

Tightly Coupled LiDAR-IMU Odometry with Online Adaptation of a Learning-Based IMU Motion Model

Kazushi Fujimoto^{*1,2}, Kenji Koide^{*2}, Taku Okawara^{*2}, Aoki Takanose^{*2},
Shuji Oishi^{*2}, Masashi Yokozuka^{*2}, and Nobuhiko Nishio^{*1}

Abstract—In this work, we propose a LiDAR-IMU odometry algorithm based on the tight coupling of LiDAR-IMU state estimation and online learning of an IMU motion model. The learning-based IMU motion model in the proposed method learns sensor displacements from motion patterns, which avoids double integration of IMU data and providing robust motion prediction. It is combined with the LiDAR-IMU state estimation on a unified factor graph that allows us to use the rich geometric information in LiDAR data as supervision to train the IMU motion model online and overcome challenging situations where LiDAR data degrade and unseen motion models and devices. In addition, we propose an efficient on-factor-graph online learning of the neural network-based IMU motion model that reduces the computational burden of the optimization process. Consequently, the proposed framework achieves real-time online learning and odometry optimization. In the experiments, we demonstrated that the proposed method exhibits high resilience even in challenging environments such as out-of-domain mobility and differences in calibration settings, IMU devices, and sampling rates.

I. INTRODUCTION

Robust and accurate odometry estimation is essential for autonomous systems including self-driving cars and service robots. LiDAR odometry is one of the most promising approaches for motion prediction because rich geometrical and structured information provided by a LiDAR allows us to establish reliable motion estimation. However, LiDAR-based approaches degrade in environments with limited geometric features (e.g. tunnels and flat terrain) or when point clouds are degraded by fog and dust. Recent works employ a tight coupling of LiDAR and IMU constraints that jointly minimizes LiDAR and IMU observation errors on a unified objective function to achieve accurate and robust state estimation in challenging situations [1]–[4].

In environments where LiDAR data are degraded, LiDAR-IMU odometry systems need to rely more heavily on IMU-based estimation. Therefore, IMU-based motion prediction is a critical factor for maintaining motion estimation performance. However, typical IMU-based motion prediction using pure integration of linear acceleration and angular velocity measurements accumulates errors rapidly. In particular, translation estimation requires double integration of linear

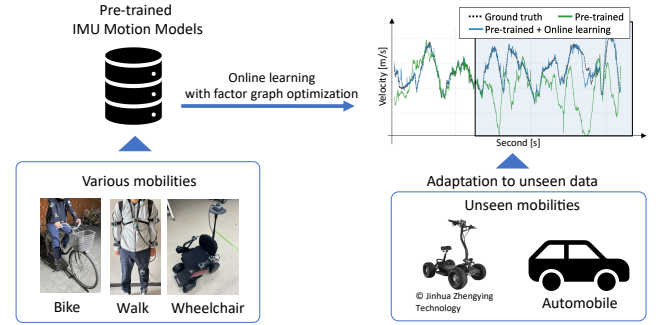


Fig. 1. The concept of proposed method. Our proposed method performs online learning of the IMU motion model along with LiDAR-IMU state estimation, enabling us to generalize to previously unseen motion models that are not included in the training data.

acceleration, which leads to significant accuracy degradation over a short period. In the context of PDR (Pedestrian Dead Reckoning), pattern-matching-based approaches (e.g., step detection, step length estimation [5], [6]) have been proposed to suppress integration errors, and learning-based approaches that implicitly learn walking motion patterns from large-scale training data have shown promising results [7], [8]. These methods estimate the number of steps and the step length from the wave patterns appearing in the IMU data within a temporal window and enables predicting velocity or displacement without integration. It provides more robust motion prediction than pure integration when the motion pattern is known a priori.

Learning-based IMU odometry has been applied to a wide range of motion models beyond PDR, including legged robot and vehicle [9], [10]. Motion models for each mobility exhibit unique motion patterns, such as vibrations caused by bumps that occur sequentially from the front wheels to the rear wheels. However, these models often show degraded performance when applied to different mobility types or IMU models [11].

One of the effective solutions to this problem is online learning. This technique enables high generalization performance because it can adapt to the target on the fly. However, tight coupling of LiDAR-IMU state estimation and online learning poses challenges in terms of real-time performance. In contrast to the conventional online adaptation of low-dimensional IMU bias with factors related to IMU integration, recent learning-based IMU models involve a large number of parameters, and it is difficult to perform tightly coupled online learning of a neural network in real

This work was supported by the Japan Science and Technology Agency (JST) K Program, Grant Number [JPMJKP23G2].

¹Department of Information Science and Engineering, Ritsumeikan University 2-150, Iwakura-cho, Ibaraki-shi, Osaka, 567-0816, Japan
kazushi@ubi.ise.ritsumei.ac.jp

²Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology Tsukuba Central 2, 1-1-1, Umezono, Tsukuba, Ibaraki, 305-8568, Japan k.koide@aist.go.jp

time.

In this paper, we propose a LiDAR-IMU odometry estimation method that tightly couples online learning of IMU motion model and LiDAR-based state estimation on a sliding window factor graph. The method employs an IMU motion model based on ResNet [12] that estimates sensor displacements with uncertainty estimation. This model enables robust motion prediction and can be incorporated into the factor graph with the probabilistic uncertainty modeling. In addition, the proposed method performs optimization of sensor states (pose and velocity) and adaptation of the IMU motion model simultaneously on a unified objective function. To achieve real-time performance, we propose efficient and fast online adaptation by updating only the bias terms of the IMU motion model, inspired by BitFit [13]. Experimental results show that our method exhibits high robustness in challenging environments involving varying motion patterns, IMU sensor models, and sampling frequencies. The contributions of this work are as follows:

- We propose a novel LiDAR-IMU odometry method based on a tight coupling of LiDAR-IMU state estimation and online learning of the IMU motion model. The proposed architecture allows us to use the rich geometric information in LiDAR data as supervision to train the IMU motion model and overcome challenging situations where LiDAR data degrades.
- We provide an odometry estimation framework that incorporates online learning of the IMU motion model in real time by using BitFit [13] based fine-tuning for efficient and rapid adaptation. The proposed method achieves a processing time of less than 0.1 seconds (10 Hz), including all components of the pipeline such as factor graph construction, point cloud preprocessing, and optimization.
- Through experiments, we show that the proposed method can adapt to previously unseen motion models, and IMU devices with different specifications, and sampling rates without requiring prior calibration such as IMU constant bias. The method can be used easily without any adjustment or tuning.

II. RELATED WORK

A. IMU-Based Odometry Estimation

IMU-based odometry estimation has been widely studied in the field of PDR. Traditionally, some studies focused on the motion patterns of walking and estimate step counts and step length to avoid double integration [16]. In recent years, these count and step length estimation is learned automatically by using deep learning. These learning models estimate the sensor motion using only IMU data within a temporal window, without requiring prior velocity information. It means odometry estimation is based on pattern matching instead of integration that avoids accumulation of velocity errors [4], [8]. RONIN [8] is a pioneering work on learning-based PDR that provides robust motion estimation model trained on a large dataset collected using smartphones,

and TLIO [4] proposes a tightly coupled Extended Kalman Filter framework with an IMU device. These methods rely on motion pattern matching and often suffer from difference in motion patterns.

In contrast, in order to achieve high generalization, some studies estimate IMU bias evolutions and reduce the integration errors. Buchanan et al. [11] incorporated IMU bias that is estimated by LSTM or transformers into factor graph optimization. Zhou et al. [17] estimated IMU bias using a diffusion model. Methods based on IMU bias estimation enables improving the accuracy of IMU data integration with generalization capability. Nevertheless, these works only target a specific IMU model and changes in hardware configuration can significantly degrade estimation performance. In addition, if a wrong bias is estimated due to misconfiguration of sensors or unseen errors, the velocity estimation diverges through integration.

B. Odometry Estimation with Online Learning

Online learning is an effective way to solve the issue of generalization. Tartan IMU [10] proposes IMU foundation models based on ResNet [12] that was trained on a large-scale IMU dataset covering a wide range of mobility types, and adapt this model online using SLAM supervision. While such foundation models show good performance and generalization, they require a high computation load and are difficult to run on a onboard computer. In addition, online learning and the trajectory estimation are decoupled: online adaptation is performed separately from the trajectory optimization. Okawara et.al [18] propose a novel online learning method. This method implemented training in factor graphs and create tightly coupled learning-based IMU models and other constraints. It can perform odometry estimation and training simultaneously which allows for learning that considers uncertainty. In this paper, we use ResNet-based models to achieve versatility in motion models and tightly coupled online training method proposed by Okawara for robustness and online execution.

C. Parameter Efficient Fine-Tuning

PEFT (Parameter Efficient Fine-Tuning) is often used in large models such as LLMs. Large models require enormous computational resources for fine-tuning. To avoid this issue, many studies focus on balancing parameter reduction and learning efficiency. There is LoRA [19] that is one of the most popular fine-tuning methods. LoRA updates weight matrices through adding multi low rank matrix products. BitFit [13] updates only the bias parameters of the model and reduces the number of parameters for fine-tuning significantly. In the proposed method, we utilize BitFit during online learning to enable a real-time system.

III. METHODOLOGY

Fig. 2 shows an overview of the proposed method. The overall system design follows [18] and further extends its scope of application. In addition, Fig. 3 illustrates the structure of the factor graph used for optimization. The

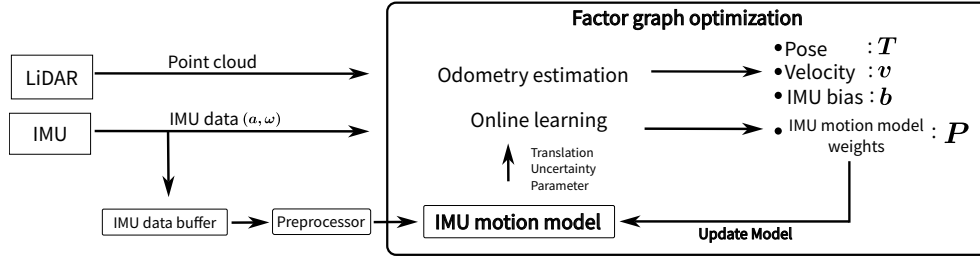


Fig. 2. Overview of the proposed method. This method is based on tight coupling of LiDAR-IMU state estimation and online learning of IMU motion model on a sliding window factor graph. The model is trained online through optimization and provide a robust motion prediction. Proposed method supplies supervision to other factors in the rich geometrical environment.

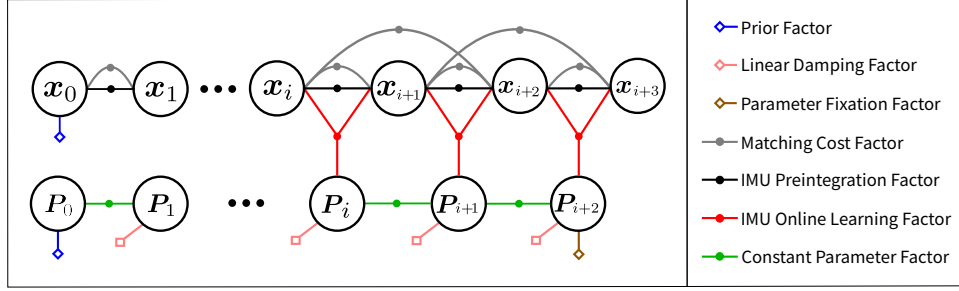


Fig. 3. Proposed Factor Graph. The factor graph consists of three major components. The first component is the matching cost factor [14]. It produces low-uncertainty constraints and supplies supervision to other factors in the rich geometrical environment. The second component is the IMU preintegration factor [15] that performs motion prediction for rotational dynamics in our proposed method. The third component is the IMU online learning factor, enabling online learning.

proposed method optimizes sensor trajectory and network bias in IMU motion model simultaneously with factor graph optimization. There are three types of factors. Matching cost factor serves as a point-cloud-based constraints and provide strong supervision for online learning in the rich geometrical environment. IMU preintegration factor is based on IMU integration and introduced primarily to predict rotation. The third factor related to online learning provides robust motion prediction in the LiDAR degraded environments.

A. Problem Statement

The proposed method takes as input LiDAR point cloud $\mathcal{P}_t$ and IMU data (linear acceleration $\mathbf{a}_t$ and angular velocity $\boldsymbol{\omega}_t$) at time t . The state $\mathbf{X}_t$ is defined as a concatenation of sensor state $\mathbf{x}_t$ and parameters of the IMU motion model $\mathbf{P}_t$:

$$\mathbf{X}_t = [\mathbf{x}_t, \mathbf{P}_t], \quad (1)$$

$$\mathbf{x}_t = [\mathbf{T}_t, \mathbf{v}_t, \mathbf{b}_t], \quad (2)$$

where $\mathbf{T}_t = [\mathbf{R}_t | \mathbf{t}_t] \in SE(3)$ is the sensor pose at time of t , $\mathbf{v}_t \in \mathbb{R}^3$ is the sensor translational velocity and $\mathbf{b}_t \in \mathbb{R}^6$ is IMU bias.

B. Building blocks: IMU and point cloud error factors

IMU preintegration factor: To efficiently incorporate IMU observation constraints into the factor graph, we employ the conventional preintegration technique [15]. Given linear acceleration $\mathbf{a}_t$ and angular velocity $\boldsymbol{\omega}_t$, the sensor state $\mathbf{x}_t$

evolves over a time interval Δt as follows:

$$\mathbf{R}_{t+\Delta t} = \mathbf{R}_t \exp((\boldsymbol{\omega}_t - \mathbf{b}_t^\omega - \boldsymbol{\eta}_t^\omega) \Delta t), \quad (3)$$

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \mathbf{g} \Delta t + \mathbf{R}_t (\mathbf{a}_t - \mathbf{b}_t^a - \boldsymbol{\eta}_t^a) \Delta t, \quad (4)$$

$$\mathbf{t}_{t+\Delta t} = \mathbf{t}_t + \mathbf{v}_t \Delta t + \frac{1}{2} \mathbf{g} \Delta t^2 + \frac{1}{2} \mathbf{R}_t (\mathbf{a}_t - \mathbf{b}_t^a - \boldsymbol{\eta}_t^a) \Delta t^2, \quad (5)$$

Where $\mathbf{g}$ is the gravity and $\boldsymbol{\eta}_t^a, \boldsymbol{\eta}_t^\omega$ represents noise on $\mathbf{a}_t$ and $\boldsymbol{\omega}_t$. By integrating Eqs 3 to 5 between timesteps of x_i and x_j , we obtain relative state increments, $\Delta \mathbf{R}_{ij}$, $\Delta \mathbf{t}_{ij}$, $\Delta \mathbf{v}_{ij}$, and compute the IMU motion error as follows:

$$\begin{aligned} e^{\text{IMU}}(\mathbf{x}_i, \mathbf{x}_j) = & \|\log(\Delta \mathbf{R}_{ij}^\top \mathbf{R}_i^\top \mathbf{R}_j)\|^2 \\ & + \|\Delta \mathbf{t}_{ij} - \mathbf{R}_i^\top (\mathbf{t}_j - \mathbf{t}_i - \mathbf{v} \Delta t_{ij} - \frac{1}{2} \mathbf{g} \Delta t_{ij}^2)\|^2 \\ & + \|\Delta \mathbf{v}_{ij} - \mathbf{R}_i^\top (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \Delta t_{ij})\|^2 \end{aligned}$$

Matching cost factor: We use Voxelized GICP [14] as a geometric constraint between point clouds. This factor takes two point clouds $\mathcal{P}_i, \mathcal{P}_j$ and their poses $\mathbf{T}_i, \mathbf{T}_j$ and computes the geometric consistency error between these point clouds. This constraint error function is defined as follows:

$$\begin{aligned} e^M(\mathcal{P}_i, \mathcal{P}_j, \mathbf{T}_i, \mathbf{T}_j) = & \sum_{\mathbf{p}_k \in \mathcal{P}_i} e^{D^{2D}(\mathbf{p}_k, \mathbf{T}_i^{-1} \mathbf{T}_j)}, \\ e^{D^{2D}}(\mathbf{p}_k, \mathbf{T}_{ij}) = & \mathbf{d}_k^\top (\mathbf{C}_k' + \mathbf{T}_{ij} \mathbf{C}_k \mathbf{T}_{ij}^\top)^{-1} \mathbf{d}_k, \end{aligned} \quad (6)$$

Each point of $\mathbf{p}_k \in \mathcal{P}_i$ is modeled as a normal distribution $\mathbf{p}_k = (\mathbf{u}_k, \mathbf{C}_k)$ representing the local surface shape around $\mathbf{p}_k$. Similar $\mathbf{p}_k' = (\mathbf{u}_k', \mathbf{C}_k')$ is the distribution of points in the corresponding voxel in $\mathcal{P}_j$ and $\mathbf{d}_k = (\mathbf{u}_k' - \mathbf{T}_{t-1,t} \mathbf{u}_k)$ is the residual between $\mathbf{u}_k$ and $\mathbf{u}_k'$.

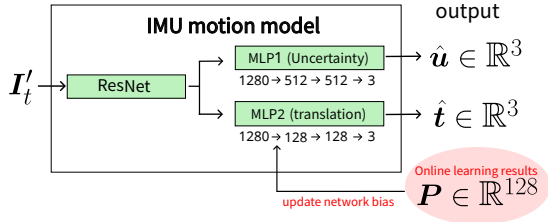


Fig. 4. IMU motion model architecture: The IMU motion model is built with a ResNet backbone and two MLP output heads, which predict translation and uncertainty. This model can be easily incorporated into the factor graph because of probabilistic uncertainty modeling.

C. Learning-based IMU Motion Model

Preprocess: We employ linear interpolation and normalization as preprocessing to handle varying IMU sampling rates and to provide uniform model inputs that stabilize inference. For the IMU data obtained for a window, linear interpolation is applied to each of the six axes ($a_x, a_y, a_z, \omega_x, \omega_y, \omega_z$) to handle missing data and IMU sampling-rate jitters. Let the result of linear interpolation at step j be $\hat{i}_j = [a_x(j), a_y(j), a_z(j), \omega_x(j), \omega_y(j), \omega_z(j)]$. Then, we denote by I_j the set of $\hat{i}_j$ within the window. In this paper, we set the delta of interpolation is 0.01s, resulting in a window size of 2.95s. These values were determined empirically. Then, standardization is defined as follows: $I'_j = s(I_j)$. s is a function that standardizes the input such that the mean is 0 and standard deviation is 1 for each channel.

While some studies report using gravity-aligned coordinate systems makes the separation of gravity and motion easier [7], [8], we observed that the separate gravity alignment results in a degraded performance when the gravity direction estimation becomes incorrect. It requires careful handling of gravity direction estimation and is difficult to use in practical situations. We thus intentionally use raw IMU data without gravity alignment.

Offline Pre-training: We follow T-LIO's model architecture and training procedure [7]. The architecture of the learning model is shown in Fig. 4. IMU learning model uses a ResNet18 and MLP1 (covariance head) and MLP2 (translation head). The input to the model between steps i and j is preprocessed data I'_j , and the outputs are translation $\hat{t}_{ij}$ and uncertainty $\hat{u}_{ij}$. The ground-truth data t_{gt} is calculated by using the pose T'_i, T'_j obtained by a reference trajectory (Eq. 7) and the loss function is defined as L2 norm (Eq. 8). The model infers the relative translation over a time interval of 0.1s.

$$t_{gt} = T_i'^{-1} T_j' \quad (7)$$

$$\mathcal{L}_1(t_{gt}, \hat{t}_{ij}) = \frac{1}{n} \sum_{i=1}^n \|t_{gt} - \hat{t}_{ij}\|_2 \quad (8)$$

In addition, the uncertainty is modeled as a diagonal covariance matrix and defined as Eq.9. The loss function for learning uncertainty is defined as negative log-likelihood and shown in Eq.10. (We refer readers to [7] for a detailed

description of the loss function).

$$\hat{\Sigma}_{ij}(\hat{u}_{ij}) = \text{diag}(e^{2\hat{u}_{x_{ij}}}, e^{2\hat{u}_{y_{ij}}}, e^{2\hat{u}_{z_{ij}}}) \quad (9)$$

$$\mathcal{L}_2(t_{gt}, \hat{\Sigma}_{ij}, \hat{t}_{ij}) = \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{2} \log \det(\hat{\Sigma}_{ij}) + \frac{1}{2} \|t_{gt} - \hat{t}_{ij}\|_{\hat{\Sigma}_{ij}}^2 \right) + \text{Cst} \quad (10)$$

D. Online Learning

IMU Online Learning Factor: This factor provides constraints about the sensor displacement and the bias parameters of the IMU motion model. To achieve real-time online learning, it is necessary to limit the number of parameters being updated. The proposed method employs two key strategies.

The first is reducing the model size. The IMU motion model should output not only translation but also rotational elements ideally. However, it would be necessary to increase the model size in that case and it may fail to meet real-time constraints. Therefore, we impose model constraints on the translation component and balance between reducing the number of parameters and maintaining system robustness. We formulated a factor for translation, because translation estimation undergoes double integration and accumulates errors more rapidly than rotation, whereas pure integration of angular velocities provides accurate rotation estimation.

The second strategy is updating only the bias terms using BitFit [13]. Fine-tuning techniques such as LoRA adapt neural network weights by introducing trainable low-rank matrices while keeping the original weights frozen. However, these methods still introduce a large number of parameters even for a small MLP. For example, even two rank matrices for a 128-unit MLP require $128 \times 2 \times 2 = 512$ parameters, which is much larger than state estimation parameters (e.g., pose, velocity, and IMU bias). It makes the system difficult to operate within strict real-time constraints. Therefore, we employ the technique introduced in BitFit that only adapts the bias terms of MLPs. In our model, online learning is performed by optimizing the bias parameters of the first layer of the translation MLP2 (128 units).

The residual is defined as Eq.11. Where MLP_1 (Eq.13) is a function that infers translation and MLP_2 (Eq.14) is a function that infers uncertainty. The MLP_1 and MLP_2 share the same feature encoding with ResNet. We can assume that the LiDAR constraints provide accurate pose estimation in environments with rich geometric features. Therefore, pose variables T can be considered as supervision for online learning and P is updated online through factor graph optimization.

$$e^{m_p}(T_i, T_j, P_j) = r_{ij}^{m_p \top} \hat{\Sigma}_{ij}(\hat{u}_{ij})^{-1} r_{ij}^{m_p}, \quad (11)$$

$$r_{ij}^{m_p} = t_{ij} - \hat{t}_{ij}, \quad (12)$$

$$\hat{t}_{ij} = \text{MLP}_1(P_j, I'_j), \quad (13)$$

$$\hat{u}_{ij} = \text{MLP}_2(I'_j), \quad (14)$$

Constant Parameter Factor: This factor constraints the IMU motion model parameters $\mathbf{P}$ between consecutive steps to be similar and change gradually. The residual is defined as.

$$e^{\text{cp}}(\mathbf{P}_i, \mathbf{P}_j) = \mathbf{r}_{ij}^{\text{cp}\top} \mathbf{Q}_{ij}^{\text{cp}-1} \mathbf{r}_{ij}^{\text{cp}} \quad (15)$$

$$\mathbf{r}_{ij}^{\text{cp}} = \mathbf{P}_j - \mathbf{P}_i \quad (16)$$

where $\mathbf{Q}^{\text{cp}}$ is a covariance matrix with diagonal elements set to 4.0×10^{-4} and other elements set to 0. This factor can suppress overfitting by enforcing smoothness between adjacent parameters.

Parameter Fixation Factor: This factor is used to fix online learning parameters in LiDAR degeneration scenes. If this factor is not used and the sensor encounters environment where LiDAR odometry accuracy degrades, the online learning may corrupt, and thus we avoid the corruption by fixing the motion model parameters. When the minimum eigenvalue of the information matrix of the latest point cloud matching cost factor falls below 10^{-4} , we use fixation strategy. This threshold was determined empirically, and it may need to be re-tuned when using a different LiDAR model. $\mathbf{P}_f$ is a value before LiDAR odometry degrades. The residual is defined as follows.

$$e^f(\mathbf{P}_j) = \begin{cases} \mathbf{r}_j^f \mathbf{Q}_j^{f-1} \mathbf{r}_j^f, & \text{if LiDAR odometry degrades,} \\ 0, & \text{otherwise,} \end{cases} \quad (17)$$

$$\mathbf{r}_j^f = \mathbf{P}_j - \mathbf{P}_f. \quad (18)$$

where $\mathbf{Q}^f$ is a covariance matrix with diagonal elements set to 1.0×10^{-8} and other elements set to 0.

E. Factor Graph Optimization

Finally, according to all error residuals, the state of sensor is optimized with Fixed-lag-smoother (window size 2.5s) and iSAM2 optimizer [20]. Prior factor $e^{\text{Prior}}(\mathbf{x}_0)$, $e^{\text{Prior}}(\mathbf{P}_0)$ provide initial values for each variable. $\mathbf{P}_0$ use the parameters that are trained offline.

$$\begin{aligned} e^{\text{all}} = & e^{\text{Prior}}(\mathbf{x}_0) + e^{\text{Prior}}(\mathbf{P}_0) \\ & + \sum_{i=1} e^M(\mathcal{P}_{i-1}, \mathcal{P}_i, \mathbf{T}_{i-1}, \mathbf{T}_i) \\ & + \sum_{i=1} e^{\text{IMU}}(\mathbf{x}_{i-1}, \mathbf{x}_i) \\ & + \sum_{i=1} e^{\text{mP}}(\mathbf{T}_{i-1}, \mathbf{T}_i, \mathbf{P}_i) \\ & + \sum_{i=1} e^{\text{cp}}(\mathbf{P}_{i-1}, \mathbf{P}_i) \\ & + \sum_{i=1} e^f(\mathbf{P}_i) \end{aligned} \quad (19)$$

F. Data Collection:

IMU models must achieve sufficient generalization capability to adapt to previously unseen motion patterns during online learning. Though achieving such versatility requires diverse training data, collecting such data is time-consuming. To address this problem, we use the multi sensor devices



Fig. 5. Data collection. We use multiple sensors, which enables the efficient collection of data representing diverse motion patterns.

simultaneously and reduce the time it takes for data collection. The data collection process is shown in Fig. 5. We used Livox-Mid360 which is equipped with both LiDAR and IMU sensors and collect various motion patterns, including walking (6 devices $\times$ about 0.5 hours), electric-wheelchair motion (4 devices $\times$ about 2.0 hours), and bicycling (4 devices $\times$ about 1.3 hours). The sampling rates were set to 10 Hz for the LiDAR and 200 Hz for the IMU. The point cloud data are fed to 3D LiDAR SLAM (GLIM [3]) to generate the reference sensor trajectories data. Data collection was conducted in environments with sufficient geometric features to ensure highly accurate trajectory estimation using SLAM. Since the proposed data collection method enables self-labeling and eliminates the need for annotated supervision, it offers scalability for a large-scale deployment.

IV. EXPERIMENTS

We use GeForce RTX 4080 GPU and Ryzen 9 9950X CPU are used in all experiments.

A. Comparison methods:

We use four types of comparison methods to show the effectiveness of the proposed method in all experiments.

- LiDAR-IMU: Factor graph optimization with IMU preintegration and matching cost factor as an ablation study. This is equivalent to removing factors related to the learning model from the proposed method.
- Ours w/o online learning: Same proposed method without online learning as an ablation study. This approach can be considered as integrating learning-based motion pattern odometry models into a factor graph framework.
- FAST-LIO2 [2]: State-of-the-art LiDAR-IMU odometry algorithm, which utilizes an iterated Kalman filtering for the tight fusion of LiDAR and IMU observations.
- DLIO [21]: State-of-the-art LiDAR-IMU odometry based on direct point cloud registration with IMU constraints.

B. Evaluation of robustness to LiDAR dropouts

Experimental settings: We evaluated the robustness of odometry estimation algorithms on the NTU sequences collected from an ATV (All-Terrain Vehicle) in the MCD dataset [22]. To simulate a challenging situation where point cloud data degrade, we reduced the frame rate of point clouds by regularly dropping LiDAR frames (3.3, 2, 1, 0.5 Hz). This is

a very challenging setup that requires odometry estimation algorithms to deal with occasional data interruptions. During the first 33 seconds, which are used for system initialization, all methods receive point cloud at 10 Hz. After initialization, the frame rate of point clouds is reduced. The IMU motion model is trained only on in-house Livox-Mid360 data (Walk, electric wheelchair, and Bike) and tested on the OS1 LiDAR and IMU (100 Hz) sensor data, with the test data collected using an ATV. This setup represents a challenging evaluation scenario involving unseen mobility and variations in calibration settings, IMU devices, and sampling rates.

Evaluation metric: We use three evaluation metrics: absolute trajectory error (ATE), relative trajectory error (RTE) and a metric based on RTE to evaluate the system resilience. We name this metric Within-0.5m RTE Rate (W0.5-RTE), which measures the ratio of trajectory sections over the full sequence that have RTE ($m/\Delta m$) within 0.5 m. When W0.5-RTE is 1.0, it means that the estimated trajectory has RTEs smaller than 0.5 m at all points along the trajectory. When the error rapidly accumulates, the system becomes unable to recover. Once this happens, the subsequent estimation remains corrupted, leading to a smaller value of this metric. A larger value of this metric indicates that the optimization remained stable for a longer duration, and thus serves as a measure of resilience.

Experimental result: The results for each sequence of the MCD dataset are shown in Table I, and the corresponding results are visualized in Fig. 6. Even when the LiDAR rate decreases, the proposed method demonstrates high robustness. Comparison methods that rely on pure IMU integration velocity errors grow significantly and recovery becomes impossible, once the estimation fails. In contrast, the proposed method does not rely on IMU integration solely, preventing error accumulation and avoiding estimation failures. One possible reason for the lower accuracy of the proposed method at higher LiDAR rates (e.g., ntu day 01 at 3.3 Hz) is that it does not incorporate a keyframe-based registration strategy. The estimated velocities in ntu.day.10 are shown in Fig. 7. It can be seen that conventional LiDAR-IMU methods (LiDAR-IMU, FAST-LIO2) become corrupted once the velocity estimation error becomes large is made in the estimation. On the other hand, which methods with learning-based IMU motion models (ours) show high robustness. Ours without online learning the methods tend to underestimate the velocity. This is because the motion model is pre-trained on a dataset containing only low-speed motion data (walking, bike, and electric chairs), and it failed to generalize to the on-vehicle high-speed sequences. This results demonstrate the capability of the proposed method to adapt to unseen mobility types and different LiDAR and IMU models.

C. Evaluation of real-time performance

Real-time performance is evaluated based on the time required for each optimization step. To satisfy the real-time constraint at a LiDAR rate of 10 Hz, the optimization time per iteration must be smaller than 0.1 s. The optimization time is summarized in the Table II. The optimization time

is below 0.1 s in almost all intervals, confirming that the proposed method is capable of running in real-time.

D. Qualitative evaluation of online learning

We tested the proposed method on a bicycle using a model pre-trained on walking and electric-wheelchair data to evaluate online learning qualitatively. In this evaluation, the value of Q^{cp} is set to 2.5×10^{-5} . Fig. 8 illustrates the time-varying difference between the LiDAR odometry translation and the model output translation. The decreasing error over time indicates that online learning is proceeding appropriately. The temporal increases in error are caused by exceptional operations such as initialization or exit operation (the procedures required to start or terminate data collection).

E. Visualizing IMU motion model parameters transition

Fig. 9 illustrates the transition of the online-learned parameters, plotted using one-dimensional PCA on a sequence including (walk, bike, and electric wheelchair). This indicates that the system adapts to each motion model through online learning.

V. CONCLUSIONS

We proposed a LiDAR-IMU odometry estimation framework. We fine-tuned an IMU learning model online using factor graphs and PEFT. In contrast to integration-based methods, the IMU model based on motion patterns avoids the accumulation of drift error. The results show that our approach maintains strong LiDAR-IMU odometry performance in complex environments, even with low-rate point clouds and IMUs with completely different sampling rates and product types.

REFERENCES

- [1] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.
- [2] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.
- [3] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "GLIM: 3D range-inertial localization and mapping with GPU-accelerated scan matching factors," *Robotics and Autonomous Systems*, vol. 179, p. 104750, 2024.
- [4] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 3144–3150.
- [5] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, 2013, pp. 225–234.
- [6] I. Klein and O. Asraf, "StepNet—Deep learning approaches for step length estimation," *Ieee Access*, vol. 8, pp. 85 706–85 713, 2020.
- [7] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, "Tlio: Tightly learned inertial odometry," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5653–5660, 2020.
- [8] S. Herath, H. Yan, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 3146–3152.
- [9] R. Buchanan, M. Camurri, F. Dellaert, and M. Fallon, "Learning inertial odometry for dynamic legged robot state estimation," in *Conference on robot learning*. PMLR, 2022, pp. 1575–1584.
- [10] S. Zhao, S. Zhou, R. Blanchard, Y. Qiu, W. Wang, and S. Scherer, "Tartan imu: A light foundation model for inertial positioning in robotics," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 22 520–22 529.

TABLE I
ODOMETRY ESTIMATION RESULTS ON MCD-DATASET.

Method	sequence	W0.5-RTE $\uparrow$ / ATE $\downarrow$ / RTE (m / Δm) $\downarrow$			
		3.3 Hz	2 Hz	1 Hz	0.5 Hz
Ours	ntu_day_01	1.0 / 27.7 / 0.059	1.0 / 12.8 / 0.057	1.0 / 18.6 / 0.052	0.87 / 38.7 / 0.110
LiDAR-IMU		1.0 / 25.6 / 0.059	1.0 / 19.0 / 0.058	0.79 / - / -	0.02 / - / -
Ours w/o online learning		1.0 / 26.8 / 0.063	1.0 / 15.3 / 0.059	0.98 / 40.4 / 0.093	0.56 / - / -
FAST-LIO2		1.0 / 1.1 / 0.124	0.97 / 16.3 / 0.211	0.19 / - / -	0.24 / - / -
DLIO		0.59 / 51.5 / 0.773	N/A	N/A	N/A
Ours	ntu_day_02	1.0 / 2.8 / 0.059	1.0 / 2.2 / 0.058	1.0 / 2.0 / 0.049	1.0 / 2.4 / 0.070
LiDAR-IMU		1.0 / 2.6 / 0.055	1.0 / 2.4 / 0.057	1.0 / 1.83 / 0.048	0.03 / - / -
ours w/o online learning		1.0 / 3.4 / 0.068	1.0 / 3.5 / 0.063	1.0 / 2.0 / 0.051	0.61 / 30.8 / 0.601
FAST-LIO2		1.0 / 0.3 / 0.096	1.0 / 0.2 / 0.113	0.94 / 0.47 / 0.157	0.36 / - / -
DLIO		0.96 / 0.9 / 0.211	N/A	N/A	N/A
Ours	ntu_day_10	1.0 / 12.8 / 0.055	1.0 / 2.6 / 0.049	1.0 / 6.2 / 0.047	0.73 / 20.9 / 0.123
LiDAR-IMU		1.0 / 5.7 / 0.054	1.0 / 16.4 / 0.053	0.04 / - / -	0.04 / - / -
Ours w/o online learning		1.0 / 24.7 / 0.062	1.0 / 12.0 / 0.053	0.93 / 19.8 / 0.148	0.38 / - / -
FAST-LIO2		1.0 / 3.5 / 0.101	0.99 / 3.1 / 0.151	0.45 / - / -	0.30 / - / -
DLIO		0.55 / 3.5 / 0.395	N/A	N/A	N/A
Ours	ntu_night_04	1.0 / 11.0 / 0.052	1.0 / 8.8 / 0.051	1.0 / 14.2 / 0.052	0.80 / 36.0 / 0.240
LiDAR-IMU		1.0 / 16.5 / 0.051	1.0 / 11.1 / 0.051	1.0 / 8.0 / 0.052	0.05 / - / -
Ours w/o online learning		1.0 / 8.8 / 0.058	1.0 / 4.4 / 0.055	0.96 / 14.2 / 0.087	0.43 / 57.4 / 0.888
FAST-LIO2		0.83 / 1.6 / 0.267	0.76 / 2.9 / 0.374	0.23 / - / -	0.31 / - / -
DLIO		0.78 / 1.5 / 0.340	N/A	N/A	N/A
Ours	ntu_night_08	1.0 / 3.6 / 0.056	1.0 / 10.2 / 0.051	1.0 / 6.1 / 0.045	0.71 / 52.9 / 0.308
LiDAR-IMU		0.90 / 9.7 / 0.056	1.0 / 10.9 / 0.050	1.0 / 6.0 / 0.046	0.04 / - / -
Ours w/o online learning		0.90 / 13.6 / 0.061	1.0 / 6.9 / 0.054	0.96 / 32.9 / 0.101	0.52 / 94.1 / 0.737
FAST-LIO2		1.0 / 3.0 / 0.116	0.98 / 3.2 / 0.167	0.75 / 23.4 / 1.139	0.36 / - / -
DLIO		0.70 / 6.2 / 0.499	N/A	N/A	N/A
Ours	ntu_night_13	1.0 / 16.0 / 0.079	1.0 / 10.7 / 0.070	1.0 / 12.4 / 0.081	0.68 / 54.7 / 0.409
LiDAR-IMU		1.0 / 3.8 / 0.079	1.0 / 5.4 / 0.068	0.83 / 16.5 / 0.191	0.10 / - / -
Ours w/o online learning		0.95 / 9.1 / 0.088	1.0 / 18.4 / 0.087	0.95 / 29.7 / 0.138	0.57 / 57.36 / 0.861
FAST-LIO2		1.0 / 0.8 / 0.144	0.97 / 1.8 / 0.185	0.37 / - / -	0.53 / - / -
DLIO		0.67 / 4.1 / 0.457	N/A	N/A	N/A

Results with an ATE exceeding 100 m are denoted by “-”, indicating estimation corruption.

N/A indicates cases where the accumulated estimation error became excessively large, making numerical computation of the evaluation metric infeasible.

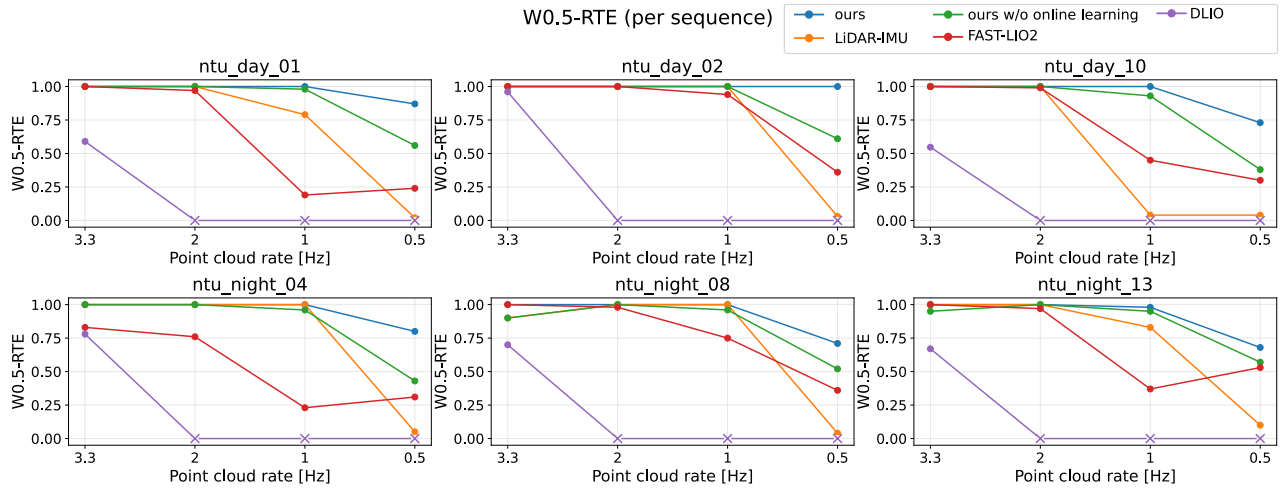


Fig. 6. W0.5-RTE result. The proposed method maintains high resilience even when the point cloud frame rate decreases.

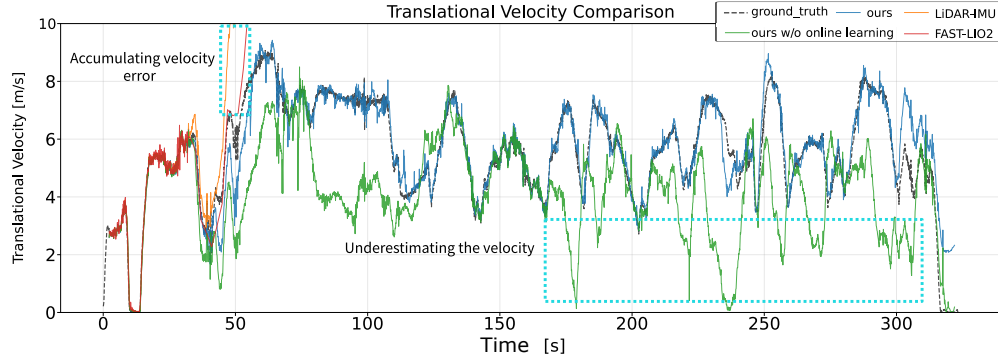


Fig. 7. Estimated velocities in ntu_day_10. Accumulating velocity error: In the integration-based method LiDAR-IMU and FAST-LIO2 errors to accumulate over time and can eventually lead to estimation failure. In contrast, the learning-based methods ours and ours w/o online learning suppress the accumulation of errors and achieve more stable velocity estimation. Underestimating the velocity: Ours w/o online learning tends to underestimate velocities because the model is pre-trained on relatively slow mobility data such as walking, electric wheelchairs, and bicycles. On the other hand, the proposed method (ours) performs online learning, allowing the model to adapt to the unseen motion model of the environment and achieve accurate velocity estimation.

TABLE II
PROCESSING TIME RESULT.

Sequence	The average processing time[s] per sequence			
	3.3 [Hz]	2 [Hz]	1 [Hz]	0.5 [Hz]
ntu_day_01	0.0451	0.0488	0.0512	0.0539
ntu_day_02	0.0475	0.0501	0.0517	0.0510
ntu_day_10	0.0468	0.0505	0.0512	0.0521
ntu_night_04	0.0430	0.0446	0.0472	0.0514
ntu_night_08	0.0468	0.0495	0.0509	0.0528
ntu_night_13	0.0447	0.0495	0.0494	0.0500

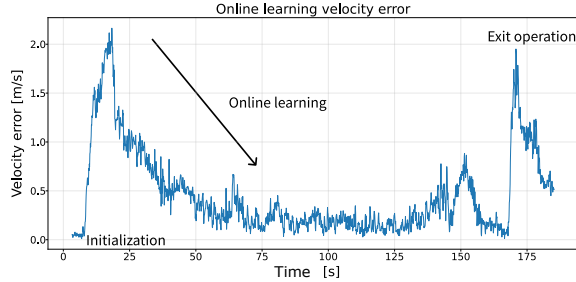


Fig. 8. Velocity errors for the online learned IMU motion model. The vertical axis plots the difference between the LiDAR odometry translation and the model output translation, and the horizontal axis represents the time frame. We demonstrate online learning for bike using a model pre-trained on electric wheelchairs and walking. Over time, the error decreases as the system adapts to the unknown motion model (bike), through online learning.

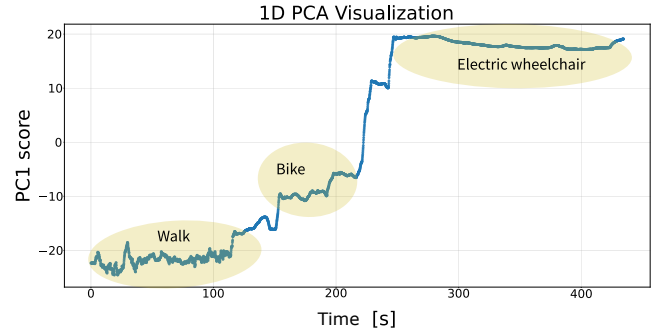


Fig. 9. Transition of IMU motion model parameters (1D PCA). We visualize the parameters learned online during transitions from walking to bicycling and then to an electric wheelchair. It is shown that the parameters for each motion model exhibit similar distributional characteristics.

- [11] R. Buchanan, V. Agrawal, M. Camurri, F. Dellaert, and M. Fallon, "Deep imu bias inference for robust visual-inertial odometry with factor graphs," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 41–48, 2022.
- [12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [13] E. B. Zaken, Y. Goldberg, and S. Ravfogel, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2022, pp. 1–9.
- [14] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Globally consistent 3D LiDAR mapping with GPU-accelerated GICP matching cost factors," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8591–8598, 2021.
- [15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preinte-
- gration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Robotics: Science and Systems XI*, 2015.
- [16] X. Hou and J. Bergmann, "Pedestrian dead reckoning with wearable sensors: A systematic review," *IEEE Sensors Journal*, vol. 21, no. 1, pp. 143–152, 2020.
- [17] S. Zhou, S. Katragadda, and G. Huang, "Learning imu bias with diffusion model," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 2162–2168.
- [18] T. Okawara, K. Koide, S. Oishi, M. Yokozuka, A. Banno, K. Uno, and K. Yoshida, "Tightly-coupled LiDAR-IMU-wheel odometry with an online neural kinematic model learning via factor graph optimization," *Robotics and Autonomous Systems*, vol. 187, p. 104929, 2025.
- [19] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models," *Iclr*, vol. 1, no. 2, p. 3, 2022.
- [20] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.
- [21] K. Chen, R. Nemirow, and B. T. Lopez, "Direct lidar-inertial odometry: Lightweight lio with continuous-time motion correction," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3983–3989.
- [22] T.-M. Nguyen, S. Yuan, T. H. Nguyen, P. Yin, H. Cao, L. Xie, M. Wozniak, P. Jensfelt, M. Thiel, J. Ziegenbein *et al.*, "Mcd: Diverse large-scale multi-campus dataset for robot perception," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 22 304–22 313.