Coco-LIC: Continuous-Time Tightly-Coupled LiDAR-Inertial-Camera Odometry Using Non-Uniform B-Spline

Xiaolei Lang, Chao Chen, Kai Tang, Yukai Ma, *Graduate Student Member, IEEE*, Jiajun Lv, Yong Liu, and Xingxing Zuo, *Member, IEEE*

***Abstract*—In this letter, we propose an efficient continuous-time LiDAR-Inertial-Camera Odometry, utilizing non-uniform B-splines to tightly couple measurements from the LiDAR, IMU, and camera. In contrast to uniform B-spline-based continuous-time methods, our non-uniform B-spline approach offers significant advantages in terms of achieving real-time efficiency and high accuracy. This is accomplished by dynamically and adaptively placing control points, taking into account the varying dynamics of the motion. To enable efficient fusion of heterogeneous LiDAR-Inertial-Camera data within a short sliding-window optimization, we assign depth to visual pixels using corresponding map points from a global LiDAR map, and formulate frame-to-map reprojection factors for the associated pixels in the current image frame. This way circumvents the necessity for depth optimization of visual pixels, which typically entails a lengthy sliding window with numerous control points for continuous-time trajectory estimation. We conduct dedicated experiments on real-world datasets to demonstrate the advantage and efficacy of adopting non-uniform continuous-time trajectory representation. Our LiDAR-Inertial-Camera odometry system is also extensively evaluated on both challenging scenarios with sensor degenerations and large-scale scenarios, and has shown comparable or higher accuracy than the state-of-the-art methods.**

***Index Terms*—LiDAR-inertial-camera SLAM, localization, sensor fusion, state estimation.**

I. Introduction

A WIDE range of sensors can be applied for accurate 6-DoF motion estimation, among which LiDAR, IMU, and camera might be the most popular and widely used. Due to the inherent complementarity of such three sensors, LiDAR-Inertial-Camera Odometry (LICO) has achieved higher robustness and accuracy than those which only utilize the component sensors, especially in challenging structure-less and texture-less environments, attracting significant attention [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. However, measurements from these multi-modal sensors are usually received at different rates and different time instants, which causes difficulty in fusing them in a discrete-time odometry system. A common solution is to interpolate the measurements or estimated poses at the same time instants from different sensors for fusion [12], [13].

In contrast, the continuous-time trajectory representation can avoid the need for interpolation and naturally enables pose queries at any given time, thereby facilitating the fusion of high-rate, multi-rate, and asynchronous measurements from various sensors [14]. Most of the existing LICO systems adopt discrete-time trajectory [3], [5], [9], while there are some works adopt continuous trajectory representation [11].

In terms of parameterizing 6-DoF continuous-time trajectories, most existing methods adopt a cubic, cumulative B-spline which offers local control property [11], [14], [15], [16], [17]. In these approaches, control points of B-splines are often distributed uniformly. It has been proved that the spacing of control points greatly influences the accuracy and time performance of trajectory estimation [18]. In the case of uniform B-splines, the spacing of control points is typically predetermined, which determines the complexity of the trajectory. However, in many applications, there is no prior knowledge about the complexity of the trajectory to be estimated. Consequently, due to the inability to dynamically adjust the distribution of control points, uniform B-splines are prone to under- or over-parameterization [15]. An appealing alternative is the use of non-uniform B-splines, which allows for dynamic control point distribution instead of a fixed frequency [15], [19], [20].

In terms of fusion methods in a LIC system, the existing methods can be classified into two categories: loosely-coupled methods, which utilize the visual-inertial system to provide initial values for LiDAR scan matching [1], [2], and tightly-coupled methods, which jointly optimize LiDAR measurements with visual or inertial data [3], [6], [8], [9], [10], [11]. Generally, tightly-coupled methods tend to require higher computation to fuse raw measurements from various sensors but appear more accurate and robust. To formulate effective constraints for the estimated states by the LiDAR and the camera, data associations are required. For the LiDAR, edge and planar features [21] extracted based on curvature are commonly adopted. The extracted

Manuscript received 20 June 2023; accepted 4 September 2023. Date of publication 14 September 2023; date of current version 21 September 2023. This letter was recommended for publication by Associate Editor J. Zhang and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grant NSFC 61836015. *(Corresponding authors: Yong Liu; Xingxing Zuo.)*

Xiaolei Lang, Chao Chen, Kai Tang, Yukai Ma, Jiajun Lv, and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: jerry_locker@zju.edu.cn; chenchao1924@zju.edu.cn; kaitang@zju.edu.cn; yukaima@zju.edu.cn; lvjiajun314@zju.edu.cn; yongliu@iipc.zju.edu.cn).

Xingxing Zuo is with the Department of Computer Engineering, Technical University of Munich, 80333 Munich, Germany (e-mail: xingxing.zuo@tum.de).

The codebase of this letter will also be open-sourced at https://github.com/APRIL-ZJU/Coco-LIC.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3315542, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3315542

features from different scans are associated by finding closet neighbors [6], [21]. For the camera, extracted visual features from images are associated by optical flow tracking [22] or distinctive descriptors such as ORB [23]. Beyond intra-sensor data associations, there are also associations across sensors such as initializing or assigning the depth of visual features or pixels [10], [24]. Our previous work CLIC [11] tightly couple LIC data within the continuous-time trajectory optimization based on uniform B-splines. However, it disregards cross-sensor associations and instead relies on triangulating visual features using a lengthy sliding window of visual keyframes. This approach unavoidably involves numerous control points and adversely affects runtime efficiency of continuous-time framework.

In this work, we develop a continuous-time tightly-coupled LICO system based on the non-uniform B-spline trajectory representation, where control points are dynamically distributed and LIC data are tightly and efficiently coupled. Briefly, the contributions are summarized as follows:

- We propose a LiDAR-Inertial-Camera Odometry system termed as Coco-LIC, utilizing continuous-time trajectory parameterized by non-uniform B-splines. Compared to methods based on uniform B-splines, control points here are dynamically placed through our proposed simple but effective distribution strategy.
- We naturally fuse LIC data without any interpolation. Based on the reconstructed LiDAR point cloud map and optical-flow tracking of visual pixels, we formulate frame-to-map reprojection errors for the current image frame, excluding the depth estimation and optimization for visual pixels. This couples LiDAR and camera data in a tightly-coupled and highly-efficient way.
- We specifically verify the necessity of the non-uniform placement of control points in real-world scenarios, and prove the efficacy of our control point spacing strategy. Furthermore, the entire LICO system is extensively tested on several challenging datasets, demonstrating its real-time performance and high accuracy.

II. Related Work

A. Continuous-Time SLAM

Furgale et al. [25], [26] are among the first to present a full probabilistic derivation of the continuous-time state estimation for solving the SLAM problem based on B-splines. Afterward, they enable the joint estimation of the spatial and temporal extrinsic between different sensors by continuous-time batch optimization, which has been validated on the Visual-Inertial system and stereo LIC system [27], [28]. Lovegrove et al. [14] utilize the continuous-time formulation to incorporate measurements from rolling shutter cameras and IMU. As a well-crafted and comprehensive continuous-time calibration package for the LiDAR-Inertial system, Lv et al. [29] simultaneously estimate the intrinsic and extrinsic while addressing degenerated motions. Different from these offline and batch calibration tasks, some works start to apply the continuous-time representation to online incrementally estimate odometry for the LiDAR-Inertial system [16], [30], multi-camera systems [31], event cameras [32], rolling shutter cameras [17] and so on. With the finding of the efficient derivative computation for cumulative B-splines on Lie groups [33], continuous-time odometry tends to achieve sub-real-time or even real-time performance [11]. A dedicated marginalization strategy in slid-window estimators is also proposed for the continuous-time framework [11], [17].

Continuous-time formulation benefits the fusion of high-rate, multi-rate, and asynchronous sensors. Cioffi et al. [18] further compare the continuous-time methods with the discrete-time methods comprehensively and find the frequency of the uniformly distributed control points in B-splines matters a lot for the trajectory estimation. In real practice, the smoothness of a trajectory can vary significantly, and each segment actually requires a different density of control points to accurately model the real trajectory [34]. An adaptive metric for spacing control points is proposed in [20] by comparing the objective function cost against its expected value. Anderson et al. [34] then adopt this scheme in continuous-time pose-graph optimization and find that sharp variations in robot velocity usually indicate the requirement for more control points. Vandeportaele et al. [19] add more control points where the angular velocity or linear acceleration is larger, and vice versa. Hug et al. [15] recommend employing a more generic and non-uniform representation to prevent the under or over-parameterization. Admittedly, although the non-uniform formulation is critical for continuous-time trajectory estimation, it has only been employed in just a few batch offline applications, lacking implementation and verification in online odometry systems. Our work, to the best of our knowledge, is among the *first* to adopt non-uniform continuous-time representation in the odometry task and demonstrates its efficacy through extensive experiments.

B. LiDAR-Inertial-Camera Fusion for SLAM

As early works of LICO, [1], [2] leverage monocular or stereo VIO to initialize LiDAR scan matching in a loosely-coupled way. Built upon a factor graph and composed of two subsystems, a LIO and a VIO, LVI-SAM [3] shows robustness in various LiDAR-degenerated or visual-failure scenarios. Similarly, characterized by an IMU-centric estimator, Super-Odometry [4] works well with an extra IMU odometry subsystem. Yet, the following tightly-coupled methods maintain only one state vector and show brilliant performance. LIC-fusion [5], [6] and CLIC [11] both tightly fuse LiDAR features, sparse visual features, and IMU measurements within a sliding window, while the former adopts MSCKF [35] filter based framework [36] and the latter is based on continuous-time optimization framework. Wisth et al. [7] develop a unified multi-modal landmark tracking method and fuse IMU measurements with visual and LiDAR landmarks. R2LIVE [8] executes Error-State Iterated Kalman Filter (ESIKF) update whenever a LiDAR scan or a camera image comes and continues to optimize camera poses and visual landmarks in the window. Furthermore, R3LIVE [9] and FAST-LIVO [10] maximally reuse the map points selected from the global LiDAR map to conduct frame-to-map photometric updates in an ESIKF framework, which avoids the triangulation and optimization for visual features over multiple keyframes.

Unlike R3LIVE [9] and FAST-LIVO [10], our method diverges by simultaneously fusing data from three sensors within a fixed time interval, rather than fusing LiDAR-Inertial or Visual-Inertial data separately.

III. Methodology

We first introduce the convention in this letter. We denote the 6-DoF rigid transformation by ${}^{A}_{B}\mathbf{T} \in SE(3) \in \mathbb{R}^{4\times4}$, which

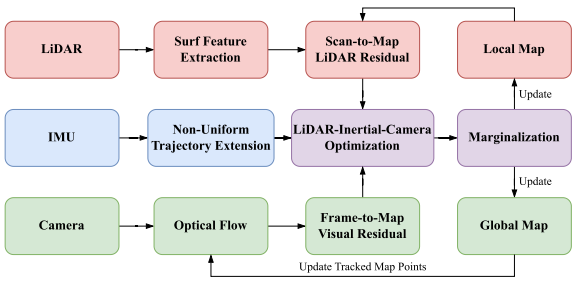

Fig. 1. Pipeline of Coco-LIC.

transforms the point ${}^{B}\mathbf{p}$ in the frame $\{B\}$ into the frame $\{A\}$. ${}^{A}_{B}\mathbf{T} = \begin{bmatrix} {}^{A}_{B}\mathbf{R} & {}^{A}\mathbf{p}_{B} \\ \mathbf{0} & 1 \end{bmatrix}$ consists of rotation ${}^{A}_{B}\mathbf{R} \in SO(3)$ and translation ${}^{A}\mathbf{p}_{B} \in \mathbb{R}^3$. $\mathrm{Exp}(\cdot)$ maps Lie Algebra to Lie Group and $\mathrm{Log}(\cdot)$ is its inverse operation. $(\cdot)_\wedge$ maps a 3D vector to the corresponding skew-symmetric matrix, while $(\cdot)_\vee$ is its inverse operation. The control points in the time interval $[t_a, t_b)$ are denoted as $\mathbf{\Phi}(t_a, t_b)$.

A. System Overview

Fig. 1 shows the pipeline of Coco-LIC. In the beginning, we assume the system is stationary and initialize the IMU bias and gravity by IMU measurements similar to [36]. From now on, the continuous-time trajectory is extended and optimized every Δt (0.1 in this letter) seconds based on the frequency of LiDAR. Suppose we have estimated the trajectory before $t_{\kappa-1}$, we will then estimate the trajectory in $[t_{\kappa-1}, t_\kappa)$ once the LIC data in this time interval is ready, where $t_\kappa = t_{\kappa-1} + \Delta t$. We first dynamically decide the number of control points in the next Δt seconds utilizing the proposed adaptive non-uniform technique (Section III-C). Subsequently, the newly added control points are initialized and further optimized in a manner of factor graph (Section III-D), using LiDAR planar points [21], IMU raw measurements and the latest image in $[t_{\kappa-1}, t_\kappa)$. Finally, after marginalization, we update the local and global LiDAR map for scan-to-map LiDAR association and frame-to-map visual association (Section III-E), respectively.

In summary, we aim to estimate the following states by LIC data over $[t_{\kappa-1}, t_\kappa)$:

$$\begin{aligned} \mathcal{X}^\kappa &= \{\mathbf{\Phi}(t_{\kappa-1}, t_\kappa),\ \mathbf{x}^\kappa_{I_b}\}, \\ \mathbf{x}^\kappa_{I_b} &= \{\mathbf{b}^{\kappa-1}_g,\ \mathbf{b}^{\kappa-1}_a,\ \mathbf{b}^{\kappa}_g,\ \mathbf{b}^{\kappa}_a\}, \end{aligned} \tag{1}$$

which include control points $\mathbf{\Phi}(t_{\kappa-1}, t_\kappa)$ and IMU bias $\mathbf{x}^\kappa_{I_b}$. $\mathbf{b}_g$ and $\mathbf{b}_a$ indicate gyroscope bias and accelerometer bias, respectively. The IMU biases during $[t_{\kappa-1}, t_\kappa)$ are assumed to be $\mathbf{b}^{\kappa-1}_g$ and $\mathbf{b}^{\kappa-1}_a$ for simplicity. They are under Gaussian random walk and evolve to $\mathbf{b}^{\kappa}_g$ and $\mathbf{a}^{\kappa}_g$ at time instant t_κ. We denote all LiDAR planar points in $[t_{\kappa-1}, t_\kappa)$ as $\mathcal{L}_\kappa$, all IMU raw data as $\mathcal{I}_\kappa$, and the latest image frame as $\mathcal{F}_\kappa$.

B. Non-Uniform Continuous-Time Trajectory Formulation

In this letter, we adopt two non-uniform cumulative B-splines to separately parameterize the 3D rotation and the 3D translation. The 6-DoF poses at time $t \in [t_i, t_{i+1})$ of a continuous-time trajectory of degree k are denoted by:

$$\begin{aligned} \mathbf{R}(t) &= \mathbf{R}_{i-k} \cdot \prod_{j=1}^{k} \mathrm{Exp}\left(\lambda_j(t) \cdot \mathrm{Log}\left(\mathbf{R}^{-1}_{i-k+j-1}\mathbf{R}_{i-k+j}\right)\right), \\ \mathbf{p}(t) &= \mathbf{p}_{i-k} + \sum_{j=1}^{k} \lambda_j(t) \cdot \left(\mathbf{p}_{i-k+j} - \mathbf{p}_{i-k+j-1}\right), \\ \boldsymbol{\lambda}(t) &= \widetilde{\mathbf{M}}^{(k+1)}(i)\mathbf{u}, \mathbf{u} = \begin{bmatrix} 1 & u & \cdots & u^k \end{bmatrix}^\top, u = \frac{t - t_i}{t_{i+1} - t_i}, \end{aligned} \tag{2}$$

where $\mathbf{R}_x$ and $\mathbf{p}_x$ denote control points ($x \in \{i-k, \ldots, i\}$) [37]. t_{i-1} and t_i represent any two adjacent knots of B-splines. [38] The knots $\{t_0, t_1, t_2, \ldots\}$ of B-splines are non-uniformly distributed together with the control points, thus the cumulative blending matrix $\widetilde{\mathbf{M}}^{(k+1)}(i)$ is non-constant which is the main difference between uniform and non-uniform B-splines [39]. $\lambda_j(t)$ is an element of vector $\boldsymbol{\lambda}(t)$ with index j. Cubic B-spline is adopted in this letter, which implies $k = 3$, and the blending matrix $(\mathbf{M}^{(4)}(i))^\top$ is derived as follows [33], [39]:

$$\begin{bmatrix} m_{0,0} & 1 - m_{0,0} - m_{0,2} & m_{0,2} & 0 \\ -3m_{0,0} & 3m_{0,0} - m_{1,2} & m_{1,2} & 0 \\ 3m_{0,0} & -3m_{0,0} - m_{2,2} & m_{2,2} & 0 \\ -m_{0,0} & m_{0,0} - m_{3,2} - m_{3,3} & m_{3,2} & m_{3,3} \end{bmatrix}, \tag{3}$$

$m_{a,b}$ denotes the element in row a, column b, computed by knots $t_{i-2} \sim t_{i+3}$ [39]. $\widetilde{\mathbf{M}}^{(4)}(i)$ can be derived accordingly:

$$\widetilde{\mathbf{M}}^{(k+1)}(i)_{m,n} = \sum_{s=m}^{k} \mathbf{M}^{(k+1)}(i)_{s,n}. \tag{4}$$

The continuous-time trajectory of IMU in the global frame $\{G\}$ is denoted as ${}^{G}_{I}\mathbf{T}(t) = [{}^{G}_{I}\mathbf{R}(t), {}^{G}\mathbf{p}_I(t)]$. In this letter, the extrinsics between LiDAR/camera and IMU are pre-calibrated, so we can handily get LiDAR trajectory ${}^{G}_{L}\mathbf{T}(t)$ and camera trajectory ${}^{G}_{C}\mathbf{T}(t)$. B-splines provide closed-form analytical derivatives w.r.t. temporal variables [27], [33], leading to easily computed angular velocity and linear acceleration. The time derivatives of the B-splines can be derived as:

$$\begin{aligned} {}^{G}_{I}\dot{\mathbf{R}}(t) &= {}^{G}_{I}\mathbf{R}(t) \cdot \left({}^{I}\boldsymbol{\omega}(t)\right)_\wedge \\ &= \mathbf{R}_{i-k}\left(\dot{\mathbf{A}}_1\mathbf{A}_2\mathbf{A}_3 + \mathbf{A}_1\dot{\mathbf{A}}_2\mathbf{A}_3 + \mathbf{A}_1\mathbf{A}_2\dot{\mathbf{A}}_3\right) \end{aligned} \tag{5}$$

$${}^{G}\mathbf{a}(t) = {}^{G}\ddot{\mathbf{p}}_I(t) = \sum_{j=1}^{k} \ddot{\lambda}_j(t) \cdot \left(\mathbf{p}_{i-k+j} - \mathbf{p}_{i-k+j-1}\right) \tag{6}$$

where $\dot{\mathbf{A}}_j = \mathrm{Exp}(\dot{\lambda}_j(t) \cdot \mathrm{Log}(\mathbf{R}^{-1}_{i-k+j-1}\mathbf{R}_{i-k+j}))$.

C. Adaptive Non-Uniform Technique

New control points should be added to extend the trajectory, the number of which is set as a constant value in uniform-spline-based methods. However, such uniform placement might fail to precisely model the trajectory segments with violent motions, or introduce redundant control points where the motion is steady, leading to an increase of the computational cost. To address the issue of the uniform B-spline not being able to flexibly accommodate trajectory complexity online, we employ an adaptive non-uniform technique analogous to [19]. Specifically, we

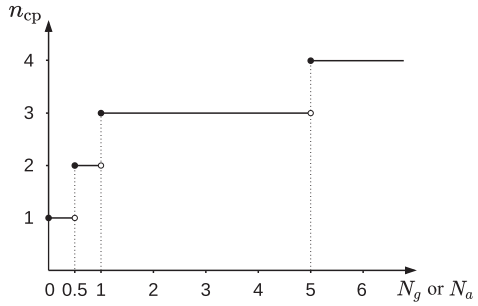

Fig. 2. Number of control points to add depends on IMU. N_g and N_a might correspond to the different values of n_{cp}, while we choose the larger one.

try to get the norm of average angular velocity N_g and linear acceleration N_a in the global frame in $[t_{\kappa-1}, t_\kappa)$:

$$N_g = \frac{1}{n}\left\| \sum_i^n {}^G_I\mathbf{R}_{m_i}\, {}^I\boldsymbol{\omega}_{m_i} \right\|,$$
$$N_a = \frac{1}{n}\left\| \sum_i^n \left({}^G_I\mathbf{R}_{m_i}\, {}^I\mathbf{a}_{m_i} - {}^G\mathbf{g} \right) \right\|, \tag{7}$$

where ${}^I\boldsymbol{\omega}_{m_i}$ and ${}^I\mathbf{a}_{m_i}$ are angular velocity and linear acceleration measured by IMU, respectively. n denotes the number of IMU measurements. The rotation ${}^G_I\mathbf{R}_{m_i}$ is integrated from $t_{\kappa-1}$ using raw IMU measurements. ${}^G\mathbf{g} = \begin{bmatrix}0 & 0 & 9.8\end{bmatrix}^T$ is the gravity in the global frame. Both N_g and N_a reflect the intensity of the motion. We then determine the number n_{cp} of control points to add according to Fig. 2, which is proven to be effective and generic in our experiments. The new control points are all initially assigned with the value of the last control point in the latest optimization. They are uniformly distributed in $[t_{\kappa-1}, t_\kappa)$ with the time interval divided uniformly by $\frac{1}{n_{\text{cp}}}$. Then we can query the pose at any time instant over $[t_{\kappa-1}, t_\kappa)$. Subsequently, the newly added control points are further optimized by solving a factor graph optimization as (13) with only IMU factors and a prior factor specified in Section III-D, where we only optimize $\boldsymbol{\Phi}(t_{\kappa-1}, t_\kappa)$ and keep the other states constant.

D. LiDAR-Inertial-Camera Optimization

1) Scan-to-Map LiDAR Factor: In the continuous-time framework, we estimate the trajectory of a whole scan instead of merely estimating a single pose, which enables simultaneous motion distortion removal and trajectory estimation [16]. Consider a LiDAR planar point ${}^L\mathbf{p}$ in $\mathcal{L}_k$ measured at time t, we can transform it to the global frame by

$${}^G\hat{\mathbf{p}} = {}^G_L\mathbf{R}(t)\,{}^L\mathbf{p} + {}^G\mathbf{p}_L(t). \tag{8}$$

We find for ${}^G\hat{\mathbf{p}}$ five closest neighbor points in the local LiDAR map, and fit a 3D plane using these neighbor points. The local LiDAR map is constructed atop keyscans that are selected by time and space [16]. Thus the point-to-plane error can be defined as:

$$\mathbf{r}_L = {}^G\mathbf{n}_\pi^\top {}^G\hat{\mathbf{p}} + {}^Gd_\pi, \tag{9}$$

where ${}^G\mathbf{n}_\pi$ and ${}^Gd_\pi$ denote the unit normal vector and the distance of the plane to the origin respectively. Jacobians of $\mathbf{r}_L$ w.r.t. control points can be found in [11], the main difference lies in $\boldsymbol{\lambda}(t)$ where the cumulative blending matrix is not constant according to (2).

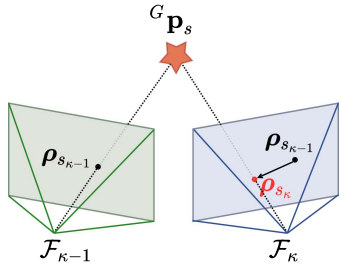

Fig. 3. Projection of a LiDAR map point across image frames. ${}^G\mathbf{p}_s$ in the global LiDAR map is fixed during optimization.

2) Frame-to-Map Visual Factor: Similar to [9], [10], we associate 3D map points in the global LiDAR map (detailed in Section III-E) with the image by projection. Suppose We have associated the map points $\mathcal{P} = \{{}^G\mathbf{p}_1, \ldots, {}^G\mathbf{p}_m\}$ in the last image frame $\mathcal{F}_{\kappa-1}$ with the corresponding pixels $\hat{\varrho}_{\kappa-1} = \{\hat{\boldsymbol{\rho}}_{1_{\kappa-1}}, \ldots, \hat{\boldsymbol{\rho}}_{m_{\kappa-1}}\}$. We then track these pixels $\hat{\varrho}_{\kappa-1}$ into frame $\mathcal{F}_k$ by KLT sparse optical flow [22] and get tracked pixels $\varrho_\kappa = \{\boldsymbol{\rho}_{1_\kappa}, \ldots, \boldsymbol{\rho}_{m_\kappa}\}$. Afterward, RANSAC based on the fundamental matrix and PnP are successively leveraged to remove outlier associations, and we finally get a group of map points $\mathcal{P}$ successfully associated with pixels in the current image frame $\mathcal{F}_k$. Consider a map point ${}^G\mathbf{p}_s$ observed in frame $\mathcal{F}_k$ with the optical-flow tracked pixel $\boldsymbol{\rho}_{s_k} = \begin{bmatrix}u_s & v_s\end{bmatrix}^\top$. As shown in Fig. 3, the reprojection error of the tracked LiDAR point in frame $\mathcal{F}_k$ is defined as:

$$\mathbf{r}_C = \pi_c\left(\frac{{}^C\hat{\mathbf{p}}_s}{\mathbf{e}_3^\top {}^C\hat{\mathbf{p}}_s}\right) - \begin{bmatrix}u_s \\ v_s\end{bmatrix}, \quad {}^C\hat{\mathbf{p}}_s = {}^G_C\mathbf{T}^\top(t)\,{}^G\mathbf{p}_s, \tag{10}$$

where $\mathbf{e}_i$ is a 3×1 vector with its i-th element to be 1 and the others to be 0. $\pi_c(\cdot)$ denotes the projection function which transforms a point on the normalized image plane to a pixel. Note that we also apply Cauchy robust kernel to the reprojection error in optimization to further suppress outliers. In such a fashion based on the optical flow tracking of existing map points with known depth, we can avoid triangulation and sliding window optimization of visual features, thus keeping a short sliding window within $[t_{\kappa-1}, t_\kappa)$ for high efficiency and accuracy, without numerous control points involved.

3) IMU Factor and Bias Factor: We seamlessly use raw IMU measurements to formulate IMU factors like [11], [17], instead of using IMU preintegration. For IMU data in $\mathcal{I}_\kappa$ at time t, we define the following IMU factor:

$$\mathbf{r}_I = \begin{bmatrix} {}^I\boldsymbol{\omega}(t) - {}^I\boldsymbol{\omega}_m + \mathbf{b}_g^{\kappa-1} \\ {}^I\mathbf{a}(t) - {}^I\mathbf{a}_m + \mathbf{b}_a^{\kappa-1} \end{bmatrix}, \tag{11}$$

and bias factor based on the random walk process:

$$\mathbf{r}_{I_b} = \begin{bmatrix} \mathbf{b}_g^\kappa - \mathbf{b}_g^{\kappa-1} \\ \mathbf{b}_a^\kappa - \mathbf{b}_a^{\kappa-1} \end{bmatrix}, \tag{12}$$

where ${}^I\boldsymbol{\omega}_m, {}^I\mathbf{a}_m$ are the raw measurements of angular velocity and linear acceleration at time t, respectively. ${}^I\mathbf{a}(t)$ and ${}^I\boldsymbol{\omega}(t)$ can be computed from the derivatives of the continuous-time trajectory in (5) and (6) by:

$${}^I\mathbf{a}(t) = {}^G_I\mathbf{R}^\top(t)\left({}^G\mathbf{a}(t) - {}^G\mathbf{g}\right),$$
$${}^I\boldsymbol{\omega}(t) = \left({}^G_I\mathbf{R}^\top(t)\cdot {}^G_I\dot{\mathbf{R}}(t)\right)_\vee,$$

where the computation of ${}^I\boldsymbol{\omega}(t)$ has been further sped up using recurrence relation [33] without costly computing ${}^G_I\dot{\mathbf{R}}(t)$.

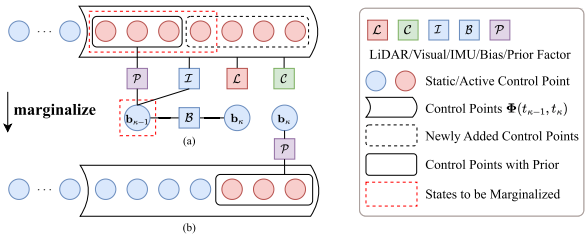

Fig. 4. (a) Factor graph. (b) Marginalization.

4) Optimization and Marginalization: To estimate the aforementioned states $\mathcal{X}^\kappa$, we jointly fuse LIC data in a factor graph as displayed in Fig. 4 and formulate the following nonlinear least-squares problem:

$$\arg\min_{\mathcal{X}^\kappa}\Bigg\{\sum\|\mathbf{r}_L\|^2_{\Sigma_L}+\sum\|\mathbf{r}_C\|^2_{\Sigma_C}+\sum\|\mathbf{r}_I\|^2_{\Sigma_I} +\sum\|\mathbf{r}_{I_b}\|^2_{\Sigma_{I_b}}+\sum\|\mathbf{r}_{\text{prior}}\|^2_{\Sigma_{\text{prior}}}\Bigg\}, \tag{13}$$

which is solved by the Levenberg-Marquardt algorithm in Ceres Solver [40] and sped up by analytical derivatives. $\mathbf{r}_{\text{prior}}$ is the prior factor from marginalization. Σ_L, Σ_C, Σ_I, Σ_{I_b}, Σ_{prior} are the corresponding covariance matrices.

After the optimization is done for the newly added control points and IMU bias, we marginalize states that will not be involved in the next Δt seconds, that is $[t_\kappa, t_{\kappa+1})$, and obtain the prior factor which constrains the states as follows:

$$\mathcal{X}^\kappa_{prior}=\left\{\mathbf{\Phi}(t_{\kappa-1},t_\kappa)\cap\mathbf{\Phi}(t_\kappa,t_{\kappa+1}),\ \mathbf{b}^\kappa_g,\mathbf{b}^\kappa_a\right\},$$

where $\mathbf{\Phi}(t_{\kappa-1},t_\kappa)\cap\mathbf{\Phi}(t_\kappa,t_{\kappa+1})$ represents the control points shared by the trajectory segments in $[t_{\kappa-1},t_\kappa)$ and $[t_\kappa,t_{\kappa+1})$.

E. LiDAR Map for Visual Factor

We maintain a global LiDAR map stored in voxels [9], [10] for the fast query of map points associated with images. The voxel resolution is 0.1 m in our experiments. After the optimization of the trajectory in $[t_{\kappa-1},t_\kappa)$, the global LiDAR map is updated with LiDAR planar points from $\mathcal{L}_\kappa$. For high-quality association, tracked map points in $\mathcal{P}$ with large reprojection error on $\mathcal{F}_\kappa$ are removed. We then project map points in the current FoV onto $\mathcal{F}_\kappa$ and add new successfully associated map points into $\mathcal{P}$. We ensure tracked map points are evenly distributed on images and prioritize retaining map points closer to the camera to alleviate the occlusion problem.

IV. Experiments

In the experiments, we first compare and analyze the trajectory estimation accuracy and the time cost of uniform and non-uniform methods based on LiDAR-Inertial data. Second, we evaluate the proposed Coco-LIC in degraded sequences and compare it with several typical open-source LIO, VIO, and LICO methods. Finally, we conduct experiments on a large-scale dataset to further assess accuracy and time consumption in outdoor environments.

All experiments are executed on a desktop PC with an Intel i7-8700 CPU @ 3.2 GHz and 32 GB RAM. In one dataset, we keep the same parameters of Coco-LIC for all sequences for a fair comparison. We run all experiments six times and take the average values as results. The noise parameters of the IMU are taken from the datasheet. Also, the estimated poses of Coco-LIC used for evaluation by evo [41] are queried from the continuous time trajectory at 100 Hz.

TABLE I
Description of the Sequences

Seq	Duration (second)	Length (m)	Description
Smooth1	78	19	move gently in a square shape
Smooth2	117	17	move gently in a butterfly shape
Smooth3	154	23	move gently in a circle shape
Violent1	74	20	large angular velocity
Violent2	91	115	large linear acceleration
Violent3	121	78	large angular velocity and linear acceleration
Hybrid1	91	37	smooth-violent-smooth
Hybrid2	137	69	smooth-violent-smooth
Hybrid3	104	64	smooth-violent-smooth-violent-smooth

A. Comparison of Uniform and Non-Uniform Placement

We collect LiDAR-Inertial data by a sensor rig comprising a 16-beam 3D LiDAR Velodyne VLP-16[1] at 10 Hz and an Xsens MTi-300 IMU[2] at 400 Hz, with ground-truth data at 120 Hz recorded by a motion capture system. The accuracy of the motion capture is at the millimeter level. Note that cameras experience motion blur under violent motions leading to the poor quality of visual data, while extremely intense movements are contained in this experiment, thus we here only use LiDAR-Inertial data, which is sufficient for non-uniform verification. In real-world applications, motion can be divided into three modes, that is, *smooth*, *violent*, and *hybrid:*

- *smooth* case: ground vehicles moving on smooth roads, and cleaning robots working in office park.
- *violent* case: quadruped robots repeatedly impacting the ground, and aerial robots quickly avoiding obstacles.
- *hybrid* case: humans holding a scanning device to build a map of the environment.

We gather data with a total of nine sequences detailed in Table I, including three motion patterns mentioned above.

Table II summarizes the RMSE of APE and the time consumption for different control point distributions in LIO. Here, *uni-x* means there is x number of control points per Δt seconds using uniform B-splines, while *non-uni* represents dynamically placing control points by the adaptive non-uniform technique specified in Section III-C.

1) Smooth Sequences: uni-1 achieves the highest accuracy, and increasing the number of control points can lead to slightly lower accuracy or even failure to estimate the trajectory (*uni-16*), which suggests that fewer control points are sufficient to accurately model the trajectory with lower complexity for smooth motions, while too many control points might cause over-fitting. Additionally, the time consumption for optimization significantly increases with more control points, probably due to the increase in the dim of states. In contrast, *non-uni* adaptively places control points at a density of 1 per Δt seconds in most cases, obtaining comparable accuracy and computation time to *uni-1*.

2) Violent Sequences: The trajectories exhibit high complexity under large angular velocity and linear acceleration, which

[1][Online]. Available: https://velodyneLiDAR.com/vlp-16.html
[2][Online]. Available: https://www.xsens.com/hubfs/Downloads/Leaflets/MTi-300.pdf

TABLE II
THE RMSE OF APE RESULTS AND THE TIME CONSUMPTION FOR OPTIMIZATION OF LIO WITH DIFFERENT CONTROL POINT DISTRIBUTIONS (UNIT: METERS/MILLISECONDS)

	Smooth1	*Smooth2*	*Smooth3*	*Violent1*	*Violent2*	*Violent3*	*Hybrid1*	*Hybrid2*	*Hybrid3*
uni-1	0.011 / **16.71**	**0.027** / 19.12	0.018 / 16.81	0.355 / **17.32**	fail	fail	fail	fail	fail
uni-2	0.012 / 24.06	0.048 / 28.50	0.018 / 24.35	0.060 / 24.84	0.079 / **22.92**	0.140 / **23.59**	0.164 / 23.93	0.059 / 24.70	0.038 / 25.68
uni-3	0.012 / 29.76	0.050 / 35.07	0.018 / 29.48	0.054 / 30.60	0.052 / 28.64	0.112 / 29.95	0.105 / 25.09	0.059 / 30.75	0.035 / 32.49
uni-4	0.012 / 31.48	0.048 / 38.83	0.018 / 31.05	0.052 / 34.05	0.051 / 31.54	0.102 / 33.45	0.101 / 32.05	0.060 / 34.54	0.035 / 36.79
uni-5	0.012 / 32.00	0.051 / 40.29	0.019 / 31.08	0.054 / 35.33	0.051 / 35.37	0.104 / 34.98	0.107 / 32.71	0.060 / 36.39	0.036 / 38.93
uni-8	0.013 / 34.07	0.052 / 42.94	0.018 / 32.87	0.052 / 37.28	0.060 / 37.14	0.118 / 36.91	0.113 / 34.84	fail	0.037 / 41.91
uni-16	0.049 / 38.80	fail	fail	fail	fail	fail	fail	fail	fail
non-uni	**0.011** / 16.84	0.028 / **19.06**	**0.018** / **16.78**	**0.038** / 22.56	**0.051** / 29.14	**0.100** / 27.45	**0.100** / **20.27**	**0.059** / **24.12**	**0.035** / **24.54**

The best results are marked in bold. Here, *uni-x* means x number of control points per Δt seconds, while *non-uni* means dynamically placing control points by the adaptive technique specified in Section III-C. *fail* means large RMSE over 1 m.

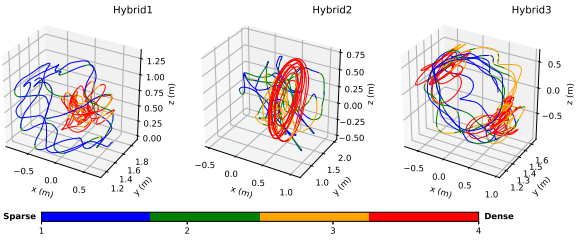

Fig. 5. Trajectories estimated by LIO with non-uniform distribution of the control points on hybrid sequences. The different colors of the trajectories correspond to different densities of control points. From blue to red, control points change from sparse to dense.

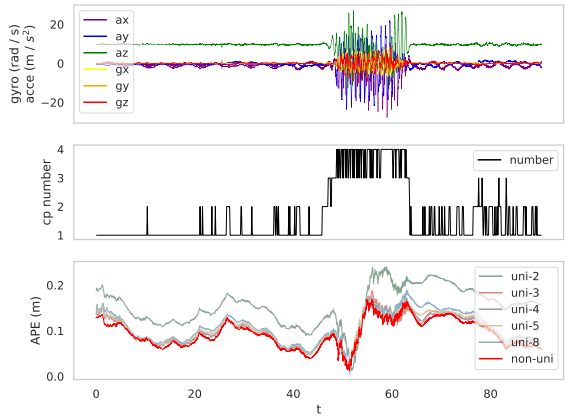

Fig. 6. Top shows the IMU data in *Hybrid1* and the middle shows the number of control points set per Δt with time by adaptive technique. The bottom shows the pose error curve over time based on different distributions of the control points and the red one denotes the result of *non-uni*.

uni-1 fails to adapt to, resulting in poor pose estimation. Increasing the number of control points can improve accuracy, but still, excess ones will decrease accuracy and increase computation time. On the other hand, *non-uni* adjusts the density of control points based on inertial data at each time interval, achieving the optimal accuracy with moderate computation cost.

3) Hybrid Sequences: It is inadequate for *uni-1* to accurately fit trajectory segments with intense motions, causing the failure of the estimation. Adding adequate control points can enhance the accuracy, whereas it appears to be unnecessary and incurs excess computation consumption for simple and smooth motion profiles. By adding control points during intense motions and reducing them during smooth motions, as displayed in Fig. 5, our *non-uni* attains the highest accuracy at almost the lowest time cost. Fig. 6 also depicts that *non-uni* maintains a small pose error throughout the whole trajectory owing to the adaptive non-uniform technique.

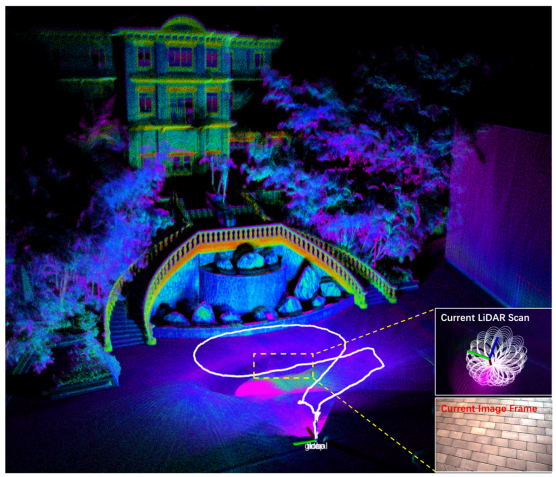

Fig. 7. Odometry and mapping result of Coco-LIC on the sequence *degenerate_seq_00*, where severe LiDAR degeneration happens when Livox Avia faces the ground for a while. Coco-LIC overcomes the degradation and succeeds in returning to the origin.

B. Robustness and Accuracy Evaluation of LiDAR-Inertial-Camera Odometry

We test our LICO (Coco-LIC) on both challenging datasets and large-scale datasets, comparing it against four LIC methods: LVI-SAM [3], R3LIVE [9], FAST-LIVO [10], CLIC [11] and a LIO system FAST-LIO2 [42], as well as a VIO system VINS-Mono [12]. Since LVI-SAM only supports rotating LiDAR and R3LIVE only supports solid-state LiDAR, Table III misses the results of LVI-SAM and Table IV misses R3LIVE. During the experiments, the loop closure module is disabled for pure odometry evaluation, and the reported results are the average of 6 runs. Note that, CLIC assigns three control points every Δt seconds, as described in [11], whereas Coco-LIC dynamically places control points.

1) Challenging Degenerate Dataset: We validate the robustness of Coco-LIC in all challenging sequences provided in R3LIVE and FAST-LIVO, using Livox Avia[3] LiDAR at 10 Hz and its internal IMU at 200 Hz, along with a camera at 15 Hz. These sequences exhibit severe degradation, such as the solid-state LiDAR with small FoV facing the ground or walls, and the camera facing textureless surfaces or capturing blurry images

[3][Online]. Available: https://www.livoxtech.com/avia

TABLE III
THE START-TO-END DRIFT ERROR ON ALL CHALLENGING SEQUENCES OF R3LIVE AND FAST-LIVO (UNIT: METERS/DEGREES). *DEGENERATE_SEQ_00*, *DEGENERATE_SEQ_01*, AND *DEGENERATE_SEQ_02* ARE FROM R3LIVE

	Degeneration Type(*)	FAST-LIO2	VINS-Mono	R3LIVE	FAST-LIVO	CLIC	Coco-LIC
degenerate_seq_00	L	9.019 / 3.441	0.807 / 12.736	0.035 / **0.405**	0.420 / 3.621	0.031 / 0.578	**0.016** / 0.428
degenerate_seq_01	L	3.161 / 15.632	2.455 / 3.523	0.114 / 0.536	0.881 / 2.248	1.287 / 1.293	**0.062** / **0.262**
degenerate_seq_02	L, V1	fail	fail	**0.065** / **1.124**	3.855 / 9.303	2.457 / 2.823	0.122 / 1.283
LiDAR_Degenerate	L	0.798 / 1.816	1.644 / 2.572	8.733 / **1.861**	0.049 / 2.025	6.811 / 5.911	**0.045** / 2.497
Visual_Challenge	L, V2	2.957 / 2.707	fail	0.234 / 0.751	**0.043** / **0.408**	5.262 / 4.537	0.166 / 0.889

(*) L denotes LiDAR faces the ground or walls. V1, V2 denote camera faces a white wall and undergoes motion blur, resp.
Lidar_Degenerate and Visual_Challenge are from FAST-LIVO. The best results are marked in bold. *fail* means the start-to-end drift error is larger than 10 meters.

TABLE IV
THE RMSE (M) OF APE RESULTS ON URBANNAV DATASET. FOR SIMPLICITY, THE SEQUENCES URBANNAV-HK-MEDIUM-URBAN-1, URBANNAV-HK-DEEP-URBAN-1, AND URBANNAV-HK-HARSH-URBAN-1 ARE REFERRED TO AS *MEDIUM*, *DEEP*, AND *HARSH* RESPECTIVELY

	Medium (3.64km)	*Deep* (4.51km)	*Harsh* (3.04km)
FAST-LIO2	6.734	6.335	2.820
VINS-Mono	102.582	63.894	fail
LVI-SAM	7.456	8.607	25.684
FAST-LIVO	7.331	7.159	2.787
CLIC	6.923	6.001	2.415
Coco-LIC	**6.031**	**5.688**	**2.189**

TABLE V
THE AVERAGE TIME CONSUMPTION (MILLISECONDS) OF DIFFERENT MODULES OF FAST-LIO2, CLIC, COCO-LIC ON THE SEQUENCE *MEDIUM*

	FAST-LIO2	CLIC	Coco-LIC
LiDAR Association	24.14	32.90	31.46
Visual Association	-	0.74	18.90
Optimization	1.07	29.05	9.09

due to aggressive motions, as partly shown in Fig. 7. The ground truth is not provided, but the rig starts and ends at the same position.

Table III shows the start-to-end drift error of various methods. VINS-Mono fails to work in the *Visual_Challenge* sequence due to intense motions and varying illumination, which makes it fail to stably track the visual features. The *LiDAR_Degenerate* and *degenerate_*xx sequences suffer from serious degradation of LiDAR point cloud, and when Livox Avia LiDAR faces a plane, FAST-LIO2 lacks constraints in certain DoFs, leading to insufficient pose estimation. R3LIVE and FAST-LIVO can handle various degenerations and have low start-to-end drift in most sequences, but there are also cases where both methods experience significant drift. In contrast, by tightly coupling LIC data, Coco-LIC achieves plausible performance in all sequences and minimal drift in several sequences. Besides, it is worth noting that although CLIC fuses all information at a time, it still fails to work in LiDAR-degenerated scenarios. This is because CLIC enforces the majority of control points in the visual sliding window to be fixed, resulting in no full exploitation of visual data [11].

2) Large-Scale Dataset: To quantitatively evaluate our approach, we conduct experiments on the large-scale UrbanNav dataset [43]. The dataset includes a 32-beam 3D LiDAR Velodyne HDL-32E[4] at 10 Hz, a stereo camera at 15 Hz (only use the left camera in this experiment), and an Xsens MTi-10 IMU at 400 Hz, collected in urban areas with many dynamic objects by a human-driving vehicle.

Table IV shows the RMSE of APE of the aforementioned methods. VINS-Mono performs poorly due to incorrect feature tracking caused by moving objects. LIO and LICO achieve satisfactory results in the absence of degraded LiDAR scenes. However, FAST-LIVO has slightly higher trajectory errors, possibly due to its sparse direct visual alignment being affected by moving objects and lighting changes. Notably, in highly-dynamic motion scenarios, the accuracy of CLIC and Coco-LIC is higher attributed to their continuous-time trajectory representation, which effectively handles LiDAR distortion and efficiently fuses high-rate IMU data. Additionally, Coco-LIC reuses high-quality map points for visual factors, resulting in better performance than CLIC.

We further investigate the time consumption of competitive methods and Coco-LIC on the sequence *Medium* with a duration of 785 seconds. Table V summarizes the result. *LiDAR Association* refers to updating the local LiDAR map (global LiDAR map in the case of FAST-LIO2) and finding associations for LiDAR surf points. For CLIC, *Visual Association* means visual feature extraction and tracking, and for Coco-LIC, it means updating the global LiDAR map and associating map points with the image frames. *Optimization* represents executing ESIKF update or factor graph optimization. FAST-LIO2 shows brilliant efficiency due to the adoption of ESIKF and ikd-Tree [42]. Compared to the closest work, CLIC, the proposed Coco-LIC is involved fewer control points and excludes the depth estimation process, which leads to less consuming time. Overall, all three compared methods are able to achieve real-time performance. Coco-LIC consumes around 639 seconds on the entire sequence. Note that the current implementation of Coco-LIC can be further optimized for efficiency, such as by refining the map management strategy to accelerate the association.

V. CONCLUSION AND FUTURE WORK

This letter presents Coco-LIC, a continuous-time LiDAR-Inertial-Camera odometry that tightly integrates information from LiDAR, IMU, and camera using non-uniform B-splines. The method achieves higher accuracy in pose estimation with moderate time consumption by placing more control points where motion is aggressive and fewer control points where motion is smooth. Additionally, it relies on the LiDAR map points for formulating frame-to-map visual factors. This eliminates the need for depth estimation and optimization by multiple visual keyframes, which benefits the efficiency of our continuous-time estimator. Real-world dataset experiments demonstrate

[4][Online]. Available: https://velodynelidar.com/products/hdl-32e

the importance of non-uniform control point placement and the effectiveness of our non-uniform continuous-time method. Robustness and accuracy evaluations show that Coco-LIC outperforms other state-of-the-art LIC Odometry, even in severe degenerate scenarios. In the future, it is worthwhile investigating more efficient map management strategies and incorporating complementary sensors like event cameras.

REFERENCES

[1] J. Zhang and S. Singh, "Laser–visual–inertial odometry and mapping with high robustness and low drift," *J. Field Robot.*, vol. 35, no. 8, pp. 1242–1264, 2018.

[2] W. Shao, S. Vijayarangan, C. Li, and G. Kantor, "Stereo visual inertial LiDAR simultaneous localization and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 370–377.

[3] T. Shan, B. Englot, C. Ratti, and D. Rus, "LVI-SAM: Tightly-coupled LiDAR-visual-inertial odometry via smoothing and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 5692–5698.

[4] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super odometry: IMU-centric LiDAR-visual-inertial estimator for challenging environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8729–8736.

[5] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "LIC-fusion: LiDAR-inertial-camera odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 5848–5854.

[6] X. Zuo et al., "LIC-fusion 2.0: LiDAR-inertial-camera odometry with sliding-window plane-feature tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5112–5119.

[7] D. Wisth, M. Camurri, S. Das, and M. Fallon, "Unified multi-modal landmark tracking for tightly coupled LiDAR-visual-inertial odometry," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1004–1011, Apr. 2021.

[8] J. Lin, C. Zheng, W. Xu, and F. Zhang, "R^2 LIVE: A robust, real-time, LiDAR-inertial-visual tightly-coupled state estimator and mapping," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7469–7476, Oct. 2021.

[9] J. Lin and F. Zhang, "R^3 live: A robust, real-time, RGB-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 10672–10678.

[10] C. Zheng, Q. Zhu, W. Xu, X. Liu, Q. Guo, and F. Zhang, "FAST-LIVO: Fast and tightly-coupled sparse-direct LiDAR-inertial-visual odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 4003–4009.

[11] J. Lv, X. Lang, J. Xu, M. Wang, Y. Liu, and X. Zuo, "Continuous-time fixed-lag smoothing for LiDAR-inertial-camera slam," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 4, pp. 2259–2270, Aug. 2023.

[12] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[13] 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.*, 2020, pp. 5135–5142.

[14] S. Lovegrove, A. Patron-Perez, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," *Int. J. Comput. Vis.*, vol. 113, no. 3, pp. 208–219, 2015.

[15] D. Hug and M. Chli, "HyperSLAM: A generic and modular approach to sensor fusion and simultaneous localization and mapping in continuous-time," in *Proc. IEEE Int. Conf. 3D Vis.*, 2020, pp. 978–986.

[16] J. Lv, K. Hu, J. Xu, Y. Liu, X. Ma, and X. Zuo, "CLINS: Continuous-time trajectory estimation for LiDAR-inertial system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 6657–6663.

[17] X. Lang, J. Lv, J. Huang, Y. Ma, Y. Liu, and X. Zuo, "Ctrl-VIO: Continuous-time visual-inertial odometry for rolling shutter cameras," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11537–11544, Oct. 2022.

[18] G. Cioffi, T. Cieslewski, and D. Scaramuzza, "Continuous-time vs. discrete-time vision-based SLAM: A comparative study," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2399–2406, Apr. 2022.

[19] B. Vandeportaele, P.-A. Gohard, M. Devy, and B. Coudrin, "Pose interpolation for rolling shutter cameras using non uniformly time-sampled b-splines," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2017, pp. pp–286.

[20] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1360–1367.

[21] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst.*, 2014, pp. 1–9.

[22] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[24] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 4973–4980.

[25] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 2088–2095.

[26] P. Furgale, C. H. Tong, T. D. Barfoot, and G. Sibley, "Continuous-time batch trajectory estimation using temporal basis functions," *Int. J. Robot. Res.*, vol. 34, no. 14, pp. 1688–1710, 2015.

[27] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1280–1286.

[28] J. Rehder, R. Siegwart, and P. Furgale, "A general approach to spatiotemporal calibration in multisensor systems," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 383–398, Apr. 2016.

[29] J. Lv, X. Zuo, K. Hu, J. Xu, G. Huang, and Y. Liu, "Observability-aware intrinsic and extrinsic calibration of LiDAR-IMU systems," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3734–3753, Dec. 2022.

[30] 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 Trans. Robot.*, vol. 38, no. 2, pp. 978–997, Apr. 2022.

[31] A. J. Yang, C. Cui, I. A. Bârsan, R. Urtasun, and S. Wang, "Asynchronous multi-view SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 5669–5676.

[32] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-time visual-inertial odometry for event cameras," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1425–1440, Dec. 2018.

[33] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers, "Efficient derivative computation for cumulative b-splines on lie groups," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11148–11156.

[34] S. Anderson, F. Dellaert, and T. D. Barfoot, "A hierarchical wavelet decomposition for continuous-time SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 373–380.

[35] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 3565–3572.

[36] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4666–4672.

[37] F. Shi, *Computer-Aided Geometric Design and Non-Uniform Rational B-Splines: CAGD and NURBS*. Beijing, China: Beihang Univ. Press, 1994.

[38] J. Lowther and C.-K. Shene, "Teaching B-splines is not difficult!," *ACM SIGCSE Bull.*, vol. 35, no. 1, pp. 381–385, 2003.

[39] K. Qin, "General matrix representations for B-splines," in *Proc. IEEE Pacific Graph. 6th Pacific Conf. Comput. Graph. Appl.*, 1998, pp. 37–43.

[40] S. Agarwal, K. Mierle, and T. C. S. Team, *Ceres Solver*. 2022. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[41] M. Grupp, "evo: Python package for the evaluation of odometry and SLAM," 2017. [Online]. Available: https://github.com/MichaelGrupp/evo

[42] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

[43] L.-T. Hsu et al., "UrbanNav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas," in *Proc. 34th Int. Tech. Meeting Satell. Division Inst. Navigation*, 2021, pp. 226–256.