

Dynamic-Aware Point-Line Visual SLAM with Loop-Consistent Semantic Mapping

Bo Jiang¹, Mengcheng Xu^{2*}, Chenglong Li³, Xi Wen⁴, Xuanzhen Chen⁵

College of Flight Technology, Civil Aviation Flight University of China, Guanghan 618307, China^{1,3}

College of Air Traffic Management, Civil Aviation Flight University of China, Chengdu 610000, China^{2,4}

Institute of Electronic and Electrical Engineering, Civil Aviation Flight University of China, Chengdu 610000, China⁵

Abstract—Simultaneous localization and mapping (SLAM) plays an important role in autonomous vehicles and Unmanned Aerial Vehicles (UAVs) when satellite navigation is unreliable or unavailable. However, its performance often degrades in complex scenes with moving objects, weak texture, and map inconsistencies caused by loop closure. To address these issues, this paper presents PLCS-SLAM, a visual localization and mapping system for dynamic and low-texture environments. The system combines depth-aware semantic gating with probabilistic geometric verification to suppress dynamic outliers while preserving valid static features. A tightly coupled point-line optimization framework is introduced, in which continuous motion confidence is propagated from the front-end to the back-end information matrix and line features are used to strengthen structural constraints. A keyframe-anchored semantic octree mapping module is further developed to reduce structural ghosting and maintain map consistency during pose updates. Experiments on the TUM RGB-D benchmark, KITTI, and a real-world UAV dataset show lower trajectory errors than several evaluated baselines, while paired map evaluations indicate reduced local structural misalignment on KITTI.

Keywords—Visual SLAM; UAV Navigation; point-line fusion; dynamic environments; low-texture scenes; semantic mapping; loop closure

I. INTRODUCTION

Visual simultaneous localization and mapping (vSLAM) has been widely applied in autonomous vehicles and UAVs, particularly in environments with weakened GNSS signals [1]–[4]. Due to the low cost, ease of integration, and rich environmental data provided by visual sensors such as monocular and stereo cameras, visual SLAM has seen extensive development, including point feature-based methods [5], [6] and direct methods [7]–[9]. However, these conventional visual SLAM systems heavily rely on the static environment assumption, requiring the relative positions of objects within the scene to remain constant. In the real world, these methods are prone to failure. The presence of dynamic entities such as pedestrians and vehicles, or low-texture conditions typical of indoor corridors and tunnels, easily disrupts feature association and pose estimation, leading to a significant decline in system robustness and map reliability. For autonomous vehicles and UAVs, robustness in such conditions is essential. A localization system should remain reliable under dynamic interference, and the generated map should stay consistent and reusable after long-range motion and loop closure. These two aspects are closely connected. Once unstable observations are introduced

into tracking and mapping, drift tends to accumulate, data association becomes less reliable, and map maintenance becomes more difficult. This issue is especially common in urban traffic scenes and low-altitude UAV flights, where dynamic foreground objects and weakly textured backgrounds often occur together.

To address this challenge, most studies have explored multiview geometry [10], [11], multisensor fusion [12], and semantic segmentation based on deep learning [13]. Some researchers have further incorporated object dynamics directly into the optimization framework [14]–[16]. However, these approaches exhibit critical limitations in complex scenarios. Purely geometric and fusion methods often struggle with highly dynamic interference or require complex parameter tuning. Semantic approaches are effective in identifying potentially movable objects, yet they are not always reliable in practice. Temporarily stationary objects [13], such as parked vehicles, are often removed together with truly dynamic objects, while some moving objects may still escape detection. In addition, many methods still rely on fixed geometric thresholds, which overlook the nonlinear scaling of depth sensor errors. This makes distant static features more likely to be rejected, further weakening the stability of the system.

The problem becomes more serious in low-texture environments. When reliable point features are insufficient, line features can provide useful geometric structure and help maintain observability. However, conventional point-line SLAM methods also face two practical difficulties. First, line extraction and descriptor matching usually introduce extra computational cost. Second, line observations are easily corrupted when they are partially occluded by moving objects. In this case, treating the whole line segment as valid may introduce outliers, whereas rejecting it entirely may remove useful structural information. A more effective strategy is therefore needed to make point-line fusion useful in dynamic and low-texture scenes.

Besides localization robustness, map consistency and long-term reusability are equally critical for autonomous navigation tasks. The majority of existing map models are purely geometric [17], [18], lacking high-level semantic understanding of the environment and thus failing to support the decision-making requirements of high-level tasks. Even when some systems incorporate semantic labels, their underlying voxels lack awareness of historical poses. Consequently, after loop closure, these discrete voxels cannot be synchronously calibrated with the updated camera poses, leading to severe artifacts such as structural misalignments and double-wall ghosting.

*Corresponding author (Mengcheng Xu, monchxu@cafuc.edu.cn)

To address these issues, this paper proposes PLCS-SLAM, a tightly coupled visual SLAM system designed for autonomous vehicles and UAVs operating in dynamic and low-texture environments. The proposed system combines dynamic suppression, feature tracking, and map maintenance within a multithreaded framework. By passing structural constraints and dynamic weights from the front-end to the back-end and the mapping module, PLCS-SLAM improves pose estimation and maintains a more reliable global map. The main contributions of this paper are summarized as follows:

- 1) A dual-channel dynamic suppression strategy is developed for visual SLAM in dynamic scenes. The method combines depth-aware semantic gating with probabilistic geometric verification, allowing the system to suppress moving outliers while retaining static features that are often removed by semantic-only filtering.
- 2) A tightly coupled point-line optimization framework is proposed to improve localization robustness in low-texture and dynamically occluded environments. By combining efficient geometry-guided fast matching in the front-end with adaptive information weighting in the back-end, the proposed method makes better use of structural constraints when reliable point features are insufficient.
- 3) A keyframe-anchored semantic octree mapping module is designed to maintain map consistency after loop closure. Instead of treating the voxel map as a rigid structure, the proposed strategy updates historical voxels together with corrected poses, which helps reduce structural misalignment and ghosting artifacts in the global map.

The rest of this paper is structured as follows. Section II reviews the related work. Section III outlines the system architecture. Section IV gives the details of the proposed method. Section V details the experimental results, and Section VI concludes the paper and discusses potential directions for future work.

II. RELATED WORK

A. Dynamic Object Processing in SLAM

Recent advancements in visual and VI-SLAM have increasingly focused on dynamic object interference, particularly for the autonomous navigation of vehicles and UAVs in real-world environments. To tackle this challenging issue within the context of VI-SLAM, Song et al. [12] proposed rejecting feature points associated with dynamic objects by refining the BA process using pose priors estimated from IMU pre-integration. They employed a grouping strategy to mitigate the impact of temporarily stationary objects during loop closure detection. Similarly, CD-SLAM [19] builds on ORB-SLAM3 [20] by running a parallel YOLOv5 detection thread guided by IMU priors. Although this successfully filters dynamic features to stabilize tracking, these IMU-fusion frameworks typically demand precise initialization and tedious hyperparameter tuning, hindering their scalability in complex dynamic environments.

In V-SLAM, features are commonly extracted from static regions of dynamic scenes for tracking, mapping, and opti-

mization. Earlier work, such as Wang et al. [21], mainly relied on optical flow to detect motion. Co-Fusion [22] adopts an object-level representation by separating foreground objects from the background and tracking them independently in dynamic scenes. Fan et al. [10] address this problem from a geometric perspective by using multiview reprojection errors to distinguish static objects from dynamic ones, and Cañovas et al. [11] follow a similar idea with a supersurfel-based representation.

However, when dynamic objects account for most of the scene, multiview geometry-based methods often degrade significantly and may even fail in pose estimation. This limitation has led many studies to combine deep learning with geometric methods. DynaSLAM [13] and DS-SLAM [23], for example, couple instance segmentation with motion consistency checks to reject dynamic outliers, and VDO-SLAM [24] further incorporates object tracking. More recent work, such as NGD-SLAM [25], improves dynamic scene handling through a tighter integration of neural and geometric modules. Nevertheless, most existing pipelines still depend on binary masks and fixed geometric thresholds. Such designs are less robust to nonlinear depth errors and semantic segmentation errors, and may discard useful static features, leading to feature starvation.

B. Point-Line SLAM in Low-Texture Environments and Dynamic Environments

Traditional visual SLAM estimates camera pose by minimizing reprojection or photometric errors. In low-texture or dynamic environments, the number of reliable pixels often decreases significantly, which degrades tracking performance. In such cases, line features provide useful geometric constraints because they are common in many environments and remain detectable even in texture-poor regions.

Gómez-Ojeda et al. [26] proposed PL-SLAM, which integrates point and line features throughout the full pipeline, including visual odometry, keyframe selection, and BA. This improves robustness and mapping quality in scenes where point features are sparse or unevenly distributed. In visual-inertial systems, PL-VIO [27] and PL-VINS [28] further extend this idea by tightly coupling point and line features, leading to lower localization error than point-only methods in low-texture and structured environments.

Point-line features have also been incorporated into SLAM systems designed for dynamic scenes. PLDS-SLAM [29] combines semantic priors with geometric checks on points and lines, and uses Bayesian dynamic-feature tracking to reject unstable observations. DPL-SLAM [30] uses object detection, Lucas-Kanade optical flow, and epipolar constraints to handle features within and outside detected object regions. Its regional rejection strategy aims to remove moving features while limiting the loss of static observations. DynPL-SLAM [31] further explores stereo point-line estimation in dynamic scenes, introducing dynamic feature detectors and a spatial line residual for pose optimization.

Reliability over time is also considered in visual-inertial methods. TRAIL-VIO [32] considers the reliability of point-line observations over time in UAV flights through dynamic

and low-texture environments. Semantic cues and IMU measurements contribute to this assessment, while inertial prediction supports feature association. Its line processing also retains usable portions of segments partly affected by dynamic objects.

Partial occlusion remains a concern when using line features in dynamic scenes. A segment may contain both valid structural information and observations affected by motion. Keeping the whole segment can introduce outliers, but discarding it may remove useful constraints. Observation weighting and local segment handling offer ways to address these cases, although more detailed processing must also account for the computational cost of line extraction and matching.

C. Semantic Mapping and Octree-based Map Representation

Besides localization, map representation is equally important for autonomous navigation. Traditional SLAM systems usually build purely geometric maps and therefore provide limited support for tasks that require semantic understanding.

To enrich map representation, many studies have introduced semantic mapping frameworks that combine occupancy information with semantic labels. Hornung et al. [33] proposed OctoMap, which represents volumetric occupancy with an octree structure and log-odds updates. Its multiresolution design preserves spatial information while maintaining efficient memory usage. Based on this framework, Yu et al. [23] developed semantic octree mapping methods that combine pixel-level semantic segmentation with voxel occupancy estimation. With motion consistency checking and semantic filtering, these methods reduce the influence of dynamic objects and improve label stability.

Efficiency and scalability have also been studied in octree-based mapping. Vespa et al. [34] and Reijgwart et al. [35] investigated multiresolution fusion strategies for large-scale environments, using a unified hierarchical structure to maintain both TSDF and volumetric occupancy. Min et al. [36] further improved update efficiency with OctoMap-RT, which applies GPU-based parallel ray tracing to accelerate occupancy integration.

Although these studies have improved semantic mapping in terms of representation, efficiency, and scalability, most existing semantic mapping methods still use rigid voxel structures. Since the map is often maintained separately from the SLAM pose graph, it cannot be corrected consistently during loop closure, which may result in structural misalignment and ghosting artifacts.

III. SYSTEM OVERVIEW

PLCS-SLAM adopts a tightly coupled multithreaded architecture to improve robustness in dynamic and low-texture environments. As illustrated in Fig. 1, the framework comprises dual-channel dynamic suppression, front-end pose tracking, local mapping, loop closure detection, and dense map creation. Through shared data association and cross-thread interaction, these modules jointly support robust pose estimation and loop-consistent 3D mapping. The specific theoretical formulations and algorithmic designs of these core modules are detailed in Section IV.

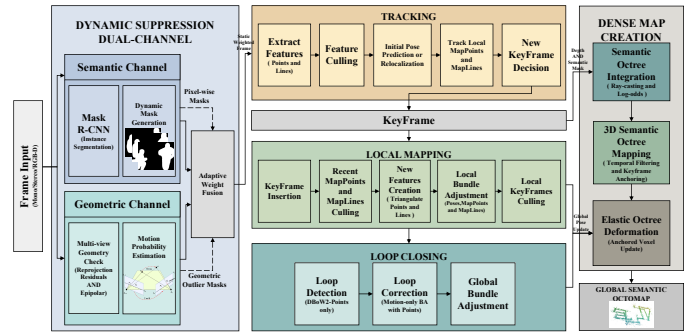


Fig. 1. Overall architecture of PLCS-SLAM, showing confidence-weighted point-line estimation and keyframe-anchored semantic mapping.

The dual-channel dynamic suppression module first processes each incoming frame. The semantic branch provides low-latency instance segmentation, while the geometric branch estimates depth-adaptive motion probabilities. Their outputs are fused to generate a static-weighted frame for the tracking thread. The tracking thread then extracts point and line features and applies soft gating to suppress dynamic outliers. To address degraded observability without introducing substantial computational overhead, a geometry-guided fast line-matching strategy is further employed. The filtered point and line landmarks are subsequently used to initialize camera pose estimation.

For map maintenance, the local mapping module triangulates new features and performs local BA. Here, spatial lines utilize a minimal orthogonal parameterization. Continuous dynamic weights from the front-end scale the optimization information matrix, marginalizing dynamic features. Concurrently, the loop closure thread detects loop candidates via point-based BoW matching. Upon validation, it triggers global BA to reduce accumulated drift.

Finally, to facilitate long-term map reusability and support navigation-oriented tasks, the system incorporates a parallel semantic mapping module. It constructs a hierarchical octree map asynchronously using depth maps and semantic masks. Rather than maintaining a rigid voxel grid, it employs a keyframe-anchored elastic deformation mechanism. When the loop closure thread updates the global poses, the mapping module updates the anchored voxels accordingly, thereby reducing structural ghosting artifacts.

IV. PROPOSED METHOD

A. Dual-Channel Adaptive Dynamic Suppression

To overcome the limitations of misclassification and missed detections inherent in purely semantic networks, a dual-channel strategy integrating a depth-adaptive semantic channel with multiview geometric verification is proposed.

1) *Depth-Aware Adaptive Semantic Gating*: As shown in Fig. 2, the semantic branch uses a modified Mask R-CNN [37] to generate instance masks for potentially movable objects. MobileNetV3 [38] replaces the original ResNet backbone, and BiFPN [39] replaces FPN. The Box Head incorporates global average pooling, while the Mask Head uses depthwise separable convolutions. These changes aim to reduce the computational cost of instance segmentation. Section V-G describes

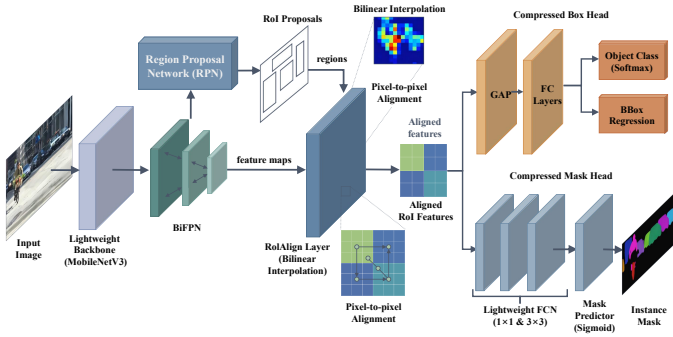


Fig. 2. Modified Mask R-CNN architecture with a MobileNetV3 backbone and a BiFPN neck. Global average pooling is introduced into the Box Head, and depthwise separable convolutions are used in the Mask Head.

the weight initialization and training procedure, and Table XII reports the segmentation accuracy and inference latency.

During inference, the network processes each input frame I_k and outputs a set of instances, written as $\{(c_j, s_j, M_{k,j})\}_{j=1}^{N_k}$. Here, c_j is the predicted category, s_j is the confidence score, and $M_{k,j} \in \{0, 1\}^{H \times W}$ is the corresponding binary mask. Let $\mathcal{C}_{dyn}$ denote the predefined set of potentially movable classes. The base semantic mask M_k^{sem} is obtained by taking the union of all high-confidence masks associated with categories in $\mathcal{C}_{dyn}$.

$$M_k^{sem} = \bigcup_{\substack{j: c_j \in \mathcal{C}_{dyn} \\ s_j > \tau_c}} M_{k,j} \quad (1)$$

After the base mask is obtained, a soft-gating strategy is applied rather than uniformly rejecting all features in and around the detected dynamic regions. Features well inside the mask are assigned a higher rejection tendency, whereas those near mask boundaries or associated with temporarily stationary objects are down-weighted rather than removed directly. The final rejection decision is deferred until semantic information is fused with multiview geometric consistency. Moreover, the pixel-space impact of segmentation errors varies with object depth. A given physical error produces a larger projection deviation at close range and a smaller one at greater distance. Therefore, a fixed dilation radius is not suitable for objects at different depths. To address this issue, a depth-adaptive soft-gating mechanism is introduced, in which the dilation radius $r(z_i)$ is determined by the depth of each feature point.

$$r(z_i) = \text{clamp} \left(\frac{f \cdot \delta}{z_i}, r_{min}, r_{max} \right) \quad (2)$$

where, f denotes the focal length, δ is the estimated bound on physical edge error, and clamp restricts the radius to a predefined range to avoid extreme values. In this way, the dilation radius can better adapt to nearby large objects while avoiding excessive expansion into the static background around distant small objects. When valid depth is unavailable, the dilation radius is conservatively set to r_{min} , while motion discrimination is mainly handled by the epipolar geometric branch.

Based on this observation, the conventional exponential decay model is replaced with a piecewise linear function to compute the semantic weight $w_{sem}^{(p_i)}$ for each point feature.

$$w_{sem}^{(p_i)} = \begin{cases} 0, & p_i \in M_{k,j}, \\ \max \left(0, 1 - \frac{s_{conf}}{\tau} \right), & p_i \in \mathcal{B}_{k,j}^{(i)}, \\ 1, & \text{otherwise.} \end{cases} \quad (3)$$

where, $\mathcal{B}_{k,j}^{(i)} = \text{dilate}(M_{k,j}, r(z_i)) \setminus M_{k,j}$ denotes the depth-adaptive boundary region outside the original mask. Here, dilate denotes the dilation operation that expands the original mask $M_{k,j}$ by a radius of $r(z_i)$. Feature points inside the original mask are assigned zero semantic confidence, but are retained until semantic and geometric cues are fused. Points falling within $\mathcal{B}_{k,j}^{(i)}$ are down-weighted linearly according to the dynamic classification confidence s_{conf} and a normalization coefficient τ . Points outside the dilation region are regarded as static background, so their weights are kept unchanged.

When a line segment passes through different semantic regions, rejecting it as a whole can remove useful background structure. An overlap-rate-based strategy is therefore adopted for piecewise clipping and adaptive weighting. Each line segment is represented by a set of uniformly sampled pixels defined by its endpoints. To account for uncertainty near the mask boundary, the dilation radius is determined from the depth at the midpoint of the line segment, and the overlap ratio is then computed between the sampled line segment and the dilated dynamic mask.

$$\phi_{l_m} = \frac{1}{T} \sum_{t=1}^T \mathbb{I}\{u_m(t) \in \text{dilate}(M_k^{sem}, r(z_{mid}))\} \quad (4)$$

where, T is the total number of sampled points along the line segment, and $\mathbb{I}(\cdot)$ denotes the indicator function, which takes the value 1 if a sampled point lies inside the dilated mask and 0 otherwise.

Instead of a nonlinear decay model, a linear truncated mapping is used to assign the initial semantic weight to each line feature:

$$w_{sem}^{l_m} = \max \left(0, 1 - \frac{\phi_{l_m}}{\tau_{overlap}} \right) \quad (5)$$

The semantic weight decreases linearly with the mask overlap ratio. Once the overlap ratio exceeds a predefined threshold, the line segment is discarded. For segments that intersect the core mask region, piecewise clipping is further applied by removing the sampled intervals inside the mask and preserving only the subsegments that remain in the static background. A residual subsegment is kept for subsequent matching and BA optimization only if its length exceeds a minimum geometric threshold.

2) *Geometric Consistency Verification Based on Probabilistic Modeling and Depth Adaptation:* The geometric channel provides a second stage of dynamic suppression to compensate for the limitations of semantic priors. To address issues such as dynamic objects erroneously classified as static by

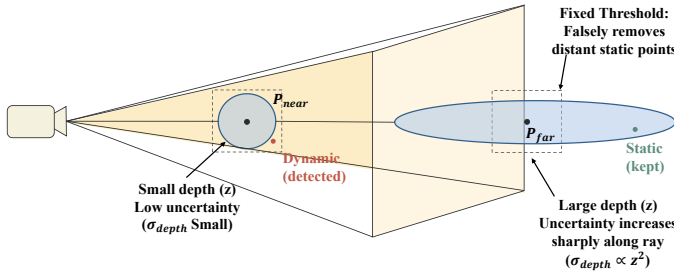


Fig. 3. Probabilistic discriminant model based on depth-adaptive covariance. The observation uncertainty ellipse (blue region) expands nonlinearly with respect to depth z . Compared to a fixed threshold (dashed box), this model successfully preserves distant static features while retaining high sensitivity to proximal dynamic features.

the semantic network and the missed detection of temporarily stationary objects, the geometric channel performs a secondary verification of the true motion states of observed features using multiview geometric constraints to complement semantic priors. During geometric verification, traditional fixed-distance threshold methods often exhibit poor adaptability in multiscale scenarios, and fundamental matrix calculations are highly susceptible to interference from dynamic objects. To overcome these challenges, a probabilistic discriminant model based on depth-adaptive covariance is proposed, coupled with an epipolar geometric verification mechanism.

For RGB-D and stereo input frames, traditional fixed-distance thresholds neglect the physical characteristic that sensor measurement errors increase nonlinearly with depth. Consequently, geometric consistency is evaluated through a hypothesis test based on the chi-square distribution, using a depth-adaptive probabilistic model of projection residuals. For a map point in a keyframe, its projection residual e_{proj} in the current frame is defined as the difference between its predicted pixel coordinates and the actual measured coordinates. Taking into account both the feature extraction noise on the image plane and the measurement uncertainty inherent in depth sensors, a depth-adaptive observation covariance matrix $\Sigma_i(z)$ is constructed as follows.

$$\Sigma_i(z) = \sigma_{img}^2 \mathbf{I}_{2 \times 2} + J_z (\kappa z_i^2)^2 J_z^T. \quad (6)$$

where, σ_{img} denotes the image feature extraction noise, typically set to 1.0 pixel, and $\mathbf{I}_{2 \times 2}$ is the identity matrix. The depth measurement uncertainty is modeled as $\sigma_z(z_i) = \kappa z_i^2$, where κ is a sensor-dependent coefficient determined by the corresponding depth or disparity uncertainty model. The term J_z denotes the Jacobian of the projection function with respect to depth. Accordingly, $J_z \sigma_z^2(z_i) J_z^T$ propagates the depth variance onto the image plane, as illustrated in Fig. 3.

Based on this covariance matrix, the system computes the Mahalanobis distance of the residual, utilizing it as a normalized statistical test statistic r_{geo}^2 .

$$r_{geo}^2 = e_{proj}^T \Sigma_i(z)^{-1} e_{proj} \quad (7)$$

Assuming that the projection residuals of static points follow a Gaussian distribution, the squared Mahalanobis distance r_{geo}^2 follows a Chi-square distribution with two degrees of freedom. Rather than applying a fixed truncation threshold, the

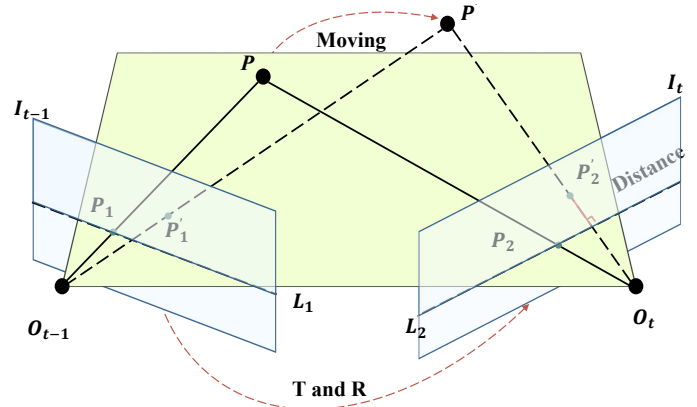


Fig. 4. Epipolar geometric verification for regions lacking depth data. Dynamic features are identified by the perpendicular epipolar distance D_{epi} (red line) between the observed point P'_2 and its predicted epipolar line L_2 .

system maps this distance to a continuous probability under the static hypothesis and uses it as the geometric weight. In this way, motion consistency can be evaluated more robustly across different depths, reducing the chance that distant static background points are classified as dynamic due to amplified depth noise:

$$w_{geo} = P(\chi_2^2 > r_{geo}^2) = \exp\left(-\frac{r_{geo}^2}{2}\right) \quad (8)$$

When the system operates in monocular mode or when observation points fall within the depth sensor's blind spots, the above depth probabilistic model automatically degrades into an epipolar geometric verification mechanism. During the estimation of the fundamental matrix F , conventional methods are highly susceptible to interference from dominant dynamic objects, resulting in severe degradation of the epipolar constraints. To address this vulnerability, a semantics-guided fundamental matrix estimation strategy is proposed. Leveraging the prior masks generated by the semantic channel, the system exclusively samples a subset of feature points, denoted as P_{static} , located within safe background regions. Subsequently, the RANSAC algorithm is employed to robustly solve for the static fundamental matrix F_{static} .

$$l = F_{static} x_{ref} \quad (9)$$

For any arbitrary feature point x_{cur} in the current frame, its point-to-epipolar-line distance D_{epi} to the corresponding epipolar line is calculated:

$$D_{epi} = \frac{|x_{cur}^T F_{static} x_{ref}|}{\sqrt{(F_{static} x_{ref})_1^2 + (F_{static} x_{ref})_2^2}} \quad (10)$$

As shown in Fig. 4, a static point is expected to lie near its corresponding epipolar line in the current frame. A large point-to-line distance suggests a violation of the rigid-motion assumption and indicates that the point is likely to belong to a dynamic region. Based on this, a smooth truncated Gaussian

weighting function is defined:

$$w_{geo}^{mono} = \begin{cases} 1, & D_{epi} < \tau_{epi} \\ \exp\left(-\frac{(D_{epi} - \tau_{epi})^2}{2\sigma_{epi}^2}\right), & D_{epi} \geq \tau_{epi} \end{cases} \quad (11)$$

This function imposes an inlier tolerance threshold τ_{epi} , typically set to a maximum of two pixels. When the error falls within this tolerance range, the geometric constraint is assigned full confidence; once the error exceeds this bound, the weight undergoes a Gaussian decay modulated by the bandwidth parameter σ_{epi} . Finally, the system integrates w_{sem} and w_{geo} to make the ultimate decision regarding feature retention or rejection.

Since semantic models often misclassify temporarily stationary objects as dynamic, a geometry-dominated, continuous fusion mechanism is proposed to recover features associated with these objects. First, for features with ambiguous motion states, a fusion weight w_{base} is computed.

$$w_{base} = \lambda w_{sem} + (1 - \lambda) w_{geo} \quad (12)$$

where, $\lambda \in [0, 1]$ controls the relative contribution of the semantic and geometric terms. The coefficient λ is set to 0.4, giving slightly greater weight to geometric consistency. To elevate the confidence of reliable static points, a logistic modulator $\alpha(w_{geo})$ is employed to smooth the transition, especially when w_{geo} approaches a high static bound τ_{static} .

$$\alpha(w_{geo}) = \frac{1}{1 + \exp(-k(w_{geo} - \tau_{static}))} \quad (13)$$

where, k controls the transition steepness. The final dynamic weight w_{dyn} is then computed:

$$w_{dyn} = (1 - \alpha(w_{geo})) w_{base} + \alpha(w_{geo}) \quad (14)$$

As w_{geo} increases, w_{dyn} gradually approaches 1, which indicates stronger confidence in the static assumption. The resulting joint weight is introduced as a continuous scalar in the information matrix and is used in both front-end feature filtering and back-end BA.

B. Confidence-Weighted Point-Line Optimization

To address observability degradation, PLCS-SLAM adopts a tightly coupled optimization framework that jointly exploits point and line features. ORB point features and LSD line features are extracted in parallel from each image. Because LBD descriptor matching is relatively expensive, a geometry-guided fast matching strategy is introduced for line features. Rather than using exhaustive global matching, the method takes the inter-frame pose prior obtained from point-feature tracking to project line features from the previous frame into the current frame and form a search band around each projection. Descriptor distance and angular consistency are evaluated only within this band, which reduces computation and improves matching reliability.

After this initial matching step, the joint dynamic weights from the semantic and geometric channels are incorporated into the observation model to suppress the influence of dynamic objects on structural constraints. Soft gating and feature

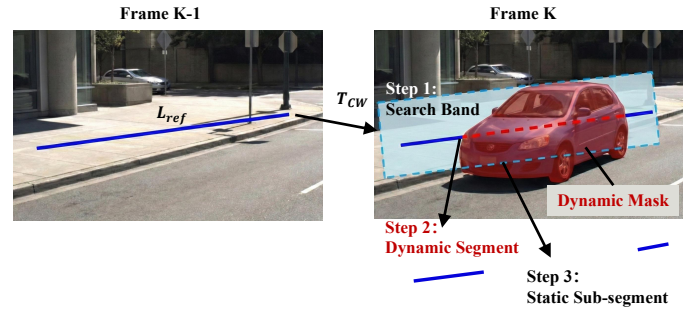


Fig. 5. Illustration of geometry-guided fast matching and dynamic segment clipping. Pose priors define the epipolar search band for efficient matching, while dynamic masks remove the portions of line segments covered by moving objects and preserve the remaining static subsegments for optimization.

clipping are then applied. For line features crossing dynamic regions, the overlap ratio between the sampled points and the dynamic mask is computed, and piecewise clipping is performed accordingly. A residual subsegment is retained for subsequent triangulation and optimization only when it lies fully in the static background and passes the dual-channel consistency check, as illustrated in Fig. 5.

Although the front-end performs projection and clipping based on line endpoints, the back-end uses a minimal orthogonal representation for optimization and map maintenance. This resolves the overparameterization and unobservability of translations along the line direction inherent in traditional 6DoF endpoint representations for three-dimensional lines. Specifically, a three-dimensional line $\mathcal{L}$ is first represented in Plücker coordinates as $\mathcal{L} = (\mathbf{n}, \mathbf{d})^T \in \mathbb{R}^6$, where $\mathbf{n} = \mathbf{P} \times \mathbf{Q}$ and $\mathbf{d} = \mathbf{Q} - \mathbf{P}$. Since $\mathbf{n}^T \mathbf{d} = 0$, the two vectors can be factorized into an orthonormal representation as

$$\begin{aligned} \begin{bmatrix} \mathbf{n} & \mathbf{d} \end{bmatrix} &= \mathbf{U} \begin{bmatrix} \omega_1 & 0 \\ 0 & \omega_2 \\ 0 & 0 \end{bmatrix}, \\ \mathbf{W} &= \frac{1}{\sqrt{\omega_1^2 + \omega_2^2}} \begin{bmatrix} \omega_1 & -\omega_2 \\ \omega_2 & \omega_1 \end{bmatrix}, \end{aligned} \quad (15)$$

where, $\mathbf{U} \in SO(3)$ describes the orientation of the line, with $\omega_1 = \|\mathbf{n}\|$ and $\omega_2 = \|\mathbf{d}\|$, while $\mathbf{W} \in SO(2)$ encodes their normalized relative magnitudes. The spatial line can therefore be represented by the orthonormal pair $(\mathbf{U}, \mathbf{W})$. Although the Plücker vector contains six components, the orthogonality constraint and homogeneous scale leave only four independent degrees of freedom for a three-dimensional line. The orthonormal representation is therefore used in the back-end optimization.

When formulating observation constraints on the image plane, the system reprojects the lines back into endpoint format to compute residuals. Let the projected endpoints of the k -th spatial line in the i -th frame be p_{ik} and q_{ik} . Given the observed two-dimensional infinite line parameters $l_{ik} = [l_1, l_2, l_3]^T$, the perpendicular distance residual from the endpoints to the line

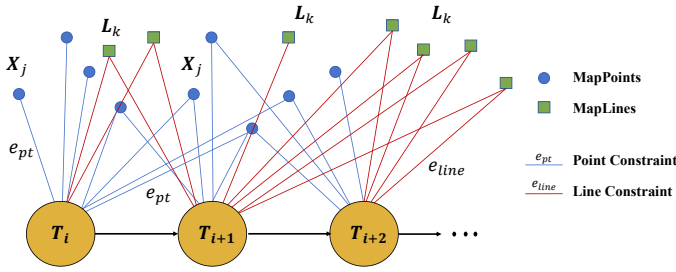


Fig. 6. Factor graph for tightly coupled point-line optimization. The system jointly minimizes point and line reprojection residuals, dynamically balanced by an adaptive information matrix.

is defined as follows:

$$e_{line}(k, i) = \begin{bmatrix} d(p_{ik}, l_{ik}) \\ d(q_{ik}, l_{ik}) \end{bmatrix} = \begin{bmatrix} \frac{p_{ik}^T l_{ik}}{\sqrt{l_1^2 + l_2^2}} \\ \frac{q_{ik}^T l_{ik}}{\sqrt{l_1^2 + l_2^2}} \end{bmatrix} \quad (16)$$

To accurately quantify the confidence of this residual and balance the numerical disparity between point and line features, the system constructs an adaptive information matrix Ω_{line} for line feature observations. This weight depends on the image pyramid level and is directly proportional to the length len_{ik} of the observed line segment.

$$\Omega_{line}^{-1} = \frac{\sigma_{line}^2}{\gamma \cdot len_{ik}} \cdot \mathbf{I}_{2 \times 2} \quad (17)$$

This formulation assigns larger weights in the information matrix to longer observed line segments, since they generally have lower measurement noise. Here, σ_{line}^2 denotes the base extraction noise associated with the corresponding image pyramid level, and γ controls the balance between point and line features. In this way, short and noisy line segments contribute less to the objective, improving the robustness of pose estimation in low-texture regions.

On this basis, a joint nonlinear least-squares problem is defined over the local sliding window, as represented by the factor graph in Fig. 6. The variables include camera poses T_i , map points X_j , and map lines L_k , where the line features are parameterized in orthogonal form.

The Huber robust cost function $\rho(\cdot)$ is used to reduce the influence of outliers:

$$\min_{T, X, L} \left(\sum_{i,j \in \mathcal{O}_{pt}} \rho(e_{pt}^T \Omega_{pt} e_{pt}) + \sum_{i,k \in \mathcal{O}_{line}} \rho(e_{line}^T \Omega_{line} e_{line}) \right) \quad (18)$$

During optimization, the motion confidence estimated in the front-end is propagated to the back-end through the joint dynamic weight. This weight serves as a continuous scalar for rescaling the base information matrix and forming the effective observation information matrix.

$$\Omega_{pt}^{effective} = w_{dyn} \cdot \Omega_{pt} \quad (19)$$

$$\Omega_{line}^{effective} = w_{dyn} \cdot \Omega_{line} \quad (20)$$

For features with high confidence under the static assumption, the effective information matrix $\Omega_{pt}^{effective}$ stays close to the original Ω_{pt} , preserving the contribution of reliable geometric constraints. For features near uncertain mask boundaries, the corresponding information matrix is attenuated accordingly. The resulting joint objective is solved with the Schur complement, which first marginalizes the large number of map point and line variables thereby preserving sparsity and enabling efficient optimization under joint point-line constraints.

C. Loop-Consistent Semantic Octree Mapping Based on Keyframe Anchoring

A parallel semantic mapping thread based on a hierarchical sparse octree [33] is introduced. Through recursive subdivision of the 3D space, the map is represented at multiple resolutions.

The mapping thread is activated by keyframes and asynchronously retrieves the poses, depth maps, and semantic masks generated by the front-end from a buffer queue. Before fusion, the depth map is filtered with the joint static mask produced by the front-end dual-channel mechanism, so that pixels in dynamic regions are removed early. A log-odds update rule is then used to suppress depth quantization noise and residual dynamic interference. In addition, upper and lower clamping thresholds are applied to help misclassified voxels return more quickly to an unknown or inactive state, improving the robustness of map correction.

While updating the spatial geometric occupancy states, the system overcomes semantic label flicker induced by view-point variations and occlusions within instance segmentation networks under multiview observations. To achieve this, the system maintains a continuous semantic distribution vector for each voxel, denoted as $S_v = [S_1, S_2, \dots, S_N]^T$, where N represents the total number of semantic categories. When a pixel carrying semantic label k projects into voxel v , the system executes an incremental probabilistic fusion process as follows:

$$S_v^{(t)}[k] = S_v^{(t-1)}[k] + \eta \cdot w_{proj} \cdot C_{conf} \quad (21)$$

Here, $S_v[k]$ denotes the cumulative confidence that voxel v belongs to the k -th object category. η represents the update step size, while C_{conf} is the classification confidence output by the neural network. The term w_{proj} denotes the geometric projection weight.

$$w_{proj} = \max(0, \cos \theta) \frac{1}{1 + (z/z_{max})^2} \quad (22)$$

where, θ is the viewing angle, z is the observation depth, and z_{max} is the valid sensor range. This weight decreases for observations characterized by longer projection distances or larger incident angles, which typically exhibit higher marginal noise levels. The final semantic category of the voxel is determined by the maximum value within the distribution vector.

$$C_{final} = \operatorname{argmax}_{i \in \{1, \dots, N\}} (S_v[i]) \quad (23)$$

Through this multiview consistency fusion strategy, even if occasional segmentation errors occur in specific keyframes,

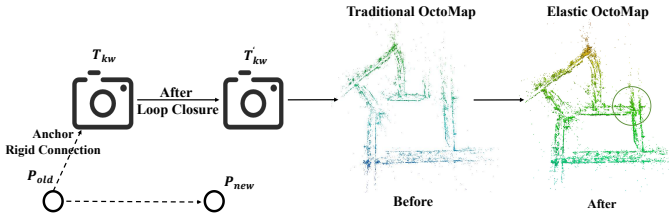


Fig. 7. Keyframe-anchored map correction. Occupied voxel centers are transformed according to changes in their anchor poses and reassigned to octree cells. The local views compare structural alignment before and after correction.

as long as the majority of observations within the temporal window remain correct, the maximum value in the cumulative vector converges to the true category.

To reduce structural misalignment after loop closure, the semantic octree map is corrected using keyframe anchors, as shown in Fig. 7. Each occupied voxel is assigned to the keyframe that creates it. Later observations may update its occupancy state and accumulated semantic scores, but the anchor index remains unchanged. A back-end pose update therefore changes the transformation associated with this anchor rather than assigning the voxel to another keyframe. Pose records are retained for all keyframes referenced by the map.

At each correction event, the mapping thread takes a snapshot of the map together with the anchor poses last applied to it. For a voxel anchored to keyframe k , let $\mathbf{T}_{kw}$ denote the world-to-camera transformation associated with the current map state, and let $\mathbf{T}'_{kw}$ denote the updated back-end pose. The voxel center is transformed as

$$\tilde{\mathbf{P}}_{\text{new}} = (\mathbf{T}'_{kw})^{-1} \mathbf{T}_{kw} \tilde{\mathbf{P}}_{\text{old}}, \quad (24)$$

where, $\tilde{\mathbf{P}}_{\text{old}}$ is the homogeneous voxel-center coordinate in the current map, and $\tilde{\mathbf{P}}_{\text{new}}$ is its transformed coordinate before assignment to an octree cell. This transformation preserves the center's coordinates in the anchor keyframe's camera frame: the old pose maps the center into that frame, and the inverse of the updated pose maps it back into the world frame.

New spatial keys are computed from the transformed centers, and the corresponding voxel states are inserted into a new octree at the same resolution. The existing tree is left unchanged during the rebuild. Once insertion is complete, inner-node occupancy values are updated, and the new tree replaces the previous map under synchronization. The updated anchor poses are then stored with the corrected map. These stored poses provide the starting point for the next correction, so a pose change already incorporated into the map is not applied again. The spatial index can thus be rebuilt without reintegrating all historical depth observations.

Reassignment may place several source voxels in the same destination cell. In this case, the destination retains the anchor and creation-order identifier of the earliest-created source voxel. Its occupancy log-odds is set to the maximum among the source values, while the accumulated semantic score vectors are summed. Each source voxel contributes once. Ordinary observation fusion therefore preserves the voxel's existing anchor; a spatial collision instead retains one of the source

anchors. This rule combines stored voxel states and should not be interpreted as Bayesian fusion of independent observations.

The rebuild transfers only occupied voxel states. Previously stored free-space nodes are omitted, leaving cells that receive no occupied state unknown until subsequent sensor observations update them. Correction therefore relocates occupied structures without recovering the historical free-space evidence from ray tracing. The resulting map may differ from one reconstructed by reintegrating all observations at the corrected keyframe poses.

For voxels that combine observations from several keyframes, Eq. (24) uses only the pose change of the stored anchor. Each fused voxel moves as a single entity. This single-anchor approximation is generally more suitable when the contributing keyframes undergo similar corrections. When those corrections differ substantially, one rigid transformation cannot represent the separate movements of all contributing observations. Reassignment to octree cells also introduces discretization effects, and collisions may further coalesce spatial detail. Some local misalignment and residual surface error may consequently remain after correction.

V. EXPERIMENTAL RESULTS

To evaluate the accuracy and robustness of PLCS-SLAM in complex environments, experiments are performed on the public TUM RGB-D and KITTI datasets together with a self-collected real-world dataset. Comparisons are also made with several state-of-the-art SLAM methods. Trajectory accuracy is assessed using the root mean square error (RMSE) of the translational absolute trajectory error (ATE). Referred to as ATE RMSE throughout the paper, this measure summarizes the positional deviation between the estimated trajectory and the ground truth over each evaluated sequence. Values are reported in meters.

Most experiments were carried out on the same computing platform running Ubuntu 18.04.6 LTS, equipped with an Intel Core i9-14900HX CPU, 32 GB of RAM, and an NVIDIA GeForce RTX 4060 GPU. Table I lists the main parameters used in PLCS-SLAM. Most of them are kept unchanged across the experimental sequences. The parameters related to depth uncertainty are set according to the depth or disparity model of the sensor, whereas the other thresholds are generally kept the same. In the sensitivity analysis, one parameter is varied at a time, while the remaining parameters are kept at the values in Table I.

A. Evaluation on TUM Dataset

The TUM RGB-D dataset [40] provides ground-truth camera trajectories through a high-precision motion capture system and is widely used for evaluating visual SLAM methods in indoor dynamic environments. The sequences are typically divided into sitting (s) and walking (w) scenarios. Sitting sequences involve only slight local motion, such as hand or torso movement, and therefore represent low-dynamic scenes. Walking sequences, by contrast, contain pedestrians moving across the camera view and producing occlusions, which makes them more challenging and more strongly dynamic. Based on the four camera motion patterns in the TUM fr3 subset, seven representative sequences covering both low- and

TABLE I. MAIN PARAMETER SETTINGS OF PLCS-SLAM

Module	Parameter	Value
Semantic gating	τ_c	0.60
	δ	0.03 m
	$[r_{min}, r_{max}]$	[3, 15] px
Line clipping	τ	1.0
	$\tau_{overlap}$	0.50
	minimum segment length	20 px
	σ_{img}	1.0 px
Geometric verification	κ	$1.28 \times 10^{-3} \text{ m}^{-1}$ (sensor-dependent)
	τ_{epi}	2.0 px
Weight fusion	σ_{epi}	1.0 px
	λ	0.40
	τ_{static}	0.80
	k	10
Point-line BA	γ	0.02 px^{-1}
Semantic mapping	voxel resolution	0.10 m
	η	1.0

TABLE II. TYPES OF CAMERA MOVEMENT

SEQUENCE	DESCRIPTION
halfsphere	the camera moves following the trajectory of a 1-meter diameter half sphere
xyz	the camera moves along the x-y-z axes
rpy	the camera rotates over roll, pitch and yaw axes
static	the camera is kept static manually

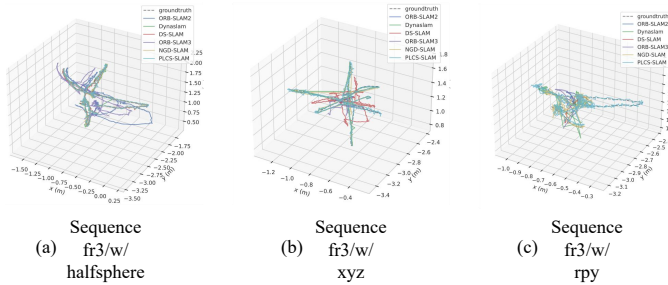


Fig. 8. Trajectory of PLCS-SLAM and baseline methods against the ground truth on three representative walking sequences.

high-dynamic conditions were selected for comparison. This selection is intended to assess the robustness and generalization ability of PLCS-SLAM. Detailed descriptions of the camera motion types are given in Table II.

Fig. 8 compares the trajectories estimated by PLCS-SLAM and five baseline methods on three representative walking sequences from the TUM dataset. Compared with the sitting sequences, these sequences involve more frequent pedestrian motion and stronger dynamic interference. ORB-SLAM2 and ORB-SLAM3 exhibit noticeable trajectory drift, while DS-SLAM and DynaSLAM achieve better results but still show local deviations during rapid motion. NGD-SLAM is generally more stable, although its accuracy decreases when large moving objects occupy a substantial part of the camera view. In contrast, the trajectories produced by PLCS-SLAM remain generally closer to the ground truth on all three sequences.

Table III reports the ATE RMSE results for all test sequences, where the best values are highlighted in bold. In the s sequences, the performance differences among the methods

TABLE III. COMPARISON OF ATE RMSE [M] OF PLCS-SLAM AGAINST SOTA BASELINES ON THE TUM DATASET

SEQUENCE	ORB-SLAM2	DYNASLAM	DS-SLAM	ORB-SLAM3	NGD-SLAM	PLCS-SLAM
w-halfsphere	0.269	0.022	0.021	0.220	0.024	0.020
w-xyz	0.276	0.016	0.190	0.280	0.015	0.013
w-rpy	0.158	0.058	0.146	0.154	0.036	0.030
w-static	0.012	0.008	0.007	0.011	0.008	0.006
Mean of w	0.179	0.026	0.091	0.166	0.021	0.017
s-halfsphere	0.033	0.039	0.013	0.054	0.020	0.018
s-xyz	0.014	0.015	0.009	0.012	0.009	0.010
s-static	0.012	0.010	0.005	0.007	0.007	0.004
Mean of s	0.020	0.021	0.009	0.024	0.012	0.011
Overall average	0.111	0.024	0.056	0.105	0.017	0.014

are small because stable background features are still sufficient. In contrast, the w sequences are more challenging and show clearer differences. Under these conditions, the geometric baselines degrade noticeably, while early semantic methods still retain some drift. NGD-SLAM performs strongly with effective dynamic masking, but PLCS-SLAM further improves the results by incorporating line features to compensate for the loss of valid point features during dynamic rejection. As a result, PLCS-SLAM achieves the lowest overall average error of 0.014 m, which is about 15.13% lower than that of NGD-SLAM. These results suggest that combining continuous probability weighting with structural constraints improves pose estimation robustness under strong dynamic interference.

1) *Repeated-Run Comparison with NGD-SLAM*: PLCS-SLAM and NGD-SLAM were each evaluated ten times on the four TUM walking sequences to examine variation across runs. The parameters of each method remained fixed. Every run started with a new process and an empty map, using the same input data, camera calibration, and trajectory evaluation settings. Each completed run produced one translational ATE RMSE value for the statistical analysis.

Table IV lists the mean ATE RMSE, its sample standard deviation across runs, and the number of successful runs. A run was classified as failed if it ended before the sequence was completed or encountered unrecoverable tracking loss. All ten runs completed for both methods on every sequence. Each mean and standard deviation was therefore calculated from ten trajectory-level RMSE values. The difference between the means is denoted by $\Delta = \bar{e}_{\text{NGD}} - \bar{e}_{\text{PLCS}}$, so positive values favor PLCS-SLAM.

The mean ATE RMSE is lower for PLCS-SLAM on all four sequences. The largest difference is on w-rpy: NGD-SLAM gives 0.036153 m and PLCS-SLAM gives 0.029686 m, a difference of 0.006467 m. On w-halfsphere, the means are 0.023892 m and 0.020382 m, respectively. The difference is smaller on w-xyz, at 0.001892 m, and on w-static, at 0.001333 m. Across the four sequences, the observed reductions in mean error range from about 1.3 to 6.5 mm.

The standard deviations show a less uniform pattern. PLCS-SLAM has lower variation on w-xyz and w-rpy; on w-rpy, its standard deviation is 0.016064 m, compared with 0.024264 m for NGD-SLAM. On w-halfsphere and w-static, its standard deviations are slightly higher. The lower mean errors thus do not indicate lower variation across runs on every sequence.

Two-sided Welch's *t*-tests were used to compare the mean errors, with Holm correction applied to the four comparisons. No statistically significant difference was found at the 0.05

TABLE IV. REPEATED-RUN COMPARISON OF NGD-SLAM AND PLCS-SLAM ON THE TUM WALKING SEQUENCES

Sequence	NGD-SLAM		PLCS-SLAM		Δ (m)
	Mean $\pm$ SD (m)	Successful runs (/10)	Mean $\pm$ SD (m)	Successful runs (/10)	
w-halfsphere	0.023892 $\pm$ 0.009977	10	0.020382 $\pm$ 0.010289	10	0.003510
w-xyz	0.015279 $\pm$ 0.007768	10	0.013387 $\pm$ 0.006901	10	0.001892
w-rpy	0.036153 $\pm$ 0.024264	10	0.029686 $\pm$ 0.016064	10	0.006467
w-static	0.007588 $\pm$ 0.003521	10	0.006255 $\pm$ 0.003958	10	0.001333

TABLE V. COMPARISON OF ATE RMSE [M] ON THE KITTI DATASET IN THE MONOCULAR SETTING.

SEQUENCE	ORB-SLAM2	DYNASLAM	ORB-SLAM3	PLCS-SLAM (MONO)
00	6.02	7.55	7.28	6.26
02	26.77	26.29	27.47	21.21
03	1.51	1.81	0.87	0.62
04	1.62	0.97	0.65	0.29
05	6.15	4.60	5.83	3.53
06	14.61	14.74	16.38	12.55
07	2.87	2.36	3.20	2.01
08	47.85	40.28	57.82	31.24
09	6.62	3.32	9.60	2.36
10	8.80	6.78	7.66	6.61

TABLE VI. COMPARISON OF ATE RMSE [M] ON THE KITTI DATASET IN THE STEREO SETTING.

SEQUENCE	ORB-SLAM2	DYNASLAM	ORB-SLAM3	PLDS-SLAM	PLCS-SLAM (STEREO)
00	1.31	1.43	0.98	3.58	0.96
01	10.40	9.46	9.54	2.31	8.45
02	5.63	6.49	5.91	3.80	5.12
03	0.38	0.62	0.31	2.74	0.25
04	0.18	0.25	0.17	1.18	0.22
05	0.39	0.85	0.43	2.89	0.36
06	0.63	0.81	0.74	2.23	0.42
07	0.52	0.51	0.49	1.58	0.46
08	3.45	3.49	3.08	3.95	3.14
09	3.25	3.44	1.87	3.09	1.84
10	1.06	1.18	1.13	2.24	0.83

For existing SLAM systems, results reported in the original publications are used where available.

level. The smaller mean errors observed for PLCS-SLAM should therefore be interpreted alongside the variation across runs.

B. Evaluation on KITTI Dataset

The KITTI visual benchmark dataset [41] was collected on an autonomous driving platform with stereo cameras and LiDAR, and includes large-scale outdoor scenes such as urban, rural, and highway environments. Compared with indoor datasets, it is characterized by longer observation ranges, larger illumination changes, and more frequent dynamic objects such as vehicles and pedestrians. To evaluate the generalization ability of PLCS-SLAM in large-scale dynamic outdoor environments, sequences 02, 07, 08, and 09 were selected for qualitative analysis because they contain long trajectories and complex loop closures. Quantitative evaluation was further conducted on the KITTI core sequences.

Tables V and VI report the ATE RMSE results on KITTI with monocular and stereo input, respectively. PLCS-SLAM has the lowest reported error on most sequences in the monocular comparison. The stereo comparison includes PLDS-SLAM [29], a point-line system that combines semantic priors,

TABLE VII. COMPARISON OF ATE RMSE [M] AGAINST SOTA BASELINES ON THE UAV DATASET

SEQUENCE	ORB-SLAM2	PL-SLAM	DYNASLAM	ORB-SLAM3	NGD-SLAM	PLCS-SLAM
Test01	19.37	26.77	18.93	33.26	22.48	10.57
Test02	9.50	10.64	9.66	8.56	9.45	6.56
Test03	9.08	45.10	10.75	10.17	9.96	9.66
Test04	23.16	21.53	23.02	225.16	26.99	13.93
Test05	15.08	10.80	13.16	35.82	17.32	10.25

geometric checks, and Bayesian feature tracking to reject dynamic observations.

In the stereo comparison, PLCS-SLAM has the lowest reported ATE RMSE on seven of the eleven sequences and lower errors than PLDS-SLAM on nine. PLDS-SLAM performs better on sequences 01 and 02, with reported errors of 2.31 m and 3.80 m; the corresponding values for PLCS-SLAM are 8.45 m and 5.12 m. ORB-SLAM3 is lowest on sequences 04 and 08. The comparison therefore favors different methods on different sequences.

C. Evaluation on UAV Flight Dataset

PLCS-SLAM is further evaluated on a long-range UAV flight dataset collected by the same research team and previously described in [32]. The dataset was collected with a hexarotor UAV carrying a monocular camera and a high-precision differential GPS/RTK system. The platform and several representative scenes are shown in Fig. 9. Only monocular images are used as input to the SLAM systems, while the GPS/RTK trajectory serves as ground truth. The dataset includes five flight sequences, Test01–Test05, with a total length of about 20 km. These sequences cover urban areas, lakes, forests, and suburban roads. They also contain rapid UAV rotations, large low-texture regions, and moving vehicles and pedestrians, making them suitable for evaluating SLAM performance under more demanding flight conditions.

Translational ATE RMSE was computed using evo after synchronization with the GPS/RTK ground truth. Each entry in Table VII is the mean of three completed runs. A run is marked as Fail if premature termination or unrecoverable tracking loss prevents it from completing the sequence. No full-sequence RMSE is assigned in that case. For completed runs, the measured error is retained regardless of its magnitude.

PLCS-SLAM has the lowest mean ATE RMSE on four of the five UAV sequences. Test03 is the exception, ORB-SLAM2 gives 9.08 m, compared with 9.66 m for PLCS-SLAM. Some baselines have much larger errors on individual sequences, including 45.10 m for PL-SLAM on Test03 and 225.16 m for

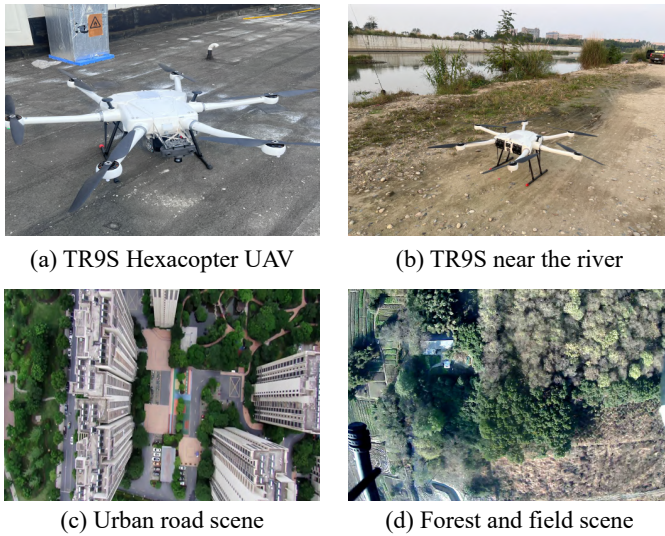


Fig. 9. Data acquisition platform and representative test scenes.

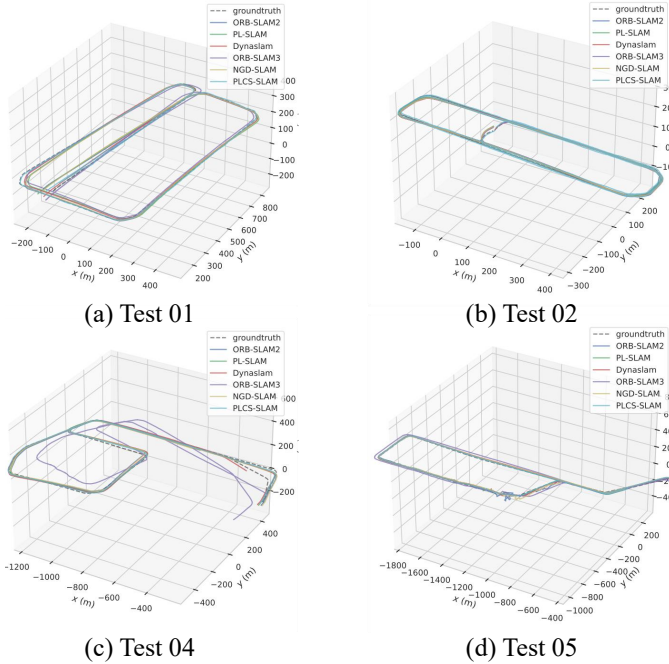


Fig. 10. Comparison of trajectories on the self-collected dataset.

ORB-SLAM3 on Test04. The ORB-SLAM3 result comes from completed runs and reflects substantial trajectory deviation.

Fig. 10 shows the trajectory comparison on the self-collected dataset. Although most baseline methods recover the general flight path, drift becomes more noticeable over long trajectories. In the more challenging Test 04 and Test 05 sequences, ORB-SLAM3 exhibits clear trajectory deviation. In contrast, PLCS-SLAM stays closer to the GPS ground truth throughout the sequences, which indicates better long-term consistency.

D. Semantic Mapping Evaluation

Fig. 11 shows the semantic octree maps reconstructed on KITTI Sequences 02 and 08, along with comparisons before and after keyframe-anchored correction on Sequences 07 and 09. The global views in Fig. 11(a)–(b) provide an overview of the reconstructed scene layout. The highlighted regions in Fig. 11(c)–(f) focus on structures observed during repeated visits.

If previously mapped voxels retain their positions after a keyframe-pose update, observations of the same structure from different visits may remain spatially separated. Such separation is visible in the highlighted regions of Fig. 11(c) and 11(e). After correction, the corresponding structures appear more closely aligned in Fig. 11(d) and 11(f).

To examine these changes quantitatively, a paired evaluation was conducted on KITTI Sequences 07 and 09. One correction event was evaluated per sequence, at frame 1052 in Sequence 07 and frame 1569 in Sequence 09. At each event, the occupied voxel map, the anchor poses last applied to it, and the updated back-end poses were saved. Two map versions were then generated from the same snapshot. The uncorrected version retained the original voxel positions, whereas the corrected version applied Eq. (24), followed by spatial reindexing and the collision rule described in the mapping method. The historical observations and back-end pose update were held fixed; only the propagation of the pose change to the voxel map differed. No new observations were fused. Both versions used a voxel resolution of 0.10 m. Table VIII reports the results for these individual events, without averaging across events or runs.

For each sequence, one static planar region was selected within the highlighted area. Each region covered a single physical surface observed during repeated visits; moving objects, vegetation, adjacent walls, and wall intersections were excluded. A reference plane was fitted to static Velodyne points transformed using the KITTI calibration and reference poses. This plane provided a geometric reference, subject to laser measurement and registration uncertainty. The two map versions were evaluated in a common coordinate system with the same reference plane, fixed three-dimensional region, and occupancy threshold. Each occupied leaf voxel contributed one sample at its cell center.

For region r , the reference plane is written as $\mathbf{n}_r^T \mathbf{p} + b_r = 0$, with $\|\mathbf{n}_r\| = 1$. The signed distance from an occupied voxel center $\mathbf{p}_{r,i}$ to the plane is

$$d_{r,i} = \mathbf{n}_r^T \mathbf{p}_{r,i} + b_r. \quad (25)$$

The point-to-plane RMSE is then calculated as

$$E_{\text{plane},r} = \sqrt{\frac{1}{N_r} \sum_{i=1}^{N_r} d_{r,i}^2}, \quad (26)$$

where, N_r is the number of evaluated occupied voxel centers. This measure captures both displacement from the reference plane and dispersion of the reconstructed surface. It does not distinguish between the two.

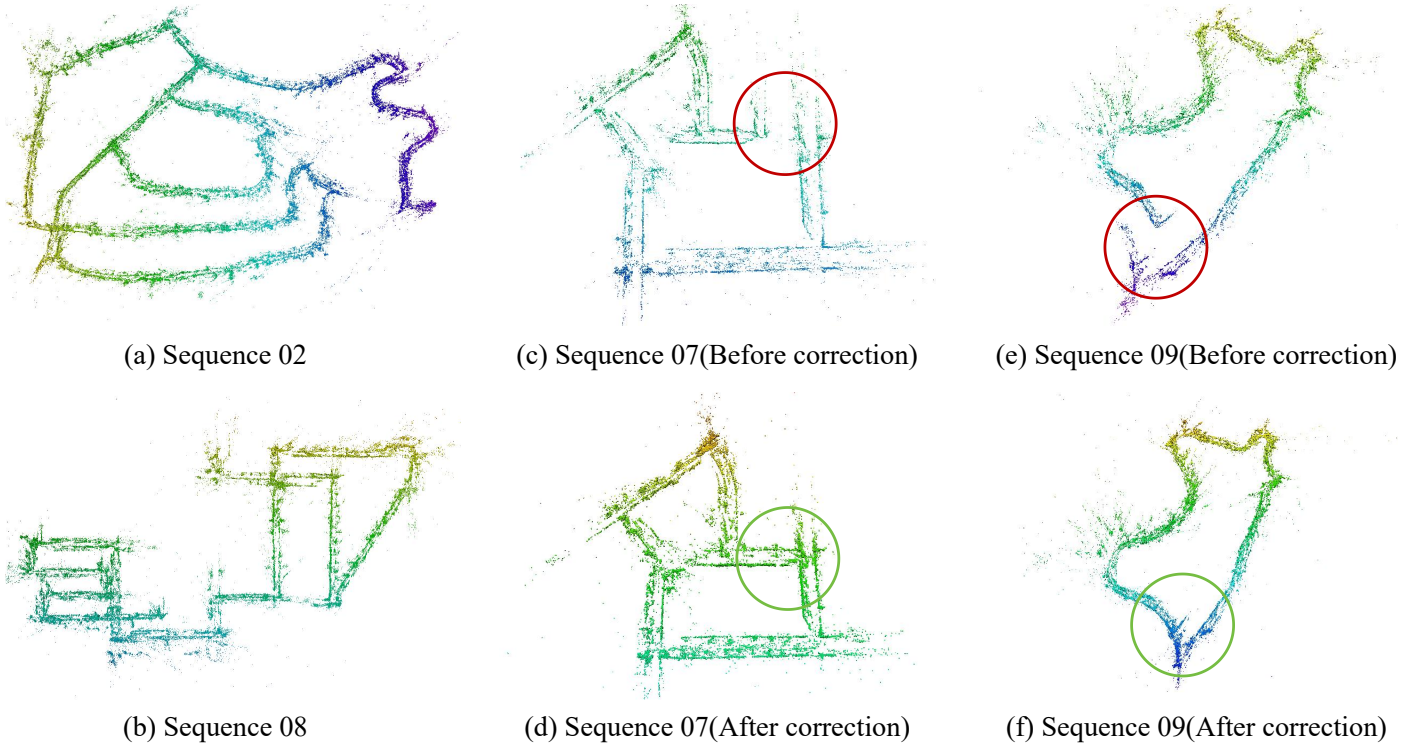


Fig. 11. Semantic octree maps on KITTI Sequences 02 and 08 (a)–(b), with comparisons before and after keyframe-anchored correction on Sequence 07 (c)–(d) and Sequence 09 (e)–(f). Circles mark regions with visible differences in structural alignment.

TABLE VIII. PAIRED EVALUATION OF ONE STATIC PLANAR REGION AT ONE SELECTED CORRECTION EVENT PER KITTI SEQUENCE. E_{plane} AND W_{90} ARE REPORTED IN METERS; N IS THE NUMBER OF EVALUATED OCCUPIED LEAF VOXELS IN EACH MAP SNAPSHOT.

Sequence	Map	$E_{\text{plane}} \downarrow$	$W_{90} \downarrow$	N
07	Uncorrected	0.68	1.72	13600
	Corrected	0.48	0.95	11413
09	Uncorrected	2.65	2.10	9960
	Corrected	1.93	1.26	7968

Surface thickness along the plane normal is also measured using

$$W_{90,r} = Q_{0.95}(d_r) - Q_{0.05}(d_r), \quad (27)$$

where, $Q_q(d_r)$ is the q -quantile of the signed-distance distribution. Thus, $W_{90,r}$ spans the central 90% of the distances. Separated surface layers may increase this width, although depth noise, voxel discretization, and surface irregularity also affect it.

Both metrics decreased after correction in the evaluated regions, as reported in Table VIII. On Sequence 07, the point-to-plane RMSE fell from 0.68 m to 0.48 m (29.41%), while surface thickness decreased from 1.72 m to 0.95 m (44.77%). Sequence 09 showed a similar trend: RMSE decreased from 2.65 m to 1.93 m, and thickness from 2.10 m to 1.26 m, corresponding to reductions of 27.17% and 40.00%, respectively. These results suggest improved alignment with the reference planes and reduced spread along their normals in the selected regions. Substantial residual error remained on Sequence 09.

TABLE IX. ABLATION STUDY OF THE DYNAMIC SUPPRESSION MODULE

DEPTH-ADAPTIVE SEMANTIC GATING	PROBABILISTIC GEOMETRIC VERIFICATION	RMSE (m)	RELATIVE PERCENTAGE REDUCTION IN RMSE
-	-	0.026	-
✓	-	0.021	19.23%
-	✓	0.024	7.69%
✓	✓	0.017	34.62%

The number of evaluated occupied voxels also decreased, from 13600 to 11413 on Sequence 07 and from 9960 to 7968 on Sequence 09. Spatial merging can account for part of this change. However, correction may also move voxels across the fixed region boundaries, so the reduction in voxel count cannot be attributed to merging alone.

E. Ablation Study

Table IX reports the ablation results for the dual-channel dynamic suppression module on highly dynamic sequences from the TUM dataset, with DynaSLAM serving as the reference baseline for semantic dynamic-object suppression. Although integrating either depth-adaptive semantic gating or probabilistic geometric verification independently improves upon the traditional baseline, their synergistic fusion achieves the lowest RMSE. This confirms their complementary roles in preserving static backgrounds while rejecting dynamic outliers.

Table X presents a controlled ablation of geometry-guided fast matching, orthonormal line representation, and adaptive information matrix scaling within PLCS-SLAM. The internal

TABLE X. CONTROLLED ABLATION OF POINT-LINE COMPONENTS IN PLCS-SLAM

GEOMETRY-GUIDED FAST MATCHING	ORTHONORMAL REPRESENTATION	ADAPTIVE INFORMATION MATRIX	MEAN ATE RMSE (m)	RELATIVE PERCENTAGE REDUCTION IN RMSE
-	-	-	21.68	-
✓	-	-	16.40	24.35%
-	✓	-	17.04	21.40%
-	-	✓	12.92	40.41%
✓	✓	✓	10.19	53.00%

All reductions are calculated relative to the internal baseline with a mean ATE RMSE of 21.68 m.

baseline disables all three evaluated components while retaining dual-channel dynamic suppression and semantic mapping. All remaining modules and settings are kept unchanged across the variants. The original PL-SLAM results in Table VII serve only as an external comparison and are not used to calculate the reductions in this ablation. Each reported error is the arithmetic mean of the ATE RMSE values over the five UAV sequences, Test01–Test05.

The internal baseline has a mean ATE RMSE of 21.68 m. Adding geometry-guided fast matching reduces it to 16.40 m, while orthonormal line representation alone gives 17.04 m, corresponding to reductions of 24.35% and 21.40%, respectively. Adaptive information matrix scaling gives the largest reduction among the single-component variants. Its mean error is 12.92 m, or 40.41% below the baseline. This suggests that using front-end motion confidence to weight observations in back-end optimization can help reduce trajectory error. With all three components enabled, the mean ATE RMSE falls to 10.19 m, a reduction of 53.00%.

Dynamic suppression, semantic mapping, and the other settings remain unchanged throughout this comparison. The 53.00% reduction therefore reflects the combined effect of the three evaluated point-line components under these conditions. The fully enabled variant is the complete PLCS-SLAM system, so its mean error matches the result in Table VII. The two ablation studies address different parts of the framework: the TUM results suggest complementary gains from the two dynamic suppression channels, while the UAV results show the additional accuracy gains obtained from the point-line components with dynamic suppression held fixed.

F. Parameter Sensitivity Analysis

Parameter sensitivity was evaluated on the TUM fr3/walking_xyz and fr3/walking_rpy sequences, which contain moving pedestrians under different camera motions. Five values were tested for each parameter $\lambda \in \{0.20, 0.30, 0.40, 0.50, 0.60\}$, $\tau_{static} \in \{0.60, 0.70, 0.80, 0.90, 0.95\}$, and $\tau_{overlap} \in \{0.30, 0.40, 0.50, 0.60, 0.70\}$. Only one parameter was varied in each test. The other parameters remained at the values in Table I, and the same trajectory evaluation procedure was used throughout.

Each configuration was run three times. Table XI lists the mean ATE RMSE of completed runs and the number of successful runs out of three. Incomplete runs were counted as failures and excluded from the mean; where only one run completed, its error is reported directly. The reference configuration is $\lambda = 0.40$, $\tau_{static} = 0.80$, and $\tau_{overlap} = 0.50$.

The same reference results appear in all three parameter groups for comparison.

Varying λ changes the relative contributions of semantic and geometric weights. On fr3/walking_xyz, the mean ATE RMSE falls from 0.019 m at $\lambda = 0.20$ to 0.013 m at 0.40, before rising to 0.018 m at 0.60. The trend is similar on fr3/walking_rpy, with errors of 0.048 m, 0.030 m, and 0.041 m at these settings. Among the tested values, 0.40 gives the lowest mean error on both sequences. All runs complete at this setting, whereas one run fails on each sequence at 0.20 and on fr3/walking_xyz at 0.50 and 0.60.

For τ_{static} , fewer runs complete at 0.60 and 0.70 than at the higher tested values. At 0.60, only one run completes on each sequence, with errors of 0.025 m on fr3/walking_xyz and 0.041 m on fr3/walking_rpy. Two runs complete on each at 0.70. At 0.80, all three runs complete and the mean errors reach their lowest reported values in this group, 0.013 m and 0.030 m. Increasing the parameter to 0.90 or 0.95 retains full completion, but the mean errors rise on both sequences.

For $\tau_{overlap}$, the results are close at 0.40 and 0.50. Both give a reported mean ATE RMSE of 0.013 m on fr3/walking_xyz; on fr3/walking_rpy, the corresponding values are 0.031 m and 0.030 m. All three runs complete at both settings. Larger values are associated with higher errors and, on some runs, failure to complete the sequence. At 0.70, the mean errors are 0.019 m and 0.041 m, with two successful runs on fr3/walking_xyz and one on fr3/walking_rpy.

Within the tested ranges, the settings in Table I give low mean errors, and all three runs complete on both sequences. These results support retaining the selected values for the evaluated conditions. The errors at $\tau_{overlap} = 0.40$ are also close to those at 0.50. Since the means exclude incomplete runs, the RMSE values should be considered together with the completion counts.

G. Segmentation Network Training and Evaluation

The modified network uses MobileNetV3-Large as its backbone, with a width multiplier of 1.0. Its BiFPN consists of three repeated fusion cells, each with 64 channels per feature level. The Box Head incorporates global average pooling, and the Mask Head uses depthwise separable convolutions. For training, the backbone was initialized with the ImageNet-1K weights provided by TorchVision. The BiFPN, RPN, Box Head, and Mask Head were initialized separately: new convolutional layers used Kaiming initialization, while fully connected prediction layers used a zero-mean normal distribution with a standard deviation of 0.01. All biases were set to zero.

The complete network was trained on the COCO 2017 train2017 split [42]. All 80 foreground categories were assigned contiguous labels, with one additional label for the background. The mapping to the original COCO category identifiers was retained for evaluation. When constructing semantic priors, person, bicycle, car, motorcycle, bus, truck, and train are treated as potentially movable categories. These labels provide motion priors; the final feature weights also depend on geometric verification.

TABLE XI. SENSITIVITY OF ATE RMSE TO THE THREE PARAMETERS ON TWO TUM WALKING SEQUENCES

Parameter	Value	fr3/walking_xyz		fr3/walking_rpy	
		ATE RMSE (m)	Successful runs (/3)	ATE RMSE (m)	Successful runs (/3)
λ	0.20	0.019	2	0.048	2
	0.30	0.014	3	0.035	3
	0.40	0.013	3	0.030	3
	0.50	0.016	2	0.033	3
	0.60	0.018	2	0.041	3
τ_{static}	0.60	0.025	1	0.041	1
	0.70	0.019	2	0.037	2
	0.80	0.013	3	0.030	3
	0.90	0.016	3	0.034	3
	0.95	0.019	3	0.036	3
$\tau_{overlap}$	0.30	0.015	3	0.033	3
	0.40	0.013	3	0.031	3
	0.50	0.013	3	0.030	3
	0.60	0.015	3	0.036	2
	0.70	0.019	2	0.041	1

Training ran for 26 epochs using SGD on a single GPU, with a batch size of two images. Momentum was set to 0.9 and weight decay to 10^{-4} . The initial learning rate was 0.0025, with a tenfold reduction after epochs 16 and 22. Images were resized to a target shorter side of 640 pixels, subject to a maximum longer side of 1024 pixels, while preserving their aspect ratio. Random horizontal flipping was applied with a probability of 0.5, and automatic mixed precision was enabled.

All learnable components were optimized during training. The running statistics of the pretrained backbone's batch normalization layers, however, were kept fixed. After each epoch, the checkpoint was evaluated on COCO 2017 val2017, and the one with the highest validation mask AP was selected. Training did not use any TUM, KITTI, or UAV evaluation images.

Segmentation accuracy was measured on COCO 2017 val2017 using the COCO mask evaluation protocol, covering all 80 object categories. Mask AP was computed over IoU thresholds from 0.50 to 0.95 in increments of 0.05. Mask AP₅₀ used a single IoU threshold of 0.50.

For comparison, the reference model was Mask R-CNN with a ResNet-50-FPN backbone and the TorchVision COCO_V1 checkpoint. Both networks used the same validation images, resizing rule, and category definitions. Latency was measured on a platform equipped with an Intel Core i9-14900HX processor and an NVIDIA GeForce RTX 4060 Laptop GPU. Both models ran with a batch size of one in FP32 precision.

Latency measurements were taken after warm-up, with GPU synchronization to account for asynchronous execution. The timed operations covered image resizing and normalization, CPU-to-GPU transfer, network inference, mask post-processing, and transfer of the required outputs back to the CPU. Disk loading and visualization were excluded. Table XII summarizes the segmentation accuracy, parameter counts, and mean latency.

The modified network achieved a mask AP of 35.50% and

TABLE XII. INSTANCE SEGMENTATION ACCURACY AND INFERENCE COST

Network	Mask AP (%)	Mask AP ₅₀ (%)	Parameters (M)	Mean segmentation latency (ms)
Mask R-CNN	31.74	53.40	44.40	43.67
Modified Mask R-CNN	35.50	55.30	23.91	36.10

a mask AP₅₀ of 55.30%, which were 3.76 and 1.90 percentage points higher than those of the reference model, respectively. It also used fewer parameters. The parameter count decreased from 44.40 million to 23.91 million, a reduction of 46.15%. On the tested platform, mean segmentation latency decreased from 43.67 ms to 36.10 ms, representing a reduction of 17.33%.

H. Runtime and Memory Evaluation

Module profiling and system-level benchmarking used the platform and segmentation settings described in Section V-G, with separate runs under the same algorithm configuration. The evaluated sequences were TUM freiburg3_walking_rpy, KITTI Sequence 00, and UAV Test01. Each sequence was run three times, restarting the process and clearing the map before each run. Timing began after model loading, warm-up, and successful SLAM initialization. Visualization and file saving were disabled.

Per-frame and keyframe-related modules were profiled on the TUM sequence, while loop closure and voxel correction were measured on KITTI Sequence 00. Module timings excluded image decoding. The segmentation pipeline followed the timing boundaries defined in the preceding subsection. Table XIII lists the execution units and mean durations calculated from the recorded invocations across the three runs. The segmentation latency in Table XII was measured on COCO val2017, whereas the corresponding module timing in Table XIII was measured on the TUM sequence.

Point-line matching and pose estimation had the longest mean duration among the per-frame modules, at 33.42 ms.

TABLE XIII. PROCESSING TIME OF INDIVIDUAL MODULES

Module	Execution unit	Mean (ms)
Semantic segmentation pipeline	Frame	25.32
ORB extraction	Frame	10.65
LSD and LBD computation	Frame	26.48
Dynamic verification and clipping	Frame	26.79
Point-line matching and pose estimation	Frame	33.42
Local mapping and BA	Invocation	104.88
Semantic octree fusion	Keyframe update	238.09
Loop closure processing	Event	165.27
Voxel correction	Event	102.51

TABLE XIV. SYSTEM THROUGHPUT, POSE LATENCY, AND MEMORY USAGE

Dataset	Input size	Throughput (FPS)	Mean pose latency (ms)	Peak RAM (GiB)	Peak GPU memory (GiB)
TUM	640 × 480	8.72	116.43	13.32	1.64
KITTI	1241 × 376	8.03	139.58	22.73	3.15
UAV	1280 × 720	7.61	161.41	26.83	6.84

Dynamic verification and clipping, LSD and LBD computation, and semantic segmentation took 26.79 ms, 26.48 ms, and 25.32 ms, respectively, while ORB extraction required 10.65 ms. The mapping stages required more time per invocation but ran less frequently, with semantic octree fusion operating on keyframe updates and loop-related corrections triggered by events. Their overall cost depended on the invocation frequency. Since some modules overlapped in execution, their durations could not be added directly to obtain pose latency.

For system-level benchmarking, images were replayed offline without delays based on the dataset timestamps and decoded on demand. The input queue held at most one waiting frame. When it was full, the reader waited before submitting the next image. Every processed frame underwent segmentation using its own mask; masks from earlier frames were not reused. No frames were intentionally dropped, and tracking failures were recorded separately.

Frame identifiers linked the timestamp at queue entry to the time when the corresponding pose became available, using the same monotonic clock. Pose latency included waiting after submission but excluded decoding and waiting for queue space before submission. Throughput was calculated from the processed frame count and the elapsed time between the first submission and completion of the last frame. Loading and decoding within this interval contributed to the elapsed time. Measurement durations, processed frame counts, and valid-pose counts were retained in the logs.

Memory was sampled every 100 ms until pending mapping tasks had completed. At each sampling time, host memory was calculated as the sum of the RSS values reported by psutil for the SLAM process and any separate segmentation process. GPU process memory was obtained through NVIDIA NVML and summed for the specified processes. Table XIV reports throughput and mean pose latency averaged over the three runs, together with the highest sampled memory totals in GiB.

The TUM sequence achieved the highest throughput and

lowest mean pose latency. Throughput was lower on KITTI and UAV Test01, with UAV Test01 also showing the longest latency and highest memory usage. Differences in input size, scene content, and the amount of data processed may partly explain these results. Map growth can further increase memory requirements and the cost of mapping and optimization.

VI. CONCLUSION

This paper presented PLCS-SLAM, a visual localization and mapping framework for autonomous vehicles and UAVs operating in dynamic and low-texture environments. The proposed method improves localization robustness by combining dual-channel dynamic verification with tightly coupled point-line optimization, so that useful static features can be preserved and geometric constraints can still be maintained when point observations become unreliable. To improve map quality after loop closure, a keyframe-anchored octree mapping strategy was further introduced to correct historical voxels together with pose updates, thereby reducing ghosting and improving global consistency. Experiments on the TUM, KITTI and a real-world UAV dataset show that PLCS-SLAM improves trajectory accuracy over several baselines and reduces local map misalignment on KITTI. Future work will focus on integrating IMU measurements to improve robustness and exploring deployment on embedded platforms for real-time operation.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (NSFC) Joint Fund of Civil Aviation Research under grant U2333214, the Sichuan Science and Technology Program under grant 2026YFHZ0275, the Sichuan Flight Engineering Technology Research Center Project under grant GY2024-11C and the Civil Aviation Administration of China Safety Capacity Building Project under grant MHAQ2024033.

REFERENCES

- [1] Y. Wang, Y. Tian, J. Chen, K. Xu, and X. Ding, "A survey of visual SLAM in dynamic environment: the evolution from geometric to semantic approaches," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–21, 2024.
- [2] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A survey on active simultaneous localization and mapping: state of the art and new frontiers," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1686–1705, 2023.
- [3] K. Ebadi *et al.*, "Present and future of SLAM in extreme environments: the DARPA SubT challenge," *IEEE Trans. Robot.*, vol. 40, pp. 936–959, 2024.
- [4] Z. Yu, L. Zhu, and G. Lu, "Tightly coupled fusion of VINS and motion constraint for autonomous vehicle," *IEEE Trans. Veh. Technol.*, vol. 71, no. 6, pp. 5799–5810, 2022.
- [5] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE and ACM Int. Symp. Mixed Augmented Reality (ISMAR)*, 2007, pp. 225–234.
- [6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [7] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Proc. German Conf. Pattern Recognit. (DAGM)*, 2010, pp. 11–20.

- [8] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: dense tracking and mapping in real time," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2011, pp. 2320–2327.
- [9] G. Graber, T. Pock, and H. Bischof, "Online 3D reconstruction using convex optimization," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, 2011, pp. 708–711.
- [10] Y. Fan, H. Han, Y. Tang, and T. Zhi, "Dynamic objects elimination in SLAM based on image fusion," *Pattern Recognit. Lett.*, vol. 127, pp. 191–201, 2019.
- [11] B. Cañovas, M. Rombaut, A. Nègre, D. Pellerin, and S. Olympi-eff, "Speed- and memory-efficient dense RGB-D SLAM in dynamic scenes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 4996–5001.
- [12] S. Song, H. Lim, A. J. Lee, and H. Myung, "DynaVINS: a visual-inertial SLAM for dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 11 523–11 530, 2022.
- [13] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, 2018.
- [14] R. Long, C. Rauch, T. Zhang, V. Ivan, and S. Vijayakumar, "Rigid-Fusion: robot localisation and mapping in environments with large dynamic rigid objects," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3703–3710, 2021.
- [15] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: tightly coupled multi-object tracking and SLAM," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5191–5198, 2021.
- [16] K. Qiu, T. Qin, W. Gao, and S. Shen, "Tracking 3-D motion of dynamic objects using monocular visual-inertial sensing," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 799–816, 2019.
- [17] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [18] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 834–849.
- [19] S. Wen, S. Tao, X. Liu, A. Babiarz, and F. R. Yu, "CD-SLAM: a real-time stereo visual-inertial SLAM for complex dynamic environments with semantic and geometric information," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–8, 2024.
- [20] C. Campos, R. Elvira, J. J. Gómez Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: an accurate open-source library for visual, visual-inertial, and multi-map SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [21] Y. Wang and S. Huang, "Towards dense moving object segmentation-based robust dense RGB-D SLAM in dynamic scenarios," in *Proc. Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, 2014, pp. 1841–1846.
- [22] M. Rünz and L. Agapito, "Co-Fusion: real-time segmentation, tracking and fusion of multiple objects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 4471–4478.
- [23] C. Yu, Z. Liu, X. J. Liu *et al.*, "DS-SLAM: a semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2018, pp. 1168–1174.
- [24] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: a visual dynamic object-aware SLAM system," *arXiv preprint arXiv:2005.11052*, 2020.
- [25] Y. Zhang, M. Bujanca, and M. Luján, "NGD-SLAM: towards real-time dynamic SLAM without GPU," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2025, pp. 3467–3473.
- [26] R. Gómez-Ojeda, F. A. Moreno, D. Zúñiga-Noël, D. Scaramuzza, and J. González-Jiménez, "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.
- [27] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, "PL-VIO: tightly coupled monocular visual-inertial odometry using point and line features," *Sensors*, vol. 18, no. 4, p. 1159, 2018.
- [28] Q. Fu, J. Wang, H. Yu, I. Ali, F. Guo, Y. He, and H. Zhang, "PL-VINS: real-time monocular visual-inertial SLAM with point and line features," *arXiv preprint arXiv:2009.07462*, 2020.
- [29] C. Yuan, Y. Xu, and Q. Zhou, "PLDS-SLAM: point and line features SLAM in dynamic environment," *Remote Sens.*, vol. 15, no. 7, p. 1893, 2023.
- [30] Z. Lin, Q. Zhang, Z. Tian, P. Yu, and J. Lan, "DPL-SLAM: enhancing dynamic point-line SLAM through dense semantic methods," *IEEE Sensors J.*, vol. 24, no. 9, pp. 14 596–14 607, 2024.
- [31] B. Zhang, Y. Dong, Y. Zhao, and X. Qi, "DynPL-SLAM: a robust stereo visual SLAM system for dynamic scenes using points and lines," *IEEE Trans. Intell. Veh.*, 2024.
- [32] Y. Liu, R. Qu, M. Xu, C. Li, and H. Jiang, "A reliability-aware visual-inertial odometry for dynamic and low-texture environments," *Int. J. Adv. Comput. Sci. Appl.*, vol. 17, no. 5, pp. 1020–1030, 2026.
- [33] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.
- [34] E. Vespa, N. Funk, P. H. J. Kelly, and S. Leutenegger, "Adaptive-resolution octree-based volumetric SLAM," in *Proc. Int. Conf. 3D Vis. (3DV)*, 2019, pp. 654–662.
- [35] V. Reijgwart, C. Cadena, R. Siegwart, and L. Ott, "Efficient volumetric mapping of multi-scale environments using wavelet-based compression," *arXiv preprint arXiv:2306.01279*, 2023.
- [36] H. Min, K. M. Han, and Y. J. Kim, "OctoMap-RT: fast probabilistic volumetric mapping using ray-tracing GPUs," *IEEE Robot. Autom. Lett.*, vol. 8, no. 9, pp. 5696–5703, 2023.
- [37] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2961–2969.
- [38] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 1314–1324.
- [39] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 10 778–10 787.
- [40] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2012, pp. 573–580.
- [41] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [42] T. Y. Lin, M. Maire, S. Belongie *et al.*, "Microsoft COCO: common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.