CLINS: Continuous-Time Trajectory Estimation for LiDAR-Inertial System

Jiajun Lv[1,†], Kewei Hu[1,†], Jinhong Xu[1], Yong Liu[1], Xiushui Ma[2], Xingxing Zuo[1]

***Abstract*— In this paper, we propose a highly accurate continuous-time trajectory estimation framework dedicated to SLAM (Simultaneous Localization and Mapping) applications, which enables fuse high-frequency and asynchronous sensor data effectively. We apply the proposed framework in a 3D LiDAR-inertial system for evaluations. The proposed method adopts a non-rigid registration method for continuous-time trajectory estimation and simultaneously removing the motion distortion in LiDAR scans. Additionally, we propose a two-state continuous-time trajectory correction method to efficiently and efficiently tackle the computationally-intractable global optimization problem when loop closure happens. We examine the accuracy of the proposed approach on several publicly available datasets and the data we collected. The experimental results indicate that the proposed method outperforms the discrete-time methods regarding accuracy especially when aggressive motion occurs. Furthermore, we open source our code at `https://github.com/APRIL-ZJU/clins` to benefit research community.**

I. Introduction

Multi-sensor fusion plays an essential role in simultaneous localization and mapping (SLAM) algorithms with its complementarity and robustness, and it has been widely deployed in autonomous navigation, scene reconstruction, mixed reality, etc. In this paper, we study the high-accuracy continuous-time trajectory estimation with a fusion of LiDAR and IMU measurements. Most existing methods process LiDAR and IMU measurements in a discrete-time fashion. The LiDAR scan collecting points when the laser heads rotate around a mechanical axis, thus motion distortion is unavoidable when the LiDAR does not keep static in the data collecting process. Discrete-time based methods undistort the LiDAR points into the start time instant of the LiDAR scan by interpolations. IMU measurements are interpolated and integrated to formulate relative poses constraints between discrete LiDAR scans. Discrete-time based methods have several inherent limitations that hurt the estimation. Firstly, in practice, different sensors do not get measurements at the same frequency, let alone the same time instants. Interpolations has to be employed to fuse measurements from different sensors, which introduces non-negligible errors. Secondly, it is unlikely to leverage the raw measurements directly, i.e. the raw LiDAR points and raw IMU measurements. Raw LiDAR points are undistorted into the specific time instants to constitute LiDAR scans, while IMU measurements are assembled to get integrated relative pose measurements. These phenomenons result from the intractable super-high-frequency raw sensor measurements, which requires huge amount of pose variable to be estimated if they are utilized in a direct way. The above-mentioned difficulties can be summarized as the discrete pose representation fails to meet the system's demand of high temporal resolution. Recently, the continuous-time based method, which models the trajectory as a function of time and supports querying poses at any timestamp that naturally solves the problem of integrating asynchronous and high-frequency data. With those sound properties, continuous-time based method has been applied to many areas, such as visual-inertial navigation system [1], [2], event camera [3], rolling-shutter camera [1], actuated LiDAR [4], intrinsic and extrinsic calibration between sensors [5], [6].

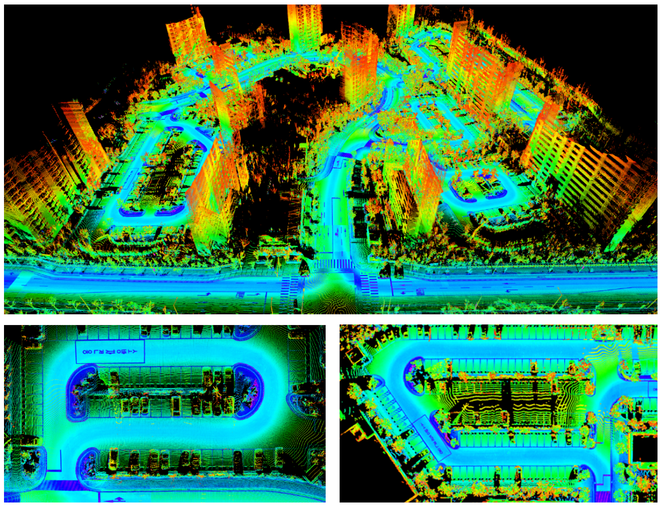


Fig. 1: Dense 3D reconstruction of Kaist-Urban-07 dataset by simply assembling 2D LiDAR scans from SICK LMS-511 with the estimated continuous-time trajectory from CLINS. The global trajectory is estimated with 3D LiDAR, Velodyne VLP-16 and Xsens IMU, MTi-300. 2D LiDAR scans is accumulated for reconstruction due to its high density.

This paper proposes a complete continuous-time trajectory estimation framework for LiDAR-inertial system. We summarize the contributions as follows:

- We propose a continuous-time trajectory estimator, which now supports the fusion of 3D LiDAR points and inertial data, and it is easy to expand and fuse data from other asynchronous sensors at arbitrary frequencies.
- We propose a two-stage continuous-time trajectory correction method to efficiently and effectively tackle loop closures.
- The proposed approach is extensively evaluated on

[1] April Lab, Zhejiang University, Hangzhou, China. (Yong Liu is the corresponding author, email: yongliu@iipc.zju.edu.cn)
[2] NingboTech University, Ningbo, China.
[†] Jiajun Lv and Kewei Hu contribute equally to this work.

several publicly available datasets and our collected datasets, and compared to several state-of-the-art methods. We further make the code open-sourced. To the best of our knowledge, this is the first open-sourced continuous-time LiDAR-inertial trajectory estimator.

II. Related Works

Continuous-time state estimation for solving SLAM problem is firstly systematically derived in [5] by Furgale et al. They firstly employ the continuous-time batch optimization on calibrating the rigid transformation between camera and IMU in visual-inertial system, and successively explore to calibrate temporal offset between the camera and IMU [7], shutter timings for rolling-shutter camera [8], spatio-temporal extrinsics between a LiDAR and a stereo-visual-inertial system [9]. Rehder et al. summarize their works and develop a general framework [10] for general spatio-temporal calibration between diverse sensors and evaluate on several combinations of different sensors in support of the generality claim. In addition to the calibration applications of continuous-time state estimation, Furgale et al. also present the details of theoretical derivations and effective implementation using B-splines in the well-known Kalibr calibration toolbox [11]. Sommer et al. further investigate the initialization and analytical Jacobians of B-splines on Lie group [11]. This series of works has made a significant contribution to the continuous-time state estimation.

Recently, continuous-time trajectory method has been employed in LiDAR odometry. In [12], [13], researchers propose a continuous SLAM solution with a spinning 2D laser scanner which is relatively dense and friendly for surfel-based registration. Alismail et al. propose continuous-ICP [4] that explicitly accounts for sensor motion during registration which improves the accuracy compared to rigid registration. Since global batch optimization of continuous-time trajectory is highly time-consuming, Droeschel et al. [14] present a hierarchical refinement structure that optimizes a single firing sequence based sub-graph firstly and incorporate sub-graph when optimizing the allocentric pose graph. Park et al. [15] adopt a map-centric method which introduces map deformation to remove the need for global trajectory optimization. Instead of adjusting the global continuous trajectory directly when loop closure happens, we propose a two-state based continuous trajectory correction method. We firstly perform a discrete-time pose graph optimization involved with key-scan poses; then the continuous-time trajectory is aligned with the optimized key-scan poses while maintaining local shape via original local velocities.

III. Representation of Continuous-time Trajectory

First of all, we introduce notations used in this paper. We denote the 6-DoF rigid transformation by ${}^{B}_{A}\mathbf{T} \in SE(3) \in \mathbb{R}^{4\times4}$, which transforms the point ${}^{A}\mathbf{p} \in \mathbb{R}^3$ in the frame $\{A\}$ into frame $\{B\}$. ${}^{B}_{A}\mathbf{T} = \begin{bmatrix} {}^{B}_{A}\mathbf{R} & {}^{B}\mathbf{p}_A \\ \mathbf{0} & 1 \end{bmatrix}$ consists of rotational part ${}^{B}_{A}\mathbf{R} \in SO(3)$ and translational part ${}^{B}\mathbf{p}_A \in \mathbb{R}^3$. For simplicity, we omit the homogeneous conversion in the rigid transformation by ${}^{B}\mathbf{p} = {}^{B}_{A}\mathbf{T}{}^{A}\mathbf{p}$. $\mathrm{Exp}(\cdot)$ maps tangent vector in $\mathbb{R}^3$ to Lie group $SO(3)$, and $\mathrm{Log}(\cdot)$ is its inverse operation. $(\cdot)_\vee$ is used to map the elements in the Lie algebra to tangent vectors.

Uniform B-Spline Basics	
Order	k
Knot distance	Δt
Coeff. Vec. $(^1)$	$\boldsymbol{\phi}(u) = \mathbf{M}^{(k)}\mathbf{u}$
Cumul. Coeff. Vec.$(^{1,2})$	$\boldsymbol{\lambda}(u) = \widetilde{\mathbf{M}}^{(k)}\mathbf{u}$
Position in Cumulative Form	
Control point	$\boldsymbol{\Phi}_p = \{\mathbf{p}_i\} \in \mathbb{R}^3, i \in [0, n]$
Distance	$\mathbf{d}^i_j = \mathbf{p}_{i+j} - \mathbf{p}_{i+j-1} \in \mathbb{R}^3$
Position	$\mathbf{p}(u) = \mathbf{p}_i + \sum_{j=1}^{k-1} \lambda_j(u) \cdot \mathbf{d}^i_j$
Velocity	$\mathbf{v}(u) = \sum_{j=1}^{k-1} \dot{\lambda}_j(u) \cdot \mathbf{d}^i_j$
Acceleration	$\mathbf{a}(u) = \sum_{j=1}^{k-1} \ddot{\lambda}_j(u) \cdot \mathbf{d}^i_j$
Orientation in Cumulative Form	
Control point	$\boldsymbol{\Phi}_R = \{\mathbf{R}_i\} \in SO(3), i \in [0, n]$
Distance	$\mathbf{d}^i_j = \mathrm{Log}\left(\mathbf{R}^{-1}_{i+j-1}\mathbf{R}_{i+j}\right) \in \mathbb{R}^3$
Position	$\mathbf{R}(u) = \mathbf{R}_i \cdot \prod_{j=1}^{k-1} \mathrm{Exp}\left(\lambda_j(u) \cdot \mathbf{d}^i_j\right)$
Velocity	$\boldsymbol{\omega}(u) = (\mathbf{R}^\top\dot{\mathbf{R}})_\vee$

$(^1)$ $t \in [t_i, t_{i+1})$, $\quad u(t) = s(t) - i$, $\quad s(t) := (t - t_0)/\Delta t$.

$(^2)$ Note that $\lambda_0(u) \equiv 1$.

TABLE I: Summarization of uniform B-Spline based continuous-time trajectory representation.

B-spline is smooth (C^2 continuity in case of cubic spline) and has local continuity. Most importantly, it has closed-form analytic derivatives which is easy to match against IMU measurements. To this end, we adopt two separate groups of B-splines to parameterize the 3D translation and 3D rotation, $\mathbf{p}(t) \in \mathbb{R}^3$ and $\mathbf{R}(t) \in SO(3)$. This formulation is termed as split representation of continuous-time trajectory in [16], [2]. A comprehensive overview of the B-spline can be found in [17]. Here we also summarize the definition of continuous trajectory representation by uniform B-splines in Tab. I. Specifically, spline matrix $\mathbf{M}^{(k)}$ and cumulative matrix $\widetilde{\mathbf{M}}^{(k)}$ are constant for uniform B-spline [17]. For translational trajectory representation in $\mathbb{R}^n$, the basic form and the cumulative form of the B-spline representation are equivalent and can be converted to each other. However, this is not the case in the non-Euclidean space $SO(3)$, The orientational trajectory representation is feasibly represented in the cumulative form. Tab. I lists cumulative form trajectory in $\mathbb{R}^3$ and $SO(3)$ that are adopted in this paper. Taking derivative of splines with respect to time, we can get velocity and acceleration. As shown in Tab. I, linear velocity $\mathbf{v}(t)$ and linear acceleration $\mathbf{a}(t)$ are in global frame while angular velocity $\boldsymbol{\omega}(t)$ is in local frame.

The continuous-time trajectory of IMU in global frame

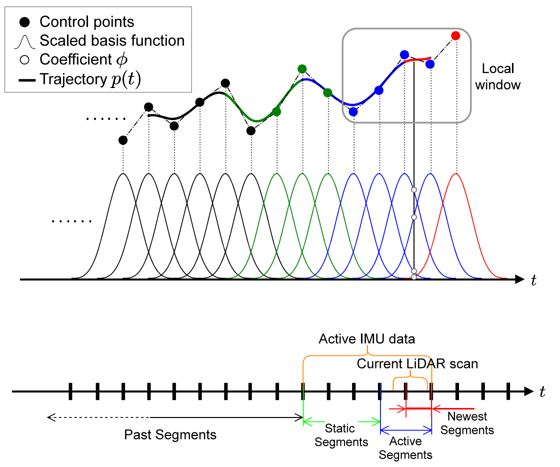


Fig. 2: An illustration of the proposed continuous-time trajectory for LiDAR-inertial system based on cubic B-splines.

$\{G\}$ is denoted as

$$ {}^G_I\mathbf{T}(t) = \begin{bmatrix} \mathbf{R}(t) & \mathbf{p}(t) \\ \mathbf{0} & 1 \end{bmatrix}, \tag{1} $$

where we omit the superscripts and subscripts for simplicity. With the known pre-calibrated extrinsic transformation ${}^I_L\mathbf{T}$ between IMU and LiDAR using toolbox LI-Calib [6], the trajectory of LiDAR could be calculated as

$$ {}^G_L\mathbf{T}(t) = {}^G_I\mathbf{T}(t) {}^I_L\mathbf{T}. \tag{2} $$

Due to the locality of the B-spline basis, for $t \in [t_i, t_{i+1})$, ${}^G_I\mathbf{T}(t)$ is only controlled by knots at $\{t_i, t_{i+1}, \cdots, t_{i+k-1}\}$ with their corresponding control points set $\mathbf{\Phi}(t_i, t_{i+1})$:

$$ \begin{aligned} \mathbf{\Phi}(t_i, t_{i+1}) &= \mathbf{\Phi}_R(t_i, t_{i+1}) \cup \mathbf{\Phi}_p(t_i, t_{i+1}) \\ &= \{\mathbf{R}_i, \cdots, \mathbf{R}_{i+k-1}\} \cup \{\mathbf{p}_i, \cdots, \mathbf{p}_{i+k-1}\}. \end{aligned} \tag{3} $$

IV. Methodology

Fig. 2 illustrates the definitions involved in the proposed continuous-time trajectory estimator of LiDAR-inertial system. We define the *active segments* of time based on the data-collection time interval of current LiDAR scan. The control point sequence $\mathbf{\Phi}_{\text{active}}$ corresponding to the active segments are denoted as *active control points* (blue dots and red dots). The subset of basis functions corresponding to $\mathbf{\Phi}_{\text{active}}$ are denoted as active basis functions (blue curves and red curves). The active trajectory (in blue and red) is determined by the active control points in a local region. Furthermore, except the active segments of time, the time segments involved with the active basis functions are notated as *static segments*. Accordingly, *static control points* $\mathbf{\Phi}_{\text{static}}$ (green dots) and static basis functions (green curves) are defined.

In the proposed system, after the coming of a new LiDAR scan, new control points (red dots) are added into the state vector to parameterize the extended trajectory. We utilize the integrated IMU poses to initialize these new control points (see Sec. IV-A). Subsequently, we extract LOAM features [18] including the edge points and planar points from current scan to make data association with local submap, which is composed of plenty of selected key-scans based on spatio-temporal distance. Finally, given the raw IMU measurements in the local window and the associated LiDAR features, we estimate the continuous-time trajectory in a batch optimization fashion. The state estimation problem could be formulated as a maximum a posteriori (MAP) problem. With the assumption of independent Gaussian noise corruption on the raw sensor measurements, we can estimate the continuous-time trajectory by solving a non-linear least squares (NLLS) problem.

A. Initialization

When new LiDAR scan arrives, new control points are added to extend the existing trajectory. We integrate discrete IMU data to obtain high-frequency rotation, position and linear velocity estimations, which are denoted as $\mathbf{R}_{I_m}, \mathbf{p}_{I_m}, \mathbf{v}_{I_m}$ at time t_m, respectively. We can minimize the following cost function to initialize the newly added control points $\mathbf{\Phi}_{\text{new}}$

$$ \begin{aligned} \arg\min_{\mathbf{\Phi}_{\text{new}}} \sum \Big(\left\| \text{Log}(\mathbf{R}_{I_m}^\top \mathbf{R}(t_m)) \right\| + \left\| \mathbf{p}(t_m) - \mathbf{p}_{I_m} \right\| + \\ \left\| \mathbf{v}(t_m) - \mathbf{v}_{I_m} \right\| \Big). \end{aligned} \tag{4} $$

B. Non-rigid Registration in Local Window

For new-coming scan $\mathcal{S}_k$, measuring during time interval $[t_k, t_k + \Delta T]$, where t_k is the timestamp of the first point in scan $\mathcal{S}_k$ and ΔT is the period of completing a LiDAR scan, it is essentially a non-rigid point cloud in $\mathcal{S}_k$ if there is external motion while sensor is scanning. Therefore traditional registration algorithms, which compute the relative transformation between two rigid point clouds, are not applicable or have degraded performance. To tackle this problem, we propose a non-rigid registration method which estimates the continuous-time trajectory in current scan.

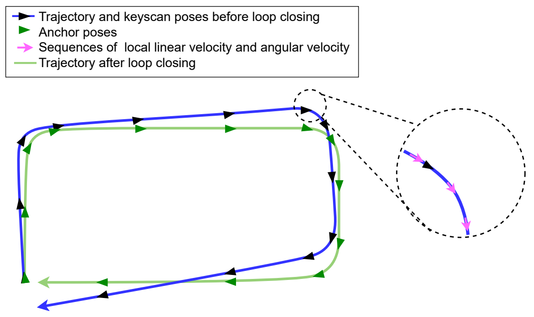
Specifically, each point in $\mathcal{S}_k$ is transformed to a unified frame $\{L_k\}$

$$ {}^{L_k}\mathbf{x}_{kj} = {}^G_L\mathbf{T}(t_k)^\top {}^G_L\mathbf{T}(t_k + \tau_j) {}^{L_{kj}}\mathbf{x}_{kj} \tag{5} $$

where τ_j is relative timestamp of point ${}^{L_{kj}}\mathbf{x}_{kj}$ with respect to the start time of current scan. Similar to [18], in order to keep the computation tractable, we extract the planar and edge features from the raw point cloud by computing local curvature. These extracted features are used in the following registration.

Local submap consists of some selected key-scans that are distributed in time or space. Note that when the changes of system' pose are over a certain extent, a key-scan is selected and all the points in that key-scan are undistorted into the start of the LiDAR scan by using the non-active trajectory. To find current features' correspondences in the local submap, we transform current features into the frame of local submap and make correspondences based on closet neighbors [18].

Considering LiDAR data degenerate in some structureless environments while the high-frequency IMU data are not



Trajectory and keyscan poses before loop closing
Anchor poses
Sequences of local linear velocity and angular velocity
Trajectory after loop closing

Fig. 3: An illustration of the two-stage continuous-time trajectory correction method for loop closures.

affected, we tightly couple IMU measurements with LiDAR features to constrain the trajectory. Thus the non-rigid registration problem can be defined as: given the associated LiDAR features in current scan and inertial measurements in active segments and static segments, estimate the active control points $\mathbf{\Phi}(t_k, t_k + \Delta T)$ of the trajectory and the biases of IMU. Specifically, we can solve this problem by minimizing the following objective function

$$\arg\min_{\mathcal{X}} \sum \|\mathbf{r}_{\mathcal{L}}\|_{\mathbf{\Sigma}_{\mathcal{L}}} + \|\mathbf{r}_a\|_{\mathbf{\Sigma}_a} + \|\mathbf{r}_w\|_{\mathbf{\Sigma}_w} \tag{6}$$

where $\mathcal{X} = \{\mathbf{\Phi}(t_k, t_k + \Delta T), \mathbf{b}_a, \mathbf{b}_g\}$, and $\mathbf{b}_a$, $\mathbf{b}_w$ are the bias of accelerator and gyroscope, respectively. $\mathbf{r}_{\mathcal{L}}, \mathbf{r}_a, \mathbf{r}_w$ are residual errors associated to LiDAR features and IMU measurements, respectively. $\mathbf{\Sigma}_{\mathcal{L}}, \mathbf{\Sigma}_a, \mathbf{\Sigma}_w$ are the corresponding covariance matrices. The residuals are defined as

$$\mathbf{r}_{\mathcal{L}} = \pi\left({}^G_L\mathbf{T}(t_k + \tau_j)^{L_{kj}}\mathbf{x}_{kj}\right) \tag{7}$$

$$\mathbf{r}_a = {}^G_I\mathbf{R}^\top(t_m)\left(\mathbf{a}(t_m) - {}^G\mathbf{g}\right) - \mathbf{a}_m + \mathbf{b}_a \tag{8}$$

$$\mathbf{r}_\omega = \boldsymbol{\omega}(t_m) - \boldsymbol{\omega}_m + \mathbf{b}_w \tag{9}$$

where project function $\pi(\cdot)$ is a point-to-plane projection for planar features and a point-to-line projection for edge features. $\mathbf{a}_m, \boldsymbol{\omega}_m$ are inertial measurements at time t_m. Note that static control points are involved in the optimization process but remain unchanged during the optimization. We use the Levenberg–Marquardt method implemented in Ceres Solver [19] to solve the above non-linear problem.

C. Trajectory Correction

Accumulative estimation drift is unavoidable in odometry systems, we also correct the estimations when loop closure occurs to mitigate the drift. Normallly, global optimization is required for smooth and consistent trajectory correction. Considering the fact that it is really time-consuming to perform global optimization on the continuous trajectory with abundant of knots, we propose a two-stage trajectory correction method that is effictive but computationally friendly. At stage one, when loop closures are detected, we perform a pose-graph optimization over the involved discrete poses of key-scans to eliminate cumulative draft. This stage is same with the traditional descrete-time based method [20]. At stage two, we try to update the control points with the updated poses of key-scans. This operation can be interpreted as projecting the continuous trajectory to

Error	Sequence	LOAM	LIO-SAM	LIOM	CLINS
Trans-lation RMSE (m)	fast1	0.4469	0.1058	0.0529	**0.0436**
	fast2	0.2023	0.1557	0.0663	**0.0616**
	mid1	0.1740	0.1486	0.0576	**0.0488**
	mid2	0.1010	0.0952	0.0874	**0.0731**
	slow1	0.0606	0.0727	0.0318	**0.0295**
	slow2	0.0666	0.0674	0.0435	**0.0376**
Rotation RMSE (rad)	fast1	0.1104	0.1047	**0.0537**	0.0565
	fast2	0.0763	0.1022	0.0574	**0.0538**
	mid1	0.0724	0.1020	**0.0523**	0.0567
	mid2	0.0617	0.0789	0.0567	**0.0538**
	slow1	0.0558	0.0698	0.0496	**0.0438**
	slow2	0.0614	0.0715	**0.0530**	0.0570

TABLE II: RMSE of translational and rotational estimation in the 6 sequences with motion varies from fast to slow.

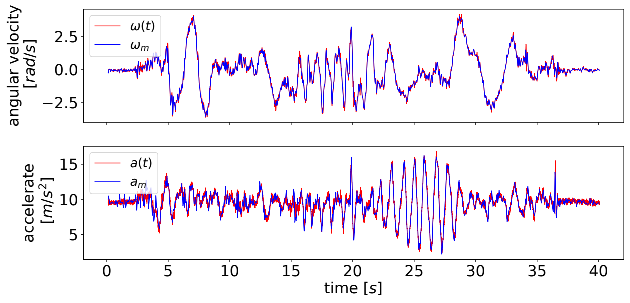


Fig. 4: The linear acceleration and angular velocity fitting results on *fast1* sequence. Only the z-axis components are shown. Red is from the derivatives of the estimated continuous-time trajectory, while blue is from the raw IMU measurements.

some anchored poses and preserving the local shape of the trajectory. In general, the estimation results of the inertial-aided LiDAR odometry is accurate in local regions, so we compute the local linear and angular velocities from the original continuous-time trajectory at certain time instants. Those velocities can be used to locally constrain the shape of the trajectory. To this end, the potential problem at state two to be solved could be explicitly formulated as

$$\begin{aligned}\arg\min_{\mathbf{\Phi}_{\text{update}}} &\sum \left(\left\|\mathrm{Log}(\hat{\mathbf{R}}_k^\top \mathbf{R}(t_k))\right\| + \|\mathbf{p}(t_k) - \hat{\mathbf{p}}_k\|\right) + \\ &\sum \left(\left\|\mathbf{R}(t_j)^\top \mathbf{v}(t_j) - \hat{\mathbf{v}}_j\right\| + \|\boldsymbol{\omega}(t_j) - \hat{\boldsymbol{\omega}}_j\|\right)\end{aligned} \tag{10}$$

where $\hat{\mathbf{R}}_k, \hat{\mathbf{p}}_k$ are the updated poses of key-scan at time t_k after the pose-graph optimization. And $\hat{\mathbf{v}}_j, \hat{\boldsymbol{\omega}}_j$ are the computed linear velocity and angular velocity at time t_j from the trajectory before correction, respectively. Fig. 3 illustrates the proposed two-stage based trajectory correction method.

V. Experimental Results

We qualitatively and quantitatively evaluate the proposed method in a series of publicly available datasets as well as the datasets we collected. Considering that there is no open-source continuous-time trajectory estimation method for LiDAR inertial system, we compare the proposed method

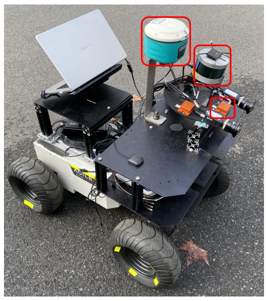


Fig. 5: The unmanned ground vehicle with self-assembled sensors rigidly mounted. Sensors with red box are used to collect YQ sequences in campus.

Sequence	Distance [km]	Duration [s]	Average Velocity [m/s]
YQ-01	3.26	2471	1.32
YQ-02	0.95	690	1.38
Kaist-Urban-07	2.54	570	4.46
Kaist-Urban-08	1.56	307	5.08

TABLE III: Some specifications of YQ and Kaist-Urban sequences.

with LIO-SAM [21] and LIOM [22] (abbreviation of LIO-mapping). In the following experiments, LIO-SAM without loop correction is notated as LIO-SAM(odom), and CLINS(odom) for CLINS without loop correction.

A. *Trajectory Estimation in Room-scale Scenes*

We evaluate the proposed CLINS on the publicly available datasets[1] provided in [22] with ground truth to examine the representation capability of our continuous-time trajectory formulation and the estimation accuracy of positions and orientations. The knot distance Δt of B-spline is set as 0.05 second to deal with the highly-dynamic motion. Tab II summarizes the root mean square error (RMSE) of the estimations from different methods. Note that the experimental results for LOAM [18] and LIOM in Tab. II are acquired directly from the paper [22]. It is also important to note that the output pose estimation of LIOM is around 5 Hz and LIO-SAM in 10 Hz, while for CLINS, we query and evaluate the estimated pose at 100Hz from the obtained continuous-time trajectory . From Tab. II we can see that CLINS provides more accurate translation estimation in all sequences while the accuracy of rotation is similar between LIOM and CLINS. Fig. 4 shows the fitting results of the estimated trajectory on *fast1* sequence compared to the raw IMU measurements. in which the angular velocity varies between -4.87 rad and 6.18 rad, while the acceleration reaches a maximum of 5.8 m/s^2. The experimental results on *fast1* sequence in Tab. II and Fig. 4 show that our proposed method can not only achieve the high accuracy in trajectory estimation, but also well fits the derivatives of the trajectories to the inertial measurements.

[1]Available at https://drive.google.com/drive/folders/1dPy667dAnJy9wgXmlnRgQZxQF_ESuve3

Method	YQ-01	YQ-02	Kaist-Urban-07	Kaist-Urban-08
LIO-SAM (odom)	8.857	3.215	1.288	3.524
CLINS(odom)	5.917	3.15	1.383	3.907
LIOM	3.931	**0.881**	1.515	16.277
LIO-SAM	**2.220**	2.487	1.972	3.951
CLINS	2.311	1.509	**0.562**	**1.133**

TABLE IV: Summary of RMSE(m) of APE results on YQ and Kaist-Urban sequences. The best results are shown in bold.

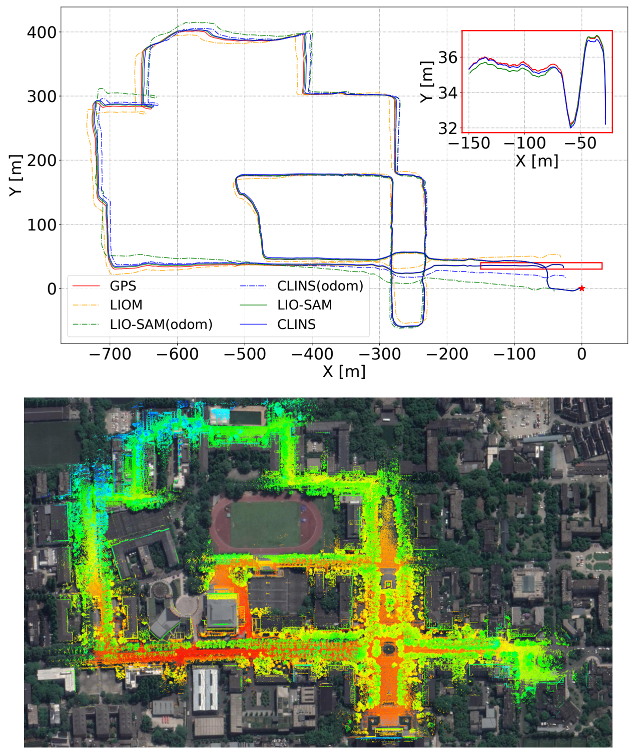


Fig. 6: Top: Trajectory comparison with different methods on YQ-01 sequence. The red star indicates the start position, and the end part of the trajectory are shown in zoom view. Bottom: Mapping results of CLINS with loop correction using YQ-01 Sequence. The map is consistent with the Google Earth imagery.

B. *Trajectory Estimation and Scene Mapping in Large-scale Scenes*

Vehicle platform. We further investigate the accuracy of the CLINS with or without loop correction on outdoor long-distance sequence, YQ-01, YQ-02, Kaist-Urban-07 and Kaist-Urban-08. The first two sequences are collected on campus by ourselves using a Velodyne VLP-16 LiDAR, an Xsens-300 IMU and JingLing-K50 RTK-GPS, and all these sensors are rigidly mounted on a small vehicle as shown in Fig. 5. The last two sequences are from Kaist Urban datasets [23], we leverage the IMU measurements and the data from the 3D LiDAR mounted on the vehicle at a tilt of about 45 degrees. Tab. III lists distances and durations of the above mentioned four sequences. Due to the slow motion

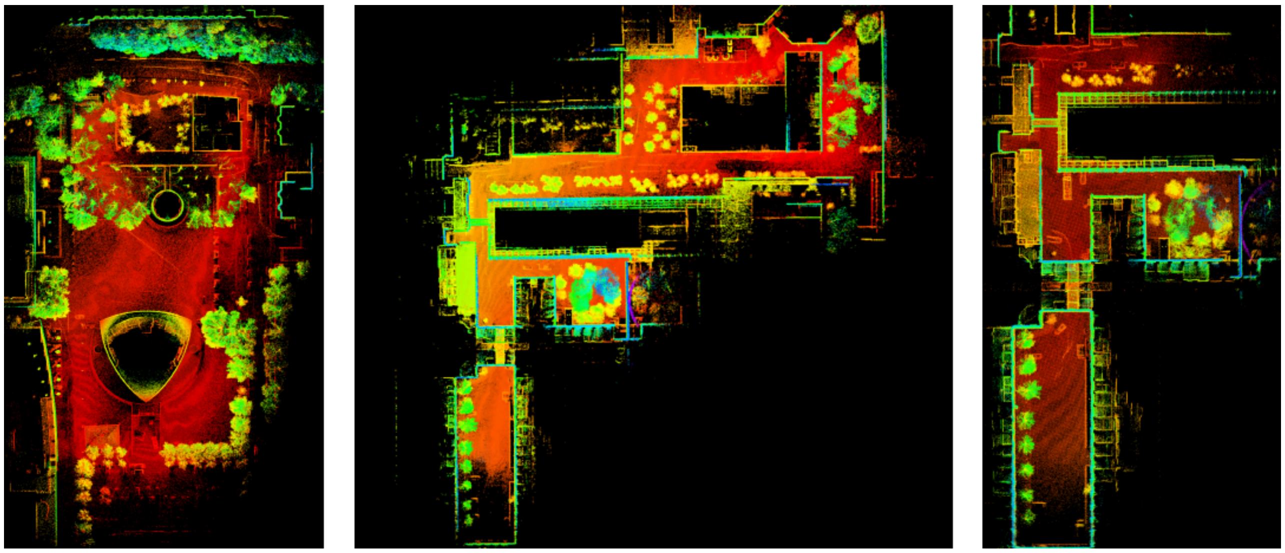


Fig. 7: Mapping results of CLINS using the *garden*, *walking* and *small campus* datasets from left to right, respectively. All are colored with reflective intensity.

of the on-board data, we set the knot distance Δt as 0.1 second. We compute absolute pose error (APE) [24] with the provided ground truth to compare the CLINS, LIO-SAM and LIOM. Tab. IV summarizes RMSE results, and the proposed CLINS shows promising accuracy in the experiments. Fig. 6 illustrates the trajectory comparison results with different methods on YQ-01 sequence. Additionally, the bottom figure in Fig. 6 shows the mapping result of CLINS using YQ-01 sequence. The map, about 700m$\times$500m in size, is consistent with the Google Earth imagery.

Handheld device. Considering that the main advantage of CLINS is non-rigid registration, we conduct qualitative experiments on open source handheld datasets provided from LIO-SAM[2]. Fig. 7 shows the mapping results for *garden*, *walking* and *small campus* sequences from left to right, respectively. Note that, during collection of *walking* sequence, aggressive motion both in translation and rotation are coupled.

C. Application of Global Continuous Trajectory

In this section, we introduce an interesting application of global continuous trajectory. Generally, a 2D LiDAR-inertial system is challenging to determine 6D pose no matter in the vehicle platform or the handheld platform. With the assistance of the estimated globally consistent continuous trajectory, 2D LiDAR can provide highly accurate reconstructions. Fig. 1 shows the 3D reconstruction result of Kaist-Urban-07 using 2D LiDAR data from SICK LMS-511. Note that global continuous trajectory is provided by CLINS with 3D LiDAR-inertial system without the participation of 2D LiDAR.

[2]Available at https://drive.google.com/drive/folders/1gJHwfdHCRdjP7vuT556pv8atqrCJPbUq

D. Implementation details

We adopt the flexible least squares solver Ceres to iteratively solve the NLLS problem and compute derivatives automatically. Typically the non-rigid registration converges within four or five iterations, consuming about 200ms. The computation of jacobians takes most of the computation time and real-time performance can be obtained by using analytical derivatives and exploring the efficient derivative computation for B-Spline [17].

VI. Conclusion

In this paper, we propose a continuous-time trajectory estimation approach for LiDAR inertial system, termed CLINS. To the best of our knowledge, this is the first open-source continuous time LiDAR-inertial trajectory estimation method. In addiction, we propose a two-state continuous-time trajectory correction method to efficiently and effectively cope with the computationally-intensive loop closure correction. We compare CLINS against several open-source state-of-the-art discrete-time based algorithms. The experiments indicate that CLINS outperforms the discrete-time based methods regarding accuracy, which is even more significant in the case of aggressive motion due to the non-rigid registration in our method. There are still potential future works to improve the accuracy and efficiency of the system. For example, it is interesting to investigate reducing the number of static control points in the local window. Instead of automatic derivation, using analytical jacobians and taking full advantage of the recurrence relations [17] in the spline computation to reduce computational effort is also worth exploring.

VII. Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant 61836015. We like to thank Hangyu Wu for his assistance with visualizing the results.

References

[1] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras." in *BMVC*, vol. 2, no. 5, 2013, p. 8.

[2] H. Ovrén and P.-E. Forssén, "Trajectory representation and landmark projection for continuous-time structure from motion," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 686–701, 2019.

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

[4] H. Alismail, L. D. Baker, and B. Browning, "Continuous trajectory estimation for 3d slam from actuated lidar," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6096–6101.

[5] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2088–2095.

[6] J. Lv, J. Xu, K. Hu, Y. Liu, and X. Zuo, "Targetless calibration of lidar-imu system based on continuous-time batch estimation," *arXiv preprint arXiv:2007.14759*, 2020.

[7] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.

[8] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1360–1367.

[9] J. Rehder, P. Beardsley, R. Siegwart, and P. Furgale, "Spatio-temporal laser to visual/inertial calibration with applications to hand-held, large scale scanning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 459–465.

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

[11] P. Furgale, C. H. Tong, T. D. Barfoot, and G. Sibley, "Continuous-time batch trajectory estimation using temporal basis functions," *The International Journal of Robotics Research*, vol. 34, no. 14, pp. 1688–1710, 2015.

[12] R. Zlot and M. Bosse, "Efficient large-scale three-dimensional mobile mapping for underground mines," *Journal of Field Robotics*, vol. 31, no. 5, pp. 758–779, 2014.

[13] L. Kaul, R. Zlot, and M. Bosse, "Continuous-time three-dimensional mapping for micro aerial vehicles with a passively actuated rotating laser scanner," *Journal of Field Robotics*, vol. 33, no. 1, pp. 103–132, 2016.

[14] 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.

[15] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic lidar fusion: Dense map-centric continuous-time slam," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1206–1213.

[16] A. Haarbach, T. Birdal, and S. Ilic, "Survey of higher order rigid body motion interpolation methods for keyframe animation and continuous-time trajectory estimation," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 381–389.

[17] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers, "Efficient derivative computation for cumulative b-splines on lie groups," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 148–11 156.

[18] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9, 2014.

[19] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[20] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[21] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," *arXiv preprint arXiv:2007.00258*, 2020.

[22] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3144–3150.

[23] 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.

[24] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.