2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# CPBA-LIWO: Continuous-time LiDAR-Inertial-Wheel Odometry Based on Probabilistic Bundle Adjustment

Song Wu[1], Yunzhou Zhang[1]*, Yuezhang Lv[1], Wu Li[1], Sizhan Wang[1], Su Yan[1], Hengwang Ding[1]

***Abstract*— LiDAR-based odometry is widely used in ground robot localization. However, current methods encounter challenges in accuracy and robustness due to structural degradation, system observational error, and accumulated error. To address the above issues, we propose CPBA-LIWO, a continuous-time LiDAR-Inertial-Wheel (LIW) odometry based on probabilistic bundle adjustment (PBA) within a sliding window. This method constructs a general wheel model, which is used for the complementary fusion of LiDAR, IMU and wheel data through a continuous-time trajectory using a B-spline curve, thereby improving the robustness of the system in structurally degraded environments. Furthermore, to improve the accuracy of long-distance odometry, we propose a probabilistic model for the voxel plane and implement a sliding-window voxel PBA backend based on this model. The experimental results on the M2DGR-plus and KAIST datasets demonstrate that our method outperforms state-of-the-art LiDAR-based odometry in terms of accuracy and robustness.**

## I. INTRODUCTION

LiDAR-based odometry is commonly used for localization and mapping in GPS-denied environments, such as indoors or urban canyons. However, existing LiDAR-based odometry methods often experience reduced accuracy during aggressive motion and long-distance localization due to degradation of environmental structure, observational error from LiDAR, and accumulated error from incremental map updates. Therefore, developing robust, long-distance odometry suitable for ground robots remains a research-worthy topic.

Current LiDAR-based odometry methods [1]–[7] often fuse IMU data to obtain a better initial pose estimate. However, due to IMU noise and accumulated error, these algorithms may fail in harsh environments. In contrast, wheel encoders [8] can directly provide linear velocity and angular velocity, effectively maintaining odometry scale in scenarios with rapid motion and LiDAR degradation. Moreover, robot motion can cause motion distortion in the LiDAR point cloud, which is typically compensated with the assistance of IMU [1]–[5], but the high-frequency sampling nature of LiDAR [9]–[13] is not well utilized. Therefore, this paper proposes a continuous-time odometry approach, modeling the trajectory using B-spline to eliminate point cloud motion distortion and then efficiently coupling IMU, wheel, and raw LiDAR data with a general wheel model for ground robots.

*The corresponding author of this paper
[1]Song Wu, Yunzhou Zhang, Yuezhang Lv, Wu Li, Sizhan Wang, Su Yan, Hengwang Ding are with College of Information Science and Engineering, Northeastern University, Shenyang 110819, China zhangyunzhou@mail.neu.edu.cn

This work was funded by National Natural Science Foundation of China (No. 61973066) and Major Science and Technology Projects of Liaoning Province(No. 2021JH1/10400049).

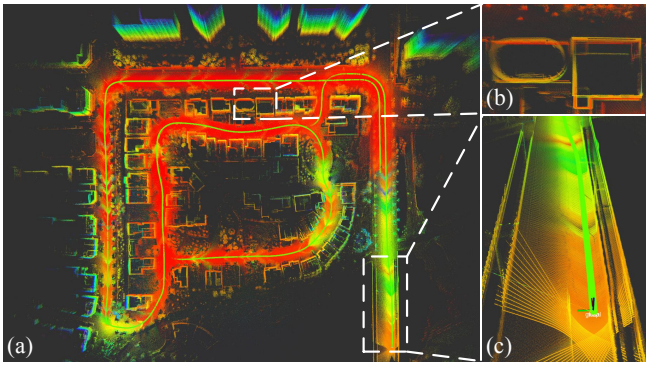


Fig. 1. CPBA-LIWO achieves robust localization and accurate mapping by constructing PBA based on a continuous-time trajectory through LIW fusion. (a) Our global map is constructed using two LiDARs in the urban08 sequence of the KAIST dataset. (b) A close-up of buildings repeatedly scanned over a long distance without loop closure shows the consistency of the mapping. (c) A close-up of a bridge shows that CPBA-LIWO can robustly localize and map in LiDAR-degraded situations.

Compared to visual odometry, which can easily establish data associations between frames through feature matching, the sparsity of point clouds makes this challenging for LiDAR [14]. However, a multi-frame data association backend is crucial for reducing accumulated error in odometry. Some approaches [14]–[16] improve the consistency of odometry by constructing a voxel-based bundle adjustment (BA) backend. However, they do not account for the impact of LiDAR observational error, which arises from the observation angle and the flatness of the object itself. Inspired by CTA-LO [12], which builds a spot model for LiDAR point to enhance registration accuracy further, we propose a probabilistic model of voxel plane based on the spot model and construct a continuous-time probabilistic bundle adjustment (PBA) backend accordingly.

To this end, we propose CPBA-LIWO, a long-distance and robust continuous-time LiDAR-IMU-Wheel odometry capable of providing robust localization and accurate mapping (Fig. 1). The main contributions of this paper are as follows:

- We propose a continuous-time LiDAR-Inertial-Wheel odometry framework, which effectively ensures the robustness of the odometry in harsh environments by integrating a general wheel constraint.
- We propose a probabilistic model for the voxel plane and implement a continuous-time sliding window PBA based on this model, further improving the odometry accuracy during long-distance operation.
- Extensive experiments on the M2DGR-plus and KAIST datasets verify the efficacy of our method against state-of-the-art LiDAR-based odometry.

## II. RELATED WORK

In this section, we review the works on LiDAR-based odometry and bundle adjustment with probabilistic model.

### A. LiDAR-based Odometry

Discrete-time LiDAR-based odometry has achieved significant success in robot localization and mapping. Methods like LOAM [17] and LeGO-LOAM [18] realize point cloud registration by constructing a constant velocity motion model. In contrast, FAST-LIO2 [3], LiLi-OM [1], iG-LIO [6], SR-LIO [19] and Semi-Elastic-LIO [20] tightly integrate IMU pre-integration into state estimation, which is more robust compared to the constant velocity model. LIO-Vehicle [21], designed for autonomous driving, builds on LIO-SAM [5] by constructing a bicycle model to acquire vehicle data. However, discrete-time odometry methods introduce unnecessary errors by linearly interpolating sensor data like IMU.

Instead, some researchers use continuous-time LiDAR-based odometry to recover the motion trajectory, thereby eliminating motion distortion and further improving the accuracy of LiDAR registration. CT-ICP [22] considers that the LiDAR frame is continuous and recovers the trajectory through linear interpolation. LIWO [8] builds a sliding window optimization framework that integrates IMU and wheel data based on CT-ICP. However, the motion trajectory fitted by linear interpolation is not smooth enough and does not align with the actual motion patterns of robots. Therefore, some methods use B-spline to achieve trajectory parameterization that aligns more with actual motion. Among them, CTA-LO [12] constructs a B-spline-based LiDAR odometry, and CLINS [13] fuses LiDAR-IMU data in the B-spline trajectory, but they fail under LiDAR degradation. Based on LIO-Vehicle [21], He *et al*. [23] build a B-spline-based LiDAR-Inertial-Vehicle odometry suitable for vehicles, but it is only applicable to vehicle platforms.

### B. Bundle Adjustment With Probabilistic Model

On the other hand, LiDAR observations are affected by systematic ranging errors [24]. VoxelMap [25] introduces an adaptive voxel map representation and obtains optimal poses by estimating the uncertainty of points. CTA-LO [12] models the uncertainty of LiDAR points through a beam model to reduce the impact of ranging errors on registration. Meanwhile, SLICT [26] constructs voxel maps based on surfel features, which better represent real-world scenarios. However, the above methods ignore the impact of the incremental error of the map on the global consistency of the system.

At the same time, some methods reduce accumulated error by constructing BA [14], [15]. BALM [14], for example, uses LOAM [17] as a frontend and builds a sliding-window BA in the backend to eliminate accumulated error, but it requires good initial pose alignment from the frontend. HBA [16] reduces the time required to construct a globally consistent map by implementing a hierarchical LiDAR BA, but the time required for global BA increases with the accumulation of data. However, they do not comprehensively consider LiDAR observational error and map increment cumulative error.

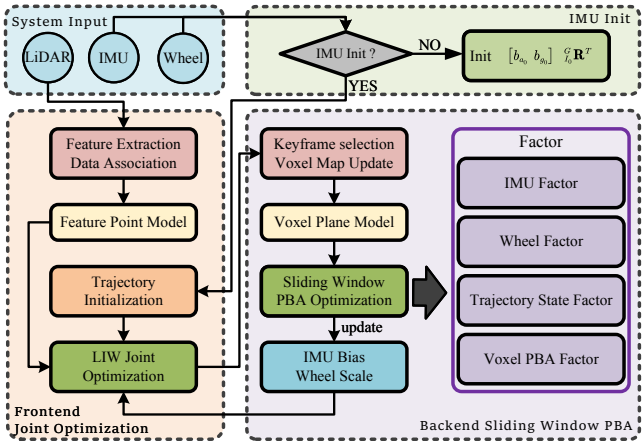


Fig. 2. Overview of our proposed CPBA-LIWO approach. The system first completes IMU initialization with the assistance of wheel, then builds a single-frame continuous-time LIW optimization frontend through the spot model, and finally builds a sliding-window-based continuous-time LIW unified PBA backend through keyframe selection and probabilistic model of voxel plane. At the same time, the IMU bias and wheel scale are updated at the backend and provided to the frontend.

## III. METHODOLOGY

In this section, we describe our LiDAR-Inertial-Wheel odometry in detail. First, we construct a general wheel model and implement wheel-assisted IMU dynamic-static initialization (See III-B). Then, we construct a probabilistic model of the voxel plane based on the spot model of the LiDAR point (See III-C). Finally, with the B-spline trajectory curve, we construct a continuous-time LiDAR-Inertial-Wheel odometry based on PBA (See III-D). Fig. 2 shows an overview of our method.

### A. Preliminary

#### 1) Notations

We denote the IMU, wheel, LiDAR, and global reference frame as $\{I\}$,$\{W\}$,$\{L\}$,$\{G\}$, and use ${}^{A}(\cdot)$ to represent data in frame $\{A\}$. ${}^{A}_{B}\mathbf{T} \in SE(3)$ is used to represent a 6-DOF rigid transformation, and ${}^{A}_{B}\mathbf{T}$ represents the transformation of point ${}^{A}\boldsymbol{p} \in \mathbb{R}^3$ in frame $\{A\}$ to frame $\{B\}$, where ${}^{A}_{B}\mathbf{T} = \begin{bmatrix} {}^{A}_{B}\mathbf{R} & {}^{A}_{B}\mathbf{p} \\ 0 & 1 \end{bmatrix}$ consists of a relative rotation ${}^{A}_{B}\mathbf{R} \in SO(3)$ and a relative translation ${}^{A}_{B}\mathbf{p} \in \mathbb{R}^3$.

#### 2) Continuous-time uniform B-spline trajectory

B-spline curves adjust their shape by modifying control points, and this adjustment only affects local regions. This property makes the B-spline more flexible and convenient to use. We constructed a uniform cubic B-spline trajectory [13] on $SO(3)$ for 3D rotation and on $\mathbb{R}^3$ for translation, which adheres to the laws of motion and better parameterizes the robot's motion trajectory. Specifically, the translation ${}^{G}_{I}\mathbf{p}(t)$ in time $t \in [t_i, t_{i+1})$ is represented by the translation control points $\mathbf{p}_i, \mathbf{p}_{i+1}, \cdots, \mathbf{p}_{i+d-1}$ of the $d$-order B-spline uniformly distributed in time:

$$ {}^{G}_{I}\mathbf{p}(t) = \mathbf{p}_i + \sum_{j=1}^{d-1} \lambda_j(u(t)) \cdot (\mathbf{p}_{i+j} - \mathbf{p}_{i+j-1}) \quad (1) $$

where $u(t) = (t - t_i) / (t_{i+1} - t_i)$, and $\lambda_j(u(t))$ is only related to the order and $j$.

Similarly, for ${}_I^G\mathbf{R}(t)$ under $SO(3)$, it can be represented as:

$${}_I^G\mathbf{R}(t) = \mathbf{R}_i \cdot \prod_{j=1}^{d-1} Exp\left(\lambda_j(u(t)) \cdot Log\left(\mathbf{R}_{i+j-1}^{-1} \cdot \mathbf{R}_{i+j}\right)\right) \quad (2)$$

where $\mathbf{R}_i, \mathbf{R}_{i+1}, \cdots, \mathbf{R}_{i+j-1}$ is the rotation control point on $SO(3)$, $Exp(\cdot)$ is the exponential map of a vector to the Lie group, and its inverse operation is $Log(\cdot)$.

The uniform B-spline curves provide closed analytical derivatives, allowing the system to fuse high-frequency asynchronous IMU, wheel, and continuous LiDAR data. By taking the derivatives of ${}_I^G\mathbf{p}(t)$ and ${}_I^G\mathbf{R}(t)$, the velocity ${}^G\boldsymbol{v}(t)$ and acceleration ${}^G\boldsymbol{a}(t)$ in frame $\{G\}$ and the angular velocity ${}^I\boldsymbol{w}(t)$ in the frame $\{I\}$ can be obtained, which can easily integrate the original data of IMU and wheel.

### B. General Wheel Model

#### 1) Wheel scale model

Based on encoders and other sensors, wheel data (linear velocity and angular velocity) can be obtained with kinematic or dynamic models of different ground-wheeled robots, such as two-wheel differential or Mecanum wheel robots. However, due to internal and external factors such as tire pressure, transmission ratio, and ground flatness, there may be a deviation between the wheel measurement data and the actual motion data. Therefore, for the original wheel measurement data ${}^W\hat{\boldsymbol{u}} = [v_{tx}\ v_{ty}\ w_{tz}]$ obtained at time $t$, We construct a wheel adaptive scale constraint $\boldsymbol{k}_W = \begin{bmatrix} k_{vx} & k_{vy} & k_{wz} \end{bmatrix}$ ($\boldsymbol{k}_{W_0} = [1\ 1\ 1]$) to eliminate the deviation. Therefore, the actual wheel data is given as:

$${}^W\hat{\boldsymbol{v}}_t = [k_{vx}v_{tx} \quad k_{vy}v_{ty} \quad 0]^T \quad (3)$$

$${}^W\hat{\boldsymbol{w}}_t = [0 \quad 0 \quad k_{wz}w_{tz}]^T \quad (4)$$

Among them, the adaptive scale constraint $\boldsymbol{k}_W$ will participate in the global optimization, and the real scale of wheel will be updated through LiDAR observation.

#### 2) Wheel-assisted IMU dynamic-static initialization

The IMU measurement data contains gravity acceleration and IMU internal parameters. It is necessary to initialize the IMU bias $[\boldsymbol{b}_{a_0}\ \boldsymbol{b}_{g_0}]$ and the posture ${}_{I_0}^G\mathbf{R}^T$ of $\{I_0\}$ relative to $\{G\}$ to remove them. The acceleration data of the 6-axis IMU contains its own acceleration and gravity acceleration. When it is stationary, its acceleration is zero, and the initial bias $[\boldsymbol{b}_{a_0}\ \boldsymbol{b}_{g_0}]$ can be obtained through the acceleration data. However, when the robot is moving, it cannot complete its own initialization through the acceleration data of the IMU. Therefore, we achieve the dynamic-static joint initialization of the 6-axis IMU by using continuous-time trajectory fusion of asynchronous wheel data and IMU data.

We achieve joint optimization of $[\boldsymbol{b}_{a_0}\ \boldsymbol{b}_{g_0}]$, ${}_{I_0}^G\mathbf{R}^T$ and trajectory control points through B-spline trajectory and rigid transformation $\begin{bmatrix} {}_W^I\mathbf{R} & {}_W^I\mathbf{p} \end{bmatrix}$ between IMU and wheel. In the initial period $[t_0, t_k)$ of the system, we estimate the following state variables:

$$\boldsymbol{\chi}_{init} = \left\{\boldsymbol{\Phi}_0, [\boldsymbol{b}_{a_0}\ \boldsymbol{b}_{g_0}], {}_{I_0}^G\mathbf{R}^T\right\} \quad (5)$$

where $\boldsymbol{\Phi}_0$ is corresponding trajectory control point, the asynchronously collected IMU data $\left({}^I\hat{\boldsymbol{a}}_{t_I}, {}^I\hat{\boldsymbol{w}}_{t_I}\right)$ and wheel data $\left({}^W\hat{\boldsymbol{v}}_{t_W}, {}^W\hat{\boldsymbol{w}}_{t_W}\right)$, where $t_I, t_W \in [t_0, t_k)$. Therefore, we construct the IMU constraint $\boldsymbol{r}_{I_i}$ and the wheel constraint $\boldsymbol{r}_{W_i}$ as:

$$\boldsymbol{r}_{I_i} = \begin{bmatrix} {}_I^G\mathbf{R}^T(t_I)\left({}^G\boldsymbol{a}(t_I) + {}^G\mathbf{g}\right) - {}^I\hat{\boldsymbol{a}}_{t_I} + \boldsymbol{b}_{a_0} \\ {}^I\boldsymbol{w}(t_I) - {}^I\hat{\boldsymbol{w}}_{t_I} + \boldsymbol{b}_{g_0} \end{bmatrix} \quad (6)$$

$$\boldsymbol{r}_{W_i} = \begin{bmatrix} {}_W^I\mathbf{R}^T \cdot \left({}_I^G\mathbf{R}^T(t_W) \cdot {}^G\boldsymbol{v}(t_W) + \left({}^I\boldsymbol{w}(t_W)\right)^\wedge {}_W^I\mathbf{p}\right) - {}^W\hat{\boldsymbol{v}}_{t_W} \\ {}_W^I\mathbf{R}^T \cdot {}^I\boldsymbol{w}(t_W) - {}^W\hat{\boldsymbol{w}}_{t_W} \end{bmatrix} \quad (7)$$

where $(\cdot)^\vee$ represents the mapping of the antisymmetric matrix to its tangent vector, and the inverse operation is $(\cdot)^\wedge$, ${}^G\mathbf{g} = [0\ 0\ 9.8]^T$ and ${}_I^G\mathbf{R}(t) = {}_{I_0}^G\mathbf{R} \cdot {}_I^{I_0}\mathbf{R}(t)$. Finally, based on the above state variables and constraints, the constructed joint initialization cost function is as follows:

$$arg\min_{\boldsymbol{\chi}_{init}} \left\{\sum \|\boldsymbol{r}_{I_i}\|_{P_{I_i}}^2 + \sum \|\boldsymbol{r}_{W_i}\|_{P_{W_i}}^2\right\} \quad (8)$$

where $\|\cdot\|^2$ is the Mahalanobis distance norm with the weighted covariance $P$. By optimizing the above cost function, we solve the IMU bias $[\boldsymbol{b}_{a_0}\ \boldsymbol{b}_{g_0}]$ and the posture ${}_{I_0}^G\mathbf{R}^T$ to achieve IMU initialization through wheel joint.

### C. LiDAR Observation Model

#### 1) Feature point probabilistic model

Due to the divergence angle of the LiDAR beam, the uncertainty of LiDAR points observed by different observations varies. In order to quantitatively analyze the uncertainty of the points, CTA-LO [12] establishes a spot model of LiDAR point. The error entropy of the feature point obtained by following the spot model of CTA-LO is:

$$S = \frac{\pi d^2}{8\cos\alpha} \quad (9)$$

where $\alpha$ is the angle between the laser ray and the characteristic plane, $d = 2\rho \cdot \tan(\beta/2) + D_0$, $\rho$ is the depth of LiDAR points, $\beta$ is the beam divergence angle, and $D_0$ is the diameter of the emission hole.

Through the error entropy of LiDAR point, we can know that the uncertainty of LiDAR point is affected by the depth of LiDAR point and the incident angle of the LiDAR beam. The probability weight of LiDAR point is constructed as:

$$W_p = \cos\alpha \cdot \frac{\rho_{\max}^2 - \rho^2}{\rho_{\max}^2 - \rho_{\min}^2}, \alpha \in [0, 90], \rho \in [\rho_{\min}, \rho_{\max}] \quad (10)$$

#### 2) Probabilistic Model Of Voxel Plane

Due to the LiDAR observational error and the difference in the flatness and point cloud density of the voxel plane, the constraint strength of different voxel planes in the adaptive voxel map constructed based on keyframes is different. Therefore, we perform probabilistic modeling on the adaptive voxel plane.

Assume that a voxel block contains point cloud data $\{{}^G\boldsymbol{p}_i = (x_i, y_i, z_i), W_{p_i}\}$ in the global frame, where $W_{p_i}$ represents the probability weights of feature points obtained through the spot model. Calculate the probability-weighted centroid $\boldsymbol{c} = (x_c, y_c, z_c)$ and the weighted covariance matrix $\mathbf{C}$ of the point cloud within this voxel:

$$\boldsymbol{c} = \frac{\sum_{i=1}^N W_{p_i}\,{}^G\boldsymbol{p}_i}{\sum_{i=1}^N W_{p_i}}, \boldsymbol{C} = \frac{\sum_{i=1}^N W_{p_i}\left({}^G\boldsymbol{p}_i - \boldsymbol{c}\right)\left({}^G\boldsymbol{p}_i - \boldsymbol{c}\right)^T}{\sum_{i=1}^N W_{p_i}} \quad (11)$$

Perform eigenvalue decomposition on the weighted covariance matrix $\mathbf{C}$:

$$\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T \tag{12}$$

where $\boldsymbol{V}$ is the eigenvector matrix and $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}, (\lambda_1 < \lambda_2 < \lambda_3) \tag{13}$$

The flatness of the point cloud within the voxel can be calculated using the eigenvalues of the voxel:

$$W_a = \frac{\lambda_2 - \lambda_1}{\lambda_3} \tag{14}$$

According to the flatness, we can judge whether the voxel has a plane feature. We only consider the voxel block of $W_a > 0.6$. To reduce the singularity of plane parameterization, the three-degree-of-freedom parameterization of the plane is realized based on the plane closest point representation [27]. Extract the eigenvector corresponding to the smallest eigenvalue $\lambda_1$ and normalize it to obtain the plane normal vector $\mathbf{n}$, then calculate the distance $d$ from the global frame origin to the weighted centroid of the plane. Therefore, the plane of the voxel block can be parameterized as:

$${}^G\mathbf{F} = \mathbf{n}d \tag{15}$$

Given $n$ voxel point cloud clusters in the voxel map that meet the plane feature conditions, for the $k$-th voxel block $\Pi_k$ with volume $V_k$, its normalized voxel density weight can be represented as the ratio of the probability-weighted density of that voxel block to the weighted average density of all voxel blocks that meet the plane feature conditions:

$$W_{d_k} = \frac{\sum_{i\in\Pi_k} W_{p_i}}{V_k} \div \frac{\sum_{k=0}^{n}\sum_{i\in\Pi_k} W_{p_i}}{\sum_{k=0}^{n} V_k} \tag{16}$$

Combining the flatness weight and plane density weight of the plane fitting, we get the weighted plane probability weight of voxel block $k$:

$$W_{q_k} = W_{a_k} \cdot W_{d_k} \tag{17}$$

### *D. Continuous-time LiDAR-IMU-Wheel Odometry*

*1) Frontend joint optimization*

When a new LiDAR frame $t \in [t_k, t_k + \Delta t)$ is received, the B-spline trajectory is extended by adding new control points. Subsequently, a unified optimization of the B-spline trajectory control points is achieved by integrating data from IMU, wheel, and the registration data of continuously scanned LiDAR features. This process enables trajectory extension and joint optimization at the frontend. The newly added control points $\boldsymbol{\Phi}_{new}(t_k, t_k + \Delta t)$ are optimized by minimizing the following cost function:

$$arg\min_{\boldsymbol{\Phi}_{new}} \left\{ \sum \|\boldsymbol{r}_{I_f}\|^2_{P_{I_f}} + \sum \|\boldsymbol{r}_{W_f}\|^2_{P_{W_f}} + \sum \|\boldsymbol{r}_{L_f}\|^2_{P_{L_f}} \right\} \tag{18}$$

where $\boldsymbol{r}_{I_f}$, $\boldsymbol{r}_{W_f}$, and $\boldsymbol{r}_{L_f}$ represent the constraint factors constructed by IMU, wheel, and LiDAR at the frontend. $\boldsymbol{r}_{I_f}$ and $\boldsymbol{r}_{L_f}$ are similar to existing work [12], [13]. The definition of $\boldsymbol{r}_{W_f}$ is the same as that of $\boldsymbol{r}_{W_i}$ in (7).

*2) Backend sliding window factor graph optimization*

Based on the factor graph optimization framework, the frontend B-spline trajectory is fused with asynchronous IMU data, wheel data, and continuous LiDAR keyframe voxel features on the backend. Specifically, within a sliding window of keyframes (15 frames in this study) that satisfy constraints from LiDAR keyframes (selected based on time, rotation, and translation constraints), the time series within the sliding window is $[t_i, t_j)$, and the time series of LiDAR keyframe $L_k$ is $[t_i, t_j)$. We estimate the following state variables:

$$\boldsymbol{\chi} = \left\{ \boldsymbol{x}_{\Phi_{ij}}, \boldsymbol{x}_{I_{b_{ij}}}, \boldsymbol{x}_{k_{W_{ij}}}, \boldsymbol{x}_{F_{ij}} \right\} \tag{19}$$

where $\boldsymbol{x}_{\Phi_{ij}} = \boldsymbol{\Phi}(t_i, t_j)$ is the trajectory control points in the sliding window, $\boldsymbol{x}_{I_{b_{ij}}} = \left\{ \left( \boldsymbol{b}_a^k, \boldsymbol{b}_g^k \right), t_k \in [t_i, t_j) \right\}$ is the IMU bias between the LiDAR keyframes, $\boldsymbol{x}_{k_{W_{ij}}} = \left\{ \boldsymbol{k}_W^k, t_k \in [t_i, t_j) \right\}$ is the wheel scale constraint between the LiDAR keyframes, $\boldsymbol{x}_{F_{ij}} = \left\{ {}^G\mathbf{F}_l \right\}$ is the voxel probability plane parameters included in the voxel map.

The sliding window factor graph optimization is constructed by constructing the IMU factor $\boldsymbol{r}_{I_b}$, wheel factor $\boldsymbol{r}_{W_b}$, voxel PBA factor $\boldsymbol{r}_{L_{ba}}$ and trajectory state factor $\boldsymbol{r}_{F_b}$. The details will be given below. Based on the factor graph shown in Fig. 3, The cost function of the backend construction is:

$$\begin{aligned} \boldsymbol{\chi} = arg\min_{\boldsymbol{\chi}} \Big\{ & \sum \|\boldsymbol{r}_{I_b}\|^2_{P_{I_b}} + \sum \|\boldsymbol{r}_{W_b}\|^2_{P_{W_b}} + \\ & \sum \|\boldsymbol{r}_{L_{ba}}\|^2_{P_{L_{ba}}} + \sum \|\boldsymbol{r}_{F_b}\|^2_{P_{F_b}} \Big\} \end{aligned} \tag{20}$$

for the least squares problem above, we iteratively solve it using Levenberg-Marquardt method of Ceres Solver [28].

For the IMU data $\left\{ \left( {}^I\hat{\boldsymbol{a}}_{t_I}, {}^I\hat{\boldsymbol{w}}_{t_I} \right), t_I \in [t_i, t_j) \right\}$ in the sliding window, when the IMU is sampled between two keyframes $t_I \in [t_k - 1, t_k)$, the IMU factor is defined as:

$$\boldsymbol{r}_{I_b} = \begin{bmatrix} {}^G_I\mathbf{R}^T(t_I)\left({}^G\boldsymbol{a}(t_I) + {}^G\mathbf{g}\right) - {}^I\hat{\boldsymbol{a}}_{t_I} + \boldsymbol{b}_a^k \\ {}^I\boldsymbol{w}(t_I) - {}^I\hat{\boldsymbol{w}}_{t_I} + \boldsymbol{b}_g^k \\ \boldsymbol{b}_a^k - \boldsymbol{b}_a^{k-1} \\ \boldsymbol{b}_g^k - \boldsymbol{b}_g^{k-1} \end{bmatrix} \tag{21}$$

The wheel data set collected in the sliding window is $\left\{ \left( {}^W\hat{\boldsymbol{v}}_{t_W}, {}^W\hat{\boldsymbol{w}}_{t_W} \right), t_W \in [t_i, t_j) \right\}$. When $t_W \in [t_k - 1, t_k)$, the wheel factor is defined as:

$$\boldsymbol{r}_{W_b} = \begin{bmatrix} {}^I_W\mathbf{R}^T \cdot {}^I\boldsymbol{v}(t_W) - {}^W\hat{\boldsymbol{v}}_{t_W} \\ {}^I_W\mathbf{R}^T \cdot {}^I\boldsymbol{w}(t_W) - {}^W\hat{\boldsymbol{w}}_{t_W} \\ \boldsymbol{k}_W^k - \boldsymbol{k}_W^{k-1} \end{bmatrix} \tag{22}$$

where ${}^I\boldsymbol{v}(t_W) = {}^G_I\mathbf{R}^T(t_W) \cdot {}^G\boldsymbol{v}(t_W) + \left({}^I\boldsymbol{w}(t_W)\right)^\wedge {}^I_W\mathbf{p}$.

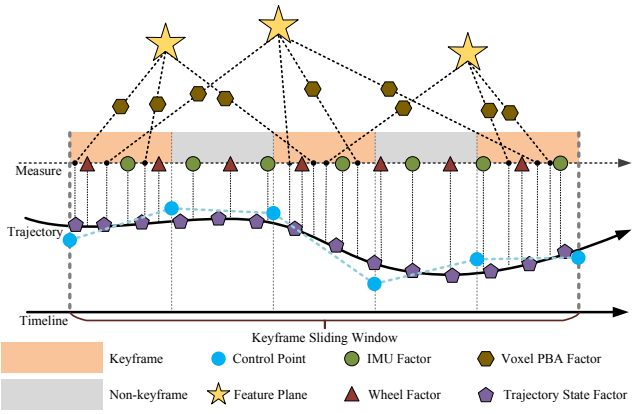


Fig. 3. Factor graph of our backend. In the keyframe-based sliding window, we uniformly optimize the trajectory control points and feature plane parameters in the sliding window through IMU factor, wheel factor, trajectory state factor, and voxel PBA factor.

In the voxel map, the set of voxel blocks that satisfy the plane feature is $\left\{V_l, {}^{G}\mathbf{F}_l, W_{q_l}\right\}$. Each voxel block $V_l$ has a plane probability weight $W_{q_l}$, the plane parameters corresponding to the voxel block is ${}^{G}\mathbf{F}_l$, and the set of feature points contained within the voxel block is $\{{}^{L}\boldsymbol{p}_{t_L} = (x_{t_L}, y_{t_L}, z_{t_L}), W_{p_{t_L}}\}$. Where ${}^{L}\boldsymbol{p}_{t_L}$ is the raw LiDAR point collected at time $t_L$. The PBA factor constructed for each feature point within the voxel is defined as:

$$\boldsymbol{r}_{L_{b\alpha}} = W_{q_l} \cdot W_{p_{t_L}} \left( \frac{{}^{G}\mathbf{F}_l}{\|{}^{G}\mathbf{F}_l\|} {}^{G}_{I}\mathbf{T}\left(t_L\right) {}^{I}_{L}\mathbf{T} \cdot {}^{L}\boldsymbol{p}_{t_L} + \|{}^{G}\mathbf{F}_l\| \right) \tag{23}$$

In the keyframe-based sliding window optimization of the backend, the trajectory of the non-keyframe has no LiDAR observation constraints. To enhance the robustness of the trajectory between non-keyframe, we construct the trajectory state factor. The B-spline trajectory initialized by LiDAR-Inertial-Wheel fusion in the frontend has stable local accuracy. By performing high-frequency ($\Delta t_F = 0.02s$) sampling of the frontend trajectory, we can obtain trajectory state constraints $\left\{ \left( {}^{G}_{I}\mathbf{R}_{t_F}, {}^{G}_{I}\mathbf{p}_{t_F}, {}^{G}\boldsymbol{v}_{t_F}, {}^{G}\boldsymbol{a}_{t_F}, {}^{I}\boldsymbol{w}_{t_F} \right), t_F = t_i + k \cdot \Delta t_F \right\}$, where $t_F \in [t_i, t_j)$ is the trajectory sampling time. Therefore, the trajectory state factor is defined as:

$$\boldsymbol{r}_{F_b} = \begin{bmatrix} \left( {}^{G}_{I}\mathbf{R}^{T}\left(t_{F-1}\right) {}^{G}_{I}\mathbf{R}\left(t_F\right) \right) \left( {}^{G}_{I}\mathbf{R}^{T}_{t_{F-1}} {}^{G}_{I}\mathbf{R}_{t_F} \right)^{T} \\ {}^{G}_{I}\mathbf{p}\left(t_F\right) - {}^{G}_{I}\mathbf{p}_{t_F} \\ {}^{G}_{I}\mathbf{R}^{T}\left(t_F\right) \cdot {}^{G}\boldsymbol{v}\left(t_F\right) - {}^{G}_{I}\mathbf{R}^{T}_{t_F} \cdot {}^{G}\boldsymbol{v}_{t_F} \\ {}^{G}_{I}\mathbf{R}^{T}\left(t_F\right) \cdot {}^{G}\boldsymbol{a}\left(t_F\right) - {}^{G}_{I}\mathbf{R}^{T}_{t_F} \cdot {}^{G}\boldsymbol{a}_{t_F} \\ {}^{I}\boldsymbol{w}\left(t_F\right) - {}^{I}\boldsymbol{w}_{t_F} \end{bmatrix} \tag{24}$$

## IV. EXPERIMENT

In this section, we evaluate our proposed CPBA-LIWO on the public datasets M2DGR-plus [29] [30] and KAIST [31]. As LIWO [8] is the only open-source among the latest LiDAR-Inertial-Wheel odometry, we compare our approach with LIWO, and with other recent LiDAR-Inertial odometry methods, including the continuous-time based method CLINS [13], the filtering based method FAST-LIO2 [3], and the sliding window optimization based method LiLi-OM [1]. Additionally, we conduct ablation experiments to test the impact of the sliding window voxel PBA and continuous-time wheel factor on the odometry performance.

To ensure a fair comparison of odometry performance, we modify all the involved algorithms to better adapt to the dataset and turn off the loop closure functionality in all the algorithms above. We use absolute trajectory error (ATE) [32] for odometry accuracy comparison and perform a rigid alignment between the estimated trajectory and the groundtruth before computing ATE. For some KAIST experimental metrics, we use the data from the original papers. All other experiments are conducted on a consumer-grade computer with an Intel Core i7-13700KF and 32 GB RAM.

### A. Datasets

*1) Mecanum wheel robot scenario:* The M2DGR-plus dataset is a campus dataset collected by a Mecanum wheel ground robot. It contains a 5Hz Robosense 16-line LiDAR, a 200Hz 6-axis IMU, and 20Hz Mecanum wheel data solved by kinematics. We use nine sequences to evaluate the odometry, which contains a variety of environments, such as street, outdoor parking, bridge, and scene switching.

*2) Long-distance urban driving scenarios:* The KAIST dataset is an urban dataset collected by a car in various long-distance environments. The car has two 10Hz Velodyne VLP-16 LiDARs, 200Hz Xsens MTi-300 IMU, and 100Hz RLS LM13 wheel encoders. The two 3D LiDARs are tilted about 45°. We use the data of two 3D LiDARs simultaneously, and the wheel data is obtained by building a two-wheel differential model for the rear wheel encoders. We use five sequences, which contain a variety of environments, such as residential areas, elevated roads, and long-distance streets.

### B. Odometry Experiments

We compare with the latest LIWO and LIO methods on M2DGR-plus and KAIST. Table I shows that our method achieves the best results on most sequences.

For the M2DGR-plus dataset, LIWO only uses velocity data from wheel. Since the LiDAR frame and wheel frame do not coincide, when the ground robot rotates, only transforming the wheel data to the LiDAR frame through rotation will cause LIWO to fail to locate. In addition, the actual velocity in the LiDAR frame is influenced by both the velocity and angular velocity in the wheel frame. Other LIO systems experience significant degradation or even failures in the *bridge1* and *bridge2* sequences, where LiDAR performance deteriorates, as well as in the *building1* and *building2* sequences with intense motion, and the *switch* sequence involving transitions between indoor and outdoor scenes. In contrast, CPBA-LIWO, by fusing IMU and wheel data through B-spline trajectory, achieves better initial trajectory estimation in environments with intense motion and LiDAR degradation, ensuring the precision of LiDAR matching and enhancing system robustness. Fig. 4 shows the trajectory of our method and other state-of-the-art odometry in the *switch* sequence. Compared with LIO, both our method and LIWO can perform robust localization under scene switches. However, the trajectory of our method is smoother and closer to the groundtruth. We also compared the mapping results of our CPBA-LIWO with existing LIO methods using the *building1* sequence. As shown in Fig. 5, even when the robot undergoes intense motion, our method still produces highly accurate mapping results, while the mapping results of FAST-LIO2 and CLINS exhibit severe ghosting. For the sequence *parking2* with more gentle motion, relatively good results can be achieved by fusing IMU and LiDAR, so CPBA-LIWO shows similar performance to FAST-LIO2.

Due to the installation method of the LiDAR in the KAIST dataset, it performs better at scanning tall buildings. However, in more open environments, the LiDAR can experience severe degradation. For instance, in the *urban08*, *urban13*, *urban14*, and *urban15* sequences, where there are scenes with missing LiDAR features, all LIO systems experience degradation or even failure. LIWO, however, integrates wheel data, ensuring scale information is preserved even in environments where LiDAR degrades. Additionally,

TABLE I

ACCURACY (ATE[M]) COMPARISON ON THE M2DGR-PLUS AND KAIST

| Datasets | Sequence | Distance | CLINS | FAST-LIO2 | LiLi-OM | LIWO | CLIO | CLIWO | CPBA-LIWO |
|---|---|---|---|---|---|---|---|---|---|
| M2DGR-plus | *bridge*1 | 35.10m | 0.510 | 0.814 | × | × | 0.458 | 0.320 | **0.264** |
| | *bridge*2 | 212.9m | 2.655 | 0.458 | 1.486 | × | 1.393 | 0.526 | **0.421** |
| | *building*1 | 46.12m | 0.135 | 0.117 | 0.128 | × | 0.078 | 0.072 | **0.069** |
| | *building*2 | 28.02m | 0.080 | 0.138 | 0.453 | 0.121 | 0.069 | 0.069 | **0.058** |
| | *parking*1 | 27.88m | 0.063 | 0.050 | 0.073 | 0.384 | 0.080 | 0.072 | **0.048** |
| | *parking*2 | 42.61m | 0.035 | **0.034** | 0.075 | 0.046 | 0.057 | 0.050 | **0.034** |
| | *street*1 | 15.00m | 0.202 | 0.211 | 0.186 | × | 0.192 | 0.196 | **0.178** |
| | *street*2 | 36.50m | 0.109 | 0.169 | 0.229 | × | 0.10 | 0.072 | **0.045** |
| | *switch* | 86.18m | 0.314 | 0.788 | × | 0.168 | 0.472 | 0.254 | **0.165** |
| KAIST | *urban*07 | 2.55km | 1.383 | 6.227 | 3.508 | 1.582 | 2.406 | 1.536 | **1.259** |
| | *urban*08 | 1.56km | 3.907 | 11.41 | × | 2.920 | 43.01 | 4.512 | **1.923** |
| | *urban*13 | 2.36km | × | × | × | 3.990 | × | 10.21 | **2.014** |
| | *urban*14 | 8.20km | × | × | × | **46.40** | × | 133.7 | 48.40 |
| | *urban*15 | 5.43km | 63.22 | × | 36.45 | 17.91 | 56.34 | 54.69 | **13.34** |

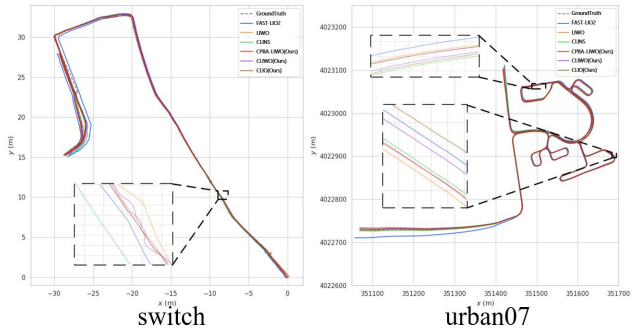


Fig. 4. Comparison of trajectories for scene-switch (*switch*) and long-distance (*urban07*) sequences.

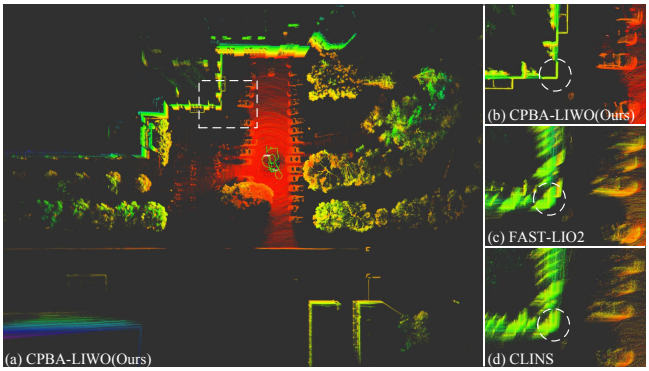


Fig. 5. Mapping result comparison using the *building1* sequence.

our CPBA-LIWO, using a unified sliding window PBA based on continuous-time LIW fusion, further ensures the odometry scale in degraded LiDAR environments. Furthermore, it enhances the robustness of the system during long-distance operation through voxel PBA. The trajectory for the long-distance movement in the *urban07* sequence is shown in Fig. 4.

### C. Ablation Study

To validate the effectiveness of the wheel factor and the LiDAR sliding window PBA in the continuous-time odometry, we conducted an ablation study. For this purpose, we constructed the following three odometry configurations to estimate trajectory ATE: 1) CLIO: Only adds constraints based on raw IMU data and LiDAR point-to-plane residual constraints within the uniform B-spline trajectory. 2) CLIWO: Builds on CLIO by adding the wheel factor. 3) CPBA-LIWO: This is our complete work, using CLIWO as the odometry frontend and constructing a unified LIW sliding window PBA in the backend.

The results are shown in Table I. While CLIO performs well in short-distance, low-intensity motion environments, it exhibits significant errors and even failures in environments with intense motion and LiDAR degradation. However, CLIWO, due to the addition of the wheel factor, ensures that the system can still operate robustly under scene-switch (*switch*) and degraded (*bridge1*) environments. Furthermore, the sliding window-based PBA added in CPBA-LIWO further guarantees the smoothness of the trajectory in long-distance environments and the global consistency of the mapping. The trajectory comparison of these three configurations in the *switch* and *urban07* sequences, as shown in Fig. 4, clearly demonstrates that in scene switching environments (*switch*), the trajectory of CLIO deviates significantly from the groundtruth. In contrast, CLIWO shows noticeable improvement, and the trajectory from CPBA-LIWO is even smoother. Additionally, in the long-distance motion environment (*urban07*), CPBA-LIWO exhibits better global consistency. We also conducte a velocity visualization experiment on CLIO and CPBA-LIWO on sequence *urban15*. As shown in Fig. 6, the velocity of CPBA-LIWO is smoother and more consistent with the kinematic laws than that of CLIO without fusion of wheel constraints.

### D. Evaluation on Computational Efficiency

The time consumption of our system is mainly focused on frontend optimization and backend PBA. In environments with intense motion, multiple iterations are required to achieve better results, and the number of planes in different environments leads to variations in time consumption. Taking 100ms of data per frame as a reference, we conducted experiments on multiple sequences from the M2DGR-plus and KAIST datasets. The average time consumption per frame for frontend optimization is 89ms, while the average time consumption per frame for backend PBA is 93ms.

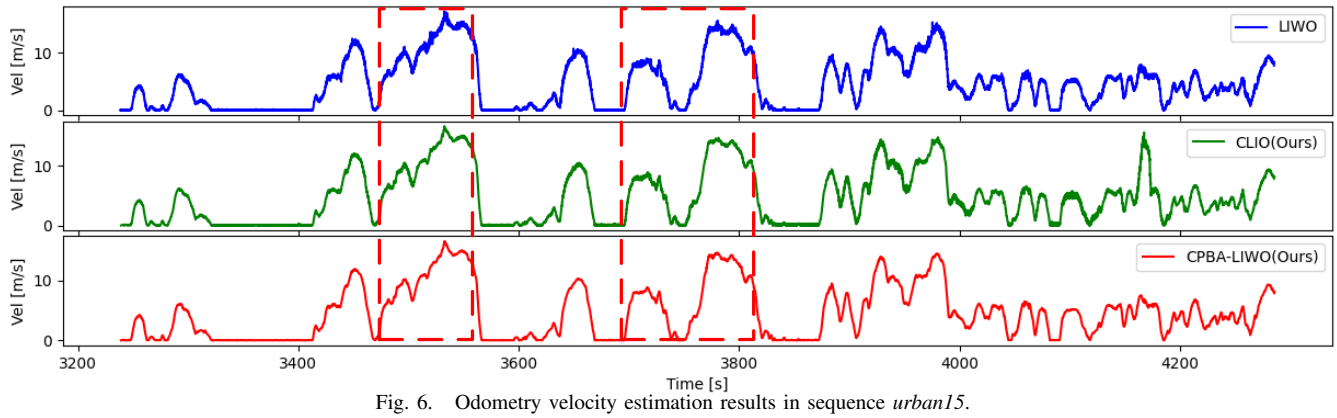

Fig. 6. Odometry velocity estimation results in sequence *urban15*.

### E. Evaluation of Velocity

The cubic B-spline trajectory, with its second-order continuous derivatives, enhances the smoothness of velocity estimation for odometry in wheeled robots. By introducing a general wheel factor to constrain the spline trajectory's velocity, the interference from IMU acceleration integration and pose differentiation on velocity estimation accuracy can be further reduced. In practice, we cannot accurately obtain real-time velocity observations of wheeled robots. However, based on kinematic principles, the velocity of a wheeled robot should be continuous and smooth, without high-frequency oscillations. Therefore, the smoother the velocity curve relative to time, the better it fits the kinematic model [8]. As shown in Fig. 6, compared to the high-frequency oscillating curve in LIWO, the curve in CLIO is much smoother, and the smoothness is further improved in CPBA-LIWO, which incorporates general wheel constraints. This indicates that embedding general velocity constraints into continuous-time B-spline trajectory significantly enhances the accuracy of odometry's velocity estimation.

## V. CONCLUSIONS

This paper proposes a continuous-time LiDAR-Inertial-Wheel odometry (CPBA-LIWO) based on PBA. It employs B-spline curve fitting to model the motion trajectory, tightly coupling IMU and wheel data while eliminating the point cloud motion distortion. A probabilistic voxel plane model is also proposed, along with a sliding window voxel PBA, to ensure global mapping consistency. Compared with several open-source state-of-the-art LiDAR-based odometry, the experiments show our method has higher robustness and accuracy. In terms of system efficiency, the frontend and backend of our method have real-time performance when running separately in a smooth motion environment. It is an exciting idea and future work to further improve our efficiency by properly downsampling the voxel map without compromising accuracy.

## REFERENCES

[1] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-lidar-inertial odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5167–5174, 2021.

[2] 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.

[3] 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.

[4] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4861–4868, 2022.

[5] 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 *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[6] Z. Chen, Y. Xu, S. Yuan, and L. Xie, "ig-lio: An incremental gicp-based tightly-coupled lidar-inertial odometry," *IEEE Robotics and Automation Letters*, 2024.

[7] K. Huang, J. Zhao, Z. Zhu, C. Ye, and T. Feng, "Log-lio: A lidar-inertial odometry with efficient local geometric information estimation," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 459–466, 2023.

[8] Z. Yuan, F. Lang, T. Xu, and X. Yang, "Liwo: Lidar-inertial-wheel odometry," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1481–1488.

[9] D. Droeschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5000–5007.

[10] C. Park, P. Moghadam, J. L. Williams, S. Kim, S. Sridharan, and C. Fookes, "Elasticity meets continuous-time: Map-centric dense 3d lidar slam," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 978–997, 2021.

[11] J. Quenzel and S. Behnke, "Real-time multi-adaptive-resolution-surfel 6d lidar odometry using continuous-time trajectory optimization," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5499–5506.

[12] Y. Lv, Y. Zhang, X. Zhao, W. Li, J. Ning, and Y. Jin, "Cta-lo: Accurate and robust lidar odometry using continuous-time adaptive estimation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 12 034–12 040.

[13] J. Lv, K. Hu, J. Xu, Y. Liu, X. Ma, and X. Zuo, "Clins: Continuous-time trajectory estimation for lidar-inertial system," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6657–6663.

[14] Z. Liu and F. Zhang, "Balm: Bundle adjustment for lidar mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.

[15] Z. Liu, X. Liu, and F. Zhang, "Efficient and consistent bundle adjustment on lidar point clouds," *IEEE Transactions on Robotics*, 2023.

[16] X. Liu, Z. Liu, F. Kong, and F. Zhang, "Large-scale lidar consistent mapping using hierarchical lidar bundle adjustment," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1523–1530, 2023.

[17] J. Zhang, S. Singh *et al.*, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[18] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-

optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.
[19] Z. Yuan, F. Lang, T. Xu, and X. Yang, "Sr-lio: Lidar-inertial odometry with sweep reconstruction," pp. 7862–7869, 2024.
[20] Z. Yuan, F. Lang, T. Xu *et al.*, "Semi-elastic LiDAR-inertial odometry," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 9855–9861.
[21] H. Xiao, Y. Han, J. Zhao, J. Cui, L. Xiong, and Z. Yu, "Lio-vehicle: A tightly-coupled vehicle dynamics extension of lidar inertial odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 446–453, 2021.
[22] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "Ct-icp: Real-time elastic lidar odometry with loop closure," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5580–5586.
[23] B. He, W. Dai, Z. Wan, H. Zhang, and Y. Zhang, "Continuous-time lidar-inertial-vehicle odometry method with lateral acceleration constraint," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3997–4003.
[24] D. D. Lichti and S. Jamtsho, "Angular resolution of terrestrial laser scanners," *The photogrammetric record*, vol. 21, no. 114, pp. 141–160, 2006.
[25] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8518–8525, 2022.
[26] T.-M. Nguyen, D. Duberg, P. Jensfelt, S. Yuan, and L. Xie, "Slict: Multi-input multi-scale surfel-based lidar-inertial continuous-time odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2102–2109, 2023.
[27] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "Lips: Lidar-inertial 3d plane slam," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 123–130.
[28] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 10 2023. [Online]. Available: https://github.com/ceres-solver/ceres-solver
[29] J. Yin, A. Li, T. Li, W. Yu, and D. Zou, "M2dgr: A multi-sensor and multi-scenario slam dataset for ground robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2266–2273, 2021.
[30] J. Yin, A. Li, W. Xi, W. Yu, and D. Zou, "Ground-fusion: A low-cost ground slam system robust to corner cases," *arXiv preprint arXiv:2402.14308*, 2024.
[31] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 642–657, 2019.
[32] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.