustment component is:

$$\phi_c(S_t, j, c_t) = \begin{cases} +\alpha_c, & \text{if } c_t > c_{th} \text{ and } j \in \{S_1, S_2\} \\ -\alpha_c/2, & \text{if } c_t < c_{th} \text{ and } j \in \{S_1, S_2\} \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

where $\alpha_c = 0.3$ is the confidence adjustment strength and $c_{th} = 0.8$ is the confidence threshold derived from precision-recall analysis on validation data. High confidence detections ($c_t > 0.8$) increase transition probabilities to confirmed states (S_1, S_2) by $\alpha_c = 0.3$, while low confidence detections reduce these probabilities by $\alpha_c/2 = 0.15$ to account for detection uncertainty.

We renormalize the probabilities after adjustment to ensure validity:

$$P_{ij}^{final} = \frac{P_{ij}^{adj}}{\sum_{k=0}^4 P_{ik}^{adj}} \quad (17)$$

Category-specific parameters accommodate different motion characteristics, as detailed in [Table 1](#).

3.3.4 Convergence Analysis

The Markov chain $P^{(0)}$ is aperiodic and irreducible, guaranteeing a unique stationary distribution π satisfying:

$$\pi^T P^{(0)} = \pi^T, \quad \sum_{i=0}^4 \pi_i = 1 \quad (18)$$

Solving the left eigenvector equation yields $\pi = [0.068, 0.445, 0.387, 0.054, 0.046]^T$, where confirmed states (S_1, S_2) dominate with 83.2% combined probability.

The convergence rate is governed by the second-largest eigenvalue $\lambda_2 = 0.782$ obtained from eigenvalue decomposition of $P^{(0)}$. The mixing time is bounded by:

$$\tau_{mix}(\epsilon) \leq \frac{\log(1/\epsilon)}{1 - \lambda_2} \approx 10.6 \text{ iterations} \quad (19)$$

At 10 Hz sampling rate, this corresponds to approximately 10 frames, confirming rapid convergence.

The five-state design achieves $\lambda_2 = 0.782$ and a mixing time of approximately 1 s at 10 Hz, which is perfectly scaled to handle the physical timescale of urban stop-and-go events (typical duration 2–5 s), ensuring the chain converges well within a single stop-and-go cycle and remains responsive to motion resumption.

Algorithm 1 formalizes the complete transition process.

Algorithm 1: Markov chain state estimation algorithm

Input: Current state S_t , velocity observation v_t , confidence observation c_t
Output: Predicted state S_{t+1}

- 1: **for** $j = 0$ to 4 **do**
 - 2: Compute bounded adjustment factor
 $\beta_{S_t, j} \leftarrow \max(0.1, \min(3.0, 1 + \omega_v \cdot \phi_v(S_t, j, v_t) + \omega_c \cdot \phi_c(S_t, j, c_t)))$
 - 3: Apply adjustment $P_{S_t, j}^{\text{adj}} \leftarrow P_{S_t, j}^{(0)} \cdot \beta_{S_t, j}$
 - 4: **end for**
 - 5: Normalize probabilities $P_{S_t, j}^{\text{final}} \leftarrow \frac{P_{S_t, j}^{\text{adj}}}{\sum_{k=0}^4 P_{S_t, k}^{\text{adj}}}$ for all j
 - 6: **return** $S_{t+1} = \arg \max_j P_{S_t, j}^{\text{final}}$
-

3.4 Detection-Guided Multi-Strategy Clustering and Removal

The clustering module addresses another critical challenge: incomplete boundary removal. Simple bounding-box methods indiscriminately delete all points within a box, creating a dilemma with two conflicting issues. First, they over-delete by removing valid static structures (e.g., ground points or road markings) that happen to fall inside the box, which degrades map fidelity. Second, they under-delete by missing dynamic points near the box boundaries, as LiDAR points from moving objects often spill slightly outside the predicted box, especially when detection confidence is low.

A standard DBSCAN algorithm with fixed parameters cannot solve both issues simultaneously because preventing over-deletion requires fine-grained filtering, while fixing under-deletion requires a broader search. To resolve this, our density-adaptive design dynamically adjusts the clustering parameters based on local point cloud density and detection confidence. Finally, a spherical region analysis applies point-level geometric checks to precisely separate the dynamic object from the surrounding static background.

Following probabilistic state estimation, dynamic point removal is performed in three progressive stages: spatial constraint removal for efficiency, density-adaptive clustering for robust segmentation, and spherical region removal for precise dynamic point elimination. This multi-strategy approach addresses the incomplete removal problem prevalent in simple threshold-based methods by combining spatial, density, and motion constraints to achieve comprehensive dynamic point identification while balancing removal accuracy with real-time performance requirements.

3.4.1 Spatial Constraint

Processing is confined to regions within dynamic detection bounding boxes, reducing computational overhead and focusing on areas likely to contain dynamic points. For each detection bounding box predicted to contain dynamic objects, we extract candidate points $\mathcal{S}_{\text{candidate}}$ and analyze their characteristics to determine the optimal clustering strategy.

We compute the volume density of points within each detection bounding box:

$$\rho = \frac{N_{\text{box}}}{V_{\text{box}}} = \frac{N_{\text{box}}}{(x_{\text{max}} - x_{\text{min}})(y_{\text{max}} - y_{\text{min}})(z_{\text{max}} - z_{\text{min}})} \quad (20)$$

where N_{box} is the number of LiDAR points within the detection bounding box, and V_{box} is the 3D volume. Density thresholds are derived from statistical analysis on 15,247 annotated object instances from independent KITTI Raw sequences (as described in [Section 3.3.2](#)). For each category c , we compute empirical

density distributions (approximated as log-normal) and extract thresholds at the 15th ($\rho_{\text{low}}^{(c)}$), 50th ($\rho_{\text{med}}^{(c)}$), and 85th ($\rho_{\text{high}}^{(c)}$) percentiles to capture natural density variations across object types.

Based on these statistics and detection confidence, regions are classified into density categories through the following decision function:

$$\text{scene_type} = \begin{cases} \text{ultra-high,} & \text{if } \rho > \rho_{\text{high}}^{(c)} \text{ and } c_{\text{det}} > 0.8 \\ \text{high,} & \text{if } N_{\text{box}} \geq N_{\text{th}}^{(c)} \text{ and } \rho > \rho_{\text{med}}^{(c)} \\ \text{ultra-sparse,} & \text{if } \rho < \rho_{\text{low}}^{(c)} \\ \text{sparse,} & \text{otherwise} \end{cases} \quad (21)$$

where c represents the object category. Category-specific density thresholds are defined in Table 2 to accommodate different object characteristics.

Table 2: Density-based scene classification thresholds.

Scene Type	Conditions	Veh	Ped	Cyc
Ultra-high	$\rho > \rho_{\text{high}}, c_{\text{det}} > 0.8$	300	450	350
High	$N_{\text{box}} \geq N_{\text{th}}, \rho > \rho_{\text{med}}$	80	120	100
	(N_{th} values)	300	200	250
Ultra-sparse	$\rho < \rho_{\text{low}}$	20	15	18
Sparse	Otherwise	—		

The spatial constraint stage outputs candidate point set $\mathcal{S}_{\text{candidate}}$ and scene type classification for subsequent clustering.

3.4.2 Density-Adaptive Clustering and Quality Validation

We employ DBSCAN with adaptive parameters. For a candidate point $\mathbf{p}_j$, its ε -neighborhood is defined as:

$$\mathcal{N}_{\varepsilon}(\mathbf{p}_j) = \{\mathbf{p}_k \in \mathcal{S}_{\text{candidate}} : \|\mathbf{p}_j - \mathbf{p}_k\|_2 \leq \varepsilon_{\text{adaptive}}\} \quad (22)$$

A point is classified as a core point if its neighborhood contains sufficient points:

$$\text{core_point}(\mathbf{p}_j) = \begin{cases} \text{true,} & \text{if } |\mathcal{N}_{\varepsilon}(\mathbf{p}_j)| \geq N_{\text{min}} \\ \text{false,} & \text{otherwise} \end{cases} \quad (23)$$

Clustering parameters are determined through detection-guided adaptation. The tolerance parameter is computed as:

$$\varepsilon_{\text{adaptive}} = \varepsilon_{\text{base}}^{(s,c)} \cdot f_{\text{conf}}(c_{\text{det}}) \quad (24)$$

where s denotes the scene type, c denotes the object category, and the base tolerance $\epsilon_{\text{base}}^{(s,c)}$ is determined according to Table 3. The confidence factor adjusts the strictness of clustering based on detection reliability:

$$f_{\text{conf}}(c_{\text{det}}) = \begin{cases} 0.8, & \text{if } c_{\text{det}} > 0.9 \text{ (high confidence)} \\ 1.0, & \text{if } 0.7 \leq c_{\text{det}} \leq 0.9 \text{ (medium confidence)} \\ 1.3, & \text{if } c_{\text{det}} < 0.7 \text{ (low confidence)} \end{cases} \quad (25)$$

Table 3: Scene and category-specific base clustering parameters.

Scene Type	$\epsilon_{\text{base}} \text{ (m)}$			N_{base}		
	Veh	Ped	Cyc	Veh	Ped	Cyc
Ultra-high density	0.20	0.15	0.18	10	5	7
High density	0.25	0.20	0.22	6	3	4
Ultra-sparse	1.50	1.20	1.35		1	
Sparse		1.00			2	

The minimum cluster size is similarly adapted:

$$N_{\text{min}} = N_{\text{base}}^{(s,c)} \cdot g_{\text{conf}}(c_{\text{det}}) \quad (26)$$

where the base cluster size $N_{\text{base}}^{(s,c)}$ is determined according to Table 3, and the confidence adjustment factor is:

$$g_{\text{conf}}(c_{\text{det}}) = \begin{cases} 1.2, & \text{if } c_{\text{det}} > 0.9 \\ 1.0, & \text{if } 0.7 \leq c_{\text{det}} \leq 0.9 \\ 0.7, & \text{if } c_{\text{det}} < 0.7 \end{cases} \quad (27)$$

After generating initial clusters with adaptive parameters, quality assessment filters unreliable clusters containing noise or multiple objects. The spatial compactness of each cluster C_i is evaluated as:

$$\text{compactness}(C_i) = \frac{1}{|C_i|} \sum_{\mathbf{p} \in C_i} \|\mathbf{p} - \text{centroid}(C_i)\|_2 \quad (28)$$

Clusters are validated against a density-adaptive compactness threshold that follows the density-compactness inverse correlation principle:

$$\text{threshold}_{\text{compact}} = 0.5 + \frac{0.3}{\sqrt{\rho}} \quad (29)$$

where larger thresholds in sparse scenarios enhance robustness, while smaller thresholds in dense scenarios improve precision. Size-based validation ensures clusters represent meaningful object segments. The maximum cluster size is defined as:

$$N_{\text{max}} = 5 \times N_{\text{min}} \quad (30)$$

Clusters are accepted based on the following criteria:

$$\text{valid}(C_i) = \begin{cases} \text{true}, & \text{if } N_{\min} \leq |C_i| \leq N_{\max} \\ & \text{and compactness}(C_i) < \text{threshold}_{\text{compact}} \\ \text{false}, & \text{otherwise} \end{cases} \quad (31)$$

This approach ensures that only coherent and appropriately sized clusters proceed to final removal, balancing computational efficiency with segmentation quality.

3.4.3 Spherical Region Removal

As the final stage, spherical region removal performs point-level classification within each validated cluster. For each cluster C_i with centroid $\mathbf{c}_i$, we define a spherical analysis region $\mathcal{R}_i(r) = \{\mathbf{p} \in \mathbb{R}^3 : \|\mathbf{p} - \mathbf{c}_i\|_2 \leq r\}$.

Within each spherical region, we evaluate three consistency measures that capture different aspects of point coherence. The first measure evaluates radial distance patterns relative to detection shape characteristics:

$$\mathcal{C}_{\text{radial}}(\mathbf{p}_j) = \exp\left(-\frac{(\|\mathbf{p}_j - \mathbf{c}_i\|_2 - \mu_{r,\text{expected}})^2}{2(\sigma_{r,\text{adaptive}})^2}\right) \quad (32)$$

where the expected radial distance and adaptive variance are guided by detection information:

$$\mu_{r,\text{expected}} = 0.4 \cdot h_{\text{shape}}(\text{bbox}) \quad (33)$$

$$\sigma_{r,\text{adaptive}} = \sigma_{r,\text{base}} \cdot (1 + 0.3 \cdot (1 - c_{\text{det}})) \quad (34)$$

where $\sigma_{r,\text{base}} = 0.3$ is the base variance and c_{det} is the detection confidence. The shape function is defined as:

$$h_{\text{shape}}(\text{bbox}) = \frac{\max(l_{\text{bbox}}, w_{\text{bbox}}, h_{\text{bbox}})}{2} \quad (35)$$

representing half of the maximum bounding box dimension. The expected radial distance is set to 40% of the object's maximum dimension based on statistical analysis, while the adaptive variance increases by up to 30% for lower detection confidence to accommodate bounding box uncertainty.

The second measure analyzes geometric coherence through local neighborhood patterns:

$$\mathcal{C}_{\text{geometric}}(\mathbf{p}_j) = \frac{1}{|\mathcal{N}_k(\mathbf{p}_j)|} \sum_{\mathbf{p}_k \in \mathcal{N}_k(\mathbf{p}_j)} \exp\left(-\frac{\|\mathbf{p}_j - \mathbf{p}_k\|_2^2}{2\tau^2}\right) \quad (36)$$

where $\mathcal{N}_k(\mathbf{p}_j)$ denotes the k -nearest neighbors ($k = 5$) of point $\mathbf{p}_j$ within the cluster, and $\tau = 0.1$ m controls the spatial scale of neighborhood coherence.

The third measure evaluates motion consistency within the same object:

$$\mathcal{C}_{\text{motion}}(\mathbf{p}_j) = \exp\left(-\frac{\|\hat{\mathbf{v}}_j - \bar{\mathbf{v}}_i\|_2^2}{2\sigma_v^2}\right) \quad (37)$$

where $\hat{\mathbf{v}}_j$ represents the estimated point velocity (approximated by the velocity of the associated tracked object $\mathbf{v}_{t,m}$ from temporal data association), $\bar{\mathbf{v}}_i$ denotes the cluster mean velocity, and $\sigma_v = 0.5$ m/s is the velocity variance threshold.

All three consistency measures are integrated through a weighted geometric mean for final classification:

$$\mathcal{S}_{\text{final}}(\mathbf{p}_j) = [\mathcal{C}_{\text{radial}}(\mathbf{p}_j)^{w_r} \cdot \mathcal{C}_{\text{geometric}}(\mathbf{p}_j)^{w_g} \cdot \mathcal{C}_{\text{motion}}(\mathbf{p}_j)^{w_m}]^{\frac{1}{w_r + w_g + w_m}} \quad (38)$$

The weighted geometric mean ensures that poor performance in any single measure significantly impacts the final score. Weights ($w_r = 0.4$, $w_g = 0.3$, $w_m = 0.3$) are optimized on SemanticKITTI validation set, emphasizing spatial consistency while balancing geometric and motion information.

Points are classified as dynamic and removed based on threshold decision:

$$\mathcal{S}_{\text{remove}}^{(i)} = \{\mathbf{p}_j \in \mathcal{R}_i : \mathcal{S}_{\text{final}}(\mathbf{p}_j) > \theta_{\text{remove}}\} \quad (39)$$

where $\theta_{\text{remove}} = 0.6$ is determined via ROC analysis on the validation set. This threshold balances false positive and true positive rates, ensuring robust dynamic point removal while preserving static structures. The complete removal process sequentially applies spatial constraint, density-adaptive DBSCAN clustering with quality validation, and spherical region analysis for precise dynamic point elimination.

3.5 System Integration and Complexity Analysis

DGMSE employs a hierarchical processing strategy where each stage refines dynamic-static classification. The Markov chain maintains temporal consistency, density-adaptive clustering ensures spatial accuracy, and spherical region analysis achieves point-level precision. Table 4 analyzes the time complexity for each module.

Table 4: DGMSE module complexity analysis.

Module	Input Scale	Time Complexity	Description
3D Detection & Data Association	K targets	$O(K^2)$	PointPillars + greedy matching
Markov Chain State Estimation	K targets	$O(K)$	Per-target state estimation
Density-Adaptive Clustering	M points/box	$O(K \cdot M \log M)$	Clustering, $M \ll N$
Overall System	N, K, M	$O(N + K^2 + K \cdot M \log M)$	Linearly dominated

With K detection targets and M candidate points where $K, M \ll N$, the overall complexity reduces to approximately $O(N)$, ensuring real-time processing. Key optimizations include early exit for sparse regions (<20 points), voxel downsampling for dense scenarios (>500 points), hierarchical processing with spatial constraint ($O(N)$) before clustering ($O(M \log M)$), and OpenMP parallelization across independent targets.

4 Results

To verify the performance of DGMSE, we conducted quantitative experimental evaluation on the SemanticKITTI dataset [45]. However, due to the small proportion of dynamic objects in SemanticKITTI, we additionally performed qualitative demonstrations on challenging KITTI Raw sequences [46] with dense dynamic content. We evaluated our method against state-of-the-art approaches and conducted ablation studies to assess the contributions of individual components.

All experiments were conducted on a laptop running Ubuntu 20.04, equipped with an Intel i5-13500H processor, 16 GB RAM, and an NVIDIA RTX 4050 GPU. We set the Markov chain parameters to $\omega_v = 0.6$ and $\omega_c = 0.4$, with a density threshold of 500 points for clustering classification.

4.1 Dataset

4.1.1 SemanticKITTI Dataset for Quantitative Evaluation

The SemanticKITTI [45] extends the KITTI dataset [46] by adding pointwise semantic labels across 28 categories, enabling precise quantitative evaluation of dynamic object removal performance. The dataset provides point cloud sequences from a 64-channel Velodyne HDL-64E LiDAR, with 30-degree vertical and 360-degree horizontal coverage, captured in diverse urban scenarios including city streets, residential areas, and highways.

However, due to the relatively small proportion of dynamic objects in typical SemanticKITTI sequences, we focused on challenging segments with substantial dynamic content: sequence 00 (frames 4390–4530), sequence 01 (frames 150–250), sequence 02 (frames 860–950), sequence 05 (frames 2350–2670), and sequence 07 (frames 630–820). These segments represent scenes with varying complexities and object densities.

4.1.2 KITTI Raw Dataset for Qualitative Validation

To complement the quantitative evaluation and demonstrate robustness in scenarios with dense dynamic content, we also performed qualitative validation on challenging KITTI Raw sequences [46]. Specifically, we used sequence 2011_09_29_drive_0071 for qualitative analysis, which contains multi-directional vehicle flows and dense pedestrian activity at a busy intersection. Additionally, sequence 2011_09_26_drive_0091_sync was used for density-adaptive clustering ablation analysis to evaluate the cyclist scenario. These sequences provide a comprehensive assessment of algorithm performance in highly dynamic environments with varying object types and motion patterns.

4.2 Evaluation Metrics

DGMSE performance is evaluated using the Preservation Rate (PR), Rejection Rate (RR), and F1 score metrics proposed by ERASOR [7]. These metrics are more suitable for dynamic object removal than precision-recall approaches.

Precision-recall metrics are less effective for dynamic object removal due to the static-dynamic imbalance in urban scenes. Incorrectly filtered static points vastly outnumber dynamic targets, leading to misleading precision values. Preprocessing variations also affect point counting consistency.

The evaluation metrics are defined as follows:

1. Preservation Rate (PR): Measures static structure retention:

$$PR = \frac{\text{Number of preserved static points}}{\text{Number of static points on the raw map}} \quad (40)$$

Higher values indicate better static point preservation.

2. Rejection Rate (RR): Measures dynamic content removal:

$$RR = 1 - \frac{\text{Preserved dynamic points}}{\text{Total dynamic points on raw map}} \quad (41)$$

Higher values indicate better dynamic point removal.

3. F1 Score: Balances preservation and rejection performance:

$$F_1 = \frac{2 \times PR \times RR}{PR + RR} \quad (42)$$

The F1 score balances preservation and rejection through harmonic averaging, capturing the trade-off between retaining static information and removing dynamic elements. This balanced measure suits real-world applications requiring simultaneous optimization of both objectives.

Point preservation assessment uses the following criterion:

$$\mathcal{K}(q, s) = \text{IVol}(q, s) \wedge \psi(q, s) \quad (43)$$

where s is the closest point to q in the processed map. Two binary functions validate preservation: $\text{IVol}(\cdot)$ checks spatial co-location in identical voxels, while $\psi(\cdot)$ verifies semantic consistency between static and dynamic labels. Voxel resolution affects accuracy, so we standardize to 0.2 m across all methods for fair comparison.

4.3 Quantitative Evaluation

We compared DGMSE with state-of-the-art dynamic object removal methods on the SemanticKITTI dataset, including offline methods (Removort [8], ERASOR [7], ERASOR2 [38]) and online methods (DynamicFilter [11]). We cited baseline performance metrics from the original publications or reproduced them using official implementations with the recommended parameters. Table 5 presents the detailed results. The method symbols in the table are defined as follows: (1) RM3 represents the results of the Removort method after three removal stages with pixel resolutions of 0.5, 0.45, and 0.4 degree, respectively; (2) RM3+RV1 represents results after adding one recovery stage to RM3 with a pixel resolution of 0.55 degree; (3) ERASOR-0.2 and ERASOR-0.4 represent different configurations of the ERASOR method with scan ratio thresholds set to 0.2 and 0.4, respectively.

Table 5: Performance comparison of different methods on the SemanticKITTI dataset.

Method	Seq.00	Seq.01	Seq.02	Seq.05	Seq.07
Removort-RM3	85.502/ 99.354 /0.919	94.221/93.608/0.939	76.319/96.799/0.853	86.900/87.880/0.874	80.689/98.822/0.888
Removort-RM3+RV1	86.829/90.617/0.887	95.815/57.077/0.715	83.293/88.371/0.858	88.170/79.981/0.839	82.038/95.504/0.883
ERASOR-0.2	93.980/97.081/0.955	91.487/ 95.383 /0.934	87.731/97.008/0.921	88.730/98.262/0.933	90.624/ 99.271 /0.948
ERASOR-0.4	80.769/98.353/0.887	79.807/95.080/0.868	78.319/98.464/0.872	72.440/97.831/0.832	82.875/98.221/0.899
ERASOR2	98.788 /98.582/ 0.987	96.879/94.629/0.957	98.523/ 99.709 / 0.991	97.582/ 98.992 /0.983	98.977 /98.459/ 0.987
DynamicFilter	90.070/91.090/0.906	87.950/87.690/0.878	88.020/86.100/0.871	90.170/84.650/0.873	87.940/86.800/0.874
DGMSE (Ours)	98.699/94.332/0.965	98.003 /95.256/ 0.966	98.624 /91.789/0.951	98.997 /98.617/ 0.988	96.051/97.555/0.968

Note: Data format: PR(%) / RR(%) / F1. Bold indicates the best value in each column.

Table 5 shows that DGMSE achieved competitive performance across all five sequences, particularly on sequences 01 and 05, where it achieved the highest F1 scores of 0.966 and 0.988, respectively. Although the offline method ERASOR2 achieves superior performance on sequences 00, 02, and 07 due to its global optimization capabilities (e.g., achieving an F1 score of 0.991 on sequence 02), DGMSE demonstrates comparable accuracy while satisfying real-time processing requirements. Notably, on sequence 05, DGMSE achieves an F1-score of 0.988, slightly outperforming ERASOR2 (0.983). Furthermore, DGMSE achieves the highest Preservation Rates (PR) on sequences 01 (98.003%), 02 (98.624%), and 05 (98.997%), indicating effective retention of static structures. This advantage stems from our detection-box-centric removal strategy, which constrains operations within 3D detection bounding boxes, avoiding unnecessary processing of external static environments.

In contrast, other baseline methods exhibit varying limitations. The online method DynamicFilter shows substantially lower performance, with DGMSE exceeding its F1 scores by 5.9% to 11.5% across all sequences. This performance gap primarily stems from DynamicFilter's incomplete removal problem

caused by fixed geometric thresholds that fail to adapt to varying object densities and motion patterns. DGMSE also exhibits superior robustness, maintaining scores between 0.951–0.988, whereas DynamicFilter fluctuates below 0.906. Removert methods show different preservation-rejection trade-offs: RM3 exhibits aggressive removal with high rejection rates but low preservation (85.5% PR on sequence 00), while the revert mechanism (RM3+RV1) improves PR to 86.8% but significantly reduces rejection rates, particularly on sequence 01 where RR dropped to 57.1%. ERASOR methods demonstrate performance variation with threshold settings, where ERASOR-0.2 shows better balance (F1-scores 0.921–0.955), while ERASOR-0.4 exhibits lower preservation but higher rejection rates.

In summary, DGMSE achieves competitive accuracy comparable to state-of-the-art offline methods while maintaining real-time processing capabilities. By leveraging the Markov chain model to accumulate observational evidence over time rather than relying on instantaneous information, the method effectively enhances stability and noise resistance in dynamic object removal. As shown in Table 5, while offline methods such as ERASOR2 achieve marginal F1 improvements through computationally expensive global optimization, DGMSE achieves comparable accuracy ($F1 > 0.95$) across all sequences under strict online, real-time constraints, without access to future frames or global map information.

4.4 Qualitative Analysis

We conduct qualitative analysis to evaluate the performance of DGMSE in challenging environments. To complement the quantitative evaluation on SemanticKITTI, we further test robustness on two complex segments from the KITTI Raw sequence 2011_09_29_drive_0071, which features higher dynamic density and more intricate interactions:

1. Dense pedestrian flow scenario (frames 170–380): characterized by mutually occluded pedestrian groups and complex motion patterns.
2. Mixed dynamic targets scenario (frames 455–600): involving multi-directional, interwoven movements of vehicles, pedestrians, and cyclists.

Although KITTI Raw does not provide pointwise semantic annotations for quantitative evaluation, it offers real-world complexity that is well suited for qualitative robustness assessment.

4.4.1 Dense Pedestrian Flow Scenario

This scenario involves dense pedestrian flow, frequent occlusions, and erratic stop-move patterns, which pose significant challenges for dynamic target identification.

Fig. 3 compares removal performance in dense pedestrian flow scenarios with frequent occlusions. Subfigures (a) and (c) show real-world camera images capturing temporal changes between frames 170–270 and 310–380, (b) presents the original map with dynamic artifacts in two test regions (green and red boxes), and (d–f) show results from ERASOR, Removert, and DGMSE, respectively.

DGMSE maintains stable identification during occlusions and stationary states through temporal memory, preserving static structure integrity.

ERASOR efficiently filters most dynamic objects but mistakenly removes static walls at corners (red arrows in Fig. 3d), even with reduced scan ratio threshold (0.05). Removert performs well in moderate dynamic scenes but exhibits incomplete removal in high-density environments, leaving residual dynamic traces as shown in Fig. 3e. This incomplete removal issue stems from its range-image-based approach, which struggles to capture spatial coherence in densely populated regions.

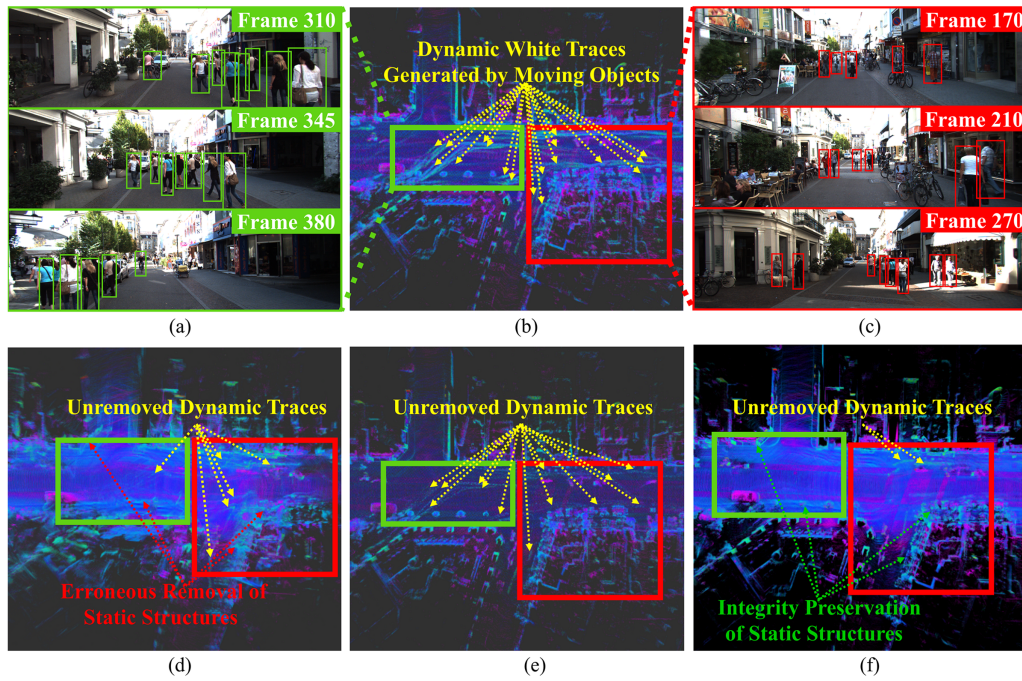


Figure 3: Dense pedestrian flow removal comparison (frames 170–270 and 310–380). (a) Temporal changes in dynamic regions. (b) Original map with dynamic artifacts. (c) Additional temporal sequence. (d) ERASOR. (e) Removert. (f) DGMSE results with static structures preserved and dynamic traces removed.

4.4.2 Multi-Type Dynamic Target Scenario

This scenario contains a busy urban intersection with multi-directional movements of vehicles, pedestrians, and cyclists. Frequent motion transitions and mutual occlusions challenge precise removal.

Fig. 4 shows performance in frame sequences 455–495 and 600–700, involving interwoven movements of vehicles, pedestrians, and cyclists with varying motion characteristics. DGMSE effectively processes multiple targets using differentiated Markov parameters for each target type, handling occlusions and state switching robustly.

ERASOR struggles in complex multi-target scenarios with overlapping vehicles and pedestrians, missing interwoven targets (yellow arrows in Fig. 4d) and erroneously removing static structures. Removert shows satisfactory performance through multi-stage processing but suffers from incomplete removal in densely populated scenes, leaving residual dynamic traces as shown in Fig. 4e. The incomplete removal problem becomes more pronounced when multiple dynamic objects with different motion patterns coexist in close proximity.

4.5 Ablation Study

We conduct ablation studies by removing key modules to compare and analyze their contributions, focusing on two core components: Markov chain state estimation and detection-guided density-adaptive clustering. For each module, we provide both qualitative and quantitative validation: (1) Qualitative visualization on carefully selected KITTI Raw sequences that exhibit representative challenge scenarios (e.g., walk-stop-walk transitions, cyclist scenarios), demonstrating the method's behavior in diverse real-world conditions; (2) Quantitative evaluation on SemanticKITTI sequence 05, which provides dense semantic annotations enabling precise computation of Preservation Rate (PR), Rejection Rate (RR), and F1 metrics

across the entire sequence. This dual-validation approach combines interpretable case-study analysis with statistically rigorous performance assessment.

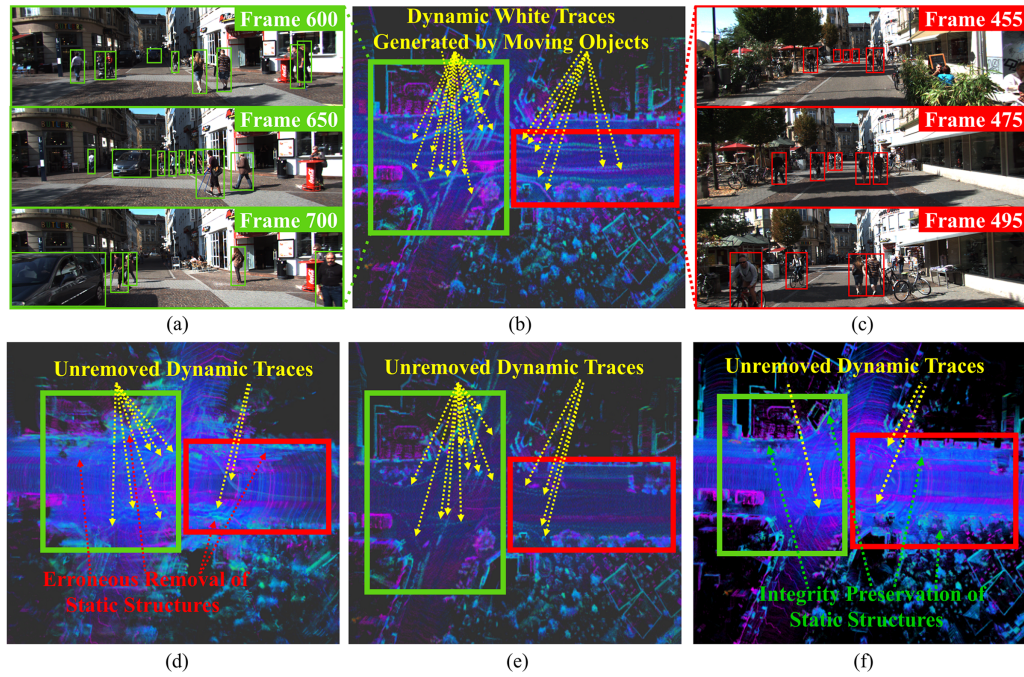


Figure 4: Multi-target removal comparison (frames 455–495 and 600–700). (a) Real-world scenario with vehicles, pedestrians, and cyclists. (b) Original map. (c) Additional temporal sequence. (d) ERASOR. (e) Removert. (f) DGMSE.

4.5.1 Markov Chain State Estimation Ablation

The effectiveness of the Markov chain model was evaluated by replacing it with an instantaneous binary classification approach. This approach determines dynamic states solely based on whether velocity exceeds a threshold, without relying on any historical information. To highlight key state transitions in the walk-stop-walk scenario, we selected frames 733–749 from KITTI Raw sequence 2011_09_29_drive_0071, capturing the transition from motion to a stationary state and the subsequent resumption of motion. From this segment, we selected three representative frames for detailed qualitative analysis: Frame 733 (normal walking), Frame 740 (short pause), and Frame 749 (resumed walking).

As illustrated in Fig. 5, the experimental setup provides a comprehensive comparison between the complete DGMSE method and its variant without the Markov chain module. The figure consists of three sections: the left section presents results from the complete method, the middle section displays real-world frame imagery from frames 733, 740, and 749, and the right section shows results from the Markov chain-ablated algorithm.

In these visualizations, red circular annotations mark pedestrian positions in the real-world scene, while blue bounding boxes (shown in both left and right panels) highlight pedestrian targets detected by the algorithm in the point cloud representation. The red and green point clouds represent the algorithm's classification of dynamic and static objects, respectively, based on the point cloud data.

The initial baseline comparison in Fig. 5a,d demonstrates that during normal pedestrian locomotion, both methods correctly classify the pedestrian as a dynamic object (represented by red point clouds). This

establishes a performance baseline for dynamic object detection, with both methods showing consistent accuracy in standard motion conditions.

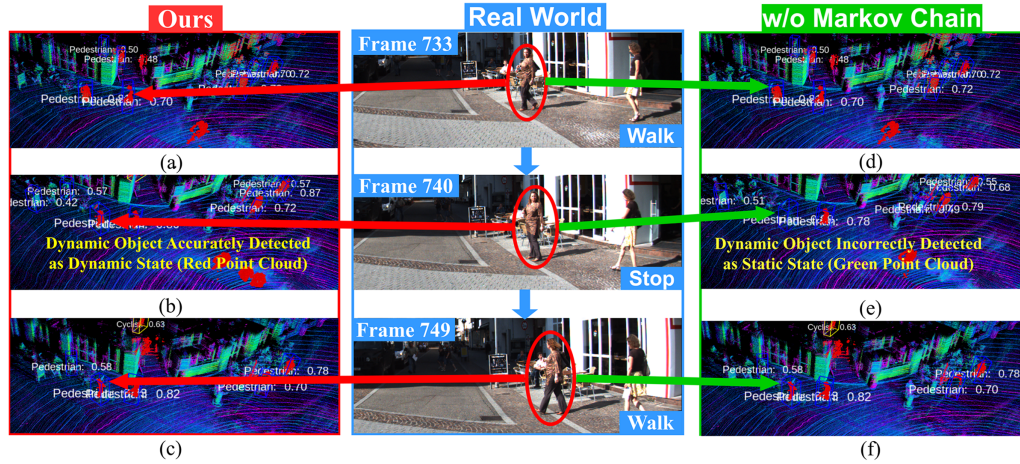


Figure 5: Ablation analysis of the Markov chain state estimation model. Left column (a–c): complete DGMSE method. Middle column: real-world frames (Frame 733, 740, 749). Right column (d–f): method without Markov chain. (a,d) Normal walking. (b,e) Short pause. (c,f) Resumed walking.

The critical distinction arises during the pedestrian pause phase, as illustrated in Fig. 5b,e. The complete method maintains the pedestrian's dynamic classification through the temporal memory mechanism in the Markov chain framework, which preserves the red point cloud representation. This behavior indicates that the temporal memory component effectively sustains dynamic state continuity during brief motion interruptions, thus reducing state oscillations. In contrast, the algorithm without the Markov chain module reveals a key limitation: it erroneously reclassifies the stationary pedestrian as a static object, as evidenced by the transition to the green point cloud representation. This misclassification highlights the crucial role of temporal memory in maintaining consistent dynamic object identification during motion transition phases.

The motion resumption phase, shown in Fig. 5c,f, reveals that both methods successfully reestablish pedestrian classification as a dynamic object upon motion restoration. However, the complete method demonstrates superior stability and consistency throughout the entire temporal sequence. By utilizing its temporal memory mechanism, it effectively prevents the classification instabilities typical of instantaneous determination approaches.

This temporal sequence analysis provides strong evidence that the temporal memory mechanism in the complete DGMSE framework is crucial for maintaining robust and consistent dynamic object recognition. The ablation study clearly shows that algorithms lacking this temporal memory component directly lead to state oscillations.

Beyond the qualitative visualization above, we quantitatively validate the contribution of the Markov chain module through ablation experiments on SemanticKITTI sequence 05 using the standard evaluation metrics. Table 6 presents the quantitative comparison between the complete DGMSE method and its variant without the Markov chain module (replaced by instantaneous velocity-based classification with a fixed threshold).

Table 6: Quantitative ablation study: markov chain state estimation on SemanticKITTI sequence 05.

Method Variant	PR (%)	RR (%)	F1
DGMSE (Complete)	98.997	98.617	0.988
w/o Markov Chain	96.234	89.156	0.925
Performance Drop	-2.763	-9.461	-0.063

The results demonstrate that removing the Markov chain module leads to significant performance degradation. The Rejection Rate (RR) drops by 9.461% (from 98.617% to 89.156%), indicating that the instantaneous classification approach struggles to maintain consistent dynamic object identification during motion transitions and temporary stops. The Preservation Rate (PR) also decreases by 2.763%, as state oscillations cause instability that affects both dynamic and static point classification. Overall, the F1 score decreases by 0.063 (from 0.988 to 0.925), confirming that the temporal memory mechanism in the Markov chain framework is crucial for robust performance.

4.5.2 Detection-Guided Density-Adaptive Clustering and Removal Ablation

We tested density-adaptive clustering by replacing it with a simple bounding box removal approach, where points within the 3D detection bounding boxes are removed. Fig. 6 compares these approaches in a cyclist scenario from KITTI Raw sequence 2011_09_26_drive_0091. Fig. 6a represents the real-world scenario, with the cyclist specifically marked by a white bounding box, while Fig. 6b shows the original map. Fig. 6c shows the result without adaptive clustering. The red markers highlight that a significant portion of the ground background has been removed, resulting in obvious gaps. Fig. 6d demonstrates our approach, where green markers indicate that only true dynamic targets are removed, while the ground structure is completely preserved.

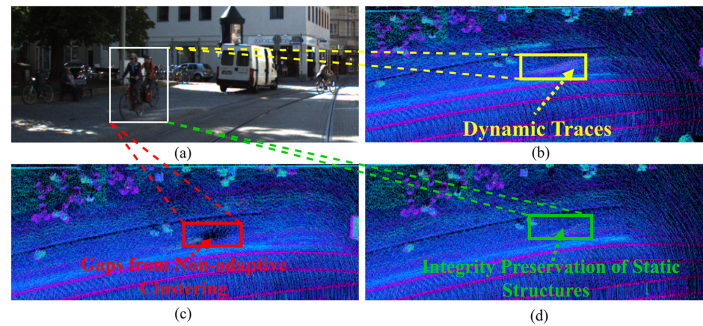


Figure 6: Ablation analysis of density-adaptive clustering in the cyclist scenario. (a) Real-world scenario. (b) Original map. (c) Without adaptive clustering showing ground removal. (d) DGMSE approach preserving ground structure while removing only dynamic targets.

In contrast, density-adaptive clustering performs geometric analysis within bounding boxes, adjusting parameters based on density to accurately identify dynamic targets. This method avoids erroneous static deletions and maintains map integrity.

To quantitatively assess the contribution of the density-adaptive clustering module, we conducted ablation experiments on SemanticKITTI sequence 05. Table 7 compares the complete DGMSE method

against a variant that uses simple bounding box removal (removing all points within detection boxes without clustering analysis).

Table 7: Quantitative ablation study: density-adaptive clustering on SemanticKITTI sequence 05.

Method Variant	PR (%)	RR (%)	F1
DGMSE (Complete)	98.997	98.617	0.988
w/o Adaptive Clustering	93.582	97.823	0.957
Performance Drop	-5.415	-0.794	-0.031

The ablation results presented in [Table 7](#) demonstrate that the density-adaptive clustering module is a primary contributor to the preservation rate (PR) of the proposed framework. Specifically, without this module, the system relies on the simple bounding-box approach, leading to a substantial PR drop of 5.415% (from 98.997% to 93.582%). This performance degradation stems from the intrinsic limitations of simple bounding-box filtering: it lacks the fine-grained spatial discrimination required to distinguish dynamic objects from adjacent static structural features (e.g., ground planes) within the detection scope. Meanwhile, the rejection rate (RR) also declines by 0.794%, demonstrating that the simple bounding-box approach cannot effectively isolate dynamic points distributed around object boundaries. While the overall F1 score decreases by only 0.031, the qualitative degradation of the reconstructed point cloud map is severe. The induced geometric discontinuities in static backgrounds, together with residual dynamic artifacts, inevitably degrade the robustness of downstream scan-matching pipelines, such as ICP and NDT algorithms. These observations strongly validate that the fine spatial granularity enabled by the density-adaptive clustering module is essential for preserving high map fidelity and suppressing trajectory drift, which is critical for high-precision LiDAR SLAM deployment.

[Table 8](#) presents a comprehensive ablation analysis examining the effects of removing individual components and both components together.

Table 8: Comprehensive ablation study on SemanticKITTI sequence 05.

Method Variant	PR (%)	RR (%)	F1
DGMSE (Complete)	98.997	98.617	0.988
w/o Markov Chain	96.234	89.156	0.925
w/o Adaptive Clustering	93.582	97.823	0.957
w/o Both Modules	91.145	85.234	0.881

When both modules are removed, performance drops dramatically: PR decreases by 7.852%, RR decreases by 13.383%, and F1 decreases by 0.107 (from 0.988 to 0.881). This demonstrates that both the Markov chain state estimation and density-adaptive clustering modules make substantial, complementary contributions to the overall system performance. The Markov chain primarily improves temporal consistency and rejection accuracy, while the adaptive clustering ensures precise spatial segmentation and preservation accuracy.

4.6 Runtime Analysis

We analyzed the computational overhead to assess the real-time capabilities of our method. [Table 9](#) presents a detailed runtime analysis.

Table 9: Runtime analysis of DGMSE modules and complete pipeline (per-frame processing).

Module	Average Time (ms)
PointPillars 3D Object Detection	20.0
Markov Chain State Estimation	0.8
Detection-Guided Multi-Strategy Clustering & Removal	7.1
Other Processing	2.1
DGMSE Total	30.0
FAST-LIO2	10.0
Complete Pipeline (DGMSE + FAST-LIO2)	40.0

Table 9 shows that DGMSE processes each frame in 30.0 ms at the module level. When integrated with FAST-LIO2 (approximately 10 ms per frame), the complete pipeline executes in approximately 40 ms. This corresponds to 25 FPS, comfortably satisfying the standard 10 Hz processing requirement of typical LiDAR SLAM systems. All measurements were obtained on a consumer-grade laptop equipped with an Intel i5-13500H processor and an NVIDIA RTX 4050 GPU. Given that this setup represents an entry-level mobile platform, deployment on more capable computing hardware typically found in autonomous vehicles is expected to further reduce latency.

Module-level analysis reveals that PointPillars detection requires 20.0 ms, accounting for 67% of the total module time. While 3D detection constitutes the primary computational bottleneck, this upfront cost is essential because the module does far more than simple localization; it provides the rich foundational priors that power all subsequent stages of DGMSE. Specifically: (1) for Markov chain state estimation, the detector provides tracking-derived velocities and confidence scores, which serve as the core inputs for observation-driven probability adjustments; and (2) for multi-strategy clustering, the bounding boxes establish spatial constraints that significantly restrict the search space, while the detection confidence and box dimensions directly guide the density-adaptive parameter tuning and spherical region analysis. By concentrating the computational load on extracting these rich priors at the front-end, the two core filtering modules remain exceptionally lightweight, requiring only 7.9 ms combined (Markov chain: 0.8 ms, clustering: 7.1 ms). Furthermore, the architecture remains fundamentally modular, allowing the detector to be seamlessly replaced with lighter alternatives if specific deployment constraints require higher scalability.

The results validate that DGMSE achieves high accuracy while satisfying real-time capabilities, making it suitable for autonomous driving and robotic navigation applications.

4.7 Odometry Benchmark

Although previous experiments have thoroughly demonstrated DGMSE's efficiency in removing dynamic point clouds, the fundamental goal of this module is to improve the accuracy of robot trajectory estimation and eliminate the detrimental impact of dynamic artifacts on the entire mapping system. To this end, this section conducts system level SLAM experiments.

Given that our previous discussions primarily focused on FAST-LIO2 [44], to further explore how DGMSE benefits algorithms with different underlying mechanisms, we intentionally introduce LIO-SAM [3] (a tightly coupled system equipped with factor graph optimization and loop closure) as an extended baseline. By integrating DGMSE as a modular frontend module (which is designed for easy integration into various SLAM frameworks) into these two representative frameworks, we investigate whether blocking dynamic

interference at the source can substantially improve the global consistency and local matching quality of the overall mapping pipeline.

Experiments are conducted on KITTI odometry sequences 05 (2.2 km, large scale urban trajectory) and 07 (0.7 km, short distance trajectory). Note that both baseline SLAM systems were executed using their default parameter configurations without per sequence optimization. Consequently, our evaluation emphasizes the relative reduction ratio of the trajectory errors, which optimally isolates the performance improvements directly attributable to the DGMSE module. Trajectory accuracy is quantitatively assessed via Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) using the *evo* toolkit [47] with Sim(3) Umeyama alignment. Both metrics are reported as RMSE in meters.

Table 10 summarizes the results, and the qualitative visualizations in Fig. 7 confirm these findings. For FAST-LIO2, which relies solely on frontend odometry without loop closure, DGMSE prevents the estimator from being misled by ghost traces of moving vehicles, reducing ATE by 42.9% on Seq. 05 (from 5.632 to 3.215 m) and by 44.1% on Seq. 07 (from 3.815 to 2.134 m), with corresponding RPE reductions of 39.2% and 42.5%. The improvement is more pronounced on Seq. 05, which traverses a densely trafficked urban corridor where dynamic interference is more severe.

Table 10: SLAM performance comparison on KITTI odometry benchmark.

Method	Seq. 05		Seq. 07	
	ATE (m)	RPE (m)	ATE (m)	RPE (m)
FAST-LIO2	5.632	1.354	3.815	1.132
FAST-LIO2 + DGMSE	3.215	0.824	2.134	0.651
LIO-SAM	2.518	1.197	0.897	1.112
LIO-SAM + DGMSE	1.526	0.398	0.485	0.384

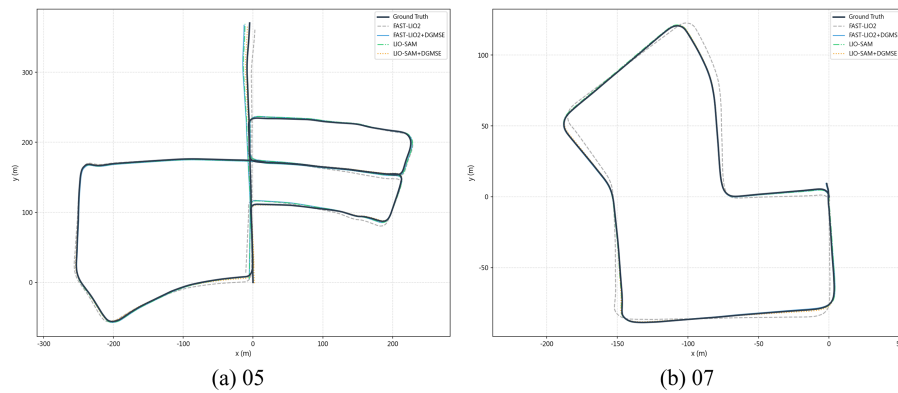


Figure 7: Qualitative comparison of estimated trajectories on KITTI sequences 05 and 07.

For LIO-SAM, despite its loop closure backend, DGMSE still reduces ATE by 39.4% on Seq. 05 (from 2.518 to 1.526 m) and by 45.9% on Seq. 07 (from 0.897 to 0.485 m). The significant RPE drop on Seq. 05 (from 1.197 to 0.398 m, a 66.7% reduction) demonstrates that dynamic point contamination corrupts frontend scan matching in ways that loop closure cannot fully compensate for. These results suggest that DGMSE provides complementary benefits to backend optimization: by filtering dynamic interference at the source, DGMSE ensures that only high quality static geometric features enter the mapping pipeline. This serves as a universal

frontend enhancement that is designed for integration into various SLAM architectures regardless of their underlying backend design.

4.8 Parameter Sensitivity Analysis

We conduct a parameter sensitivity analysis on the SemanticKITTI sequence 05 to validate the robustness of our method. We vary the velocity weight (ω_v) and confidence weight (ω_c) while keeping their sum constant. Table 11 presents stable performance across reasonable ranges.

Table 11: Parameter sensitivity analysis results on SemanticKITTI sequence 05.

ω_v	ω_c	PR (%)	RR (%)	F1
0.2	0.8	92.347	89.673	0.910
0.3	0.7	95.126	93.458	0.943
0.6	0.4	98.997	98.617	0.988
0.7	0.3	96.892	95.234	0.961
0.8	0.2	93.784	91.126	0.924

Note: Bold indicates the optimal parameter configuration.

The optimal configuration indicates that velocity information is more informative than confidence scores. Higher velocity weights ($\omega_v > 0.7$) over-emphasize motion, resulting in performance degradation ($F1 \leq 0.961$). Over-reliance on confidence ($\omega_c > 0.7$) impairs the handling of low-confidence dynamic objects ($F1 = 0.943$). The optimal setting ($\omega_v = 0.6$, $\omega_c = 0.4$) yields a 5.1% improvement in F1-score.

We evaluate different density threshold values, as shown in Table 12.

Table 12: Impact of density threshold on clustering performance.

Threshold	F1	Quality
300	0.965	Adequate
500	0.988	Optimal
700	0.975	Good
1000	0.952	Acceptable

Analysis reveals that a density threshold of 500 points offers the optimal balance between efficiency and quality, with diminishing returns beyond this threshold. Category-specific velocity thresholds (Vehicle: 1.25 m/s, Pedestrian: 0.50 m/s, Cyclist: 0.90 m/s) enhance the F1-score by 3%–5% compared to uniform thresholds.

4.9 Robustness to Detection Failures

A natural concern for any detection-driven framework is its behavior when the underlying detector produces unreliable outputs. We distinguish two failure modes and analyze how DGMSE responds to each.

For false positives (static objects misdetected as dynamic), a single erroneous detection does not trigger immediate point removal. The Markov chain inherently requires consistent multi-frame evidence to confirm a dynamic state (S_2). Specifically, the base transition probability from S_0 to S_2 is only 0.30, and our observation-driven adjustment further suppresses this transition when detection confidence is low

($c_t < c_{th}$, $\phi_c < 0$). As a result, isolated false positives are safely absorbed by the Uncertain state S_0 without causing unwarranted point deletion.

When dealing with low-confidence or inaccurate bounding boxes, the system adaptively relaxes its spatial constraints. Specifically, the confidence factor f_{conf} in Eq. (24) enlarges the DBSCAN search radius for detections with $c_{det} < 0.7$. Simultaneously, the adaptive radial variance $\sigma_{r,adaptive}$ in Eq. (34) increases by up to 30% to accommodate the bounding box uncertainty. Finally, the spherical region analysis performs a strict point-level consistency evaluation. This step ensures that even if the enclosing box is imprecise, only points that genuinely match the cluster's geometry and motion are removed.

5 Discussion

As demonstrated in Section 4.3, DGMSE achieves consistent dynamic point removal performance across five challenging sequences (F1: 0.951–0.988), outperforming the online baseline DynamicFilter by 5.9%–11.5% while remaining competitive with offline methods that exploit future frames and global map information.

The downstream SLAM evaluation further corroborates these results: integrating DGMSE as a pre-processing module into FAST-LIO2 and LIO-SAM reduces the trajectory drift by up to 45.9%, which demonstrates that eliminating dynamic interference at the raw data level suppresses error propagation that cannot be fully rectified by backend optimization alone. Importantly, DGMSE runs efficiently at 30 ms per frame (40 ms for the full pipeline at 25 FPS) and integrates seamlessly into existing odometry pipelines, striking an effective balance between map quality and real-time performance for deployment in real-world autonomous navigation systems.

Despite the above design safeguards, DGMSE inherits a fundamental limitation common to all detection-driven schemes: dynamic objects missed by the detector (false negatives) cannot be processed, and their residual point cloud traces may remain in the constructed map. This limitation is particularly prominent for small, distant, or heavily occluded targets, whose sparse point cloud observations are insufficient for the detector to yield reliable predictions. Consequently, the dynamic removal performance of the proposed method is inherently constrained by the recall of the underlying 3D detector, which constitutes an intrinsic structural limitation of the detection-centric design paradigm.

6 Conclusions

In this paper, we proposed DGMSE, a real-time dynamic object removal framework for point cloud mapping in mobile mapping and robotic navigation. The method integrates a five-state Markov chain model, which combines velocity and confidence information to maintain stable identification during motion transitions, with a detection-guided density-adaptive clustering framework that ensures precise segmentation while preserving static structures, all while satisfying real-time processing requirements.

Experimental evaluations on the SemanticKITTI and KITTI Raw datasets confirm that DGMSE achieves performance competitive with offline methods in complex dynamic scenarios while satisfying real-time constraints. The results validate the synergistic effects of the temporal memory mechanism and the spatial clustering strategy, which effectively mitigate state oscillation and eliminate ghosting artifacts, thereby ensuring robustness and precision in dynamic object removal.

Future work will explore two complementary research directions to mitigate the aforementioned false negative limitation. First, we intend to integrate lightweight geometric compensation strategies, such as ray-casting-based freespace verification and ground-plane consistency checking, to handle undetected objects without fully relying on the detection module. Second, we will incorporate LiDAR-camera multimodal fusion

(e.g., YOLO and EfficientDet) to boost the detection recall of small and occluded targets, thereby effectively reducing the false negative rate at the detection stage.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by National Natural Science Foundation of China, grant number 5207050381; Harbin Manufacturing Technology Innovation Project, grant number 2023CXRC009; and Natural Science Foundation of Heilongjiang Province, grant number PL2024C003.

Author Contributions: The authors confirm contribution to the paper as follows: conceptualization, Jiahua Kou and Chengbo Guo; methodology, Jiahua Kou; software, Jiahua Kou; validation, Jiahua Kou, Zheng Yang and Jiaxuan Cao; formal analysis, Jiahua Kou; investigation, Jiahua Kou; resources, Chengbo Guo, Yanling Guo and Shufa Sun; data curation, Jiahua Kou and Weiye Xing; writing—original draft preparation, Jiahua Kou; writing—review and editing, Chengbo Guo; visualization, Yanling Guo and Weiye Xing; supervision, Chengbo Guo, Yanling Guo and Weiye Xing; project administration, Chengbo Guo, Yanling Guo and Shufa Sun; funding acquisition, Chengbo Guo, Yanling Guo and Shufa Sun. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in SemanticKITTI dataset at <http://www.semantic-kitti.org> and KITTI Raw dataset at http://www.cvlibs.net/datasets/kitti/raw_data.php.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Thrun S, Burgard W, Fox D. Probabilistic robotics. Cambridge, MA, USA: MIT Press; 2005.
2. Zhang J, Singh S. LOAM: lidar odometry and mapping in real-time. In: Proceedings of the Robotics: Science and Systems; 2014 Jul 12–16; Rome, Italy. p. 1–9.
3. Shan T, Englot B, Meyers D, Wang W, Ratti C, Rus D. LIO-SAM: tightly-coupled lidar inertial odometry via smoothing and mapping. In: Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2020 Oct 25–29; Las Vegas, NV, USA. p. 5135–42. doi:10.1109/IROS45743.2020.9341176.
4. Qian C, Xiang Z, Wu Z, Sun H. RF-LIO: removal-first tightly-coupled LiDAR inertial odometry in high dynamic environments. In: Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2021 Sep 27–Oct 1; Prague, Czech Republic. p. 4421–8. doi:10.1109/IROS51168.2021.9636624.
5. Bai C, Xiao T, Chen Y, Wang H, Zhang F, Gao X. Faster-LIO: lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels. IEEE Robot Autom Lett. 2022;7(2):4861–8. doi:10.1109/LRA.2022.3152830.
6. Zhou P, Guo X, Pei X, Chen C. T-LOAM: truncated least squares LiDAR-only odometry and mapping in real time. IEEE Trans Geosci Remote Sens. 2022;60(3):1–13. doi:10.1109/TGRS.2021.3083606.
7. Lim H, Hwang S, Myung H. ERASOR: egocentric ratio of pseudo occupancy-based dynamic object removal for static 3D point cloud map building. IEEE Robot Autom Lett. 2021;6(2):2272–9. doi:10.1109/LRA.2021.3061363.
8. Kim G, Kim A. Remove, then revert: static point cloud map construction using multiresolution range images. In: Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2020 Oct 25–29; Las Vegas, NV, USA. p. 10758–65. doi:10.1109/IROS45743.2020.9340856.
9. Wang K, Guo J, Chen K, Lu J. An in-depth examination of SLAM methods: challenges, advancements, and applications in complex scenes for autonomous driving. IEEE Trans Intell Transp Syst. 2025;26(7):11066–87. doi:10.1109/TITS.2025.3545479.
10. Yoon DJ, Tang TY, Barfoot TD. Mapless online detection of dynamic objects in 3D lidar. In: Proceedings of the 16th Conference on Computer and Robot Vision (CRV); 2019 May 29–31; Kingston, ON, Canada. p. 113–20. doi:10.1109/CRV.2019.00024.

11. Fan T, Shen B, Chen H, Zhang W, Pan J. DynamicFilter: an online dynamic objects removal framework for highly dynamic environments. In: Proceedings of the 2022 International Conference on Robotics and Automation (ICRA); 2022 May 23–27; Philadelphia, PA, USA. p. 7988–94. doi:10.1109/ICRA46639.2022.9812356.
12. Wu R, Fang Z, Pang C, Wu X. OTD: an online dynamic traces removal method based on observation time difference. IEEE Trans Geosci Remote Sens. 2024;62:1–13. doi:10.1109/TGRS.2024.3422084.
13. Xing G, Wu K, Shao S, Zhao H, Liu Y, Lin J, et al. DO-Removal: dynamic object removal for LiDAR-inertial odometry enabled by front-end real-time strategy. IEEE Internet Things J. 2025;12(9):11553–67. doi:10.1109/JIOT.2024.3519577.
14. Park J, Cho Y, Shin Y-S. Nonparametric background model-based LiDAR SLAM in highly dynamic urban environments. IEEE Trans Intell Transp Syst. 2022;23(12):24190–205. doi:10.1109/TITS.2022.3208777.
15. Wu H, Li Y, Xu W, Kong F, Zhang F. Moving event detection from LiDAR point streams. Nat Commun. 2024;15(1):345. doi:10.1038/s41467-023-44554-8.
16. Huang F, Wen W, Zhang J, Wang C, Hsu L-T. Dynamic object-aware LiDAR odometry aided by joint weightings estimation in urban areas. IEEE Trans Intell Veh. 2024;9(2):3345–59. doi:10.1109/TIV.2023.3340218.
17. Pagad S, Agarwal D, Narayanan S, Rangan K, Kim H, Yalla G. Robust method for removing dynamic objects from point clouds. In: Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA); 2020 May 31–Aug 31; Paris, France. p. 10765–71. doi:10.1109/ICRA40945.2020.9197168.
18. Sun J, Dai Y, Zhang X, Xu J, Ai R, Gu W, et al. Efficient spatial-temporal information fusion for LiDAR-based 3D moving object segmentation. In: Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2022 Oct 23–27; Kyoto, Japan. p. 11456–63. doi:10.1109/IROS47612.2022.9981210.
19. Chen X, Milioto A, Palazzolo E, Giguère P, Behley J, Stachniss C. SuMa++: efficient LiDAR-based semantic SLAM. In: Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2019 Nov 3–8; Macau, China. p. 4530–7. doi:10.1109/IROS40897.2019.8967704.
20. Milioto A, Vizzo I, Behley J, Stachniss C. RangeNet++: fast and accurate lidar semantic segmentation. In: Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2019 Nov 3–8; Macau, China. p. 4213–20. doi:10.1109/IROS40897.2019.8967762.
21. Wang W, You X, Zhang X, Chen L, Zhang L, Liu X. LiDAR-based SLAM under semantic constraints in dynamic environments. Remote Sens. 2021;13(18):3651. doi:10.3390/rs13183651.
22. Wang F, Wu Z, Yang Y, Li W, Liu Y, Zhuang Y. Real-time semantic segmentation of LiDAR point clouds on edge devices for unmanned systems. IEEE Trans Instrum Meas. 2023;72:1–11. doi:10.1109/TIM.2023.3295479.
23. Li X, Du S, Li G, Li H. Integrating point-cloud segmentation with 3D lidar scan-matching for mobile robot localization and mapping. Sensors. 2019;20(1):237. doi:10.3390/s20010237.
24. Tian X, Zhu Z, Zhao J, Tian G, Ye C. DL-SLOT: tightly-coupled dynamic LiDAR SLAM and 3D object tracking based on collaborative graph optimization. IEEE Trans Intell Veh. 2024;9(1):1017–27. doi:10.1109/TIV.2023.3317308.
25. Jia Y, Wang T, Cao F, Chen X, Shao S, Liu L. TRLO: an efficient LiDAR odometry with 3-D dynamic object tracking and removal. IEEE Trans Instrum Meas. 2025;74:1–10. doi:10.1109/TIM.2025.3561381.
26. Zhu Z, Zhao J, Huang K, Tian X, Lin J, Ye C. LIMOT: a tightly-coupled system for LiDAR-inertial odometry and multi-object tracking. IEEE Robot Autom Lett. 2024;9(7):6600–7. doi:10.1109/LRA.2024.3402628.
27. Wang Z, Zhang Z, Kang X, Wu M, Chen S, Li Q. DOR-LINS: dynamic objects removal LiDAR-inertial SLAM based on ground pseudo occupancy. IEEE Sensors J. 2023;23(20):24907–15. doi:10.1109/JSEN.2023.3310345.
28. Peng J, Yang Q, Chen D, Yang C, Xu Y, Qin Y. Dynamic SLAM visual odometry based on instance segmentation: a comprehensive review. Comput Mater Contin. 2024;78(1):167–96. doi:10.32604/cmc.2023.041900.
29. Yin M, Qin Y, Peng J. DKP-SLAM: a visual SLAM for dynamic indoor scenes based on object detection and region probability. Comput Mater Contin. 2025;82(1):1329–47. doi:10.32604/cmc.2024.057460.
30. Peng J, Qian W, Zhang H. PPS-SLAM: dynamic visual SLAM with a precise pruning strategy. Comput Mater Contin. 2025;82(2):2849–68. doi:10.32604/cmc.2024.058028.
31. Hornung A, Wurm KM, Bennewitz M, Stachniss C, Burgard W. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. Auton Robots. 2013;34(3):189–206. doi:10.1007/s10514-012-9321-0.

32. Millane A, Taylor Z, Oleynikova H, Nieto J, Siegwart R, Cadena C. C-blox: a scalable and consistent TSDF-based dense mapping approach. In: Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2018 Oct 1–5; Madrid, Spain. p. 995–1002. doi:10.1109/IROS.2018.8593381.
33. Vizzo I, Chen X, Chebrolu N, Behley J, Stachniss C. Poisson surface reconstruction for LiDAR odometry and mapping. In: Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA); 2021 May 30–Jun 5; Xi'an, China. p. 5624–30. doi:10.1109/ICRA48506.2021.9561615.
34. Kühner T, Kümmele J. Large-scale volumetric scene reconstruction using LiDAR. In: Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA); 2020 May 31–Aug 31; Paris, France. p. 6261–7. doi:10.1109/ICRA40945.2020.9197388.
35. Schauer J, Nüchter A. The peopleremover—removing dynamic objects from 3-D point cloud data by traversing a voxel occupancy grid. *IEEE Robot Autom Lett.* 2018;3(3):1679–86. doi:10.1109/LRA.2018.2801797.
36. Pomerleau F, Krüsi P, Colas F, Furgale P, Siegwart R. Long-term 3D map maintenance in dynamic environments. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA); 2014 May 31–Jun 7; Hong Kong, China. p. 3712–9. doi:10.1109/ICRA.2014.6907397.
37. Xiao W, Vallet B, Brédif M, Paparoditis N. Change detection in 3D point clouds acquired by a mobile mapping system. *ISPRS Ann Photogramm Remote Sens Spat Inf Sci.* 2013;II-5/W2:331–6. doi:10.5194/isprsannals-II-5-W2-331-2013.
38. Lim H, Nunes L, Mersch B, Chen X, Behley J, Myung H, et al. ERASOR2: instance-aware robust 3D mapping of the static world in dynamic scenes. In: Proceedings of the Robotics: Science and Systems (RSS); 2023 Jul 10–14; Daegu, Republic of Korea.
39. Jia M, Zhang Q, Yang B, Wu J, Liu M, Jensfelt P. BeautyMap: binary-encoded adaptable ground matrix for dynamic points removal in global maps. *IEEE Robot Autom Lett.* 2024;9(7):6256–63. doi:10.1109/LRA.2024.3402625.
40. Du N, Xie L, Zhou M, Gao W, Wang Y, Hu J. Convex hull triangle mesh-based static mapping in highly dynamic environments. *IEEE Trans Instrum Meas.* 2024;73:1–14. doi:10.1109/TIM.2023.3331405.
41. Arora M, Wiesmann L, Chen X, Stachniss C. Static map generation from 3D LiDAR point clouds exploiting ground segmentation. *Robot Auton Syst.* 2023;159(6):104287. doi:10.1016/j.robot.2022.104287.
42. Duberg D, Zhang Q, Jia M, Jensfelt P. DUFOMap: efficient dynamic awareness mapping. *IEEE Robot Autom Lett.* 2024;9(6):5038–45. doi:10.1109/LRA.2024.3387658.
43. Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O. PointPillars: fast encoders for object detection from point clouds. In: Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019 Jun 15–20; Long Beach, CA, USA. p. 12689–97. doi:10.1109/CVPR.2019.01298.
44. Xu W, Cai Y, He D, Lin J, Zhang F. FAST-LIO2: fast direct LiDAR-inertial odometry. *IEEE Trans Robot.* 2022;38(4):2053–73. doi:10.1109/TRO.2022.3141876.
45. Behley J, Garbade M, Milioto A, Quenzel J, Behnke S, Stachniss C, et al. SemanticKITTI: a dataset for semantic scene understanding of LiDAR sequences. In: Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV); 2019 Oct 27–Nov 2; Seoul, Republic of Korea. p. 9296–306. doi:10.1109/ICCV.2019.00939.
46. Geiger A, Lenz P, Stiller C, Urtasun R. Vision meets robotics: the KITTI dataset. *Int J Robot Res.* 2013;32(11):1231–7. doi:10.1177/0278364913491297.
47. Grupp M. evo: python package for the evaluation of odometry and slam. 2017 [cited 2026 Jan 1]. Available from: <https://github.com/MichaelGrupp/evo>.