Dual-Mode SPL-SLAM: A Robust Semantic Point-Line SLAM System with Monocular Neural Depth and Sensor Fusion for Intelligent Transportation

Zhen Tian[1†], Zhihao Lin[1†], Dezong Zhao[1], *Senior Member, IEEE*, Christos Anagnostopoulos[2], *Member, IEEE*, Peng Wang[4], Wenjing Zhao[1], Lin Wu[1], and Qi Zhang[3,*]

***Abstract*—Simultaneous Localization and Mapping (SLAM) is critical for Intelligent Transportation Systems (ITS), yet existing solutions struggle to balance accuracy, sensor versatility, and computational efficiency. Traditional monocular methods suffer from scale drift, while RGB-D systems with heavy semantic networks often exceed the computational budgets of onboard processors. To address these conflicting constraints, we propose Dual-Mode SPL-SLAM, a unified adaptive framework capable of adapting to different intelligent driving agents. The system intelligently switches between two operating paths: Mode I (Sensor-Semantic) leverages physical depth sensors and high-precision semantic segmentation for complex dynamic environments. To further improve accuracy, we use line-feature processing and a refined epipolar-constraint error for robust detection of known and unknown dynamic objects; Mode II (Neural-Clustering) employs a lightweight neural network (LiteMono) and K-means++ clustering within the object bounding box predicted by GPU-accelerated YOLOX to recover scale and filter dynamic objects using fast LK optical flow in monocular setups. Crucially, both modes converge into a shared Adaptive Point-Line Backend, which dynamically weights features based on environmental texture to ensure robustness. Extensive experiments on KITTI dataset demonstrate that our system achieves state-of-the-art accuracy and real-time performance across diverse sensor configurations.**

***Index Terms*—Visual SLAM, Sensor Fusion, Monocular Depth Estimation, Dynamic Environment, Point-Line Features.**

I. INTRODUCTION

ROBUST perception is the cornerstone of autonomous driving in Intelligent Transportation Systems (ITS) [1]. Visual SLAM (vSLAM) has become a preferred solution due to the cost-effectiveness of cameras [2]. However, real-world deployment faces two distinct challenges: the prevalence of dynamic objects (vehicles, pedestrians) which violate the static world assumption, and the variability of sensor availability across different vehicle platforms. The choice of sensor configuration introduces a fundamental trade-off between accuracy and flexibility. RGB-D sensors provide accurate depth but have limited range and fail under strong outdoor sunlight; monocular cameras are low-cost and flexible but suffer from scale ambiguity and drift [3]. Neural monocular depth estimation can reduce drift [4], yet running both depth estimation and semantic segmentation can overload real-time onboard processors. Existing solutions usually favor one side of this trade-off: RGB-D systems [5] provide accurate short-range localization, whereas monocular systems remain scalable but drift-prone. Moreover, many dynamic SLAM methods rely on geometry alone or semantic models with limited classes [6], [7], leaving a gap for efficient, sensor-adaptive ITS localization in complex traffic scenes.

Beyond DL-based methods, point-feature SLAM systems build maps by detecting and tracking distinctive features [11]. They often fail in environments with smooth surfaces, strong lighting, or weak textures, where sparse features provide insufficient information. As a result, achieving lightweight yet information-rich 3D reconstruction remains challenging.

To resolve these conflicts, we argue against a "one-size-fits-all" approach. Instead, we propose **Dual-Mode SPL-SLAM**, a versatile system that adapts its architecture to the available hardware:

1) **Mode I: Sensor-Semantic Mode.** With RGB-D/Stereo input, depth is available from the sensor; GPU resources are therefore allocated to YOLO-seg and geometric verification. Semantic masks, LK optical flow, epipolar constraints, and line features [12] jointly support precise dynamic culling and structural localization.
2) **Mode II: Neural-Clustering Mode.** With monocular input, LiteMono recovers dense depth and YOLOX [13] supplies lightweight object boxes. To avoid a second heavy network, foregrounds are separated by K-means++ clustering over intensity and neural depth, with LK optical flow providing dynamic judgment.

Both modes feed into a unified **Adaptive Point-Line Back-**

[1]Zhen Tian, Zhihao Lin, Dezong Zhao, Wenjing Zhao, and Lin Wu are with the James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, U.K. (e-mail: 2620920z@student.gla.ac.uk, 28004001@student.gla.ac.uk, dezong.zhao@glasgow.ac.uk, wenjing.zhao@glasgow.ac.uk).

[2]Christos Anagnostopoulos is with the School of Computing Science, University of Glasgow, G12 8RZ Glasgow, U.K. (e-mail: Christos.Anagnostopoulos@glasgow.ac.uk).

[3] Qi Zhang is with the Informatics Institute at University of Amsterdam, Amsterdam, the Netherlands. (e-mail: q.zhang2@uva.nl).

[4] Peng Wang is with the School of Engineering at Westlake University in collaboration with Zhejiang University, Hangzhou, China. (e-mail: wangpeng@westlake.edu.cn).

*Corresponding author: Qi Zhang (e-mail: q.zhang2@uva.nl).

† Equal contribution

TABLE I
COMPREHENSIVE COMPARISON BETWEEN SPL-SLAM MODE I AND EXISTING POINT-LINE SLAM SYSTEMS.

Feature	PLP-SLAM [8]	PL-SLAM [9]	IPF-SLAM [10]	SPL-SLAM (Ours)
Dynamic Object Handling				
Dynamic detection approach	None	None	None	Semantic + Geometric
Label-free dynamic detection	✗	✗	✗	✓
Unknown dynamic object detection	✗	✗	✗	✓
Optical flow integration	✗	✗	✗	✓ (LK optical flow)
Epipolar constraints for dynamics	✗	✗	✗	✓
Selective feature removal	✗	✗	✗	✓ (Proportion-based)
Feature Representation & Optimization				
Line feature representation	Endpoint-based	Endpoint-based	Angle + endpoint	Hybrid error metric
Multi-feature optimization	Fixed weighting	Fixed weighting	Adaptive to camera motion	Adaptive to environment
Feature weighting approach	Manual tuning	Manual tuning	Camera motion based	Environment context based
Bundle adjustment formulation	Standard	Standard	Enhanced with angle	Hybrid with adaptive weights
Grid ID matching for efficiency	✗	✗	✗	✓
Semantic Integration				
Semantic segmentation	✗	✗	✗	✓ (YOLOv5s-seg)
Object categories	N/A	N/A	N/A	80
Semantic-geometric fusion	✗	✗	✗	✓
System Capabilities				
Outdoor environment support	Limited	Limited	Limited	✓
Indoor low-texture performance	Moderate	Good	Good	Excellent
Multi-camera support	Stereo only	Stereo only	Stereo only	Stereo and RGB-D
Real-time performance	Yes	Yes	Yes	Yes
Real-time in dynamic scenes	Degraded	Degraded	Partial	✓

✓ Feature fully supported; ✗ Feature not supported; N/A Not applicable

end, ensuring structural consistency in texture-poor environments (e.g., corridors, highways). Our contributions are:

- A switchable SLAM framework bridging high-precision sensor fusion and flexible monocular applications.
- A semantic point-line integration strategy (Mode I) that combines deep learning with epipolar constraints to handle unknown dynamic objects.
- A resource-efficient neural-clustering frontend (Mode II) that solves monocular scale drift without heavy semantic overhead.
- A unified backend with context-adaptive feature weighting, achieving state-of-the-art performance on public datasets.

II. RELATED WORKS

A. Dynamic and Semantic SLAM

Semantic segmentation provides richer priors than object detection for dynamic SLAM. DS-SLAM [5] combines SegNet masks [6] with LK optical flow, while later works employed BlitzNet [14], DeepLab v3+ [15], or motion-probability models [16] to refine segmentation and filtering. These methods retain more static keypoints than detection-based systems [17], but remain constrained by predefined labels and limited semantic classes (around 20 vs. 80+ in real environments).

Recent research seeks label-free consistency checks for dynamic–static judgment. For instance, thresholding and ambiguity constraints improve bounding-box stability, reprojection error aids in recovering static points, and feature redistribution mitigates motion blur [18], [19]. However, most of these methods still depend on object detection, limiting generalization to unknown objects.

B. Hybrid Monocular VO

Neural depth and learned feature methods have been used to reduce monocular scale ambiguity. CNNS-LAM [20], CodeSLAM [21], and DeepFactor [22] integrate learned depth or compact depth codes into SLAM, while LIFT-based VO [3] and DROID-SLAM [23] improve feature or pose estimation through deep networks. These methods improve robustness but typically require fine-tuning or substantial computation, motivating our lightweight Mode II design.

C. Point-Line SLAM

PLP-SLAM [24], PL-SLAM [9], and related stereo point-line systems improve localization in weak-texture scenes, but they provide limited dynamic-object suppression and are usually validated in controlled settings. Compared with these systems and recent implicit representations such as NeRF/Gaussian-SLAM variants [25]–[28], SPL-SLAM targets real-time ITS deployment by combining semantic/geometric dynamic filtering with adaptive point-line optimization across RGB-D, stereo, and monocular inputs.

III. SYSTEM FRAMEWORK

The proposed **Dual-Mode SPL-SLAM** framework is designed to bridge high-precision sensor-fusion SLAM and flexible monocular SLAM in intelligent transportation scenarios. As illustrated in Fig. 1, the system is not a fixed pipeline but an adaptive decision tree that configures the front-end according to the available sensor modality and computational budget. The system first receives a sequence of frames and identifies the input configuration. RGB-D/Stereo inputs activate **Mode I (Sensor-Semantic Path)**, which uses the available depth

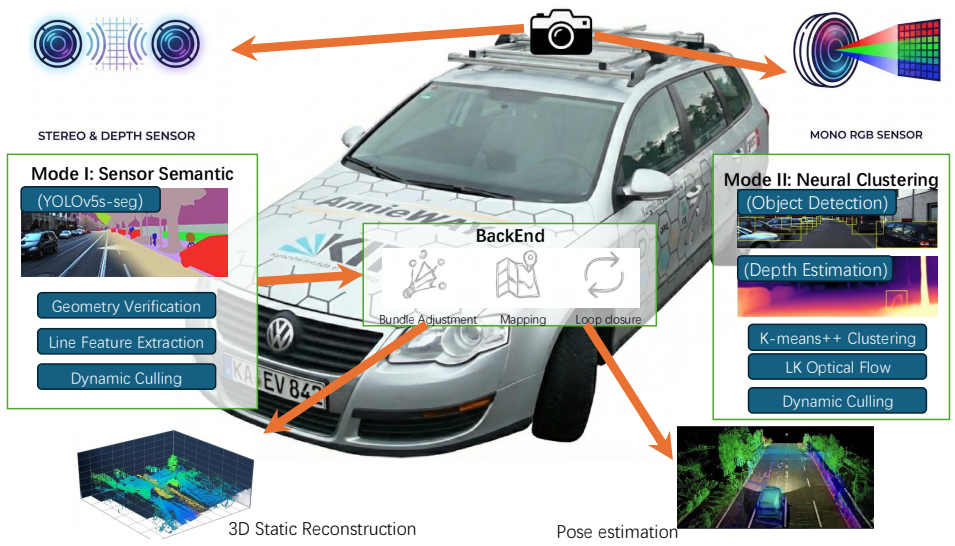


Fig. 1. Workflow of the proposed system. The vehicle depicted serves as our experimental hardware platform. Equipped with radar and a multi-sensor suite, it facilitates diverse evaluations and supports various sensor configurations, including both monocular and stereo vision setups.

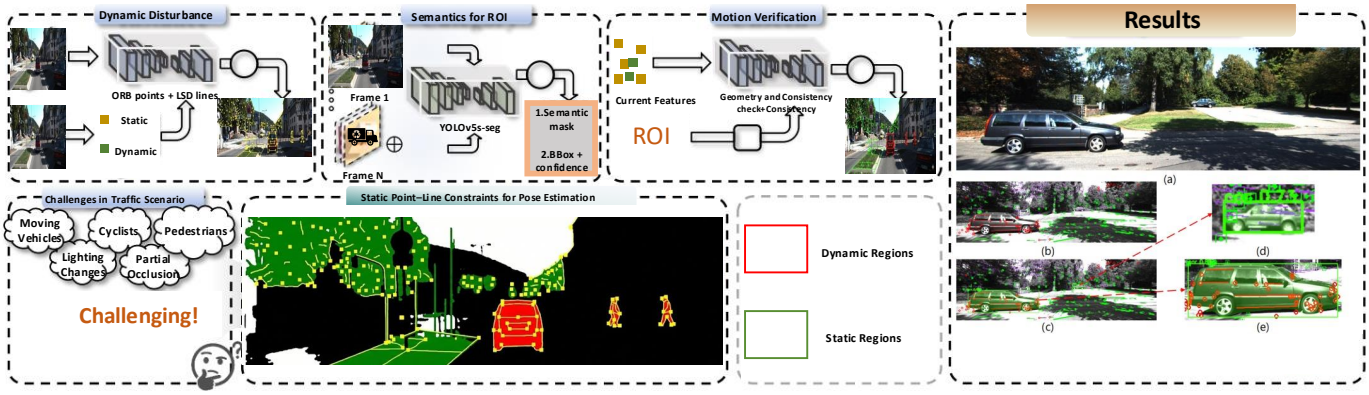


Fig. 2. Mode I semantic-geometric dynamic filtering overview. This cropped view separates the dynamic-disturbance, semantic ROI, motion-verification, and static point-line constraint components for sensor-rich RGB-D/Stereo ITS settings.

measurements and allocates GPU resources to YOLOv5s-seg, strict epipolar verification, and point-line dynamic filtering. Monocular input activates **Mode II (Neural-Clustering Path)**, where LiteMono recovers scale and lightweight K-means++ clustering replaces heavy segmentation to maintain real-time tracking. Both paths feed a unified **Adaptive Point-Line Backend**, which performs local and global BA for robust trajectory estimation and mapping across high-end autonomous vehicles and low-cost driver-assistance platforms. Mode I is further detailed below because it performs the semantic-geometric dynamic filtering used in sensor-rich settings.

IV. MODE I: SENSOR-SEMANTIC FRONT-END

As shown in Figs. 2 and 3, Mode I consists of two main stages: (i) *Potential Dynamic Area Judgment* and (ii) *Dynamic Feature Culling*. The system receives RGB and depth frames from sequential images and processes them through a semantic segmentation network to produce masks, bounding boxes, class probabilities, and confidence scores. These semantic cues are treated as coarse priors rather than final dynamic labels. Potential dynamic areas are then verified using LK optical flow and epipolar constraints, after which confirmed dynamic features are culled before point-line optimization. This design preserves useful static structures inside or near semantic boxes while removing unstable moving features.

A. Line Feature-assisted and Dynamic Feature-filtering SLAM

Building on Section III, this section details Mode I: front-end feature extraction, semantic-geometric dynamic filtering, proportion-based dynamic classification with $\tau_d = 40\%$, and the shared adaptive point-line backend used by both modes. The proposed Line Feature-assisted and Dynamic Feature-filtering SLAM system introduces several refinements. First, it employs grid-ID line matching to avoid exhaustive coordinate searches while preserving matching accuracy. Second, a multi-stage dynamic verification pipeline combines semantic masks, optical-flow consistency, and epipolar constraints, enabling robust detection of both known and unknown dynamic objects. Third, the proportion-based strategy removes an entire object region only when sufficient verified dynamic evidence is present, preserving static features near dynamic participants. Finally, two-step line optimization using reference keyframes

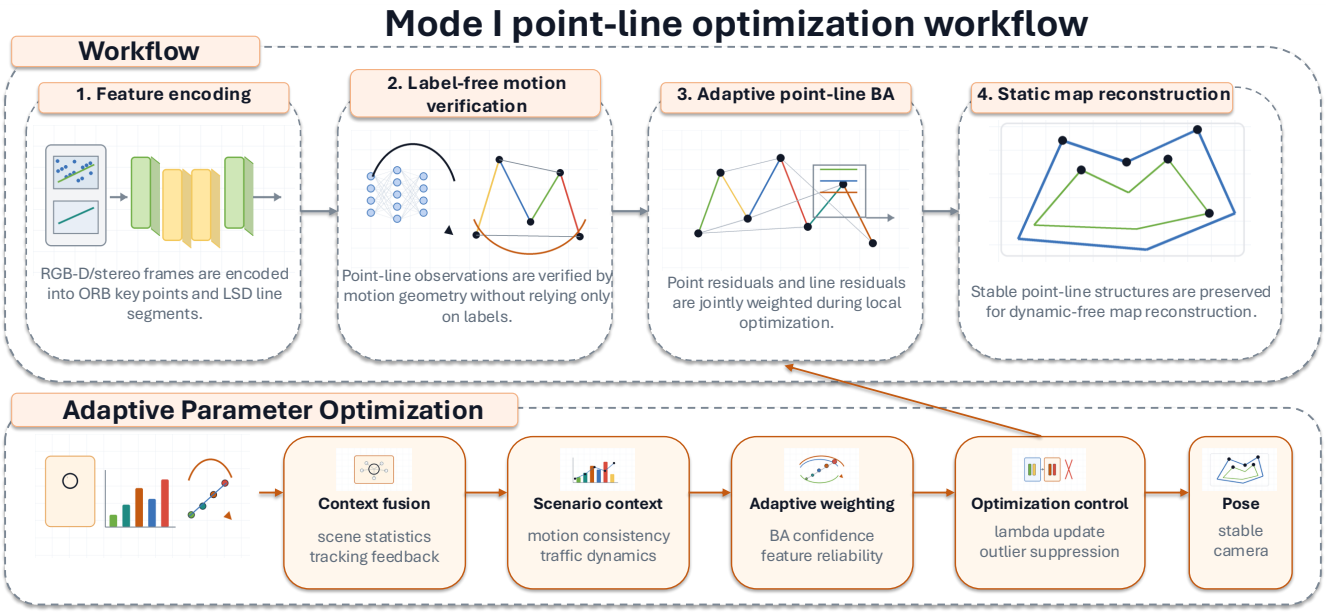


Fig. 3. Mode I point-line optimization workflow. This cropped view shows the feature-encoding, label-free motion verification, adaptive point-line bundle adjustment, static map reconstruction, and adaptive parameter optimization components using the same visual language as Fig. 2.

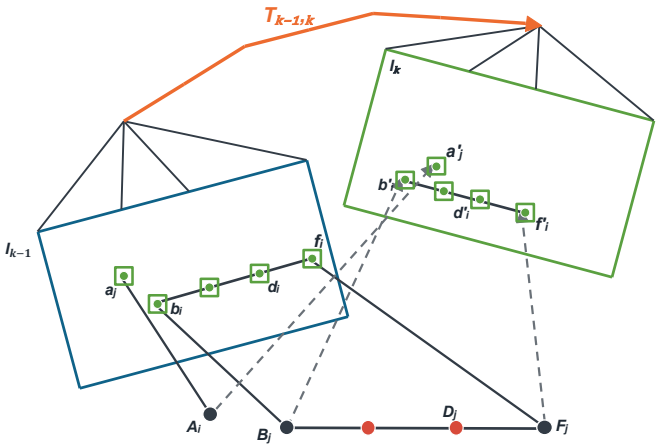


Fig. 4. Illustration of the line feature matching process in two frames.

improves robustness against sensor noise and environmental dynamics.

1) Line Features-assisted SLAM: Dynamic objects usually provide many unstable point features but fewer persistent straight lines, whereas static infrastructure such as buildings and road markings contributes reliable line cues.

a) Line Features Matching: For stereo line matching, the image is divided into regular grids and each detected line segment is assigned a grid ID according to its projected location. Candidate matches are searched only among line segments sharing the same or neighboring grid IDs, rather than over the full image. The closest descriptor pair is then selected and verified by two geometric checks: sufficient segment overlap and a small direction-angle difference. For a 3D segment with endpoints B_j and F_j, the relative pose $T_{k-1,k} \in SE(3)$ maps its image projections between consecutive frames. A valid match is therefore required to satisfy descriptor consistency, spatial-grid consistency, overlap consistency, and orientation consistency. This procedure reduces unnecessary coordinate search while preserving reliable line correspondences for pose estimation.

b) Line Feature Attributes Updating Algorithm: After a line pair is matched, Mode I updates its attributes before inserting it into the local map. First, the start and end depths of the segment are checked to reject unstable or degenerate observations. Second, the line is represented by homogeneous endpoint vectors so that it can participate in pose optimization together with point features. Third, bidirectional cosine similarity is used to filter weak line pairs, and matches with an overlap ratio above 0.75 are treated as stable. Finally, a disparity threshold removes near-degenerate stereo matches; depth is computed as $D = K/D_v$, where K is the focal-length ratio and D_v is stereo disparity or the RGB-D depth value. These updates improve co-visibility maintenance and ensure that only geometrically stable structural lines contribute to local mapping and bundle adjustment.

c) Line Features Optimizing Algorithm: Mode I optimizes line features by selecting a reference keyframe with reliable viewing geometry and low reprojection error, then updating each map line relative to the corrected reference pose. Let R_{wr} and t_{wr} transform the reference keyframe to the world frame. The start/end map-line coordinates $P_{3\mathrm{Dw_{sp}}}$ and $P_{3\mathrm{Dw_{ep}}}$ are corrected by the Sim3 transforms S_{rw} and $\mathrm{corS_{wr}}$: $\mathrm{CorP}_{3\mathrm{Dw_{sp}}} = \mathrm{corS_{wr}} \cdot (S_{\mathrm{rw}} \cdot P_{3\mathrm{Dw_{sp}}}), \mathrm{CorP}_{3\mathrm{Dw_{ep}}} = \mathrm{corS_{wr}} \cdot (S_{\mathrm{rw}} \cdot P_{3\mathrm{Dw_{ep}}})$ where $\cdot$ denotes matrix multiplication. These corrections reduce line-pose errors caused by sensor noise and dynamic interference.

2) Dynamic Features Filtering: Semantic-only dynamic SLAM [29] depends on predefined labels, geometric-only methods [30] are sensitive to slowly moving objects, and hybrid systems [5] often remain label-limited. Mode I therefore treats YOLOv5s-seg masks only as *potentially dynamic* and verifies motion using Lucas–Kanade optical flow and epipolar geometry. Fundamental-matrix constraints correct semantic

Algorithm 1 Dynamic Points Culling Algorithm

Input: Bounding boxes B_n, current key points P_n, dynamic points DP_n, threshold $\tau_d = 0.4$.
Output: Static features S_n.

$S_n \leftarrow P_n$
for each bounding box $B_i \in B_n$ **do**
  Let $P_i = \{p \in P_n \mid p \in B_i\}$ and $D_{in} = \{d \in DP_n \mid d \in B_i\}$.
  if $|P_i| = 0$ **then**
    continue
  end if
  if $|D_{in}|/|P_i| > \tau_d$ **then**
    $S_n \leftarrow S_n \setminus P_i$ ▷ Remove all points in box
  else
    $S_n \leftarrow S_n \setminus \{p \in (P_n \cap B_i) \mid \min_{d \in D_{in}} \text{dist}(p, d) \leq 15\}$
  end if
end for

false positives, and proportion-based filtering removes an entire region only when the verified dynamic ratio exceeds τ_d; otherwise, only nearby unreliable points are suppressed. This preserves static features close to moving objects and improves generalization to unknown dynamics.

a) Dynamic Feature Detection: Dynamic features are detected by extending DS-SLAM [5] with Harris corners and LK optical-flow pyramids [31]. Similar to denoising schemes in sensor processing [32], optical-flow consistency removes noisy matches and improves reliability.

In contrast to traditional Harris corners [33], the dynamic feature detection selectively omits matches located close to pixel edges or in regions exhibit significant disparity. This refinement effectively narrows down the set of potential candidate points, increasing the efficiency of detection. Besides, dynamic feature detection involves assessing the spatial relation of each retained point to its respective epipolar line. Points that deviate beyond a set distance threshold are deemed non-conforming. Therefore, the SLAM system integrated with dynamic feature detection achieves threshold-based exclusion of non-essential points, enhancing the overall accuracy of dynamic element detection.

A RANSAC-based technique [30] is used to find the fundamental matrix, which maximizes the number of inliers. The fundamental matrix is essential for ensuring accurate point matching between frames. The accurate matching is achieved by calculating the polar line for the current frame and aligning points from the previous frame based on their corresponding epipolar lines.

Assuming the coordinates of matched points in consecutive frames are denoted as p_1 and p_2, and their homogeneous counterparts as P_1 and P_2, the following relationship holds:

$$P_1 = [u_1, v_1, 1], P_2 = [u_2, v_2, 1], p_1 = [u_1, v_1], p_2 = [u_2, v_2] \quad (1)$$

where u, v represent the image coordinate components. The associated epipolar line $L_1 = [X, Y, Z] = FP_1$ where X, Y, Z are components of the line vector, and F denotes the fundamental matrix. The distance between a matched point and its corresponding epipolar line is formulated as

$$D = |P_2^\top F P_1| / \sqrt{\|X^2\| + \|Y^2\|} \quad (2)$$

where D signifies the distance. If D surpasses a predetermined threshold, the corresponding features are classified as outliers.

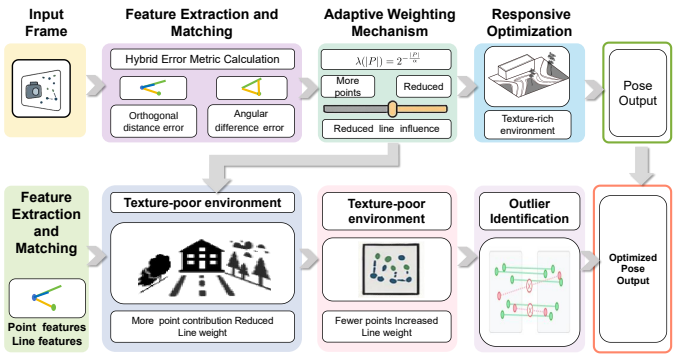


Fig. 5. Innovative Flow of the Proposed Point-Line Joint Bundle Adjustment Framework.

b) Dynamic Feature Culling: After detecting dynamic features, a feature filtering algorithm is used to retain essential data. Specifically, we employ the Euclidean distance to measure the distance between feature points. For two points (u_1, v_1) and (u_2, v_2) in a two-dimensional plane, their Euclidean distance is defined as

$$d((u_1, v_1), (u_2, v_2)) = \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2} \quad (3)$$

This distance metric is used to determine whether feature points are close enough to be considered dynamic. Key features within a 15-pixel radius of any dynamic feature are considered appended to these dynamic features and do not require further processing to determine whether they are dynamic or static objects. For scenarios involving significantly higher resolutions, the framework allows automatic scaling of the threshold based on intrinsic camera parameters. However, existing experiments indicate that the 15-pixel threshold is effective across standard resolutions, ranging from 640×480 to 1920×1080. In addition, if more than $\tau_d = 40\ \%$ of the features within a semantic object mask are verified as dynamic, the entire object is considered dynamic, and all features in that mask do not require further processing. The threshold $\tau_d = 40\ \%$ is the default setting used in all main experiments and is further justified by the sensitivity results in Table VI. The overall algorithm is illustrated in Algorithm 1.

For features outside the semantic boxes, and the dynamic points detected outside the marked region are directly removed to eliminate unstable data associations, which enhances the adaptation to the unknown objects.

B. Error-handling Point-Line Joint Bundle Adjustment

1) Point–Line Joint Bundle Adjustment: The proposed framework (Fig. 5) introduces several innovations that enhance robustness and distinguish it from existing methods. First, a hybrid error metric unifies orthogonal distance and angular difference errors, managed by a diagonal weighting matrix that suppresses outliers. Second, an adaptive weighting scheme

$$\lambda(|P|) = 2^{-\frac{|P|}{\alpha}}$$

dynamically balances point and line contributions according to the number of inlier matches. This scheme reduces line influence in texture-rich environments while increasing it in texture-poor scenes, ensuring consistent performance across diverse conditions.

In addition, both motion-only and local BA are integrated within a unified pipeline, jointly optimizing camera poses and map features for consistency. The framework also incorporates a robust outlier management strategy, further improving the reliability of erroneous point and line rejection. Together, these mechanisms allow SPL-SLAM to maintain high accuracy despite dynamic interference, feature tracking errors, or sensor noise, outperforming prior approaches that rely on fixed weights or separate optimization of points and lines.

2) Line Segment Reprojection Error Formulations: Denote the pose of the i-th camera frame as $T_{\text{iw}} \in SE(3)$ [34], which describes the pose of objects in 3D space, encompassing both position and orientation. For a line segment k observed in this frame, its 3D endpoints in the world coordinate system are represented as $P_k, Q_k \in \mathbb{R}^3$, while the corresponding 2D projections on the image plane are denoted by $p_k^i, q_k^i \in \mathbb{R}^2$. P_k and Q_k are two different points in three-dimensional space, p_k^i and q_k^i are two different points in two-dimensional space, $\mathbb{R}^3$ denotes a three-dimensional Euclidean space, and $\mathbb{R}^2$ denotes a two-dimensional Euclidean space. This paper examines various reprojection error formulations to effectively use line segments in the optimization process.

a) Orthogonal Distance Error: Inspired by [9], this formulation calculates the orthogonal distances between the projected 3D line endpoints and the observed 2D line in the image. The line on the image plane is parameterized as

$$l_k^i = p_k^i \times q_k^i / \|p_k^i \times q_k^i\| \tag{4}$$

where $p_k^i, q_k^i \in \mathbb{R}^3$ are the homogeneous coordinates of the 2D endpoints. The orthogonal distance error is defined as

$$e_{k,\perp}^i = \begin{bmatrix} l_k^{i\top} p_k' & l_k^{i\top} q_k' \end{bmatrix}^\top \tag{5}$$

where $p_k' = \pi(T_{\text{iw}}, P_k), q_k' = \pi(T_{\text{iw}}, Q_k)$ denote the homogeneous coordinates of the projected 3D endpoints, computed using the projection function $\pi : SE(3) \times \mathbb{R}^3 \to \mathbb{R}^3$:

$$\begin{aligned} \pi(T_{\text{iw}}, P_k) &= \begin{bmatrix} f_x \frac{X}{Z} + c_x & f_y \frac{Y}{Z} + c_y & 1 \end{bmatrix}^\top, \\ \begin{bmatrix} X & Y & Z \end{bmatrix}^\top &= R_{\text{iw}} P_k + t_{\text{iw}} \end{aligned} \tag{6}$$

where (f_x, f_y) represents the focal lengths, (c_x, c_y) denotes the principal point, and $R_{\text{iw}} \in SO(3), t_{\text{iw}} \in \mathbb{R}^3$ are the rotational and translational components of the camera pose T_{iw}, respectively.

b) Angular Difference Error: Adapting the idea from [10], this error function considers the angular difference between the observed and projected 2D line segments. Denote $p_k'^i, q_k'^i \in \mathbb{R}^2$ as the projected endpoints of the 3D line segment. The angular difference error is calculated as

$$e_{k,\angle}^i = \left[\arccos\left(\frac{\overrightarrow{p_k^i q_k^i} \cdot \overrightarrow{p_k'^i q_k'^i}}{\|\overrightarrow{p_k^i q_k^i}\| \|\overrightarrow{p_k'^i q_k'^i}\|} \right) \arccos\left(\frac{\overrightarrow{q_k^i p_k^i} \cdot \overrightarrow{q_k'^i p_k'^i}}{\|\overrightarrow{q_k^i p_k^i}\| \|\overrightarrow{q_k'^i p_k'^i}\|} \right) \right]^\top \tag{7}$$

c) Hybrid Error Metric: To leverage the advantages of both orthogonal and angular distances, a hybrid error function e_k^i is proposed that combines $e_{k,\perp}^i$ and $e_{k,\angle}^i$. To account for the different units and scales of these terms, a diagonal weight matrix $W_k \in \mathbb{R}^{4\times4}$ is introduced to balance their contributions: $e_k^i = W_k \begin{bmatrix} e_{k,\perp}^i & e_{k,\angle}^i \end{bmatrix}^\top$ The weight matrix W_k is tuned such that each term equals 1 at its respective outlier rejection threshold, enabling reliable outlier detection and appropriate weighting of point and line errors during BA. Line matches with $e_k^{i\top} e_k^i > \epsilon_{\text{line}}$ are classified as outliers, where ϵ_{line} is a predefined threshold.

3) Adaptive Point and Line Feature Integration in Bundle Adjustment: To adaptively react with the influence of point and line features, a weighting scheme is proposed using the factor $\lambda(|P|) = 2^{-|P|/\alpha}$, where $|P|$ is the number of inlier point matches and α is a tunable parameter. In environments with abundant point features, λ reduces the contribution of lines to mitigate the impact of potential mismatches. In low-texture environments with sparse point features, λ increases the weight of lines to compensate for the lack of reliable point matches. This adaptive approach enhances the overall robustness and accuracy of the SLAM system in various scenarios.

Considering both point matches P and line matches L, the point-line BA objective for frame i is formulated as

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \left(\sum_{j\in P} \rho(e_j^\top \Sigma_{e_j}^{-1} e_j) + \lambda(|P|) \sum_{k\in L} \rho(e_k^{i\top} e_k^i) \right) \tag{8}$$

where ω represents the pose parameters to be optimized, $\rho(.)$ denotes the robust Huber loss function, e_j is the reprojection error for the j-th point match, and Σ_{e_j} represents its associated covariance matrix.

The adaptive weighting strategy is incorporated into both motion-only BA for frame-to-frame tracking and local BA for keyframe optimization. In motion-only BA, the camera pose $T_{\text{iw}} = \{R_{\text{iw}}, t_{\text{iw}}\}$ is estimated by minimizing the following objective function:

$$\begin{aligned} T_{\text{iw}} = \arg\min_{T_{\text{iw}}} \Bigg(& \sum_{j\in\Lambda_P} \rho\left(\left\| x_j' - \pi\left(T_{\text{iw}}, X_j'\right) \right\|^2_{\Sigma_{x_j'}} \right) \\ & + \lambda(|\Lambda_P|) \sum_{k\in\Lambda_L} \rho\left((e_k^i)^\top e_k^i \right) \Bigg) \end{aligned} \tag{9}$$

where Λ_P and Λ_L are the sets of point and line matches, respectively, x_j' is the observed keypoint, X_j' is the corresponding 3D point in the world coordinate system, and $\Sigma_{x_j'}$ is the covariance matrix associated with the keypoint scale.

Similarly, local BA optimizes the keyframe poses $\{T_{\text{lw}} \mid l \in \mathcal{K}_L\}$ and 3D point positions $\{X_j \mid j \in \mathcal{P}_L\}$ using the local keyframe set $\mathcal{K}_L$ and the associated point features $\mathcal{P}_L$:

$$\begin{aligned} & \{X_j, T_{\text{lw}} \mid j \in \mathcal{P}_L, l \in \mathcal{K}_L\} \\ & = \arg\min_{X_j, T_{\text{lw}}} \left(\sum_{k\in\mathcal{K}_L\cup\mathcal{K}_F} \sum_{j\in\lambda_k} \rho\left(E_{\text{kj}}\right) + \lambda(|\lambda|) \sum_{l\in\mathcal{K}_L} \sum_{m\in L_l} \rho\left(E_{\text{lm}}\right) \right) \end{aligned} \tag{10}$$

where $\lambda = \bigcup_{k\in\mathcal{K}_L\cup\mathcal{K}_F} \lambda_k$ is the unified set of point matches, $\mathcal{K}_F$ is the set of fixed keyframes, E_{kj} is the point reprojection error in keyframe k, L_l is the set of line matches in keyframe l, and E_{lm} is the line reprojection error.

4) Integrated Motion-only and Local Bundle Adjustment: In addition to the adaptive point-line BA, the proposed SLAM system also incorporates motion-only BA and local BA for

further optimization. The projection function π_s for stereo cameras is defined as

$$\begin{aligned} \pi_s(T_{\text{iw}}, P_j) &= \begin{bmatrix} f_x \frac{X}{Z} + c_x & f_y \frac{Y}{Z} & \frac{f_x}{b}(X - b) \end{bmatrix}^\top, \\ \begin{bmatrix} X & Y & Z \end{bmatrix}^\top &= R_{\text{iw}} P_j + t_{\text{iw}} \end{aligned} \tag{11}$$

where f_x, f_y are the focal lengths, c_x, c_y represent the principal point, b is the baseline between the stereo cameras, and $P_j \in \mathbb{R}^3$ is the 3D point in world coordinates. Similarly, the projection function π_d for RGB-D cameras is defined as

$$\begin{aligned} \pi_d(T_{\text{iw}}, P_j) &= \begin{bmatrix} f_x \frac{X}{Z} + c_x & f_y \frac{Y}{Z} + c_y & Z \end{bmatrix}^\top, \\ \begin{bmatrix} X & Y & Z \end{bmatrix}^\top &= R_{\text{iw}} P_j + t_{\text{iw}} \end{aligned} \tag{12}$$

where Z represents the depth value of the 3D point P_j in camera coordinates. Motion-only BA focuses on refining the camera orientation $R_{\text{iw}} \in SO(3)$ and position $t_{\text{iw}} \in \mathbb{R}^3$. Specifically, the camera orientation is refined by minimizing the reprojection error between matched 3D points P_j in world coordinates and their corresponding image keypoints p_j^i. The corresponding images can be either stereo $p_{s,j}^i \in \mathbb{R}^3$ or RGB-D $p_{d,j}^i \in \mathbb{R}^3$, with $j \in \Lambda$ representing the set of all matches:

$$R_{\text{iw}}, t_{\text{iw}} = \arg\min_{R_{\text{iw}}, t_{\text{iw}}} \sum_{j \in \Lambda} \rho \left(\left| p_j^i - \pi \left(T_{\text{iw}}, P_j \right) \right|^2_{\Sigma_{p_j^i}} \right) \tag{13}$$

where ρ denotes the robust Huber cost function. $\Sigma_{p_j^i}$ represents the covariance matrix related to the scale of the keypoint. π is the projection function. The projection functions π_s for stereo cameras and π_d for RGB-D cameras are defined in (11) and (12), respectively.

Local BA optimizes a subset of covisible keyframes $\mathcal{K}_L$ and all points observed in those frames $\mathcal{P}_L$. The optimization is formulated as

$$P_j, T_{lw} \mid j \in \mathcal{P}_L, l \in \mathcal{K}_L = \arg\min_{P_j, T_{lw}} \sum_{k \in \mathcal{K}_L \cup \mathcal{K}_F} \sum_{j \in \lambda_k} \rho \left(E_{kj} \right) \tag{14}$$

where P_j is the 3D position of the j-th point in $\mathcal{P}_L$. T_{lw} is the pose of the l-th keyframe in $\mathcal{K}_L$. $\mathcal{K}_F$ is the set of fixed keyframes that contribute observations of $\mathcal{P}_L$ without being optimized. λ_k is the set of matches between points in $\mathcal{P}_L$ and keypoints in keyframe k. E_{kj} is the reprojection error.

In contrast to local BA, full BA optimizes all keyframes and map points simultaneously, except for the origin keyframe which remains fixed to resolve scale ambiguity. This extensive optimization refines camera poses and landmark positions across entire map, leveraging all available visual information to achieve the highest accuracy. By incorporating these various BA formulations, the proposed SLAM system can effectively optimize the camera trajectory and map structure, improving overall performance and robustness in various scenarios.

V. MODE II: NEURAL-CLUSTERING MONOCULAR FRONT-END

Fig. 6 shows Mode II. A single RGB image is processed by four parallel threads: LK optical-flow/RANSAC outlier detection, YOLOX object detection, ORB feature extraction, and LiteMono depth estimation. Points detected by LK optical flow that do not satisfy the epipolar constraint are considered dynamic candidates. The system then obtains potentially moving bounding boxes according to the proportion and count of dynamic points in each object box. Within these boxes, weighted K-means++ clustering over grayscale and neural-depth cues separates dynamic foregrounds from static backgrounds. Foreground ORB points are removed, and the remaining static features with dense depth form a pseudo-RGB-D frame for ORB-SLAM3 tracking and local optimization. This design avoids running a second semantic segmentation network while retaining geometric interpretability and real-time performance.

A. Depth estimation enhanced monocular VO

At the beginning of processing, the system feeds each frame into a neural network and uses the Lanczos kernel's high-quality resampling method to reduce the errors brought by the bilinear interpolation for image size changes. After obtaining the disparity map, the system uses the following formula to convert the disparity values into corresponding depth values:

$$Depth = \frac{mScaleFactor}{0.01 + (10.0f - 0.01f) \times disparity} \tag{15}$$

where $disparity$ is predicted by the neural network, 0.01 prevents division by zero, and $mScaleFactor$ aligns the predicted-depth median with the real-depth median:

$$\begin{aligned} mScaleFactor &= \frac{RDepth}{DepthMapFactor} \times D \\ D &= \frac{1}{0.01 + (10.0f - 0.01f) \times disparity} \end{aligned} \tag{16}$$

where $DepthMapFactor$ is the calibration parameter and $RDepth$ is the reference depth. After outlier processing, the depth map is used as the dense depth input corresponding to the RGB frame and also supports depth-aware clustering in dynamic boxes. Depth values are clipped to $[10^{-3}, 80]$ following [4], and invalid INF/NaN entries are removed before tracking. Fig. 7 shows representative KITTI depth predictions under ego-motion and dynamic vehicles. The left column shows original RGB images, while the right column visualizes the corresponding dense depth maps; brighter regions indicate shallower depth and darker regions indicate larger depth.

B. Optical flow-based moving objects detection

Mode II uses LK sparse optical flow because dense optical-flow networks impose a high computational cost when combined with monocular depth estimation. LK tracks sparse corner points between consecutive frames, and RANSAC estimates the dominant motion model from inlier correspondences. Outliers inconsistent with this model are treated as dynamic candidates. The resulting fundamental matrix defines epipolar lines for matched points and provides a lightweight geometric test for motion consistency. Let p_1, p_2 be sparse matched corners in two frames and P_1, P_2 their homogeneous forms [5]:

$$\begin{aligned} &P_1 = [u_1, v_1, 1],\ P_2 = [u_2, v_2, 1], \\ &p_1 = [u_1, v_1],\ p_2 = [u_2, v_2]. \end{aligned} \tag{17}$$

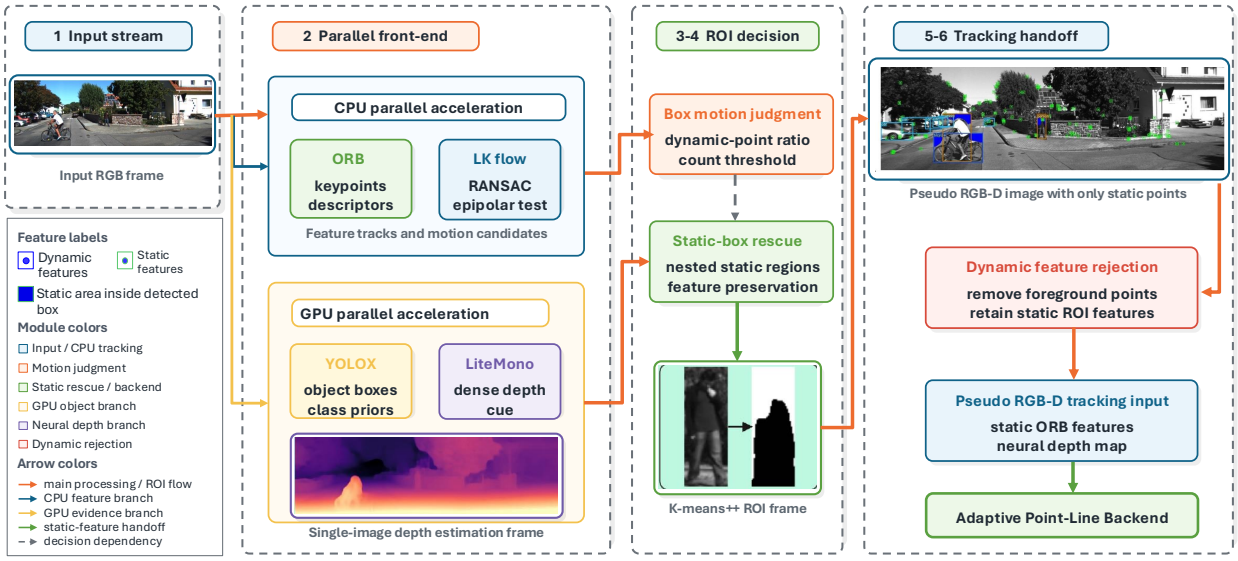


Fig. 6. Architecture of the **Mode II: Neural-Clustering Monocular Front-End**. The diagram clarifies the input stream, CPU/GPU parallel front-end, ROI motion decision, static-box rescue, dynamic feature rejection, and handoff to pseudo-RGB-D tracking.

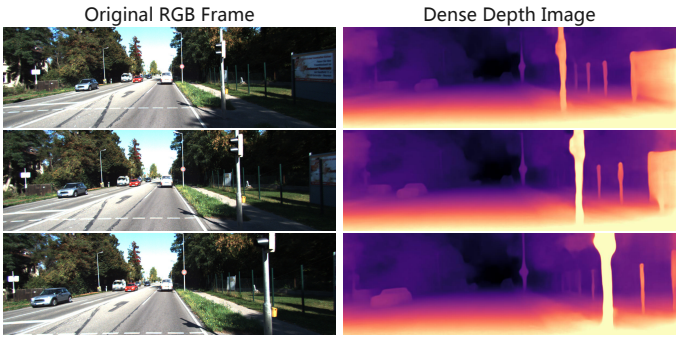


Fig. 7. Depth-estimation examples on representative KITTI odometry outdoor driving frames from sequences 00–10, following the same split used in Tables II and III. The left column shows original RGB images, and the right column shows dense depth maps predicted from the corresponding images. The continuous movement of the utility poles indicates camera ego-motion, while the distinct motion pattern of the vehicles relative to the poles indicates dynamic traffic participants.

where u, v are the coordinate values in the 2D RGB image. Then we use L_1 to represent the epipolar line, which is calculated as follows:

$$L_1 = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = FP_1 = F \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \tag{18}$$

where X, Y, Z represent the line vectors, and F represents the fundamental matrix solved by the RANSAC model. The distance between the matching point and its corresponding epipolar line is as follows:

$$D = \frac{|P_2^T F P_1|}{\sqrt{X^2 + Y^2}}, \tag{19}$$

where D is the epipolar distance. Points exceeding the threshold are marked as dynamic candidates and projected into YOLOX [13] boxes so that features on the same object share a dynamic/static state. Because RANSAC may misclassify a few static points, using only a fixed count or only a proportion threshold is unstable across boxes with different texture densities. A texture-rich static object may contain many false dynamic outliers, whereas a texture-poor moving object may contain only a few optical-flow points. We therefore combine both criteria: a box is classified as dynamic only when the dynamic-point ratio exceeds $\tau_d = 40\%$ and the dynamic-point count exceeds two, as summarized in Algorithm 2. This reduces uncertainty from sparse optical-flow matching and makes the judgment less sensitive to object texture.

Algorithm 2 Determine dynamic boxes

Input: Outliers T_M, Boxes $\mathcal{B}$, Matched points $\mathcal{P}$, threshold $\tau_d = 0.4$.
Output: Dynamic boxes $\mathcal{B}_{dyn}$, Static boxes $\mathcal{B}_{stat}$.

$\mathcal{B}_{dyn} \leftarrow \emptyset$, $\mathcal{B}_{stat} \leftarrow \emptyset$
for each box $b \in \mathcal{B}$ **do**
  Let n_T and n_P be the count of points in T_M and $\mathcal{P}$ inside b.
  if $n_T/n_P \geq \tau_d$ **and** $n_T > 2$ **then**
    $\mathcal{B}_{dyn} \leftarrow \mathcal{B}_{dyn} \cup \{b\}$
  else
    $\mathcal{B}_{stat} \leftarrow \mathcal{B}_{stat} \cup \{b\}$
  end if
end for

C. Clustering-based dynamic point rejection

After box-level motion judgment, nested or overlapping boxes are resolved by assigning each object to the smallest enclosing box and evaluating each box independently. For example, if a static object is contained within a moving-object box, points belonging to the smaller static box are retained instead of being removed together with the larger dynamic box. Static boxes inside moving boxes therefore rescue their contained points, while the remaining points in dynamic boxes are further processed by foreground-background clustering. This step is important in dense traffic scenes, where vehicles, cyclists, pedestrians, and background structures frequently overlap in the image plane. We employ k-means++ clustering to classify the foreground and background within the object detection bounding boxes. K-means++ is applied to

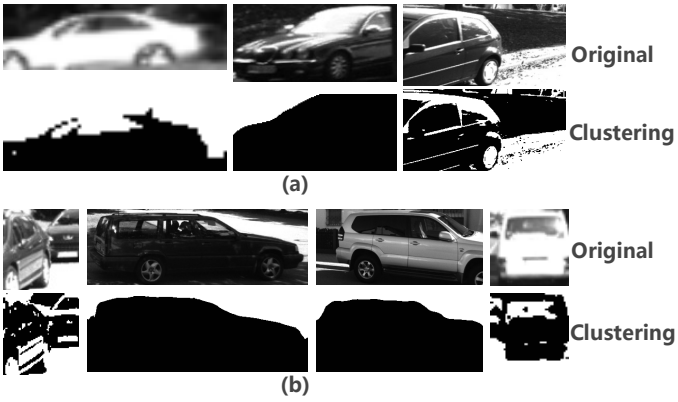


Fig. 8. Clustering results for car bounding boxes on KITTI odometry outdoor driving sequences 00–10, using the same outdoor driving split as the trajectory evaluation. (a) and (b) show representative vehicle boxes: the first row is the grayscale image of the object detection box, and the second row is the foreground-background clustering result using the grayscale image and corresponding neural depth. Black denotes foreground dynamic candidates, and white denotes retained background.

weighted grayscale and dense-depth values so that texture helps boundary recovery without dominating clustering. Pure depth clustering can be unreliable because neural depth maps are less certain than LiDAR or RGB-D measurements, while pure grayscale clustering can be disturbed by excessive texture. Combining both cues provides a more stable foreground-background separation. Lower-depth foreground candidates are shown in black and retained background in white in Figs. 8 and 9.

After foreground classification, foreground feature points are removed and all remaining points are retained for tracking. Clustering is applied only to dynamic boxes, limiting overhead as shown in Algorithm 3.

D. Integration with the Unified Backend

Unlike end-to-end learning methods, Mode II feeds cleaned static features and neural depth into the shared **Error-handling Point-Line Joint Bundle Adjustment** module in Section IV-B, preserving geometric interpretability and adaptive feature weighting.

Fig. 9. Clustering results for small traffic participants on KITTI odometry outdoor driving sequences 00–10, including pedestrian, cyclist, and motorcycle-size object boxes from the same outdoor driving split. The first row is the grayscale image of the object detection box, and the second row is the foreground-background clustering result from grayscale and neural-depth cues. Black denotes foreground dynamic candidates, and white denotes retained background.

Algorithm 3 Filter out dynamic points in a frame

Input: P: key points; $\mathcal{B}_{dyn}$: dynamic boxes; M: fg masks; $\mathcal{B}_{stat}$: static boxes.
Output: P_{out}: static key points; D_{out}: corresponding descriptors.

$P_{out} \leftarrow \emptyset, D_{out} \leftarrow \emptyset$
for each keypoint k in P **do**
  $isDynamic \leftarrow$ false
  for each box b in $\mathcal{B}_{dyn}$ **do**
    if $k \in b \wedge M(k, b) == 1$ **then** ▷ Check geometry and mask
      $isDynamic \leftarrow$ true; **break**
    end if
  end for
  for each box s in $\mathcal{B}_{stat}$ **do**
    if $k \in s$ **then**
      $isDynamic \leftarrow$ false; **break** ▷ Static box rescue
    end if
  end for
  if NOT $isDynamic$ **then**
    Append k to P_{out} and descriptor to D_{out}
  end if
end for

VI. EXPERIMENTAL EVALUATION

Experiments were executed on a Linux system with Ubuntu 18.04.6 LTS, equipped with a 12th Gen 16-thread Intel® Core™ i5-12600KF CPU, NVIDIA GeForce RTX 3070Ti GPU, and 16GB RAM. For semantic segmentation in Mode I, the YOLOv5s-seg model was chosen for its accuracy-efficiency balance. For object detection in Mode II, YOLOX was chosen for its high speed, because the monocular depth-estimation branch already consumes substantial GPU resources. The proposed system's performance was evaluated using the TUM RGB-D dataset [35], the EuRoC dataset [36], and the KITTI stereo outdoor dataset [37], with a focus on adaptability to different camera configurations. The indoor datasets were used to test runtime stability, parameter sensitivity, and dynamic feature rejection, while KITTI was used to evaluate long-range outdoor ITS performance under both stereo and monocular settings. The datasets are outlined as follows.

a) TUM RGB-D Dataset: The TUM RGB-D benchmark provides synchronized RGB, depth, and ground-truth camera trajectories for indoor dynamic scenes. We use representative walking sequences such as *w/static* and *w/rpy* to evaluate parameter sensitivity, because they contain different levels of human motion and camera rotation. These sequences are particularly useful for validating the 40% dynamic-classification threshold and the adaptive point-line weighting parameter under controlled but challenging dynamic conditions.

b) EuRoC Dataset: The EuRoC MAV dataset contains stereo-inertial sequences recorded in indoor industrial environments. We use the difficult MH 04 and MH 05 sequences to report runtime and memory behavior under high-motion conditions. Although EuRoC is not a traffic dataset, its rapid viewpoint changes and texture variation provide a useful stress test for real-time tracking stability.

c) KITTI Dataset: The KITTI dataset [38] employs colour RGB cameras mounted on a moving vehicle to capture scenes from outdoor dynamic environments. The dataset offers the flexibility to be operated in either monocular or stereo modes. Furthermore, this dataset provides precise ground truth information and the corresponding parameters tailored for various unmanned vehicle-related tasks. There are 22 image

sequences dedicated to evaluating the performance of visual SLAM, spanning a cumulative drive of 39.2 kilometres.

Only sequences 00 to 10 within the dataset provide the real trajectory data. Those sequences document various autonomous driving environments, including urban areas, suburbs, and highways. They also present different transport densities and ambient light conditions. By testing these series, we can thoroughly evaluate our work's effectiveness in outdoor dynamic settings with high mobility speeds and long duration time.

In the experimental analysis, the metrics of Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) are employed to assess results. ATE evaluates the overall trajectory alignment, while RPE measures the local accuracy over a fixed interval Δ. The estimated pose sequence is denoted by $E_1, \ldots, E_n \in SE(3)$ and the ground truth sequence is denoted by $G_1, \ldots, G_n \in SE(3)$. The ATE at a specific time t, referred to as A_t, is determined by $A_t = E_t^{-1} S G_t$ S represents the rigid transformation aligning the estimated trajectory with the ground truth scale. For RPE at time t, symbolized as R_t, is calculated by $R_t = (E_t^{-1} E_{t+\Delta})^{-1} (G_t^{-1} G_{t+\Delta})$.

A. Mode I

1) Experiments in Outdoor Environments: Mode I is compared with outdoor SLAM baselines including PLDS-SLAM [39], SD-SLAM [40], Optical-SLAM [41], Dynamic SLAM [42], RSO-SLAM [43], Light-SLAM [44], and DynaSLAM II [45]. Tables II and III report ATE RMSE over KITTI sequences 00–10, averaged over 10 trials where applicable. PLDS is structurally close to our point-line design, SD and Optical use dense detection or optical-flow based outlier filtering, Dynamic SLAM relies on semantic/environment flow, and DynaSLAM II tracks moving objects through instance segmentation. O3 serves as the baseline, and O3L adds line features without dynamic suppression. This comparison therefore covers geometry-only, semantic, optical-flow, point-line, and transportation-oriented SLAM systems.

The proposed system reduces O3 ATE by about 10%–15% across most sequences and remains robust in both static rural scenes (02, 07, 08) and dynamic traffic scenes (01, 03, 04). The gains in sequences with rich road boundaries and building facades confirm the value of line features, while the improvements in dynamic traffic scenes show that the semantic-geometric filtering can suppress moving-vehicle interference without discarding useful static structures such as parked cars and roadside infrastructure.

As shown in Fig. 10, bounding-box optical flow and epipolar constraints remove features on moving cars while preserving static features on parked cars, improving pose recovery in traffic scenes. This selective treatment is important because transportation scenes often contain semantically dynamic classes that are temporarily static, such as parked vehicles. Treating all vehicles as dynamic would unnecessarily remove stable features, whereas our method preserves them unless geometric evidence indicates motion.

2) Runtime and Parameters Analysis: Table IV shows that SPL-SLAM maintains 20 FPS on MH 04/MH 05 with less than 1 GB RAM, indicating that semantic, optical-flow, and line-feature processing remain practical for real-time deployment.

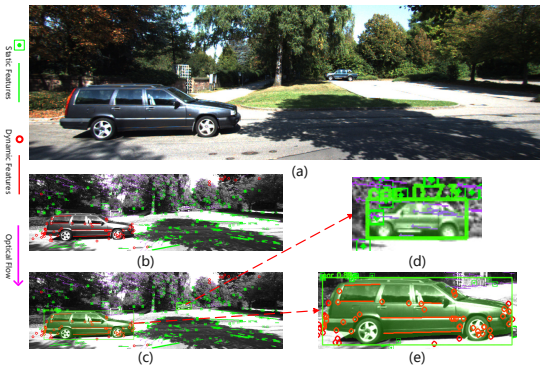


Fig. 10. Qualitative comparison on the KITTI outdoor driving dataset under dynamic traffic conditions. (a) ORB-SLAM3 baseline result, (b) trajectory estimated by the proposed system, and (c) zoomed-in view highlighting improved feature stability after dynamic feature removal.

Table V confirms that a larger α preserves useful features in low-dynamic *w/static*, while a smaller value suppresses unreliable features in high-dynamic *w/rpy*. This trend is consistent with the adaptive point-line weighting design: stable scenes benefit from retaining more structural constraints, while aggressive rotation and dynamic disturbance require faster suppression of unreliable observations.

Table VI reports the added τ_d ablation requested by the reviewer. The controlled TUM walking sequences are used here to isolate threshold effects under repeatable dynamic motion, while the selected default is then used unchanged in the KITTI ITS experiments. The default 40% threshold performs best: 30% over-removes static features, while 50% retains more dynamic outliers. This result also explains why the original default threshold was retained in the main experiments. It provides a balanced decision boundary between conservative static-feature preservation and aggressive dynamic-feature suppression.

B. Mode II

1) Experiments in Outdoor Environments: Mode II is compared with monocular ORB-SLAM3 (O), DynaSLAM [53], LIFTSLAM [3], DROID, and PVO in Table VII. The proposed depth and clustering modules reduce tracking failures in sequences 01, 07, 09, and 10 and achieve the best mean ATE. The baseline O frequently fails at the beginning of long KITTI sequences or tracks only a short segment, demonstrating the practical effect of monocular scale drift. Performance is weaker on short sequences 03/04, where limited duration reduces the benefit of dynamic-point removal and depth consistency, but long sequences confirm the advantage of scale-aware monocular tracking.

Fig. 11 compares monocular ORB-SLAM3 with the proposed depth-enhanced Mode II on KITTI sequences 00–06. The first row shows the monocular ORB-SLAM3 estimate, and the second row shows the depth-enhanced trajectory. The gray dashed line denotes ground truth, and the blue solid line denotes the estimated trajectory. The added depth branch improves scale consistency and prevents several tracking failures. Fig. 12 follows the same format and further confirms the

TABLE II
COMPARISON OF THE KITTI [38] DATASET'S MEAN ATE BY USING THE PROPOSED SYSTEM AND THE TOP VISUAL SLAM SYSTEMS.

Sequence	ATE/m									
	O3 [8]	O3L	PLDS [39]	SD [40]	Optical [41]	Dynamic [42]	Dyna II [45]	RSO-SLAM [17]	Light SLAM [44]	Ours
00	0.80	0.79	3.58	0.87	7.26	1.31	1.38	1.27	6.03	**0.72**
01	6.32	7.39	**2.31**	9.26	173.12	8.78	10.05	9.8	107.08	5.23
02	2.98	3.22	3.80	3.57	13.61	5.84	7.85	5.30	20.59	**2.81**
03	0.38	0.33	2.74	0.32	3.36	0.77	0.92	0.56	1.04	**0.26**
04	0.18	0.16	1.18	0.16	1.33	0.21	0.18	0.18	0.85	**0.14**
05	0.38	0.31	2.89	0.32	7.16	0.80	0.89	0.75	5.34	**0.27**
06	0.64	0.37	2.23	0.38	7.04	0.79	0.72	0.74	13.17	**0.32**
07	0.37	0.35	1.58	0.42	15.65	0.52	0.47	0.50	1.32	**0.30**
08	2.42	2.71	3.95	2.65	11.26	3.43	3.58	3.49	37.85	**2.11**
09	0.83	0.98	3.09	0.97	15.30	2.89	2.12	1.70	45.91	**0.75**
10	1.28	1.14	2.24	1.29	3.18	0.97	1.58	1.16	7.08	**1.09**

TABLE III
COMPARISON OF THE KITTI DATASET [38]'S MEAN ATE BY USING THE PROPOSED SYSTEM AND THE TOP LEARNING SLAM SYSTEMS. [OOM] IS SHORT FOR CUDA OUT-OF-MEMORY ON A SINGLE RTX 4090. [TL] IS SHORT FOR TRACKING LOST

Sequence	ATE/m											
	DROID-VO [23]	DPVO [46]	DROID-SLAM [23]	DPV-SLAM [47]	DPV-SLAM++ [47]	MASt3R-SLAM [48]	CUT3R [49]	Fast3R [50]	VGGT [51]	VGGT-Long [52]	Ours (Mode I)	Ours (Mode II)
00	98.43	113.21	92.10	112.80	8.30	TL	OOM	OOM	OOM	8.67	**0.72**	5.10
01	84.20	12.69	344.60	11.50	11.86	TL	OOM	OOM	OOM	121.17	**5.23**	66.22
02	108.80	123.40	107.61	123.53	39.64	TL	OOM	OOM	OOM	32.08	**2.81**	15.91
03	2.58	2.09	2.38	2.50	2.50	TL	148.07	OOM	OOM	6.12	**0.26**	6.83
04	0.93	0.68	1.00	0.81	0.78	TL	22.31	OOM	OOM	4.23	**0.14**	1.86
05	59.27	58.96	118.50	57.80	5.74	TL	OOM	OOM	OOM	8.31	**0.27**	4.35
06	64.40	54.78	62.47	54.86	11.60	TL	OOM	OOM	OOM	5.34	**0.32**	10.49
07	24.20	19.26	21.78	18.77	1.52	TL	OOM	OOM	OOM	4.63	**0.30**	2.06
08	64.55	115.90	161.60	110.49	110.90	TL	OOM	OOM	OOM	53.10	**2.11**	15.08
09	71.80	75.10	72.32	76.66	76.70	TL	OOM	OOM	OOM	41.99	**0.75**	12.89
10	16.91	13.63	118.70	13.65	13.70	TL	OOM	OOM	OOM	18.37	**1.09**	4.23

TABLE IV
REAL-TIME PERFORMANCE ANALYSIS ON HIGH-DYNAMIC SCENARIOS [36]

Sequence	RAM Usage (MB)	Frame Rate (FPS)
MH 04 (difficult)	821	20
MH 05 (difficult)	936	20

TABLE V
PARAMETER SENSITIVITY ANALYSIS OF α IN DIFFERENT DYNAMIC ENVIRONMENTS [35]

Sequence	ATE/m (RMSE)		
	α = 0.02	α = 0.06	α = 0.08
w/static	0.09	0.08	**0.05**
w/rpy	**0.035**	0.062	0.079

TABLE VI
SENSITIVITY ANALYSIS OF THE DYNAMIC CLASSIFICATION THRESHOLD [35] τ_d

Sequence	ATE/m (RMSE)		
	$\tau_d = 30\%$	$\tau_d = 40\%$	$\tau_d = 50\%$
w/static	0.061	**0.050**	0.058
w/rpy	0.044	**0.035**	0.047

The 40% column corresponds to the default threshold used in the main experiments.

benefit of scale-consistent monocular tracking in longer and more dynamic driving sequences.

2) Runtime Analysis: Table VIII reports runtime across datasets and configurations. Monocular ORB-SLAM3 averages 17.51 ms per frame, and TBB acceleration reduces this to 8.22 ms. The depth-only configuration (O+D) increases the mean time to 22.52 ms, indicating that the neural depth branch remains affordable for real-time SLAM.

Mode I runs at 40.30 ms, while the full Mode II pipeline averages 58.10 ms; within Mode II, YOLOX detection and dynamic-box judgment take about 10.34 ms and 14.55 ms. Although this is slower than pure ORB-SLAM3, it remains substantially faster than PVO (116.28 ms) and DROID-SLAM (89.85 ms), while providing dynamic-object filtering and scale-consistent monocular tracking. These results support real-time ITS use on consumer-grade hardware.

VII. CONCLUSION

This paper presented Dual-Mode SPL-SLAM for accurate, sensor-flexible, and real-time ITS localization. Mode I combines depth sensing, semantic-geometric filtering, and point-line optimization to achieve reliable tracking in dynamic environments while preserving informative static features. Mode II targets monocular platforms by recovering metric scale through lightweight neural depth and clustering-based foreground separation. Both modes are unified by an adaptive point-line backend. Experiments on KITTI, TUM RGB-D, and EuRoC show that Mode I reduces KITTI ATE by about 10%–15% over O3 and Mode II achieves the best mean ATE among monocular competitors while running at 58 ms per frame. Future work will investigate learning-based mode selection,

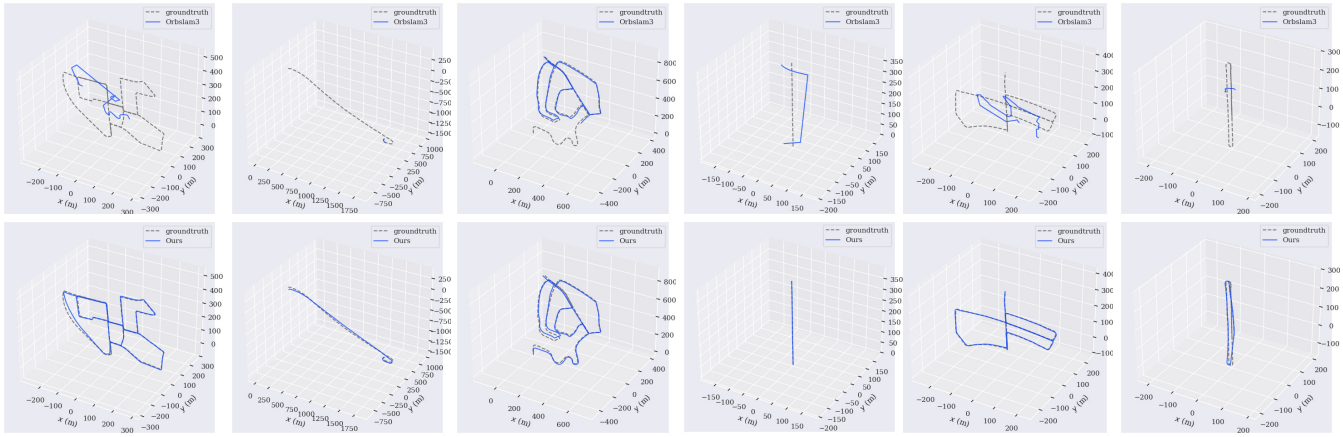


Fig. 11. Trajectory estimation comparison between ORB-SLAM3 and the proposed method on KITTI sequences 00–06. The ground-truth trajectory is shown for reference, illustrating the improved long-term tracking accuracy and scale consistency of the proposed system in outdoor ITS scenarios.

TABLE VII
BELOW ARE THE RMSE VALUES(M) FOR THE ATE ON THE KITTI DATASET [38], SOURCED FROM THE ORIGINAL PUBLICATIONS. THE RESULTS ARE COMPARED WITH LEADING VSLAM SOLUTIONS. RESULTS THAT LEAD THE PACK ARE MARKED IN BOLD, WHILE THE SUBOPTIMAL RESULTS ARE UNDERLINED.

Sequence	O	Dyna	LIFT	DROID	PVO	Ours
00	9.29	8.07	8.06	**4.86**	5.69	5.10
01	Fail	385.33	-	95.45	91.19	**66.22**
02	19.52	21.78	40.04	18.81	23.6	**15.91**
03	54.05	0.87	2.23	0.89	**0.86**	6.83
04	23.36	1.40	0.51	0.82	**0.81**	1.86
05	108.57	4.46	13.55	16.03	8.41	**4.35**
06	73.07	14.36	30.38	42.79	13.57	**10.49**
07	Fail	2.63	3.63	27.40	8.89	**2.06**
08	238.52	50.37	184.43	16.34	**6.67**	15.08
09	Fail	41.91	59.62	46.4	14.65	**12.89**
10	Fail	7.52	29.87	11.31	8.66	**4.23**
Mean	75.20	48.97	37.23	25.55	16.64	**13.18**

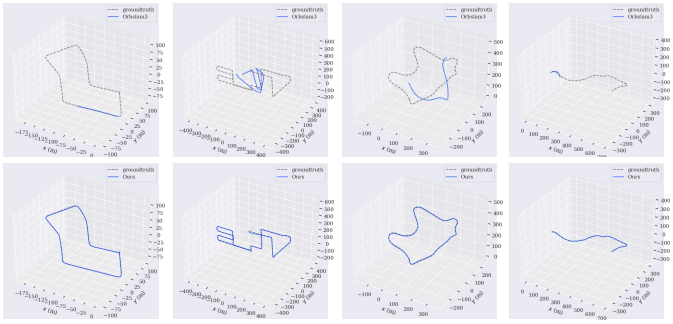


Fig. 12. Trajectory estimation comparison between ORB-SLAM3 and the proposed method on KITTI sequences 06–10. The results demonstrate superior robustness of the proposed system in long and dynamic driving sequences.

online parameter adaptation, and robustness under extreme illumination, adverse weather, and dense traffic.

TABLE VIII
COMPARISON OF MEAN COMPUTATION TIME ACROSS DIFFERENT DATASETS. ALL VALUES ARE IN [MS]. 'O' DENOTES MONOCULAR ORBSLAM3, 'O_F' IS ORBSLAM3 WITH TBB, AND 'O+D' INCLUDES DEPTH ESTIMATION. 'OURS (MODE II)' REFERS TO THE FULL NEURAL-CLUSTERING MONOCULAR PIPELINE.

Method	**TUM**	**KITTI**	**Mean**
O (ORB-SLAM3)	16.12	21.27	17.51
O_F (with TBB)	7.66	9.37	8.22
O+D	17.17	30.82	22.52
Ours (Mode I)	40.30	-	-
Ours (Mode II)	51.48	59.09	58.10
PVO	-	116.28	116.28
DROID-SLAM [23]	-	89.85	89.85

REFERENCES

[1] C. Li *et al.*, "Bridging the gap between visual servoing and visual SLAM: A novel integrated interactive framework," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 2245–2255, 2022.

[2] C. Campos, R. Elvira, J. J. Gómez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[3] H. M. S. Bruno and E. L. Colombini, "Lift-slam: A deep-learning feature-based monocular visual slam method," *Neurocomputing*, vol. 455, pp. 97–110, 2021.

[4] Y. Zhang, T. Wang, and X. Zhang, "Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 22 056–22 065.

[5] C. Yu *et al.*, "DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1168–1174.

[6] V. Badrinarayanan *et al.*, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017.

[7] C. Couprie *et al.*, "Indoor semantic segmentation using depth information," in *First International Conference on Learning Representations (ICLR 2013)*, 2013, pp. 1–8.

[8] C. Campos *et al.*, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, 2021.

[9] R. Gomez-Ojeda *et al.*, "PL-SLAM: A Stereo SLAM System Through the Combination of Points and Line Segments," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, 2019.

[10] R. Wang, K. Di, W. Wan, and Y. Wang, "Improved point-line feature based visual SLAM method for indoor scenes," *Sensors*, vol. 18, no. 10, p. 3559, 2018.

[11] R. Mur-Artal *et al.*, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[12] R. Grompone von Gioi *et al.*, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, 2010.

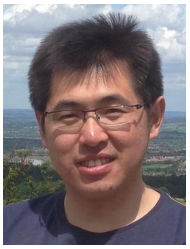
[13] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[14] N. Dvornik *et al.*, "BlitzNet: A Real-Time Deep Network for Scene Understanding," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4174–4182.

[15] Z. Hu *et al.*, "Semantic SLAM Based on Improved DeepLabv3+ in Dynamic Scenarios," *IEEE Access*, vol. 10, pp. 21 160–21 168, 2022.
[16] Y. Liu and J. Miura, "RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods," *IEEE Access*, vol. 9, pp. 23 772–23 785, 2021.
[17] R. Gao *et al.*, "Real-time SLAM based on dynamic feature point elimination in dynamic environment," *IEEE Access*, vol. 11, pp. 113 952–113 964, 2023.
[18] J. He, M. Li, Y. Wang, and H. Wang, "OVD-SLAM: An online visual SLAM for dynamic environments," *IEEE Sensors Journal*, vol. 23, no. 12, pp. 13 210–13 219, 2023.
[19] F. Min *et al.*, "COEB-SLAM: A robust VSLAM in dynamic environments combined object detection, epipolar geometry constraint, and blur filtering," *IEEE Sensors Journal*, vol. 23, no. 21, pp. 26 279–26 291, 2023.
[20] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6243–6252.
[21] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam—learning a compact, optimisable representation for dense visual slam," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2560–2568.
[22] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.
[23] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.
[24] F. Shu *et al.*, "Structure plp-slam: Efficient sparse mapping and localization using point, line and plane for monocular, rgb-d and stereo cameras," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2105–2112.
[25] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison, "Gaussian Splatting SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.
[26] Z. Zhu *et al.*, "Nicer-slam: Neural implicit scene encoding for rgb slam," *2024 International Conference on 3D Vision (3DV)*, pp. 42–52, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:256627303
[27] T. B. Martins, M. R. Oswald, and J. Civera, "Open-vocabulary online semantic mapping for slam," 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:274192471
[28] C. Yan *et al.*, "Gs-slam: Dense visual slam with 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 595–19 604.
[29] K. He *et al.*, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
[30] M. A. Fischler *et al.*, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.
[31] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, 1981, pp. 121–130.
[32] X. Chen *et al.*, "Sensing data supported traffic flow prediction via denoising schemes and ann: A comparison," *IEEE Sensors Journal*, vol. 20, no. 23, pp. 14 317–14 328, 2020.
[33] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, vol. 15, 1988.
[34] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Force control*. Springer, 2009.
[35] J. Sturm *et al.*, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.
[36] M. Burri *et al.*, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract
[37] M. Menze, C. Heipke, and A. Geiger, "Object scene flow," *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.
[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.
[39] C. Yuan *et al.*, "PLDS-SLAM: Point and Line Features SLAM in Dynamic Environment," *Remote Sensing*, vol. 15, no. 7, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/15/7/1893
[40] Q. Zhang and C. Li, "Semantic SLAM for mobile robots in dynamic environments based on visual camera sensors," *Measurement Science and Technology*, vol. 34, no. 8, p. 085202, Aug. 2023.
[41] Y. Liu and Z. Zhou, "Optical flow-based stereo visual odometry with dynamic object detection," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 6, pp. 3556–3568, 2023.
[42] S. Wen, X. Li, X. Liu, J. Li, S. Tao, Y. Long, and T. Qiu, "Dynamic SLAM: A visual SLAM in outdoor dynamic scenes," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–11, 2023.
[43] L. Qin *et al.*, "Rso-slam: A robust semantic visual slam with optical flow in complex dynamic environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 10, pp. 14 669–14 684, 2024.
[44] Z. Zhao *et al.*, "Light-slam: A robust deep-learning visual slam system based on lightglue under challenging lighting conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 26, no. 7, pp. 9918–9931, 2025.
[45] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "Dynaslam II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5191–5198, 2021.
[46] Z. Teed, L. Lipson, and J. Deng, "Deep patch visual odometry," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
[47] L. Lipson, Z. Teed, and J. Deng, "Deep patch visual slam," in *European Conference on Computer Vision*. Springer, 2025, pp. 424–440.
[48] R. Murai *et al.*, "Mast3r-slam: Real-time dense slam with 3d reconstruction priors," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 16 695–16 705.
[49] Q. Wang, Y. Zhang, A. Holynski, A. A. Efros, and A. Kanazawa, "Continuous 3d perception model with persistent state," *arXiv preprint arXiv:2501.12387*, 2025.
[50] J. Yang *et al.*, "Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass," *arXiv preprint arXiv:2501.13928*, 2025.
[51] J. Wang *et al.*, "Vggt: Visual geometry grounded transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.
[52] K. Deng, Z. Ti, J. Xu, J. Yang, and J. Xie, "Vggt-long: Chunk it, loop it, align it–pushing vggt's limits on kilometer-scale long rgb sequences," *arXiv preprint arXiv:2507.16443*, 2025.
[53] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

Zhen Tian received the B.Eng. degree in electronic and electrical engineering from the University of Strathclyde, Glasgow, U.K., in 2020. He is currently pursuing the Ph.D. degree with the James Watt School of Engineering, University of Glasgow, Glasgow, U.K. His research interests include interactive vehicle decision systems, autonomous racing decision systems, and intelligent transportation perception.

Zhihao Lin received the M.S. degree from the College of Electronic Science and Engineering, Jilin University, Changchun, China. He is currently pursuing the Ph.D. degree with the James Watt School of Engineering, University of Glasgow, Glasgow, U.K. His research interests include multi-sensor fusion SLAM systems, visual localization, and robotic perception in complex dynamic scenarios.



Dezong Zhao received the B.Eng. and M.S. degrees from Shandong University, Jinan, China, in 2003 and 2006, respectively, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2010, all in Control Science and Engineering. He is a Professor of Autonomous Systems with the James Watt School of Engineering, University of Glasgow, and a Turing Fellow with The Alan Turing Institute. He was awarded a Royal Academy of Engineering/Leverhulme Trust Research Fellowship in 2025, a Royal Society-Newton Advanced Fellowship in 2020, and an EPSRC Innovation Fellowship in 2018.

Christos Anagnostopoulos received the B.Sc. degree (Hons.) in computer science and telecommunications, the M.Sc. degree (Distinction) in advanced computing systems, and the Ph.D. degree in computing science from the University of Athens in 2002, 2004, and 2008, respectively. He is an Associate Professor of Distributed Computing and Data Engineering Systems with the School of Computing Science, University of Glasgow. His research interests include distributed computing, data engineering, distributed AI/ML, and data-centric AI.

Peng Wang received the B.E. and M.E. degrees from Huazhong University of Science and Technology in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with the School of Engineering, Westlake University, in collaboration with Zhejiang University, Hangzhou, China. His research interests include 3D computer vision, robotic perception, and visual SLAM.

Wenjing Zhao received the Ph.D. degree in Traffic Engineering from Central South University, Changsha, China, in 2022. She is currently a Postdoctoral Research Associate with the University of Glasgow, Glasgow, U.K. Her research interests include intelligent transportation systems, traffic safety, driving behaviour analysis, digital twinning, and connected and automated vehicles.

Lin Wu received the M.S. degree from Southeast University, Nanjing, China, in 2019. He is currently pursuing the Ph.D. degree with the James Watt School of Engineering, University of Glasgow, Glasgow, U.K. His research interests include machine vision, robotics, visual SLAM, and multimodal learning.

Qi Zhang received the M.S. degree from the University of Glasgow, Glasgow, U.K., in 2022. He is currently pursuing the Ph.D. degree with the Informatics Institute, University of Amsterdam, Amsterdam, the Netherlands. His research interests include 3D computer vision, robotic perception, visual localization, and SLAM in dynamic environments.