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

Jiajun Lv\textsuperscript{1,†}, Kewei Hu\textsuperscript{1,†}, Jinhong Xu\textsuperscript{1}, Yong Liu\textsuperscript{1}, Xiushui Ma\textsuperscript{2}, Xingxing Zuo\textsuperscript{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 phenomena 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].

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...
several publicly available datasets and our collected
datasets, and compared to several state-of-the-art meth-
ods. 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 prob-
lem 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 sys-
tem [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 combina-
tions 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 the
theoretical derivations and effective implementation using B-
splines in the well-known Kalibr calibration toolbox [11].
Sommer et al. further investigate the initialization and analyti-
cal 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, Droschel 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 deforma-
tion 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 \( ^B_A \mathbf{T} \in SE(3) \in \mathbb{R}^{4 \times 4} \), which transforms the point \( ^A \mathbf{p} \in \mathbb{R}^3 \) in the frame \( \{A\} \)
to frame \( \{B\} \). \( ^B_A \mathbf{T} = \begin{bmatrix} ^B_A \mathbf{R} & ^B_a \mathbf{p}_A \\ 0 & 1 \end{bmatrix} \) consists of rotational
part \( ^B_A \mathbf{R} \in SO(3) \) and translational part \( ^B_a \mathbf{p}_A \in \mathbb{R}^3 \). For

### Uniform B-Spline Basics

| Order | \( k \) |
|---|---|
| Knot distance | \( \Delta t \) |
| Coeff. Vec. \((1)\) | \( \phi(u) = M^{(k)} \mathbf{u} \) |
| Cumul. Coeff. Vec. \((1-2)\) | \( \lambda(u) = \tilde{M}^{(k)} \mathbf{u} \) |

### Position in Cumulative Form

| Control point | \( \Phi_p = \{ \mathbf{p}_i \} \in \mathbb{R}^3, i \in [0, n] \) |
| Distance | \( d_j = \mathbf{p}_{i+j} - \mathbf{p}_{i+j-1} \in \mathbb{R}^3 \) |
| Position | \( \mathbf{p}(u) = \mathbf{p}_i + \sum_{j=1}^{n} \lambda_j(u) \cdot d_j \) |
| Velocity | \( \mathbf{v}(u) = \sum_{j=1}^{n} \dot{\lambda}_j(u) \cdot d_j \) |
| Acceleration | \( \mathbf{a}(u) = \sum_{j=1}^{n} \ddot{\lambda}_j(u) \cdot d_j \) |

### Orientation in Cumulative Form

| Control point | \( \Phi_R = \{ \mathbf{R}_i \} \in SO(3), i \in [0, n] \) |
| Distance | \( d_j = \log( \mathbf{R}_{i+j} \mathbf{R}_{i+j-1} ) \in \mathbb{R}^3 \) |
| Position | \( \mathbf{R}(u) = \mathbf{R}_i \prod_{j=1}^{n} \exp( \lambda_j(u) \cdot d_j ) \) |
| Velocity | \( \dot{\omega}(u) = (\dot{\mathbf{R}}) \mathbf{v} \) |

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

\(^{(2)}\) Note that \( \lambda_0(u) = 1. \)

**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
Table I. Specifically, spline matrix \( \tilde{M}^{(k)} \) and cumulative
matrix \( \tilde{M}^{(k)} \) 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 \( \omega(t) \) is in local frame.

The continuous-time trajectory of IMU in global frame
trajectory for LiDAR-inertial system based on cubic B-splines.

Fig. 2: An illustration of the proposed continuous-time trajectory and linear velocity estimations, which are denoted as \( R_{I_m}, P_{I_m}, v_{I_m} \) at time \( t_m \), respectively. We can minimize the following cost function to initialize the newly added control points \( \Phi_{\text{new}} \)

$$
\arg\min_{\Phi_{\text{new}}} \left( \| \log(R_{I_m}^t R(t_m)) \| + \| p(t_m) - p_{I_m} \| + \| v(t_m) - v_{I_m} \| \right).
$$

\[ \text{Equation (4)} \]

B. Non-rigid Registration in Local Window

For new-coming scan \( S_k \), measuring during time interval \( [t_k, t_k + \Delta T] \), where \( t_k \) is the timestamp of the first point in scan \( S_k \) and \( \Delta T \) is the period of completing a LiDAR scan, it is essentially a non-rigid point cloud in \( S_k \) 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 \( S_k \) is transformed to a unified frame \( \{L_k\} \)

$$
L_k x_{kj} = L_k^i T(t_k)^T L_k^j T(t_k + \tau_j) L_k^j x_{kj}
$$

\[ \text{Equation (5)} \]
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 $\Phi(t_k, t_k + \Delta T)$ of the trajectory and the biases of IMU. Specifically, we can solve this problem by minimizing the following objective function

$$\arg \min \ X \sum \| r_c \| \Sigma_c + \| r_a \| \Sigma_a + \| r_w \| \Sigma_w$$

(6)

where $X = \{ \Phi(t_k, t_k + \Delta T), b_a, b_g \}$, and $b_a, b_w$ are the bias of accelerometer and gyroscope, respectively. $r_c, r_a, r_w$ are residual errors associated to LiDAR features and IMU measurements, respectively. $\Sigma_c, \Sigma_a, \Sigma_w$ are the corresponding covariance matrices. The residuals are defined as

$$r_c = \pi (G^T L_k T(t_k + \tau_j) x_j)$$

(7)

$$r_a = G^T R^T (t_m)(a(t_m) - g) - a_m + b_a$$

(8)

$$r_w = \omega(t_m) - \omega_m + b_w$$

(9)

where project function $\pi(\cdot)$ is a point-to-plane projection for planar features and a point-to-line projection for edge features. $a_m, \omega_m$ 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. Normally, 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 effective 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 drift. This stage is same with the traditional discrete-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

![Fig. 3: An illustration of the two-stage continuous-time trajectory correction method for loop closures.](image)

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

| Error   | Sequence | LOAM | LIO-SAM | LIOM | CLINS |
|---------|----------|------|---------|------|-------|
| Translation | 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 |
| RMSE (m) | 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 | fast1  | 0.1104 | 0.0407 | 0.0537 | 0.0565 |
|          | fast2  | 0.0763 | 0.1022 | 0.0574 | 0.0538 |
|          | mid1   | 0.0724 | 0.1020 | 0.0523 | 0.0567 |
| RMSE (rad) | 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 |

![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.](image)
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 $\Delta t$ 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/1dPy667dAnJy9wgXminRgQ2xQF_ESuve3

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

| 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          |

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
of the on-board data, we set the knot distance $\Delta t$ 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×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\textsuperscript{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.

### 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 addition, 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. Ovren 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. Muellger, G. Gallego, H. Rebecq, and D. Scaramuzza, “Continuous-time visual-inertial odometry for event cameras,” IEEE Transactions on Robotics, vol. 34, no. 6, pp. 1688–1710, 2015.

[4] 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.

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