Shared-Encoder Reinforcement Learning for Multitask LiDAR–Inertial SLAM

Zhaotian Deng
School of Computer Science and Information Security
Guilin University of Electronic Technology
Guilin, China
xixitiantian@mails.guet.edu.cn

Ji Li
School of Computer Science and Information Security
Guilin University of Electronic Technology
Guilin, China
liji@guet.edu.cn

Qingjiao Meng
School of Computer Science and Information Security
Guilin University of Electronic Technology
Guilin, China
3446628847@qq.com

Weixin Zhang
School of Computer Science and Information Security
Guilin University of Electronic Technology
Guilin, China
zwxxx0021@mails.guet.edu.cn

Jicheng Yao
School of Computer Science and Information Security
Guilin University of Electronic Technology
Guilin, China
yaojicheng@mails.guet.edu.cn

Chuchu Zhu
School of Computer Science and Information Security
Guilin University of Electronic Technology
Guilin, China
zhuchuchu5@163.com

***Abstract*—LiDAR–IMU SLAM in long-term operation is still exposed to IMU drift, redundant keyframe insertion and unsafe loop-closure attempts, which jointly degrade pose accuracy and increase computational load. We integrate a multi-task reinforcement learning controller into the SLAM pipeline to address these coupled failure modes. The controller observes inertial statistics over a sliding window together with estimator load indicators and lightweight loop-closure cues. A shared encoder produces a compact latent state that drives three task heads: IMU bias denoising, adaptive keyframe selection and selective loop-closure triggering. The heads output bias correction for pre-integration, a keep or skip decision for each keyframe candidate and a trigger decision for loop verification, which are injected directly into the factor-graph back-end. This design improves accuracy, stability and real-time efficiency without altering the underlying optimization structure. On KITTI, compared with the baseline, our method reduces translational error by 6% and rotational error by 0.7%.**

***Index Terms*—SLAM, LiDAR-inertial odometry, Reinforcement learning**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a foundational capability for mobile robots and unmanned systems, whose goal is to estimate ego pose while building a map in unknown environments [1]. SLAM integrates multi sensor inputs such as LiDAR, cameras, and inertial measurement units to provide continuous, real time spatial perception and localization across applications including autonomous driving, aerial inspection, and service robotics. Methodologically, SLAM has two major lines, LiDAR based and Vision based [2]. The former relies on geometric range measurements and is mature in engineering practice, whereas the latter leverages image observations with advantages in cost and weight, making it attractive for resource constrained platforms.

LiDAR–inertial SLAM typically fuses point clouds with IMU and can be grouped into two families. The first emphasizes geometric features and factor graph optimization: edge and plane features constrain the pose, followed by loop closure and global optimization in the back end, exemplified by LeGO-LOAM and LIO-SAM [3,4]. These methods achieve strong accuracy in structured scenes but are sensitive to thresholds for feature extraction, keyframe insertion, and loop triggering. The second family stresses recursive high rate fusion and tightly coupled factors, down weighting explicit feature matching; FAST-LIO is a representative line that maintains robustness under agile motion and high frequency excitation [5]. Yet these systems still depend on hand tuned filter gains, noise models, and redundancy controls. Cross-scene deployment remains limited by the overhead of expert driven parameter tuning.

To reduce reliance on static rules and manual thresholds, recent work introduces Reinforcement Learning (RL) into key SLAM decisions. In active SLAM and navigation, Wen et al. use a fully convolutional residual network to obtain obstacle depth cues, apply a Dueling DQN policy for collision avoiding path planning, and build a two dimensional map online, enabling simultaneous mapping and autonomous navigation with both static and moving obstacles [6]. Khan et al. combine reward shaping and policy optimization with pose estimation and path tracking, improving stability and accuracy in dynamic scenes [7]. These results indicate that RL can learn adaptive strategies for loop triggering, keyframe management, and trajectory control, motivating our joint policy for IMU denoising, keyframe selection, and loop detection.

RL in SLAM is evolving from pure path planning toward joint optimization of active perception and map quality, replacing fixed hyperparameters with online policies. Placed and Castellanos formulate active SLAM as a model free deep RL problem whose reward embeds utility from optimal experimental design, yielding policies that explore unknown maps and generalize to unseen layouts [8]. DA-SLAM further introduces map completeness and pose uncertainty as coupled rewards, with two cooperating agents that encourage coverage and active loop closing to improve global consistency [9]. RASLS integrates layout semantics to enhance exploration and localization robustness [10]. A transfer learning–based meta Q-learning method with prioritized replay targets sparse rewards and uncertainty, accelerating convergence and improving generalization across environments [11]. An end-to-end LiDAR active SLAM system directly consumes 3D scans and incremental maps to output linear and angular velocities, uses intrinsic rewards to boost exploration, and achieves transferable autonomous exploration and looped mapping at large scale [12]. These trends motivate our objective: unify IMU denoising, keyframe selection, and loop triggering under a single policy that adapts across scenes without manual retuning. Although these advances show that learning based agents can reason about navigation, exploration and loop usage, several gaps remain. Most existing approaches still optimize a single aspect of the pipeline, for example global exploration gain or loop triggering, rather than coordinating sensing quality, keyframe density and pose graph consistency together. As a result they do not explicitly regulate redundancy versus constraint coverage for the back end. Many methods continue to rely on manually defined thresholds or a hand edited action space, which limits awareness of internal load, local map growth and sensor degradation during high motion. Reward designs in prior work tend to emphasize coverage, collision avoidance or trajectory tracking quality, but rarely measure downstream estimator quality such as optimization stability or loop correction reliability. In practice this means that even learning enabled SLAM systems often still depend on manual adjustment of noise levels, insertion criteria and closure logic during deployment, especially in long term or high dynamic conditions.

To address these limitations, we propose a unified adaptive control framework for LiDAR-inertial SLAM in which IMU denoising, keyframe selection, and loop closure triggering are treated as a single joint decision process rather than three independent tuning problems. The system converts raw runtime signals into structured temporal observations including IMU window statistics, keyframe candidate states, and loop candidate cues and encodes them through a shared representation. Three task heads then produce, in a single forward pass, an IMU bias correction to stabilize pre-integration, a keep/skip decision to regulate keyframe density, and a trigger/skip decision to control loop insertion. The learned controller is optimized to maximize long horizon mapping quality under realistic motion, sensing noise, and computational constraints, thereby removing the need for scene-specific thresholds and enabling online adaptation to changing environments.The contributions of this work are as follows:

- A unified adaptive control framework that jointly handles IMU denoising, keyframe selection, and loop-closure triggering, replacing scene-specific heuristic thresholds with a single learned decision process.
- A shared-encoder multitask policy that converts runtime signals into structured temporal observations and generates bias correction, keyframe confidence, and loop-trigger confidence in one forward pass.
- An integrated LiDAR–inertial SLAM system that reduces redundant keyframes and unnecessary loop attempts, achieving 6% lower translational error and 0.7% lower rotational error on the KITTI dataset compared with LIO-SAM.

II. Related Works

IMU denoising is critical in LiDAR–IMU fusion because unmodeled noise and slowly varying bias can amplify drift in odometry. Recent tightly coupled systems stabilize inertial residuals and bias at the front end: LIO-SAM couples IMU pre-integration with smoothing to balance local robustness and global consistency [4]; FAST-LIO2 leverages direct point to plane tracking with high rate IMU propagation to improve transient stability [13]; LINS uses iterative Kalman updates to reduce optimization burden while maintaining real time [14]. These methods usually assume fixed noise parameters or slow self calibration and may struggle under strong dynamics or nonstationary bias. We instead produce an online bias correction from a learned policy, preserving physical modeling while adapting to operating conditions.

Keyframe selection directly trades off constraint coverage against backend cost. Mainstream LiDAR–IMU pipelines rely on heuristics: minimum time or distance spacing coupled with pose change thresholds, plus local registration quality for final keep decisions. LIO-SAM provides a robust dual-threshold rule that works well within global optimization [4]; FAST-LIO2 reduces reliance on dense keyframes via continuous high rate updates, yet still depends on expert thresholds [13]. Fixed rules are brittle when scenes, load, or sensing quality change. We recast the second stage keep decision as a policy learning problem so that keep probabilities arise from state rather than fixed cutoffs.

Loop closure constraints are essential for long term consistency. In LiDAR settings, global retrieval often uses geometric or topological signatures; Scan Context introduces a ring based descriptor that scales to large maps and pairs well with geometric verification to suppress false closures [15]. In LiDAR–IMU systems, loop triggers commonly hinge on similarity thresholds and handcrafted geometric priors; thresholds that are too low waste computation, while those that are too high miss correction opportunities. Although systems like LIO-SAM and FAST-LIO2 support loop closure, their triggers are largely heuristic [4,13]. We let the policy control loop timing and frequency based on learned cues of geometric consistency to reduce spurious attempts.

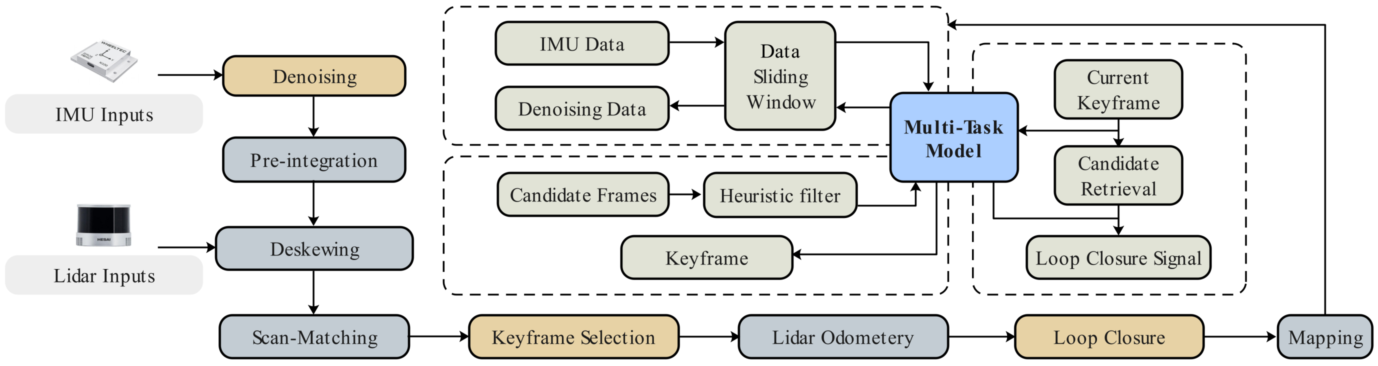

Fig. 1. Overall SLAM framework with a multi-task RL controller.

RL has been applied to LiDAR driven active exploration and mapping, showing potential to learn where and when to move without an explicit model. One line frames active SLAM as model free deep RL and embeds information gain surrogates in the reward, enabling policies that explore unseen environments and generalize across maps [16]. For large scale 3D LiDAR, end to end exploration networks have been trained to output linear and angular commands directly from scans and incremental maps, with intrinsic rewards improving efficiency and with transfer from simulation to hardware [17]. Most of these methods target navigation or exploration as a single task. By contrast, we target three runtime decisions inside LiDAR–IMU odometry—IMU denoising, keyframe selection, and loop triggering—via a multitask policy that shares an encoder and emits bias corrections and two binary actions to jointly optimize accuracy, redundancy, and global consistency.

III. METHODS

A. System Overview

This section, aligned with Fig.1, describes the online pipeline and the three gating components studied in this work. The system ingests high–rate inertial measurements and mid–rate LiDAR scans. Temporally, a short–horizon buffer aggregates raw IMU and system observations from the previous mapping step; spatially, local scan matching provides relative motion and quality indicators, which are written back to the buffer for the next decision step. The three research foci are: sliding–window denoising to stabilize IMU bias correction and improve undistortion and registration robustness; two–stage keyframe selection to balance sparsity and linearization stability while reducing redundant insertions; and dual–stream loop–closure detection that lowers unnecessary verifications under a unified real–time budget. The learned modules gate whether an attempt is worthwhile, whereas costly geometric optimization and consistency checks remain the final authority, preserving interpretability and auditability.

Let the adjacent LiDAR timestamps be t_k and t_{k+1} with current time t satisfying $t_k < t < t_{k+1}$. Inputs at time t include the raw IMU stream $z^{\mathrm{imu}}(t) = [\mathbf{a}(t), \boldsymbol{\omega}(t)]$, the system observation produced after mapping at t_k, denoted o_k^{sys}, and the online system observation available at time t, denoted $\bar{o}^{\mathrm{sys}}(t)$. The sliding window is assembled as

$$\mathcal{W}(t) = \left[\, z^{\mathrm{imu}}_{[\,t_k,\,t\,]},\ o_k^{\mathrm{sys}},\ \bar{o}^{\mathrm{sys}}(t) \,\right]$$

where $z^{\mathrm{imu}}_{[\,t_k,\,t\,]}$ denotes the ordered set of raw IMU samples from t_k to t. Features in $\mathcal{W}(t)$ are standardized using statistics estimated on training data and kept fixed at deployment, then fed to a shared temporal encoder and task heads to obtain an instantaneous IMU–bias estimate $\hat{\mathbf{b}}_t$. To enhance online stability, exponential smoothing is applied:

$$\tilde{\mathbf{b}}_t = \alpha\, \tilde{\mathbf{b}}_t^{-} + (1-\alpha)\, \hat{\mathbf{b}}_t$$

where $\tilde{\mathbf{b}}_t^{-}$ is the pre–update smoothed bias and α is the smoothing coefficient. The correction is used for the IMU segment on $[t_k,\, t]$ and injected into the back–end as a prior factor rather than overwriting raw measurements; magnitude clipping and anomaly holds are enforced. The pair $\tilde{\mathbf{b}}_t$ and o_k^{sys} is written back to the buffer for the subsequent step.

Two–stage keyframe selection follows a single–direction arbitration. The heuristic pre–filter (denoted H) first evaluates the $(k+1)$-th LiDAR frame using geometric increments and registration quality. Only if H rejects the frame, the policy head performs a second decision using the observation vector $\mathbf{x}_{k+1} = [\Delta\mathbf{p}_{k+1}, \Delta\boldsymbol{\theta}_{k+1}, q_{k+1}, o_k^{\mathrm{sys}}, \bar{o}^{\mathrm{sys}}(t)]$, yielding the insertion confidence p_{k+1}^{kf}. The overall rule keeps a single equation:

$$\mathcal{K}_{k+1} = \mathcal{K}_{k+1}^{\mathrm{H}} \cup \left\{\, k+1 \;\middle|\; [H_{k+1} = 0] \cap [\mathrm{RL}(\mathbf{x}_{k+1}) > 0.85] \,\right\}$$

where $\mathcal{K}_{k+1}^{\mathrm{H}}$ denotes frames accepted by the heuristic, H_{k+1} is its binary decision, and $\mathrm{RL}(\mathbf{x}_{k+1})$ is the policy confidence. The threshold is fixed at 0.85. This design limits the learned module to borderline cases, improving recall while constraining graph growth.

Loop–closure detection adopts a dual–stream mechanism with learning–based triggering and geometric verification. The direct stream takes relative pose increments, registration quality, the previous system observation o_k^{sys}, the current online observation $\bar{o}^{\mathrm{sys}}(t)$, and stability cues from sequential odometry to produce a trigger confidence u^{dir}. The retrieval stream obtains lightweight candidates and augments the direct–stream observation with topological similarity and coarse overlap, producing a maximal candidate confidence u^{ret}. The two streams are unified before verification by taking the maximum; verification runs only after a trigger and relies on precise alignment and consistency checks. The final rule is

$$L = \begin{cases} 1, & \max\left(u^{\text{dir}},\, u^{\text{ret}}\right) \;\geq\; \tau \;\; \cap \;\; V = 1 \\ 0, & \text{otherwise} \end{cases}$$

where L is the loop–closure decision, τ is the trigger threshold, and V indicates success of geometric verification. The approach uses low–cost context for triggering and high–cost geometry for confirmation, reducing wasted verification while maintaining interpretability under a unified real–time budget.

Training uses the KITTI sequences 0016 and 0018. The observation set comprises raw sensing and online runtime information from these sequences, including the IMU preintegration measurement $\mathbf{z}^{\text{imu}}_{[t_k,\,t]}$, the translation increment $\Delta\mathbf{p}_{k+1}$, the rotation increment $\Delta\boldsymbol{\theta}_{k+1}$, the pose quaternion $\mathbf{q}_{k+1}$, the system online observation $\mathbf{o}^{\text{sys}}_k$, and its time-continuous counterpart $\bar{\mathbf{o}}_{\text{sys}}(t)$. Supervision signals and geometric verification labels are generated offline from LIO-SAM mapping and registration results and are kept fixed during deployment.

In Fig.2, the RL pipeline comprises three layers aligned with deployment. The observation layer aggregates the system observation produced after the previous mapping step, the current online system observation, and lightweight geometric and quality features of the present LiDAR frame; statistics for normalization are estimated on training data and kept fixed at deployment with symmetric clipping for robustness. A shared temporal encoder then extracts a stable representation on a single channel and reuses hidden state across steps to respect real-time constraints and enable incremental computation. Finally, multi-task policy heads produce three outputs that mirror the runtime gates: an IMU bias estimate for denoising, a keyframe insertion confidence, and a loop closure trigger confidence. Training uses a unified loss weighting and a consistent data interface between training and deployment to avoid leakage and to preserve auditability, so that decisions made by the learned gates remain reproducible and interpretable in the full SLAM pipeline.

B. Observation Layer

In Fig. 2, the observation layer collects only causal, lightweight inputs associated with the LiDAR frame at index $k{+}1$. The current time satisfies $t_k < t < t_{k+1}$, ensuring that all features are derived solely from past or present information. The current time satisfies $t_k < t < t_{k+1}$, ensuring that all features are derived solely from past or present information. All scalar features are standardized using statistics estimated on training data and kept fixed at deployment; missing entries are masked for the downstream encoder. Observation assembly runs asynchronously off the back-end thread. We define the unified observation vector as

$$\mathbf{x}_{k+1} = \left[\, z^{\text{imu}}_{[t_k,\,t]},\; \Delta\mathbf{p}_{k+1},\; \Delta\boldsymbol{\theta}_{k+1},\; q_{k+1},\; o^{\text{sys}}_k,\; \bar{o}^{\text{sys}}(t) \,\right]$$

where $z^{\text{imu}}_{[t_k,\,t]}$ is the ordered set of raw IMU samples from t_k to t; $\Delta\mathbf{p}_{k+1}$, $\Delta\boldsymbol{\theta}_{k+1}$, and q_{k+1} are the geometric increments

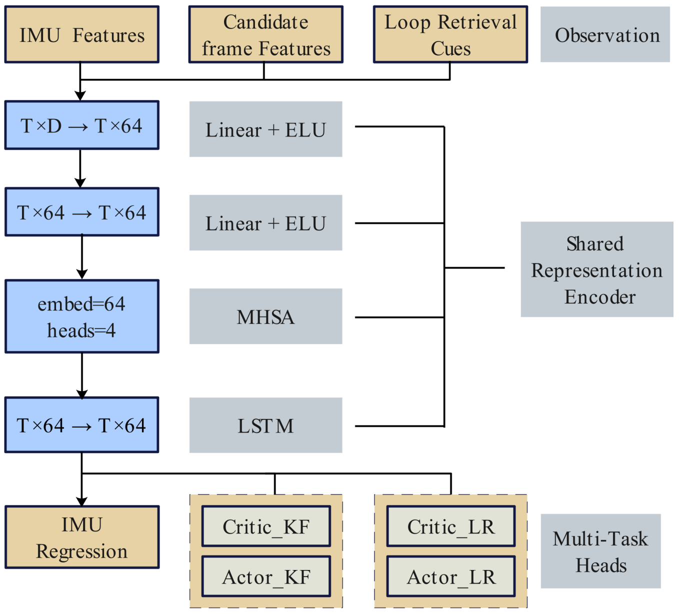

Fig. 2. Schematic of the multitask reinforcement learning model with a shared encoder and three task heads.

and registration quality of the current LiDAR frame; o^{sys}_k is the system observation produced after mapping at t_k, and $\bar{o}^{\text{sys}}(t)$ is the online system observation at time t. Under this mapping to Fig.2: *IMU Features* correspond to $z^{\text{imu}}_{[t_k,\,t]}$; *Candidate frame Features* correspond to $[\Delta\mathbf{p}_{k+1},\, \Delta\boldsymbol{\theta}_{k+1},\, q_{k+1}]$; and *Loop Retrieval Cues* correspond to $[o^{\text{sys}}_k,\, \bar{o}^{\text{sys}}(t)]$. The unified interface $\mathbf{x}_{k+1}$ preserves causality and deployment parity, and is shared by the denoising, keyframe selection, and loop-closure triggering heads.

C. Shared Temporal Encoder

This module converts the unified observation into task ready inputs under a strict causal constraint. The pipeline contains four components in sequence. First, a linear projection with ELU maps each time step to a unified channel of 64 dimensions, aligning feature scales and reducing the impact of extreme values. Second, another linear projection with ELU performs a lightweight reshaping inside the unified channel, smoothing the key–value query representations and improving numerical stability. Third, a multi-head self-attention layer with a causal mask aggregates information across time and modalities without exposing future steps; the current-frame context token and the historical IMU tokens can attend to each other so that operational status and inertial evidence are aligned. The number of heads is four and the embedding dimension is 64. Fourth, a single layer LSTM performs sequential recursion along time while reusing the previous hidden state at inference, which preserves slowly varying evidence and keeps the forward latency within a unified budget; if inputs are temporarily missing, the hidden state is held and missing entries are masked.

Inputs are organized as two types of tokens. One is the IMU sequence unfolded from t_k to t. The other is the current frame context token formed by concatenating $[\Delta\mathbf{p}_{k+1},\, \Delta\boldsymbol{\theta}_{k+1},\, q_{k+1},\, o_k^{\mathrm{sys}},\, \bar{o}^{\mathrm{sys}}(t)]$ and placed at the end of the sequence. The two token types first pass through the two linear projections with ELU for alignment, then enter the causal self attention for cross time and cross modal aggregation, and finally the LSTM for causal recursion and sequential memory. In this way, the context token absorbs the most relevant inertial evidence from history, while the IMU sequence becomes aware of the current operational state and registration quality, yielding a stable temporal representation.

The online IMU bias is obtained from the causal encoding of the IMU sequence with context. The encoded vector at the last IMU time step is taken as the compressed representation of the current inertial segment and is mapped by an affine readout

$$\hat{\mathbf{b}}_t = W_b\,\mathbf{h}_t + \mathbf{b}_b$$

where $\mathbf{h}_t$ is the 64 dimensional encoder output at time t, and $W_b, \mathbf{b}_b$ are the parameters of the bias readout. The estimate $\hat{\mathbf{b}}_t$ is then exponentially smoothed and injected into the back end as a prior factor, preserving consistency with the pre-integration noise model. The rationale for combining self attention with recurrence is that the former aligns and aggregates cross time and cross modal cues under causality, whereas the latter accumulates low frequency drift evidence, together yielding a stable online bias.

The policy observation for keyframe selection and loop closure triggering is taken from the encoded vector at the context token position and mapped by an affine readout

$$\mathbf{o}_{k+1}^{\mathrm{rl}} = W_r\,\mathbf{c}_{k+1} + \mathbf{b}_r$$

where $\mathbf{c}_{k+1}$ is the 64 dimensional encoder output at the context token, and $W_r, \mathbf{b}_r$ are the parameters of the task readout. This observation is fed to the corresponding policy head to produce either the keyframe insertion confidence or the loop closure trigger confidence, which are then used by the decision rules and the geometric verification gate described elsewhere. Throughout, only the already defined observation components are used, scalar features follow the same normalization and symmetric clipping, and missing entries are propagated by masks to ensure causality, reproducibility, and auditability.

D. Multi-Task Policy Heads

This module comprises two Bernoulli policies with separate value functions: keyframe re-screening and loop-closure triggering. Inputs are the shared-encoder readouts. The keyframe branch consumes the encoded vector formed from $[\Delta\mathbf{p}_{k+1},\, \Delta\boldsymbol{\theta}_{k+1},\, q_{k+1},\, o_k^{\mathrm{sys}},\, \bar{o}^{\mathrm{sys}}(t)]$. The loop branch consumes the encoded vector formed from $[o_k^{\mathrm{sys}},\, \bar{o}^{\mathrm{sys}}(t)]$ and the sequence context. Actions are $a_{k+1} \in \{0,1\}$ and $a_{k+1}^{\mathrm{lc}} \in \{0,1\}$. Policy outputs are p_{k+1}^{kf} and u_{k+1}^{lc}. Value functions are V^{kf} and V^{lc}. We train with PPO using $\gamma = 0.99$, $\lambda = 0.95$, clipping ratio 0.2, entropy weight 0.01, gradient clipping at 1.0. Adam is used with learning rate 3×10^{-4} linearly annealed to zero, rollout length L, minibatch size 32, and 10 epochs per update. The value loss is Huber and advantages are normalized within each batch. The keyframe threshold 0.85 and the loop trigger threshold τ are calibrated on a development set and frozen before deployment. When the heuristic already accepts or rejects a frame, or when the system is over budget, we mask actions or fall back to the heuristic and log all fields for audit.

The keyframe reward encourages insertion when motion and registration quality are sufficient while penalizing redundancy

$$r_{k+1}^{\mathrm{kf}} = a_{k+1}\big(\alpha_1\|\Delta\mathbf{p}_{k+1}\| + \alpha_2\|\Delta\boldsymbol{\theta}_{k+1}\| + \alpha_3 q_{k+1}\big) - \alpha_4 a_{k+1}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are fixed weights. The loop-closure reward promotes triggers that pass geometric verification while accounting for trigger cost

$$r_{k+1}^{\mathrm{lc}} = a_{k+1}^{\mathrm{lc}} V_{k+1} - \beta a_{k+1}^{\mathrm{lc}}$$

where $V_{k+1} \in \{0,1\}$ indicates success of the geometric verification and β is a cost weight. During training both rewards are standardized to zero mean and unit variance according to feature ranges and symmetrically clipped to $[-r_{\max}, r_{\max}]$ for stability. The joint objective sums the standard PPO objectives of the two branches with entropy regularization

$$\max_{\theta}\ \mathbb{E}\big[\,\mathcal{J}_{\mathrm{kf}}^{\mathrm{PPO}}(\theta) + \mathcal{J}_{\mathrm{lc}}^{\mathrm{PPO}}(\theta) - \lambda_{\mathrm{ent}}\,\mathcal{H}(\pi_\theta)\,\big]$$

The two heads share the encoder parameters, use fixed rollout and minibatch schedules, and apply gradient-norm clipping on shared updates to avoid dominance by a single task. At inference, the keyframe branch outputs p_{k+1}^{kf} and is fused with the heuristic by the single rule already specified, while the loop branch outputs u_{k+1}^{lc} and is maximized with the retrieval stream before geometric verification. This design preserves train–deployment parity and meets real-time constraints while improving recall and reducing false triggers.

IV. Experimental Results

A. Configuration, Environment, and Parameters

Experiments run on Ubuntu 20.04 with an RTX 4090D GPU, a 15-core Xeon Platinum 8474C CPU, 80 GB RAM, and ROS1 Noetic. Training and inference use PyTorch 1.13.1. A single temporal window and a shared encoder feed three task heads that output IMU bias correction, keyframe keep probability, and loop trigger probability in one forward pass. Key settings: learning rate 3×10^{-4}, hidden size 64, four attention heads, discount 0.99, GAE decay 0.95, gradient clip 1.0, entropy 0.01, and joint loss weights IMU/KF/LC = 0.5/1/1.

B. KITTI Dataset and Baseline

We evaluate on representative KITTI urban street and suburban highway sequences that combine dense and sparse traffic, intersections and long straightaways, visible loop closures, and fast attitude changes. These properties jointly stress keyframe density control, loop trigger timing, and IMU denoising robustness. Baselines include LIO-SAM with community used settings for feature extraction and loop detection, and Dynamic-LIO [18], which detects moving objects via label consistency between current sweep points and their voxel localized neighbors in the global map.

C. Trajectory Accuracy Results

Table I reports translation and rotation accuracy over five sequences, comparing Dynamic-LIO [18], LIO-SAM, and our method. Overall, our method shows steadier translation convergence on urban blocks and suburban straightaways; on long trajectories with loop opportunities it yields larger end point gains, mainly due to more selective loop triggers and reduced redundancy before factor graph optimization. In segments with dense turns and pronounced speed changes, the learned bias correction suppresses short lived oscillations, lowering local RPE spikes while keeping rotation accuracy comparable to the baselines. Distributionally, the translational error exhibits fewer heavy tails and large error outliers, indicating fewer spurious associations or unnecessary loop attempts.

From a systems perspective, producing three decisions in one forward pass reduces cross interference among heuristics, stabilizes linearization points and edge sets, and decreases both relinearization churn and wasted geometric verifications. In scenes with moving traffic, loop candidates obtained by retrieval are filtered more conservatively before geometry, mitigating failure modes induced by dynamic occlusions. Although transient increases in rotation error can still occur under extreme dynamics and occlusions, they do not propagate into long horizon drift; subsequent loop factors restore the trajectory to a stable band. In sum, our Method improves translation accuracy and robustness without sacrificing real time operation.

As shown in Fig.3 on KITTI 0027, Dynamic-LIO drifts in the latter part of the run and fails to return exactly to the origin. LIO-SAM is steadier yet still accumulates error along long straight segments and near the final loop. By contrast, our method tracks the reference shape more tightly with no visible closing misalignment. Translational error shows fewer heavy tails and outliers, indicating fewer spurious associations and loop attempts. Producing three decisions in one forward pass reduces heuristic interference, stabilizes linearization points and edge sets, and cuts relinearization churn and wasted verifications. With moving traffic, loop candidates are filtered more conservatively before geometry, mitigating occlusion induced failures. Occasional rotation spikes under extreme dynamics do not propagate into long horizon drift; subsequent closures restore a stable trajectory band. Overall, our method improves translation accuracy and robustness while retaining real time operation.

TABLE I
POSE ACCURACY ON KITTI SEQUENCES
$\mathrm{ATE_T}$ [M], $\mathrm{ATE_R}$ [DEG]

Seq	**Dynamic-LIO**		**LIO-SAM**		**OURS**	
	$\mathrm{ATE_T}$	$\mathrm{ATE_R}$	$\mathrm{ATE_T}$	$\mathrm{ATE_R}$	$\mathrm{ATE_T}$	$\mathrm{ATE_R}$
0101	8.154	1.363	8.261	1.342	**7.812**	**1.272**
0051	7.328	2.170	7.890	2.428	7.891	2.432
0034	3.164	2.317	2.503	2.451	**2.369**	2.450
0033	12.39	2.485	9.064	2.469	**8.601**	**2.461**
0027	8.235	2.096	2.231	2.434	**2.059**	2.482

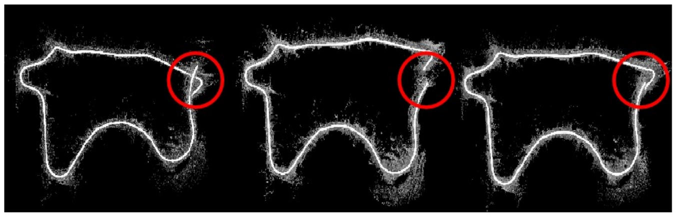

Fig. 3. Trajectory comparison on KITTI sequence 0027 using Dynamic-LIO, LIO-SAM, and our method.

D. Ablation Study

Table II contrasts four configurations: Base, IMU denoising, IMU denoising and KeyFrame policy, Full. The IMU denoising setting reduces extrapolation error in preintegration, shrinking both the mean and variance of ATE, which indicates effective smoothing of slow bias drift and high frequency noise. Adding the keyframe second stage decision further lowers ATE and reduces the number of keyframes, yielding a sparser, cleaner pose graph with smoother changes of linearization points and less oscillatory backtracking. In the Full configuration, learned loop triggering improves long horizon drift by timing closures when similarity and geometry are reliable; total loop attempts decrease while the proportion of successful closures increases, cutting down futile verifications.

Interpreted through decision cost, the sequence of gains follows a "stabilize noise, control density, then correct globally" progression: denoising secures local odometry, keyframe control alleviates graph scale and redundancy, and loop policy pushes global consistency at the right moments. Crucially, more loops are not inherently better than policy learns to trigger fewer but higher quality closures, avoiding the risk of injecting harmful constraints under weak similarity or unstable priors. In highly dynamic stretches, the Full variant may delay closures to preserve robustness but quickly converges once geometric consistency improves. These observations confirm the necessity of joint multitask learning: improvements from individual heads are not merely additive; they reinforce one another inside the factor graph optimizer to produce a more stable closed loop outcome.

TABLE II
ABLATION ON TRANSLATION ACCURACY
$\mathrm{ATE_T}$ [M]

Seq	**Base**	**IMU**	**IMU+KF**	**Full**
	$\mathrm{ATE_T}$	$\mathrm{ATE_T}$	$\mathrm{ATE_T}$	$\mathrm{ATE_T}$
0034	2.503	2.441	2.419	2.369
0033	9.064	8.891	8.840	8.601
0027	2.231	2.219	2.215	2.059

V. CONCLUSION

We presented a unified controller that inserts RL into a LiDAR–inertial SLAM backbone. IMU denoising, keyframe second stage selection, and loop closure triggering are produced in a single forward pass through a multitask observation layer, a shared temporal encoder, and three task heads. On KITTI urban and suburban sequences, the method consistently improves translation accuracy over LIO-SAM while maintaining comparable rotation accuracy, reduces redundant keyframes and unnecessary loop attempts, and preserves real time operation. Quantitatively, relative to the baseline it reduces translational error by 6% and rotational error by 0.7%. Ablations confirm a progressive benefit: stabilize sensor bias, control keyframe density, then correct globally with well timed closures, which validates the necessity of joint multitask learning.

There remain limitations. Performance depends on training coverage and reward shaping; distribution shifts across sensors, platforms, and motion profiles can still degrade generalization. In highly dynamic scenes with heavy occlusions, loop opportunities may be scarce and bias extrapolation can be stressed, occasionally yielding short lived attitude spikes. On resource limited platforms, the coordination between policy inference and back end optimization needs tighter scheduling to avoid contention for compute and memory. Future work will incorporate uncertainty aware rewards and risk sensitive objectives for more robust decisions, apply self distillation and model compression for edge deployment with bounded latency, and extend the framework to richer sensing and tasks such as semantic constraints and dynamic object suppression.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[2] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

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

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

[5] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR–inertial odometry package by tightly-coupled iterated Kalman filter," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021.

[6] S. Wen, Y. Zhao, X. Yuan, *et al.*, "Path planning for active SLAM based on deep reinforcement learning under unknown environments," *Intell. Serv. Robot.*, vol. 13, pp. 263–272, 2020.

[7] S. Khan, A. Niaz, D. Yinke, M. U. Shoukat, and S. A. Nawaz, "Deep reinforcement learning and robust SLAM based robotic control algorithm for self-driving path optimization," *Front. Neurorobot.*, vol. 18, Art. no. 1428358, 2025.

[8] J. A. Placed and J. A. Castellanos, "A deep reinforcement learning approach for active SLAM," *Appl. Sci.*, vol. 10, Art. no. 8386, 2020.

[9] M. Alcalde, M. Ferreira, P. González, F. Andrade, and G. Tejera, "DA-SLAM: Deep active SLAM based on deep reinforcement learning," in *Proc. Latin Amer. Robot. Symp. (LARS), Brazilian Symp. Robot. (SBR), and Workshop Robot. Educ. (WRE)*, 2022, pp. 282–287.

[10] C. Tian, S. Tian, Y. Kang, H. Wang, J. Tie, and S. Xu, "RASLS: Reinforcement learning active SLAM approach with layout semantic," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2024, pp. 1–8.

[11] X. Liu, S. Wen, Z. Guo, and H. Liu, "Transfer learning-based sparse-reward meta-Q-learning algorithm for active SLAM," *Expert Syst. Appl.*, vol. 299, Part A, Art. no. 129799, 2026.

[12] J. Chen, K. Wu, M. Hu, P. N. Suganthan, and A. Makur, "LiDAR-based end-to-end active SLAM using deep reinforcement learning in large-scale environments," *IEEE Trans. Veh. Technol.*, vol. 73, no. 10, pp. 14187–14200, Oct. 2024.

[13] W. Xu and F. Zhang, "FAST-LIO2: Fast direct LiDAR–inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, 2022.

[14] T. Qin, P. Li, and S. Shen, "LINS: A LiDAR–inertial state estimator for robust and efficient navigation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020, pp. 8899–8906.

[15] G. Kim and A. Kim, "Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 4802–4809.

[16] M. Tognon, A. Pretto, and E. Menegatti, "A deep reinforcement learning approach for active SLAM," *IEEE Access*, vol. 9, pp. 120814–120830, 2021.

[17] Z. Chen, X. Wang, and Y. Sun, "LiDAR-based end-to-end active SLAM using deep reinforcement learning in large-scale environments," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 1234–1241, 2023.

[18] Z. Yuan, X. Wang, J. Wu, J. Cheng, and X. Yang, "LiDAR-Inertial Odometry in Dynamic Driving Scenarios Using Label Consistency Detection," *arXiv*:2407.03590 [cs.RO], 2024, doi:10.48550/arXiv.2407.03590.