SLICT: Multi-Input Multi-Scale Surfel-Based Lidar-Inertial Continuous-Time Odometry and Mapping

Thien-Minh Nguyen, *Member, IEEE*, Daniel Duberg, *Member, IEEE*, Patric Jensfelt, *Member, IEEE*, Shenghai Yuan, and Lihua Xie, *Fellow, IEEE*

***Abstract*—While feature association to a *global map* has significant benefits, to keep the computations from growing exponentially, most lidar-based odometry and mapping methods opt to associate features with *local maps* at one voxel scale. Taking advantage of the fact that *surfels* (surface elements) at different voxel scales can be organized in a tree-like structure, we propose an octree-based global map of *multi-scale* surfels that can be updated incrementally. This alleviates the need for recalculating, for example, a k-d tree of the whole map repeatedly. The system can also take input from a single or a number of sensors, reinforcing the robustness in degenerate cases. We also propose a point-to-surfel (PTS) association scheme, continuous-time optimization on PTS and IMU preintegration factors, along with loop closure and bundle adjustment, making a complete framework for Lidar-Inertial continuous-time odometry and mapping. Experiments on public and in-house datasets demonstrate the advantages of our system compared to other state-of-the-art methods.**

***Index Terms*—Localization, mapping, sensor fusion.**

I. Introduction

THANKS to reduced cost and weight, lidar-based navigation systems have made significant progresses for autonomous systems in recent years, motivating many new advanced algorithms. In this letter, we are concerned with two main aspects of a lidar-inertial odometry and mapping (LIOM) system, namely feature extraction-association, and mapping.

For feature extraction-association (also termed the *frontend* [1]), existing methods can be grouped into three main categories. In the first one, planar and corner features are extracted based on the smoothness value, then associated with k-nearest neighbours to construct the corresponding cost factors. This method was proposed in [2] and is still widely adopted in many recent works [3], [4], [5], [6], [7]. One issue of this method is that the smoothness calculation is specialized for the lidar scans with horizontally separated rings, in environments with man-made structure. To overcome this issue, in FAST-LIO [8], [9], Xu et al. propose the *direct method*, where lidar points are directly associated with a neighborhood in the map. The method can work well with both the common spinning lidars (Ouster/Velodyne) and prism-based box-shaped lidar (Livox). We find that direct methods combined with the filter-based update scheme achieves quite good results ([8], [9], [10], [11]) for a smaller computation footprint compared to more computationally demanding iterative gradient-based optimization.

In another approach, high-abstraction features such as planes or lines [12], [13] can be extracted and treated as landmarks. The landmarks' coefficients will also be jointly estimated with the robot states, in similar fashion with visual-inertial odometry (VIO) systems. This approach has the benefit of dealing with a smaller amount of features. However, the feature extraction process is more complex and may not be able to detect planes or lines in cluttered or unstructured environments such as forests or mines.

In this work, we propose the use of surfel-based features at different resolutions for the frontend task. Different from geometric features such as planes or edges, surfels are characterized by ellipsoidal fitting parameters [14], with complexity between that of the direct method and high-abstraction features. Importantly, when surfels of one scale at the farther area of the map are still sparse, we can attempt association of the lidar points with the map at a higher scale. This allows us to balance the number of lidar factors at both close and far distance, which can reduce long-term drift. Instead of associating surfel to surfel like previous works [14], [15], [16], which requires complicated processes that involve surfelizing the input scans, then associating up-to-scale surfels using k-NN search based on high dimensional descriptors, and rejecting some associations based on timestamps and velocities [15], we propose a simpler strategy that directly matches the lidar points to surfels at all possible scales, and admit the surfel at smallest scale that satisfies all of the conditions (Section III-D) to construct the cost factor. Moreover, we also propose a continuous-time Maximum-A-Posteriori

Manuscript received 7 November 2022; accepted 9 February 2023. Date of publication 17 February 2023; date of current version 28 February 2023. This letter was recommended for publication by Associate Editor A. Nuechter and Editor S. Behnke upon evaluation of the reviewers' comments. This work was supported by Knut and Alice Wallenberg Foundation throughWallenberg AI, Autonomous Systems and Software Program (WASP) under Grant Call 10013 - Wallenberg-NTU Presidential Postdoctoral Fellowship, 2020. *(Corresponding author: Thien-Minh Nguyen.)*

Thien-Minh Nguyen, Daniel Duberg, and Patric Jensfelt are with the Division of Robotics, Perception and Learning, KTH Royal Institute of Technology, 11428 Stockholm, Sweden (e-mail: tmng@kth.se; dduberg@kth.se; patric@kth.se).

Shenghai Yuan and Lihua Xie are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: shyuan@ntu.edu.sg; elhxie@ntu.edu.sg).

To benefit the community, we release the source code and dataset at https://github.com/brytsknguyen/slict.

Digital Object Identifier 10.1109/LRA.2023.3246390

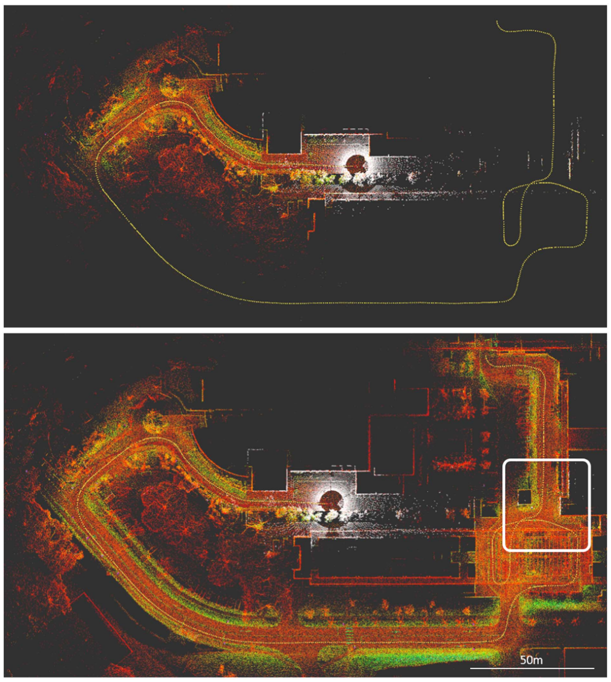


Fig. 1. While a local map (top) helps keep computation bounded, it limits the association only to recent marginalized features, which gives chance to estimation drift relative to the initial state. In contrast, with a global map, we can achieve early association with the earliest features (white box) and reinforce global consistency.

(MAP) optimization strategy, which allows us to directly use the raw pointcloud instead of deskewed points, which may contain error from the propagation process on top of the measurement noise.

On the mapping side, the update and query processes require special attention. In most conventional systems, a k-d tree structure is built to accelerate the k-NN querying process. However, as the map grows, recomputing the k-d tree on the global map is intractable. One strategy to avoid this issue is to organize the maps into keyframes and build a local map with bounded size from a finite number of nearest keyframes [3], [4], [5], [7], [14]. However this strategy is sub-optimal since there are cases where the local map misses important prior observations that were captured in previous keyframes (an example is given in Fig. 1).

To maintain efficient queries on a single global map, in [9], Xu et al. proposes the ikd-Tree framework, which can incrementally update the k-d tree of the map without recomputing the distances from scratch. Hence, Bai et al. proposed a new structure called iVox [17] that can achieve 5 ms map update time with 32-channel lidar pointcloud input. In [18], Yuan et al. proposed a structure called VoxelMap that contains coarse-to-fine voxels adaptively created based on the statistics of the points inside. We note that the aforementioned map structures are still based on a point-centric paradigm. In contrast, in this letter we propose a generalized incremental tree of surfels, based on the UFOMap framework [19].

The contributions in our work can be listed as follows:

- A full-fledged Lidar-Inertial Odometry and Mapping framework with frontend odometry, loop closure and global pose-graph optimization, capable of integrating multiple lidar inputs.
- A multi-scale point-to-surfel (PTS) association strategy and contiuous-time optimization of PTS and IMU preintegration factors on a sliding window.
- An implementation of a hierarchical multi-scale surfel map based on the UFOMap framework, enabling incremental update and efficient query of the global map.
- Extensive experiments on public and in-house datasets to validate the results and effects of the multi-scale surfel association strategy.
- We release the source code and the datasets for the benefit of the community.

Henceforth, we refer to our method as SLICT (*Surfel-based Lidar-Inertial Continous-Time Odometry and Mapping*) for short. The remaining of the letter is as follows: Section II provides some preliminary definitions for our problems, where Section II-D details the surfel tree and its implementation on the UFOMap. Section III describes in details the operation of the main blocks in our system. Section IV describes our experiments on public and in-house datasets. Section V concludes our work and provides comments for future extension.

II. Preliminary

A. Notation

In this letter for a vector $\mathbf{p} \in \mathbb{R}^3$, $\mathbf{p}^\top$ denotes its transpose, and $\lfloor \mathbf{p} \rfloor_\times$ denotes the skew-symmetric matrix of $\mathbf{p}$. For a physical quantity $\mathbf{p}$, we use the hat notation $\hat{\mathbf{p}}$ to denote the optimization-based estimate of $\mathbf{p}$. Similarly, the breve notation $\breve{\mathbf{p}}$ is used to refer to an IMU-propogated estimate of $\mathbf{p}$. To avoid convoluting definitions, we may refer to $\hat{\mathbf{p}}$, $\breve{\mathbf{p}}$ without first formally defining $\mathbf{p}$.

The robot orientation can be represented by the rotation matrix, denoted as $\mathbf{R}$, or quaternion $\mathbf{q}$. The two representations can be used interchangeably as the conversion between them is well defined.

We reserve the left superscript for the coordinate frames of a quantity. For example, ${}^{B_{t_s}}\mathbf{f}$ denotes a vector $\mathbf{f}$ whose coordinates are referenced in the robot's body at time t_s. For convenience we may also write ${}^{B_{t_s}}\mathbf{f}$ as ${}^{t_s}\mathbf{f}$.

B. State Estimate

We define a sliding window spanning a time period $[t_w, t_k]$, with M time instances (t_w, t_k included). Each t_m is associated with a state estimate $\hat{\mathcal{X}}_m$ defined as follows:

$$\hat{\mathcal{X}}_m = (\hat{\mathbf{R}}_m, \hat{\mathbf{p}}_m, \hat{\mathbf{v}}_m, \hat{\mathbf{b}}_{gm}, \hat{\mathbf{b}}_{am}) \in \mathrm{SO}(3) \times \mathbb{R}^{12}, \quad (1)$$

where $\hat{\mathbf{R}}_m \in \mathrm{SO}(3), \hat{\mathbf{p}}_m, \hat{\mathbf{v}}_m \in \mathbb{R}^3$ are respectively the state estimates of the rotation matrix, position and velocity of the robot, and $\hat{\mathbf{b}}_{gm} \in \mathbb{R}^3, \hat{\mathbf{b}}_{am} \in \mathbb{R}^3$ are the IMU gyroscope and accelerometer biases.

C. Continuous-Time MAP Optimization

The core of our system lies in solving the MAP optimization problem with the following cost function:

$$f(\hat{\mathcal{X}}) = \sum_{m=w}^{k-1} \left\| r_{\mathcal{I}}(\mathcal{I}_m, \hat{\mathcal{X}}_m, \hat{\mathcal{X}}_{m+1}) \right\|^2_{\Sigma_{\mathcal{I}}} + \sum_{m=w}^{k-1} \sum_{\mathcal{L}\in\mathcal{A}_m} \left\| r_{\mathcal{L}}(\mathcal{L}({}^{B_{t_s}}\mathbf{f}, \mathbf{n}, \mu, s), \hat{\mathcal{X}}_m, \hat{\mathcal{X}}_{m+1}) \right\|^2_{\Sigma_{\mathcal{L}}}, \tag{2}$$

where:

- $r_{\mathcal{I}}$ is the cost factor of the preintegration $\mathcal{I}_m$ constructed from the IMU samples in the interval $[t_m, t_{m+1}]$, and $\Sigma_{\mathcal{I}}$ is the corresponding covariance matrix;
- $r_{\mathcal{L}}$ denotes a lidar PTS factor based on a tuple of PTS association coefficients $\mathcal{L}({}^{B_{t_s}}\mathbf{f}, \mathbf{n}, \mu, s)$, and $\Sigma_{\mathcal{L}}$, a covariance, which is scalar in this case;
- $\mathbf{n}$, μ are the normal and mean of the underlying associated surfel (Section II-D1), and s is the normalized time stamp of raw lidar point ${}^{B_{t_s}}\mathbf{f}$, which is defined in (3).
- Finally $\mathcal{A}_m$ denotes the set of successful PTS associations in $[t_m, t_{m+1}]$ (more details are given in III-D).

It should be noted that each point ${}^{B_{t_s}}\mathbf{f}$ can be associated with multiple surfels at different scales, as long as the surfel satisfies the association predicates in Section III-D.

We refer to our method as continuous-time LIOM based on the use of continuous-time factor $r_{\mathcal{L}}(\mathcal{L}({}^{B_{t_s}}\mathbf{f}), \hat{\mathcal{X}}_m, \hat{\mathcal{X}}_{m+1})$, which is based on raw lidar point ${}^{B_{t_s}}\mathbf{f}$. Specifically, the observation ${}^{B_{t_s}}\mathbf{f}$ is coupled with an intermediary state $\hat{\mathcal{X}}_{t_s} \triangleq (\hat{\mathbf{R}}_{t_s}, \hat{\mathbf{p}}_{t_s})$ defined as a linear interpolation of $\hat{\mathcal{X}}_m, \hat{\mathcal{X}}_{m+1}$:

$$\hat{\mathbf{R}}_s = \hat{\mathbf{R}}_m \mathrm{Exp}\left(s\hat{\phi}_m\right), \ \hat{\mathbf{p}}_s = (1-s)\hat{\mathbf{p}}_m + s\hat{\mathbf{p}}_{m+1},$$
$$\hat{\phi}_m \triangleq \mathrm{Log}(\hat{\mathbf{R}}_m^{-1}\hat{\mathbf{R}}_{m+1}), \ s \triangleq \frac{t_s - t_m}{t_{m+1} - t_m}. \tag{3}$$

This categorization of continuous-time LIOM based on raw pointcloud is consistent with previous works, which may use linear interpolation [20], [21], or B-Spline interpolation [7], [14]. In contrast, discrete-time methods use the deskewed point ${}^{B_{t_m}}\mathbf{f}$ based on IMU-propagated states [3], [4], [22].

D. Surfel and UFOMap

1) Surfel's Attributes: A node i in the octree-based UFOMap corresponds to a voxel, and the node's depth indicates the voxel's scale. For leaf nodes, they are assigned depth 0, its parent has depth 1, and so forth. Fig. 2 illustrates the hierarchical relationship of the surfels/voxels on an octree at different scales. A surfel at node i in the UFOMap is defined by the set of points contained within the voxel, denoted as $\mathcal{V}_i = \{\mathbf{f}_1, \dots \mathbf{f}_{N_i}\}$, and has the following attributes:

$$N_i = |\mathcal{V}_i|, \ \mathbf{S}_i \triangleq \sum_{k=1}^{N_i} \mathbf{f}_k, \ \mathbf{C}_i \triangleq \sum_{k=1}^{N_i} \mathbf{f}_k\mathbf{f}_k^\top - \frac{1}{N_i}\mathbf{S}_i\mathbf{S}_i^\top. \tag{4}$$

Each surfel in UFOMap stores the values N_i, $\mathbf{S}_i$, $\mathbf{C}_i$. Given these attributes, we can quickly compute the mean μ_i and covariance Γ_i of $\mathcal{V}_i$, along with other quantities as follows:

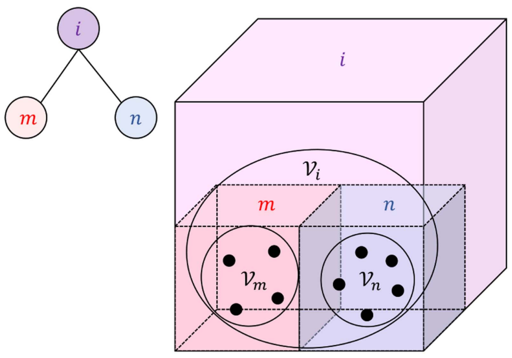


Fig. 2. Illustration of surfel and the tree-like structure: A voxel i contains a set of points $\mathcal{V}_i$, which (in this example) contains two subsets $\mathcal{V}_m, \mathcal{V}_n$, encapsulating the points in the voxels m, n at smaller scales. Each of the sets $\mathcal{V}_i, \mathcal{V}_m, \mathcal{V}_n$ define a surfel whose attributes are described in Section II-D1. As $\mathcal{V}_m \subset \mathcal{V}_i, \mathcal{V}_n \subset \mathcal{V}_i$, there is a hierarchical relationship between the surfels, which we propose to organize in an octree implemented by the UFOMap framework.

$$\mu_i \triangleq \frac{1}{N_i}\mathbf{S}_i, \ \Gamma_i \triangleq \frac{1}{N_i - 1}\mathbf{C}_i, \ \lambda_0 \le \lambda_1 \le \lambda_2,$$
$$\rho_i = 2\frac{\lambda_1 - \lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \ \mathbf{n}_i \triangleq \nu_0, \ c_i \triangleq -\nu_0^\top \mu_i.$$

where μ_i is the mean, Γ_i is the covariance with the eigenvalues $\lambda_0, \lambda_1, \lambda_2$ and $\mathbf{n}_i = \nu_0$ is the normalized eigenvector associated with λ_0; ρ_i is the so-called *planarity* value, i.e. the plane-likeness metric of the surfel.

Besides the aforementioned attributes, the *depth* and *scale* of the voxel node are also frequently queried. To initialize a surfel map in the UFOMap framework, we assign a *leaf node size*, denoted ℓ, for the voxels at the smallest scale, i.e., at depth 0. The scale of a voxel at depth D is therefore $2^D\ell$.

2) Surfel Addition: Assuming that a parent node i has two children m and n, the surfel attributes of i can be calculated via Welford's formula [23]:

$$\alpha := 1/\left[N_m N_n (N_m + N_n)\right], \ \beta := N_n\mathbf{S}_m - N_m\mathbf{S}_n,$$
$$\mathbf{C}_i \leftarrow \mathbf{C}_m + \mathbf{C}_n + \alpha\beta\beta^\top,$$
$$N_i \leftarrow N_m + N_n, \ \mathbf{S}_i \leftarrow \mathbf{S}_m + \mathbf{S}_n, \tag{5}$$

The above can be iterated for parent nodes with more than two children.

For single point update, i.e. the update at the leaf nodes when new pointcloud is added to the UFOMap, we can still use (5) by noticing that when $N_n = 1$, $\mathbf{C}_n = 0$.

III. System Description

Fig. 3 provides an overview of our system. In the next subsections we describe in details each numbered block.

A. Synchronization

Synchronization is a prerequisite to optimization based estimation methods. In this section we will describe our synchronization scheme in details. To better understand the method,

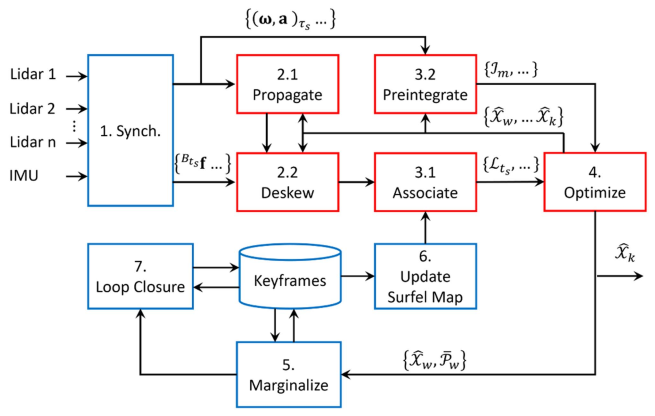

$\{(\boldsymbol{\omega}, \mathbf{a})_{\tau_s} \dots\}$
Lidar 1
Lidar 2
Lidar n
IMU
1. Synch.
2.1 Propagate
3.2 Preintegrate
$\{\mathcal{J}_m, \dots\}$
$\{\hat{\mathcal{X}}_w, \dots \hat{\mathcal{X}}_k\}$
$\{{}^{B_{t_s}}\mathbf{f} \dots\}$
2.2 Deskew
3.1 Associate
$\{\mathcal{L}_{t_s}, \dots\}$
4. Optimize
$\hat{\mathcal{X}}_k$
7. Loop Closure
Keyframes
6. Update Surfel Map
5. Marginalize
$\{\hat{\mathcal{X}}_w, \bar{\mathcal{P}}_w\}$

Fig. 3. The general workflow of the system. More details are given in Sections III-A–III-I.

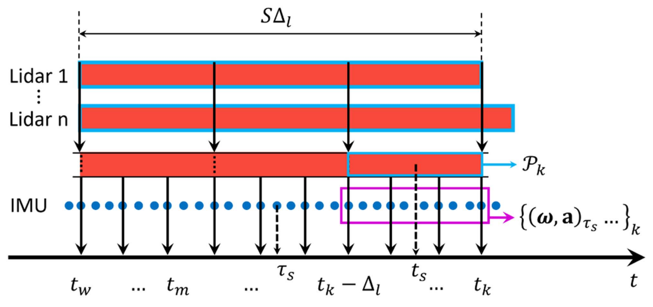


Fig. 4. Synchronization scheme on the sliding window. Detailed description is given in Section III-A.

please refer to Fig. 4 for the illustrations of the concepts introduced in this section.

The sliding window's progression is based on the *end time* of lidar messages from a so-called *primary* lidar. Thus, the end time t_k of the sliding window is also the end time of the latest scan from the primary lidar. Given Δ_l as the sweeping period of the primary lidar (typically 0.1 s), we create a so-called *window size* parameter, denoted S, for the number of lidar scans that the sliding window spans across. Hence, $t_k - t_w = S\Delta_l$, where $[t_w, t_k]$ is the time period of the sliding window introduced in Section II-B.

The messages from each lidar are stored in an ordered buffer such that data from each lidar can be treated as a continuous stream. As the window shifts from period $[t_w - \Delta_l, t_k - \Delta_l]$ to $[t_w, t_k]$, we extract data points from all the lidar streams in the period $[t_k - \Delta_l, t_k)$ and merge them into a unified pointcloud $\mathcal{P}_k$. When a new lidar pointcloud is admitted to the sliding window, we also add one or more state estimates to the sliding window. Thus the number of pointclouds can be smaller than the number of state estimates, which is different from our previous work MILIOM [4]. The addition of more than one state estimates per newly added pointcloud is to better capture the dynamics of the motion during the scan period Δ_l [24]. In practice we add 2 to 8 new state estimates to the sliding window, depending on the frequency of the IMU.

Similar to lidar, IMU samples are also stored in a buffer, and when the window slides forward for Δ_l seconds, the IMU samples in the periods $[t_k - \Delta_l, t_k]$ are also extracted and put onto the sliding window. Note that two interpolated IMU samples at the ends of each closed interval $[t_m, t_{m+1}]$ are also included. These IMU samples are used for propagation and preintegration processes in Sections III-B and III-E.

B. IMU Propagation

Given the IMU samples $\{(\boldsymbol{\omega}, \mathbf{a})_{\tau_i}, \dots\}, \tau_i \in [t_m, t_{m+1}]$, and the starting state $\check{\mathcal{X}}_m := \hat{\mathcal{X}}_m$, we can forward propagate the robot state to $\check{\mathcal{X}}_{m+1}$, or backward propagate from $\hat{\mathcal{X}}_{m+1}$ to $\check{\mathcal{X}}_m$. After this process we have a sequence of IMU-propagated states $\{\check{\mathcal{X}}_{\tau_i} \dots\}, \tau_i \in [t_m, t_{m+1}]$. This sequence can be used to initialize the new state estimate added to the sliding window, or to reduce the motion-induced distortion of the pointcloud. We explain this so-called *deskew* process in the next part.

C. Deskew (Motion Compensation)

For each lidar point ${}^{B_{t_s}}\mathbf{f}$, $t_s \in [t_m, t_{m+1}]$ we seek to transform its coordinate to the body frame at time t_{m+1}. To this end, we search for the two IMU-propagated poses closest to t_s, i.e. $\check{\mathbf{T}}_{\tau_a}, \check{\mathbf{T}}_{\tau_b}$, where $\tau_a \leq t_s \leq \tau_b$, and find the linearly interpolated pose $\check{\mathbf{T}}_{t_s}$:

$$\check{\mathbf{T}}_{t_s} = \begin{bmatrix} \mathrm{slerp}(\check{\mathbf{R}}_{\tau_a}, \check{\mathbf{R}}_{\tau_b}, u) & (1-u)\mathbf{p}_{\tau_a} + u\mathbf{p}_{\tau_b} \\ 0 & 1 \end{bmatrix}, \tag{6}$$

where $u \triangleq (t_s - \tau_a)/(\tau_b - \tau_a)$ and slerp() denotes the spherical linear interpolation operation on $\mathrm{SO}(3)$.

Given $\check{\mathbf{T}}_{t_s}$, we can transform ${}^{B_{t_s}}\mathbf{f}$ to the world frame by ${}^{W}\mathbf{f} = \check{\mathbf{R}}_{t_s}{}^{B_{t_s}}\mathbf{f} + \hat{\mathbf{p}}_{t_s}$. Hence we proceed to associating these lidar points with the surfel map.

D. Point-to-Surfel Association

The association consists of two stages. In the first stage, for each *deskewed* lidar point ${}^{W}\mathbf{f}$ we find all the nodes $\mathcal{V}_i$ matching the following predicates in the surfel map:

- The voxel depth is between 1 and $D_{\max}$ (the leaf nodes are ignored as the input pointcloud is downsampled with a spatial spacing of ℓ).
- $N_i \geq N_{\min}$ and $\rho_i > \rho_{\min}$, i.e. $\mathcal{V}_i$ should have at least $N_{\min}$ points and the planarity is sufficiently large.
- The voxel's cube intersects with a sphere of radius $r > 0$ centered at ${}^{W}\mathbf{f}$, where r is a user-defined parameter.

If a surfel $\mathcal{V}_i$ passes all of the aforementioned predicates, we can associate it with the point ${}^{W}\mathbf{f}$. The associated surfels are organized into groups by their depths. For each point ${}^{W}\mathbf{f}$, starting from the group with smallest depth (smallest scale), we find the surfel $\mathcal{V}_i$ with the shortest distance to ${}^{W}\mathbf{f}$ in that group, i.e. $d_i = \mathbf{n}_i^\top({}^{W}\mathbf{f} - \mu_i)$. If $d_i < d_{\max}$, a tuple of PTS coefficients $\mathcal{L} = ({}^{t_s}\mathbf{f}, \mathbf{n}_i, \mu_i, s)$ (s defined in (3)) will be added to the set of successful association $\mathcal{A}_m$ (defined in (2)), and the surfels at higher scales are ignored. If no suitable surfel is found at one scale, we proceed to check on the next scale. Note that the association is done on the deskewed point ${}^{W}\mathbf{f}$, but the factor is based on the original raw point ${}^{t_s}\mathbf{f}$. Given $\mathcal{L}$, the cost factor $r_{\mathcal{L}}$ in (2) can be constructed using the point-to-plane relationship:

$$r_{\mathcal{L}} = \mathbf{n}^{\top}(\hat{\mathbf{R}}_{t_s}{}^{t_s}\mathbf{f} + \hat{\mathbf{p}}_{t_s} - \mu), \tag{7}$$

where $(\hat{\mathbf{R}}_{t_s}, \hat{\mathbf{p}}_{t_s})$ is the linearly interpolated pose in (3).

E. IMU Preintegration

We refer to our previous work [4], [25] for the detailed formulation of the IMU-preintegration observation $\mathcal{I}_m$, the factors $r_{\mathcal{I}}$ and its Jacobian in optimization-based estimation.

F. Optimization

Once all of the PTS coefficients and IMU preintegrations have been prepared, we proceed to construct and optimize the cost function (2) using the ceres solver [26]. It should be noted that the steps described in Sections III-B to III-F (red boxes in Fig. 3) can be done iteratively.

G. Marginalization

After each optimization step, we check if the earliest pose estimate $\hat{\mathbf{T}}_w$ can be marginalized to become a keyframe. To this end, we search for five nearest neighbours of $\hat{\mathbf{T}}_w$ among the existing keyframes. If the Euclidean distance or rotational distance of $\hat{\mathbf{T}}_w$ to all of its neighbours exceeds a threshold, its associated deskewed pointcloud will be inserted to the surfel map. We also marginalize $\hat{\mathbf{T}}_w$ and store its deskewed pointcloud in the buffer to be reused in case of loop closure.

H. Updating the Surfel Map

When a new keyframe is admitted to the buffer, we also update the surfel map using the procedure that was introduced in Section II-D2.

I. Loop Closure

When a new keyframe $\hat{\mathbf{T}}_c$ is admitted to the buffer, we will find K nearest keyframes of $\hat{\mathbf{T}}_c$. If the time stamp of one keyframe $\hat{\mathbf{T}}_p$ and $\hat{\mathbf{T}}_m$ differs by a certain amount, it signals the return to a previously explored area. We use the ICP to calculate the relative pose ${}^{p}\bar{\mathbf{T}}_c$ between the two keyframes and store this in a buffer $\mathcal{T}$. When a new loop is admitted, we optimize the pose graph with this loop prior:

$$\begin{aligned}\min_{\hat{\mathbf{T}}_0,\dots\hat{\mathbf{T}}_c} &\sum_{k=0}^{K-1} \left\| r_1\left({}^{k}\bar{\mathbf{T}}_{k+1}, \hat{\mathbf{T}}_k^{-1}\hat{\mathbf{T}}_{k+1}\right)\right\|^2_{\Sigma_1} \\ &+ \sum_{{}^{p}\bar{\mathbf{T}}_c \in \mathcal{T}} \left\| r_2\left({}^{p}\bar{\mathbf{T}}_c, \hat{\mathbf{T}}_p^{-1}\hat{\mathbf{T}}_c\right)\right\|^2_{\Sigma_2},\end{aligned} \tag{8}$$

where ${}^{k}\bar{\mathbf{T}}_{k+1}$, ${}^{p}\bar{\mathbf{T}}_c$ are the relative pose priors, and $\mathcal{T}$ is the set of relative poses detected. After the pose graph optimization, the global map will be recomputed.

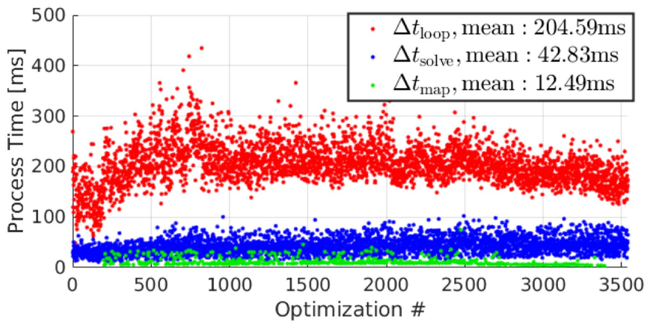


Fig. 5. Process time of SLICT in NTU VIRAL rtp_03.

IV. Experiments

We demonstrate the performance of SLICT via three data suites: NTU VIRAL [27], Newer College [28] and our in-house datasets, which respectively capture operations at different scales. We compare our methods with four other state-of-the-art (SOTA) methods. First is MARS [14], which stands for Multi-Adaptive-Resolution-Surfel with B-spline-based continuous-time optimization. Second is LIO-SAM [3], which uses conventional plane-edge feature and IMU preintegration factors, optimized by the Georgia Tech Smoothing and Mapping framework [29]. Third is VoxelMap [18], which features a probabilistic adaptive voxel mapping strategy. Fourth is FAST-LIO2, which uses the direct method for feature association and the ikd-Tree global mapping scheme. Video recording of some experiments can be viewed at the github page of SLICT.

A. NTU VIRAL Datasets

The NTU VIRAL is a multi-lidar dataset collected from an Unmanned Aerial Vehicle (UAV), with ground truth of centimeter-level accuracy obtained from a laser-tracker total station. The environments consist of indoor and outdoor spaces within a volume of 50 m radius.

For all experiments, we merge the data from the so-called horizontal and vertical lidars in the NTU VIRAL dataset and use them as the input of all methods. All experiments are run on the same computer with a Core i7-12700KF CPU. The metric used in this case is the Absolute Trajectory Error (ATE).

In these experiments we do not enable the loop closure function of SLICT and only compare the odometry result. We set sensor's parameters such as number of lines, minimum distance, IMU noises, etc, according to the dataset's meta data, while other parameters are kept as default. For SLICT, we choose a sliding window of 400 ms with 16 intervals, which encompass 4 pointclouds with 4 state estimates each.

Table I reports the result of our experiments. SLICT has the highest accuracy in the most experiments, and FAST-LIO2 has the most second best results. MARS, LIO-SAM and VoxelMap diverge in some of the experiments.

We analyze the computational load of SLICT in the NTU VIRAL rtp_02 sequence in Fig. 5. On average, it takes 205 ms to complete one cycle of the algorithm (Δt_{loop}, defined as the time from one optimization operation to another), in which solving

TABLE I
ATE of SLICT Compared With Other Methods on NTU VIRAL Datasets (Unit [m]). The Best Results are in **bold**, Second Best are underlined. 'x' Denotes Divergence

Dataset	MARS	LIO-SAM	Voxel-Map	FAST-LIO2	SLICT
eee_01	0.2471	0.0624	0.0699	<u>0.0585</u>	**0.0316**
eee_02	0.1033	0.0457	0.0506	<u>0.0318</u>	**0.0249**
eee_03	0.0927	0.0403	0.0631	<u>0.0351</u>	**0.0275**
nya_01	0.0555	2.0960	0.0508	<u>0.0305</u>	**0.0229**
nya_02	0.0624	x	0.0425	<u>0.0286</u>	**0.0227**
nya_03	0.0831	0.0468	0.0494	<u>0.0315</u>	**0.0260**
sbs_01	0.1370	0.0444	0.0535	<u>0.0324</u>	**0.0298**
sbs_02	0.1256	0.0461	0.0525	<u>0.0322</u>	**0.0291**
sbs_03	0.1588	0.0494	0.0498	<u>0.0428</u>	**0.0335**
rtp_01	x	0.2571	9.7416	<u>0.0494</u>	**0.0447**
rtp_02	0.2329	<u>0.1091</u>	2.4479	0.1151	**0.0466**
rtp_03	0.1377	0.0576	0.0792	<u>0.0543</u>	**0.0501**
tnp_01	0.0734	x	<u>0.0326</u>	0.0432	**0.0287**
tnp_02	0.0681	0.0330	<u>0.0247</u>	0.0590	**0.0201**
tnp_03	0.0665	**0.0283**	<u>0.0331</u>	0.0468	0.0383
spms_01	x	0.1620	11.3792	<u>0.0686</u>	**0.0610**
spms_02	x	0.6641	x	**0.0821**	<u>0.1000</u>
spms_03	19.8650	1.0071	x	**0.0603**	<u>0.0661</u>

TABLE II
ATE of SLICT and Other Methods on Newer College Dataset (Unit [m]). The Best Results are in **bold**, Second Best are underlined. 'x' Demotes a Divergent Experiment

Dataset	MARS	LIO-SAM	Voxel-Map	FAST-LIO2	SLICT
01_short_exp	2.1521	-	x	<u>0.3883</u>	**0.3843**
02_long_exp	6.0030	-	x	<u>0.3659</u>	**0.3496**
05_quad_dynamics	0.3729	-	17.0007	<u>0.3443</u>	**0.1155**
06_dynamic_spin	x	-	19.9993	**0.0800**	<u>0.0844</u>
07_parkland	x	-	x	<u>0.1356</u>	**0.1290**

the optimization problem takes about 43 ms (Δt_{solve}), and the rest is for deskew, association, and keyframe marginalization. When a new keyframe is created, the time to update the global map (Δt_{map}) is 13 ms on average. Since lidar input is acquired at 100 ms, currently real-time performance is not guaranteed by SLICT. Because we associate the points with surfels at five scales (from $2^1\ell$ to $2^5\ell$, where $\ell = 0.1\,m$), the computation load for association is at least a multitude that of direct method, which uses only one voxel scale. However, we think it is justifiable considering that SLICT gives higher accuracy, and real-time performance can be achieved by using a CPU that supports more threads.

B. Newer College Dataset

The Newer College Dataset consists of five sequences collected at New College, Oxford, featuring an Ouster lidar with 64 channels, 90-degree vertical field-of-view, and a built-in 100 Hz IMU. The ground truth is obtained from ICP-based registration of lidar scan with a centimeter-resolution map captured by a 3D scanner. The environment captured in the dataset features two open squares and an open park over an area of roughly 200 m $\times$ 100 m, which is several times larger than those in NTU VIRAL sequences, thus localization drift can be observed more easily.

The settings are similar to that in the NTU VIRAL dataset, however for SLICT we assign 2 state estimates for each interval since the built-in IMU of the Ouster lidar has a frequency of only 100 Hz. Note that LIO-SAM does not work in this case as it requires orientation estimate from the IMU.

In Table II, we report the ATE of the results. Again, it shows that SLICT has the best results in the most experiments, followed closely by FAST-LIO2. MARS's ATE in sequence 01, 02, 05 is similar to the reported result in [14]. As the Newer College sequences 01, 02 and 07 traverse a significantly larger area than the NTU VIRAL dataset, the ATE increases more visibly compared to the NTU VIRAL experiments. Fig. 6 shows the localization error over time of different methods in sequence 01.

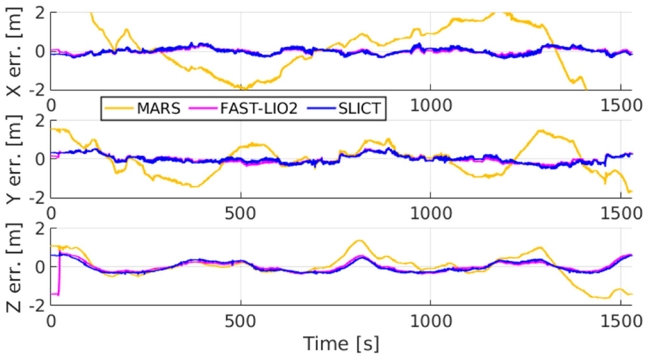


Fig. 6. Estimation error over time of the methods in the Newer College Dataset, sequence 01. We find that the error is most significant in the z direction due to the subtle changes in elevation of the area.

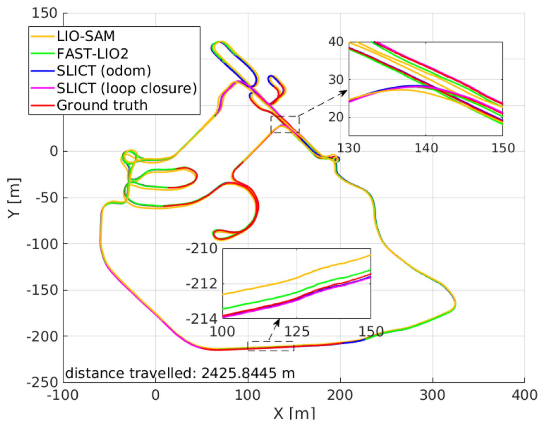
C. In-House Dataset

The in-house datasets are collected from a sensor suite consisting of an OS1-128 Lidar, an Livox Mid-70, together with a VectorNav100 IMU. The sensors are mounted on an All-Terrain-Vehicle (ATV) that travels around 1.5 km loop at the southern part of Nanyang Technical University (NTU) Campus. This covers an area of about 400 m $\times$ 200 m, which is about four times the area of the Newer College dataset. At this scale, the effect of a loop closure module can become more appreciable, thus we add new experiments of SLICT with loop closure and pose-graph optimization functions.

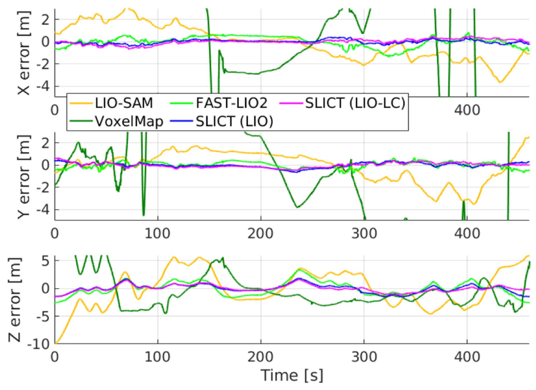
The ground truth of the dataset is constructed in a similar manner to the Newer College Dataset. Specifically, we first build a static map of the environment using the survey scanner Leica RTC360 with centimeter accuracy. We then register the ouster lidar scans with this static map to obtain the ground truth pose at the lidar frequency.

Three sequences are captured. In sequence 1, we traverse the main routes of the environments. In sequence 2, we increase the traversed distance by revisiting some of the routes. In sequence 3, we travel back and forth on the Nanyang Link route to maximize the loop closure incidents. Fig. 7 provides an overview of the routes taken.

Similar to previous experiments, we directly merge the lidar sensors into a single input and use it for all methods. Table III presents the ATE of the tested methods. Due to



(a) Sequence 2



(b) Localization error in sequence 2

Fig. 7. Localization estimates and its error in sequence 2 of the in-house dataset. The zoomed-in plots show the narrow path in the dense-vegetation section (sequences 01 and 02), and the main junction (sequence 03). The elevation difference between these two points are about 13 m.

TABLE III
ATE of SLICT and Other Methods on In-House Dataset (Unit [m]). The Best Results Are in **Bold**, Second Best Are Underlined. 'x' Demotes a Divergent Experiment. LC Denotes Experiments With Loop-Closure and Pose-Graph Optimization

Dataset	**LIO-SAM**	**Voxel-Map**	**FAST-LIO2**	**SLICT**	**LIO-SAM (LC)**	**SLICT (LC)**
seq_01	4.0678	7.8550	<u>1.7658</u>	**1.0778**	<u>1.2931</u>	**0.7437**
seq_02	3.8518	x	<u>1.2244</u>	**0.7372**	<u>0.9685</u>	**0.5401**
seq_03	x	9.5255	<u>1.1653</u>	**0.5789**	x	**0.6226**

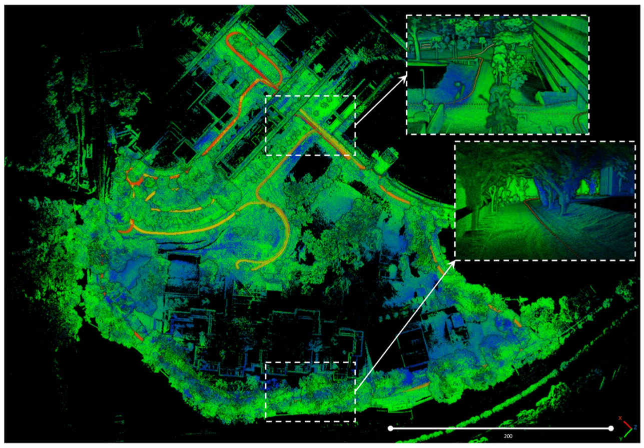


Fig. 8. The global map built by SLICT from one experiment. The zoom-in figures show the junction and area with dense vegetation in Fig. 7.

the high-speed of the ATV (up to 30 km/h), MARS diverges in all three experiments, hence is not included in Table III, and LIO SAM diverges in the last experiment. Without loop closure, SLICT still has the best accuracy, and the loop closure improves the accuracy even further. We also added result of LIO-SAM with loop closure for comparison.

Since the change in elevation of the environment is significant (up to 15 m between the highest and lowest points) the localization error is also significant, most visibly in the z dimension (Fig. 7(b)). Compared to the Newer College dataset in Fig. 6, we can see that the error in z direction is now almost doubled, which is the main contribution to the ATE in Table III. In Fig. 8, we present some views of the global map built by SLICT in sequence 2.

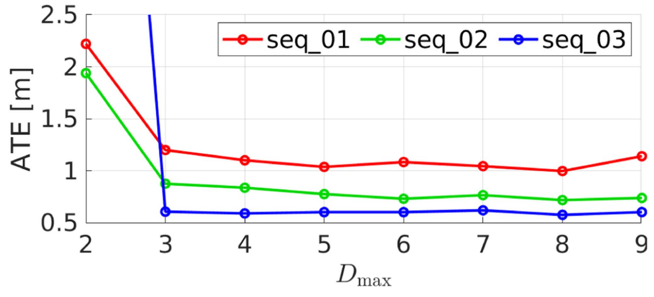


Fig. 9. Change in ATE of SLICT on the in-house datasets across the maximum depth settings.

D. Effect of the Maximum Association Depth

To better study the effect of the multi-scale association, we run SLICT with the in-house datasets at different maximum surfel depth $D_{\max}$ (Section III-D). Fig. 9 reports the ATE of these experiments with $D_{\max}$ ranging from 2 to 9. It can be seen that the ATE decreases until $D_{\max} = 5$ and varies little for $D_{\max} > 5$. As $\ell = 0.05\,\text{m}$, this means that the maximum voxel scale that can be associated is about $2^{D_{\max}}\ell = 1.6\,\text{m}$. This confirms the benefit of a multi-scale association approach.

E. Effect of the Number of State Estimates

We also study the effect of the number of state estimates per scan. Fig. 10 shows the ATE of SLICT on the in-house datasets with the number of state estimates per scan. We find that the optimal number of states for each Δ_l period is around 3 or 4 for the 400 Hz IMU (VectorNav100). However 4 is a sufficiently good choice for both the NTU VIRAL and in-house datasets. This confirms the benefits of having more than one state estimates for each lidar scan.

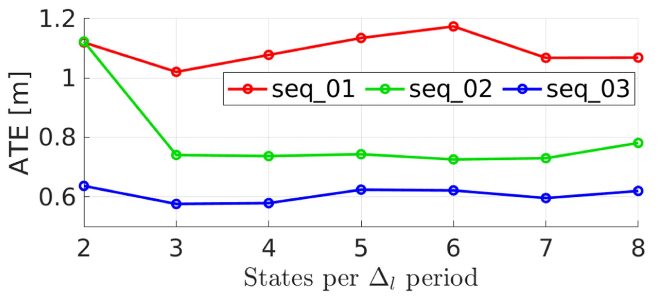


Fig. 10. Change in ATE of SLICT on the in-house datasets across different choices for number of states per scan.

V. Conclusion

In this letter we propose a full-fledged multi-input Lidar-Inertial Odometry and Mapping system called SLICT. SLICT features a multi-scale global map of surfels that can be updated incrementally using the UFOMap framework. We also propose a method to associate lidar point to surfel (PTS), and a continuous-time MAP optimization of PTS and IMU-preintegration factors. We demonstrate competitive performance on public datasets even without loop closure. To achieve a complete system, we added a simple yet effective loop closure mechanism and demonstrate its usefulness with our new datasets. The source code is released for the benefit of the community.

There remain many possibilities for extension of SLICT. For one, the PTS association still uses simple predicates, which can be made more efficient by some adaptive strategy. Moreover, the UFOMap framework allows one to integrate semantic information to the surfel, which has the potential to improve the association process. For the small drift, a basic loop closure mechanism sufficed, but more advanced methods can also be integrated in the future.

References

[1] C. Cadena et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[2] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," *Robot.: Sci. Syst.*, vol. 2, no. 9, pp. 1–9, 2014.

[3] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled Lidar inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

[4] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, "MILIOM: Tightly coupled multi-input Lidar-inertia odometry and mapping," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5573–5580, Jul. 2021.

[5] P. Chen, W. Shi, S. Bao, M. Wang, W. Fan, and H. Xiang, "Low-drift odometry, mapping and ground segmentation using a backpack LiDAR system," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7285–7292, Oct. 2021.

[6] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-LOAM: Fast LiDAR odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4390–4396.

[7] 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.

[8] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021.

[9] 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.

[10] 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.

[11] 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. Int. Conf. Robot. Automat.*, 2022, pp. 10672–10678.

[12] 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.

[13] D. Wisth, M. Camurri, and M. Fallon, "VILENS: Visual, inertial, LiDAR, and leg odometry for all-terrain legged robots," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 309–326, Feb. 2023.

[14] J. Quenzel and S. Behnke, "Real-time multi-adaptive-resolution-surfel 6D LiDAR odometry using continuous-time trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5499–5506.

[15] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-D range sensor with application to mobile mapping," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1104–1119, Oct. 2012.

[16] M. Bosse and R. Zlot, "Continuous 3D scan-matching with a spinning 2D laser," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 4312–4319.

[17] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-LIO: Lightweight tightly coupled LiDAR-inertial odometry using parallel sparse incremental voxels," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4861–4868, Apr. 2022.

[18] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8518–8525, Jul. 2022.

[19] D. Duberg and P. Jensfelt, "UFOMap: An efficient probabilistic 3D mapping framework that embraces the unknown," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6411–6418, Oct. 2020.

[20] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic LiDAR fusion: Dense map-centric continuous-time SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1206–1213.

[21] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP: Real-time elastic LiDAR odometry with loop closure," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 5580–5586.

[22] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3D LiDAR inertial odometry and mapping," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 3144–3150.

[23] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[24] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3D LiDAR-based online mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5000–5007.

[25] T.-M. Nguyen, M. Cao, S. Yuan, Y. Lyu, T. H. Nguyen, and L. Xie, "Viral-fusion: A visual-inertial-ranging-LiDAR sensor fusion approach," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 958–977, Apr. 2022.

[26] S. Agarwal and K. Mierle, "Ceres solver: Tutorial & reference," [Online]. Available: http://ceres-solver.org/

[27] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, "NTU viral: A visual-inertial-ranging-LiDAR dataset, from an aerial vehicle viewpoint," *Int. J. Robot. Res.*, vol. 41, no. 3, pp. 270–280, 2022.

[28] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The newer college dataset: Handheld LiDAR, inertial and vision with ground truth," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 4353–4360.

[29] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-RIM-CP&R-2012-002, 2012.