2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Efficient Multi-sensor Aided Inertial Navigation with Online Calibration

Woosik Lee, Yulin Yang, and Guoquan Huang

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561254

***Abstract*— In this paper, we design a versatile multi-sensor aided inertial navigation system (MINS) that can efficiently fuse multi-modal measurements of IMU, camera, wheel encoder, GPS, and 3D LiDAR along with online spatiotemporal sensor calibration. Building upon our prior work [1]–[3], in this work we primarily focus on efficient LiDAR integration in a sliding-window filtering fashion. As each 3D LiDAR scan contains a large volume of 3D points which poses great challenges for real-time performance, we advocate using plane patches, which contain the environmental structural information, extracted from the sparse LiDAR point cloud to update/calibrate the system efficiently. The proposed LiDAR plane patch processing algorithm (including extraction, data association, and update) is shown to be efficient and consistent. Both Extensive Monte-Carlo simulations and real-world datasets with large-scale urban driving scenarios have been used to verify the accuracy and consistency of the proposed MINS algorithm.**

I. Introduction

Online localization is a fundamental prerequisite of autonomous vehicles. Many algorithms have been developed thus far to achieve a high-precision consistent 3D localization using different sensors. Multi-sensor fusion is often used to achieve this goal for a number of reasons including more reliable data outcomes, more coverage, applicability, and lower equipment cost though at a higher computation cost. Among all possible navigation sensors, IMU, camera, wheel encoder, GPS, and 3D LiDAR are appealing because of their sufficient information for 3D motion estimation and good accessibility to commercial products. While it appears to be straightforward in principle to fuse all of these sensors in order to achieve good localization performance, few work has shown to fuse sensors more than three types due to their different characteristics, increasing computation, asynchronicity, and calibration issues. Moreover, accurate online multi-sensor calibration is essential for optimal sensor fusion as it may change over time during navigation. As such, in this work, we develop an efficient multi-sensor aided inertial navigation system, *MINS*, an INS aided by multi-modal sensors including camera, wheel encoder, GPS, and 3D LiDAR with the online calibration of all involving sensors while taking into account their asynchronous nature, and achieving robust accurate real-time localization performance.

When fusing these multi-modal sensors, LiDAR integration cannot be overemphasized due to its large information carried by the point clouds. Since 3D LiDAR sensors can provide more than two million points per second (e.g., HDL-64E [4]), naive fusion of all these data points likely may not be real-time. Unlike visual measurements, finding associated points between different LiDAR scans is hard because there might be no points hitting exactly same physical locations. We thus advocate to use plane patches, which contain the most dominant structural information of the point clouds and are convenient for robust data association, extracted from LiDAR point clouds to update the states and calibrate the spatiotemporal parameters between LiDAR and IMU. In particular, based on our prior work [1]–[3], we develop *MINS*, an real-time, consistent, tightly-coupled, multi-sensor aided INS estimator with efficient LiDAR plane patch tracking, while performing online spatiotemporal calibration between all the sensors. Specifically, the main contributions of this work include:

- We design *MINS*, a general real-time MSCKF [5] - based multi-sensor aided INS estimator that optimally and efficiently fuses measurements from IMU, camera, wheel encoders, GPS, and 3D LiDAR.
- We develop an efficient LiDAR feature tracking algorithm to extract, merge, and track plane patches with proper uncertainty modeling from 3D LiDAR point clouds. These plane patches are leveraged for real-time MSCKF update with online calibration.
- We evaluate the proposed *MINS* extensively in realistic simulations, analyzing detailed LiDAR processing times, calibration convergence, and estimator consistency. The proposed method is also validated on large-scale urban driving datasets.

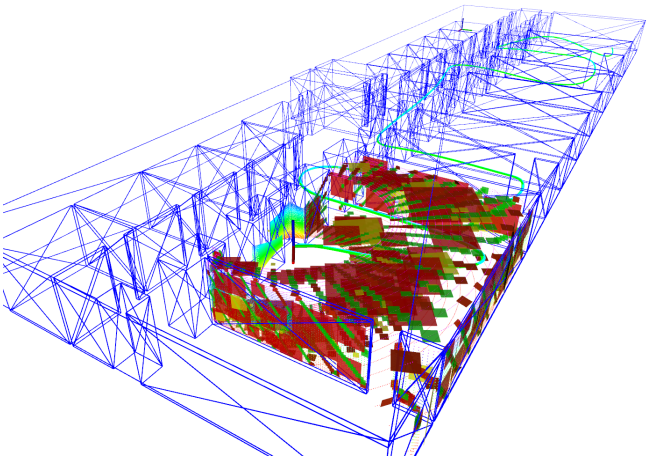

Fig. 1: The proposed *MINS* working in simulations. The green represent the plane patches extracted from the LiDAR point cloud, the red are the merged plane patches, and the yellow are the patches used to update the estimator. Green and light blue paths are the estimated and ground truth trajectories, respectively.

II. Related Work

The MSF-EKF [6] is among the first work to fuse generic relative and absolute measurements from IMU, camera, GPS,

This work was partially supported by the University of Delaware (UD) College of Engineering, the NSF (IIS-1924897), the ARL (W911NF-19-2-0226, W911NF-20-2-0098, JWS 10-051-003),

The authors are with the Robot Perception and Navigation Group (RPNG), University of Delaware, Newark, DE 19716, USA. {woosik,yuyang,ghuang}@udel.edu

and pressure sensor with online spatial extrinsic calibration. The whole processing time was bounded by a few hundred milliseconds. Hausman et al. [7] also fused IMU, camera, GPS, and ultra-wideband (UWB) measurements with extrinsic calibration within the EKF framework. However, both works can only handle a relatively small size of measurements and cannot address the temporal calibration between the sensors. Shen et al. [8] used the UKF to integrate IMU, camera, GPS, 2D LiDAR, pressure altimeter, and magnetometer without calibrating the sensor parameters and analyzing the processing times, while their recent VINS-Fusion [9] employs a loosely-coupled graph formulation to fuse IMU, camera, GPS, magnetometer, and barometer. Suhr et al. [10] fused IMU, camera, GPS, wheel along with symbolic road markings map based on a particle filter, while Meng et al. [11] fused IMU, GPS, Distance-Measuring Instruments (DMI), and LiDAR within the UKF framework. All these methods assumed known perfect calibration and have not studied computational complexity analysis.

Recently, there are research efforts primarily fusing IMU, LiDAR and camera with online calibration. For example, our prior work, LIC-Fusion 1.0 [12] and 2.0 [13] used planes extracted from LiDAR point clouds with online calibration. However, the sensors considered in that work do not include wheel encoders and GPS, and are limited to show its real-time performance through 16 channel LiDAR at 10 Hz with a smaller number of planes used for update. Our recent work [1] fused IMU, camera, and GPS with spatiotemporal calibration, [2], [14] combined IMU, camera, and wheel encoders with spatiotemporal calibration and wheel intrinsic calibration, while [15] fused IMU, Wheel, 2D LiDAR, and pre-built 2D LiDAR map within the MSCKF framework with online calibration. The summary of multi-sensor calibration algorithms is shown in Table I.

While rich literature exists on LiDAR processing, we only focus on the SLAM methods that integrate 3D LiDAR with other sensors in real-time. In particular, Xu et al. [16] used information filter to take advantage of not inverting large measurement covariance and fused IMU and LiDAR, the authors of [17]–[19] extracted LOAM [20] features from LiDAR measurements and fused with other sensors, while Zhang et al. [21] introduced a fast plane segmentation and map refinement algorithm that can save computation time and improve map quality. Pathak et al. [22] extracted planes from incoming point clouds and found correspondence efficiently leading to improved efficiency. Shan et al. [23] fused IMU with LiDAR in a graph formulation, maintaining only local LiDAR scans within a sliding window to ensure real-time performance of the system, and Maddern et al. [24] fused the stereo visual information and LiDAR point cloud within the camera field-of-view to improve image disparity estimation. While significant research efforts recently have also focused on deep learning [25]–[28], it is unclear how efficient and generalizable these data-driven approaches are.

TABLE I: Sensor usage/calibration comparison of multi-sensor systems.

Systems	IMU	Cam	GPS	Wheel	LiDAR	Other
MINS	O/O	O/O	O/O	O/O	O/O	
[6]	O/O	O/O	O/O	/	/	Press
[7]	O/O	O/O	O/O	/	/	UWB
[8]	O/	O/	O/	/	O/	Press,Mag
[9]	O/	O/	O/	/	/	Baro,Mag
[10]	O/	O/	O/	O/	/	Map
[11]	O/	/	O/	/	O/	DMI
[12], [13]	O/O	O/O	/	/	O/O	
[2], [14]	O/O	O/O	O/O	/	/	
[1]	O/O	O/O	/	O/O	/	
[15]	O/O	/	/	O/O	O/O	Map

III. MSCKF-based Multi-sensor aided INS

Before introducing our LiDAR integration method, we present the MSCKF-based multi-sensor aided INS (*MINS*) fusing IMU, camera, GPS, and wheel encoder measurements based on our prior work [1]–[3]. Specifically, at time t_k, the state vector $\mathbf{x}_k$ consists of the current inertial state $\mathbf{x}_{I_k}$ and n historical IMU pose clones $\mathbf{x}_{C_k}$ captured at camera measurement times in the global frame $\{G\}$:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{I_k}^\top & \mathbf{x}_{C_k}^\top \end{bmatrix}^\top \tag{1}$$

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_k}^\top & {}^{G}\mathbf{v}_{I_k}^\top & \mathbf{b}_g^\top & \mathbf{b}_a^\top \end{bmatrix}^\top \tag{2}$$

$$\mathbf{x}_{C_k} = \begin{bmatrix} {}^{I_{k-1}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{k-1}}^\top & \cdots & {}^{I_{k-n}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{k-n}}^\top \end{bmatrix}^\top \tag{3}$$

where ${}^{I_k}_{G}\bar{q}$ is the JPL unit quaternion [29] corresponding to the rotation ${}^{I_k}_{G}\mathbf{R}$ from $\{G\}$ to IMU frame $\{I\}$, ${}^{G}\mathbf{p}_{I_k}$ and ${}^{G}\mathbf{v}_{I_k}$ are the position and velocity of $\{I\}$ in $\{G\}$, and $\mathbf{b}_g$ and $\mathbf{b}_a$ are the biases of the gyroscope and accelerometer. We define $\mathbf{x} = \hat{\mathbf{x}} \boxplus \tilde{\mathbf{x}}$, where $\mathbf{x}$ is the true state, $\hat{\mathbf{x}}$ is its estimate, $\tilde{\mathbf{x}}$ is the error state, and the operation $\boxplus$ which maps the error state vector to its corresponding manifold [30].

A. IMU Kinematic Model

The state is propagated forward in time using the IMU linear acceleration $\mathbf{a}_m$ and angular velocity $\boldsymbol{\omega}_m$ measurements:

$$\mathbf{a}_m = \mathbf{a} + {}^{I}_{G}\mathbf{R}\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a \ , \quad \boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \tag{4}$$

where $\mathbf{a}$ and $\boldsymbol{\omega}$ are the true local acceleration and angular velocity, $\mathbf{g} \approx [0\ 0\ 9.81]^\top$ is the global gravity, and $\mathbf{n}_a$ and $\mathbf{n}_g$ are zero mean Gaussian noises. We propagate the state estimate and covariance from time t_k to t_{k+1} based on the standard inertial kinematic model $\mathbf{f}(\cdot)$ [29] under the assumption of zero noise:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{a}_m, \boldsymbol{\omega}_m, \mathbf{0}, \mathbf{0}) \tag{5}$$

$$\mathbf{P}_{k+1|k} = \boldsymbol{\Phi}(t_{k+1}, t_k)\mathbf{P}_{k|k}\boldsymbol{\Phi}(t_{k+1}, t_k)^\top + \mathbf{Q}_k \tag{6}$$

where $\hat{\mathbf{x}}_{a|b}$ denotes the estimate at time t_a formed by processing the measurements up to time t_b, and $\boldsymbol{\Phi}$ and $\mathbf{Q}$ are the state transition matrix and discrete noise covariance.

B. Camera Measurement Model

Sparse corner features are detected and tracked over a window of images associated with the cloned frames $\mathbf{x}_{C_k}$. The resulting bearing measurements, $\mathbf{z}_k$, are given by:

$$\mathbf{z}_k = \boldsymbol{\Pi}({}^{C_k}\mathbf{p}_f) + \mathbf{n}_k \tag{7}$$

$${}^{C_k}\mathbf{p}_f = {}^{C}_{I}\mathbf{R}\,{}^{I_k}_{G}\mathbf{R}({}^{G}\mathbf{p}_f - {}^{G}\mathbf{p}_{I_k}) + {}^{C}\mathbf{p}_I \tag{8}$$

where $\boldsymbol{\Pi}\left([x\ y\ z]^\top\right) = \begin{bmatrix} \frac{x}{z} & \frac{y}{z} \end{bmatrix}^\top$ is the perspective projection, ${}^{G}\mathbf{p}_f$ is the 3D point feature, and $\{{}^{C}_{I}\mathbf{R}, {}^{C}\mathbf{p}_I\}$ are the camera to IMU extrinsic transformation. Stacking all

measurements corresponding to a single feature and performing linear marginalization of the feature's position (via the nullspace projection) results in a residual [5]:

$$\tilde{\mathbf{z}}_{c_k} = \mathbf{H}_{x_k}\tilde{\mathbf{x}}_k + \mathbf{n}_{f_k} \tag{9}$$

This then can be directly used in EKF update without storing features in the state, leading to substantial computational savings and bounding the state size.

C. Wheel Measurement Model

Instead of adding clones to the state every time when the wheel encoder reading comes in, as in [1], we integrate the measurement to get the relative pose measurement between two clone times in 2D (rotation ${}^{O_{k+1}}_{O_k}\theta$ and translation ${}^{O_k}\mathbf{d}_{O_{k+1}}$), which does not increase the state vector. With that, we define the following relative transformation between the 6DOF IMU clone states leveraging the wheel-IMU extrinsics $\{{}^O_I\mathbf{R}, {}^O\mathbf{p}_I\}$ as:

$$\begin{aligned} {}^{O_{k+1}}_{O_k}\theta &= \mathbf{e}_3^\top \mathrm{Log}({}^O_I\mathbf{R}\,{}^{I_{k+1}}_{G}\mathbf{R}\,{}^{I_k}_{G}\mathbf{R}^\top {}^O_I\mathbf{R}^\top) \\ {}^{O_k}\mathbf{d}_{O_{k+1}} &= \Lambda({}^O_I\mathbf{R}\,{}^{I_k}_{G}\mathbf{R}({}^G\mathbf{p}_{I_{k+1}} + {}^{I_{k+1}}_{G}\mathbf{R}^\top {}^I\mathbf{p}_O - {}^G\mathbf{p}_{I_k}) + {}^O\mathbf{p}_I) \end{aligned} \tag{10}$$

where $\Lambda = [\mathbf{e}_1\ \mathbf{e}_2]^\top$, $\mathbf{e}_i$ is the i-th standard unit basis vector, and $\mathrm{Log}(\cdot)$ is the $SO(3)$ matrix logarithm function [31].

D. GPS Measurement Model

Besides the visual/wheel measurement update, whenever a new GPS measurement in the ENU frame $\{E\}$ is available, we use it to update the state as in [2]. In particular, the GPS measurement ${}^E\mathbf{p}_{gps_k}$ at timestep k is:

$$\mathbf{z}_{g_k} := {}^E\mathbf{p}_{gps_k} = {}^E\mathbf{p}_G + {}^E_G\mathbf{R}\,{}^G\mathbf{p}_{gps_k} + \mathbf{n}_{g_k} \tag{11}$$

$${}^G\mathbf{p}_{gps_k} = {}^G\mathbf{p}_{I_k} + {}^{I_k}_{G}\mathbf{R}^\top {}^I\mathbf{p}_{gps} \tag{12}$$

where ${}^I\mathbf{p}_{gps}$ is the GPS-IMU extrinsics, $\{{}^E_G\mathbf{R}, {}^E\mathbf{p}_G\}$ is the transformation between reference frames $\{E\}$ and $\{G\}$, and $\mathbf{n}_{g_k}$ is a white Gaussian noise. Due to the asynchronicity of the sensor, the GPS measurement time does not exactly match with the cloned times, thus we use linear interpolation [32] of the two bounding IMU poses to compute the IMU pose at the GPS measurement time.

E. Online Spatiotemporal Calibration

While we have briefly introduced different sensor integration models to estimate the state (1), we further extend our state to include the sensor intrinsics, extrinsics and time offsets among the sensors, and thus, a full calibration can be performed. We omit these details, while readers who are interested can refer to [33] for IMU-camera calibration, [2] for IMU-wheel calibration, and [1] for IMU-GPS calibration.

IV. Efficient LiDAR Measurement Update

There are two major hardships in using LiDAR measurements: real-time processing and data association. As a 3D LiDAR sensor provides a large amount of data points, it is almost impossible to track all the points in real-time. Unlike the camera measurements, finding point correspondences between different scans is very challenging because the points typically do not represent the same physical locations. To address these issues, we efficiently extract plane patches from the point cloud, and as they contain dominant structural information, we then can effectively track them over scans. To describe a plane patch ($\mathfrak{pp}$), we use both the center point $\mathbf{p}$ and Hesse normal $\mathbf{n}$ of this plane, modeled as a Gaussian:

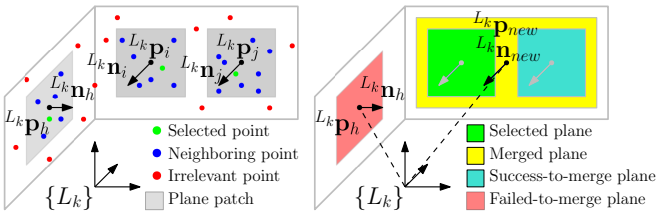

Fig. 2: Plane extraction from the point cloud (left) and plane merger (right)

$$\mathfrak{pp} = \{\mathbf{p}, \mathbf{n}\}, \quad \begin{bmatrix}\mathbf{p}\\ \mathbf{n}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\hat{\mathbf{p}}\\ \hat{\mathbf{n}}\end{bmatrix}, \mathbf{Q}\right) \tag{13}$$

where $\hat{\cdot}$ is the estimated value. Leveraging MSCKF and $\mathfrak{pp}$ representation, we are able to update the state (1) efficiently using all the LiDAR measurements collected in a local window.

In the following, we explain in detail the proposed methods to extract $\mathfrak{pp}$s from the LiDAR point cloud, merge $\mathfrak{pp}$s, find data association, and perform update/calibration.

A. Plane Patch Extraction

When a new LiDAR scan comes in, we sparsely select points and find the neighboring points for each selected point to build a local point cloud $\mathcal{P}_i,\ i \in \{1, 2, \cdots, n\}$ [see green and blue points in Fig. 2 (left)]. We utilize kd-tree to quickly find the neighboring points. Once we have $\mathcal{P}_i$, we leverage the method of [34] to fast extract $\mathfrak{pp}$, and further extend the method to compute the noise covariance of the extracted plane patches. We first express the consisting points locally, by subtracting the center point of the cloud $\mathbf{p}_c$:

$$\begin{bmatrix}x_j\\ y_j\\ z_j\end{bmatrix} = \mathbf{p}_j - \mathbf{p}_c\ ,\ \forall \mathbf{p}_j \in \mathcal{P}_i, j \in \{1, 2, \cdots, m\} \tag{14}$$

$$\mathbf{p}_c = \sum_j \mathbf{p}_j/m, \quad \mathbf{\Psi} = \begin{bmatrix}\Sigma x_j x_j & \Sigma x_j y_j & \Sigma x_j z_j\\ \Sigma y_j x_j & \Sigma y_j y_j & \Sigma y_j z_j\\ \Sigma z_j x_j & \Sigma z_j y_j & \Sigma z_j z_j\end{bmatrix} \tag{15}$$

where $\mathbf{\Psi}$ is the corresponding covariance of the point cloud distribution (scaled by m).

Now, a plane can be extracted from this point cloud parameterized as (by normalizing along z-component):

$$ax + by + d = -z \tag{16}$$

Note this expression can be singular when the z-component of the plane normal is close to zero, while this issue can be easily resolved by choosing different normalization axis. Then we can solve the following linear least-squares:

$$\begin{bmatrix}x_1 & y_1 & 1\\ x_2 & y_2 & 1\\ & \vdots & \\ x_m & y_m & 1\end{bmatrix}\begin{bmatrix}a\\ b\\ d\end{bmatrix} = \begin{bmatrix}-z_1\\ -z_2\\ \vdots\\ -z_m\end{bmatrix} \tag{17}$$

which has the following unique solution:

$$a = (\Sigma y_j z_j \times \Sigma x_j y_j - \Sigma x_j z_j \times \Sigma y_j y_j)/D \tag{18}$$

$$b = (\Sigma x_j y_j \times \Sigma x_j z_j - \Sigma x_j x_j \times \Sigma y_j z_j)/D \tag{19}$$

$$d = 0 \tag{20}$$

$$D = \Sigma x_j x_j \times \Sigma y_j y_j - \Sigma x_j y_j \times \Sigma x_j y_j \tag{21}$$

Then the plane normal $\mathbf{n}_z := [a\ b\ 1]^\top$ can be computed. Note that the computed plane passes the origin because $d = 0$, thus the center point of the cloud $\mathbf{p}_c$ also becomes the center of the $\mathfrak{pp}$ extracted. In order to avoid extracting $\mathfrak{pp}$ from a non-planar surface or ill-conditioned, we check the average point to plane distance and the size of $\mathbf{n}_c$ before being normalized to discard if they do not pass thresholds.

It is straightforward to track the plane uncertainty based on the raw point measurement noise (i.e., $\mathbf{p}_j = \mathbf{p}_{j,true} + \mathbf{n}_j,\ \mathbf{n}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_j)$):

$$\begin{bmatrix} \mathbf{n}_c \\ \mathbf{p}_c \end{bmatrix} = \sum_j \mathbf{H}_j(\mathbf{p}_j - \mathbf{n}_j) = \sum_j \mathbf{H}_j \mathbf{p}_j - \sum_j \mathbf{H}_j \mathbf{n}_j \tag{22}$$

That is, $\mathbf{Q}_c := \sum_j \mathbf{H}_j \mathbf{Q}_j \mathbf{H}_j^\top$ becomes the covariance matrix of (22) where $\mathbf{H}_j$ is the linear function for $\mathbf{p}_j$. Note that there are more robust ways to solve this problem with higher computation costs, such as eigenvalue decomposition. However, the computation speed is vital thus we chose to use the proposed method as the solution can be instantaneously computed from (15).

We register this new $\mathfrak{pp} = \{\mathbf{p}_c, \mathbf{n}_c\}$ within a new point cloud by assigning the center point $\mathbf{p}_c$ as the position of the point and $\mathbf{n}_c$ as the additional information to the point. Note that local point clouds $\mathcal{P}_i$ consisting of each $\mathfrak{pp}$ are also tracked for later merger step (IV-B). Plane patch point cloud ($\mathfrak{pppc}$) is the name of the new point cloud to distinguish it from the LiDAR point cloud. By treating a $\mathfrak{pp}$ as a point and building $\mathfrak{pppc}$, we can conveniently utilize kd-tree again to search the neighboring $\mathfrak{pp}$s from a selected $\mathfrak{pp}$.

B. Plane Patch Merging

There are practical reasons we need to merge the extracted $\mathfrak{pp}$s. If the LiDAR is in a structured environment, especially near a large wall, many of the $\mathfrak{pp}$s represent the same plane, which is redundant. This redundancy cannot be ignored because it can significantly increase computation, especially when having the same plane from several $\mathfrak{pppc}$s. Also, the more points are used to extract the $\mathfrak{pp}$ the more accurate and consistent it can be, if the points are on the same plane. Therefore we try to merge the $\mathfrak{pp}$s that are on the same plane after extracting it from the LiDAR point cloud.

To that end, we first sparsely select $\mathfrak{pp}$s from $\mathfrak{pppc}$ and find each neighboring $\mathfrak{pp}$s using kd-tree. Once the neighboring $\mathfrak{pp}$s are found, we iteratively test the "same plane hypothesis" by checking the Mahalanobis-distance of the following residual between the selected $\mathfrak{pp}_s = \{\mathbf{n}_s, \mathbf{p}_s\}$ and each neighboring $\mathfrak{pp}_n = \{\mathbf{n}_n, \mathbf{p}_n\}$:

$$\mathbf{r}_m = \begin{bmatrix} \mathbf{n}_s - \mathbf{n}_n \\ \mathbf{n}_s^\top(\mathbf{p}_s - \mathbf{p}_n) \end{bmatrix} \tag{23}$$

Specifically, we compute:

$$\gamma = \mathbf{r}_m(\mathbf{H}_s \mathbf{Q}_s \mathbf{H}_s^\top + \mathbf{H}_n \mathbf{Q}_n \mathbf{H}_n^\top)^{-1} \mathbf{r}_m^\top \tag{24}$$

where $\mathbf{H}_s$ and $\mathbf{H}_n$ are the Jacobian matrix respect to each noise of $\mathfrak{pp}_s$ and $\mathfrak{pp}_n$, respectively. γ is compared against a threshold given by the 95 % of the χ^2 distribution. The

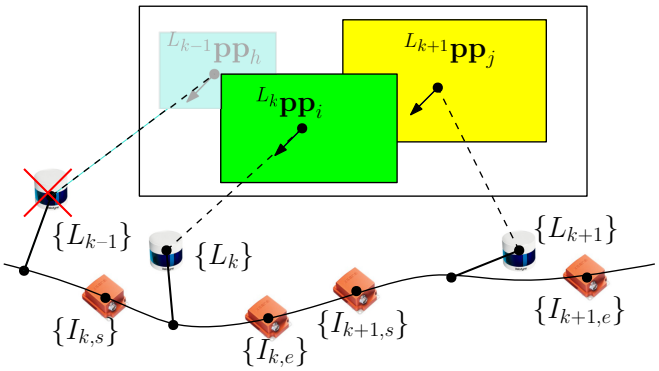

Fig. 3: Selection of LiDAR measurements for the update. $\{I\}$ and $\{L\}$ indicate the historical IMU poses and LiDAR frames at measurement time, respectively. In this example, the LiDAR measurement at t_k and t_{k+1} are collected for the update, while at t_{k-1} is not because the IMU poses cannot bound the measurement.

physical meaning of the first raw of the residual (23) is the parallelism of the planes' normal, and the second row is the point to plane distance. After testing all the $\mathfrak{pp}_n$, the original LiDAR points of the $\mathfrak{pp}_n$ passed the test and $\mathfrak{pp}_s$ are collected without duplication, a new plane patch is created from the collected LiDAR point cloud, and the used $\mathfrak{pp}_s$ and $\mathfrak{pp}_n$ are removed from the cloud. Fig. 2 (right) shows how the merging between the patches is processed for a single $\mathfrak{pp}_s$ case. This operation is repeated until all the $\mathfrak{pp}_s$s are processed, and the whole merging step can be repeated a few times depending on the structure of the environment.

C. Data Association

Unlike the previous steps that all the operations were done within the same reference frame, $\mathfrak{pp}$s have to be transformed frame to frame during the data association/update process which requires the state information. However, due to the asynchronicity among the sensors, our state does not have an IMU pose at exact LiDAR measurement time to express the measurement. We use the linear interpolation [32] of the bounding clones to express the IMU pose at the measurement time, leading to substantial computational savings compare to adding clones to the state every time LiDAR measurement comes in. Thus we first collect all the $\mathfrak{pppc}$s that can find the bounding IMU poses for further steps, which is visually illustrated in Fig. 3.

Assuming that $^{L_k}\mathfrak{pppc}$, which is measured in LiDAR frame $\{L\}$ at t_k is the oldest $\mathfrak{pppc}$ among the collected ones, we want to find the association of $^{L_k}\mathfrak{pp}_i \in {}^{L_k}\mathfrak{pppc}$. For each collected $\mathfrak{pppc}$, we use kd-tree to find the closest $\mathfrak{pp}$ to $^{L_k}\mathfrak{pp}_i$, and check the residual between them. The residual for testing $^{L_k}\mathfrak{pp}_i$ and $^{L_{k+1}}\mathfrak{pp}_j \in {}^{L_{k+1}}\mathfrak{pppc}$, for example, can be described as:

$$\mathbf{r}_a = \begin{bmatrix} {}^{L_{k+1}}\mathbf{n}_j - {}^{L_{k+1}}_{L_k}\mathbf{R}\,{}^{L_k}\mathbf{n}_i \\ {}^{L_{k+1}}\mathbf{n}_i^\top({}^{L_{k+1}}\mathbf{p}_{L_k} + {}^{L_{k+1}}_{L_k}\mathbf{R}\,{}^{L_k}\mathbf{p}_i - {}^{L_{k+1}}\mathbf{p}_j) \end{bmatrix} \tag{25}$$

Note that unlike Eq. (23), transformation terms $\{{}^{L_{k+1}}_{L_k}\mathbf{R}, {}^{L_{k+1}}\mathbf{p}_{L_k}\}$ are needed in Eq. (25) to match the reference frames. The transformation terms can be computed using linear interpolation of the bounding poses:

$${}^{L_{k+1}}_{L_k}\mathbf{R} = {}^{L}_{I}\mathbf{R}\,{}^{I_{k+1}}_{G}\mathbf{R}\,{}^{I_k}_{G}\mathbf{R}^\top\,{}^{L}_{I}\mathbf{R}^\top \tag{26}$$

$${}^{L_k}\mathbf{p}_{L_{k+1}} = {}^{L}_{I}\mathbf{R}\,{}^{I_k}_{G}\mathbf{R}({}^{G}\mathbf{p}_{I_{k+1}} + {}^{I_{k+1}}_{G}\mathbf{R}^\top\,{}^{I}\mathbf{p}_L - {}^{G}\mathbf{p}_{L_k} - {}^{I_k}_{G}\mathbf{R}^\top\,{}^{I}\mathbf{p}_L) \tag{27}$$

$$ {}^{I_k}_{G}\mathbf{R} = \text{Exp}(\lambda \text{Log}({}^{I_{k,e}}_{G}\mathbf{R}\, {}^{I_{k,s}}_{G}\mathbf{R}^{\top}))\, {}^{I_{k,s}}_{G}\mathbf{R} \tag{28} $$

$$ {}^{G}\mathbf{p}_{I_k} = (1-\lambda)\, {}^{G}\mathbf{p}_{I_{k,s}} + \lambda\, {}^{G}\mathbf{p}_{I_{k,e}} \tag{29} $$

$$ \lambda = (t_k + {}^{I}t_L - t_{k,s})/(t_{k,e} - t_{k,s}) \tag{30} $$

where $\text{Exp}(\cdot)$ is the $SO(3)$ matrix exponential function [31], $\{{}^{L}_{I}\mathbf{R}, {}^{I}\mathbf{p}_L\}$ is the LiDAR-IMU extrinsic, ${}^{I}t_L$ is the time offset, and $t_{k,s}$ and $t_{k,e}$ are the bounding IMU pose times of measurement time t_k, respectively. If the residual is smaller than threshold, we consider the two $\mathfrak{pp}$s represent the same plane. Note that we only check residuals here; Mahalanobis-distance test can be statistically more consistent but heavy due to its matrix inversion-related process. The possible false associations will be filtered-out within the next update step. We iteratively find all the associations of ${}^{L_k}\mathfrak{pp}$ among the collected $\mathfrak{ppc}$s.

D. Plane Measurement Update and Calibration

While the $\mathfrak{pp}$ representation of planes can provide a fast data association, there are several computation-related problems because of its non-minimal representation if naively used for update. As a plane has 3DOF, while $\mathfrak{pp}$ uses a 6DOF vector $(\mathbf{p}, \mathbf{n})$ to represent such a plane, this over-parameterization may significantly increase the computation burden *and* the numerical instability. We, therefore, change the plane representation from plane patch ($\mathfrak{pp}$) to closest point ($\mathfrak{cp}$) [35], a minimal representation of a plane and can formulate compact residual function to perform efficient MSCKF update.

Specifically, $\mathfrak{cp}$ can be considered as a 3D point that resides on the plane and is the closest to the measured frame's origin. This $\mathfrak{cp}$ representation can be described using the Hesse normal vector $\mathbf{n}$ and distance scalar d that has the following relation:

$$ {}^{L_k}\mathbf{\Pi} = {}^{L_k}\mathbf{n}\, {}^{L_k}d \tag{31} $$

$$ \begin{bmatrix} {}^{L_k}\mathbf{n} \\ {}^{L_k}d \end{bmatrix} = \begin{bmatrix} {}^{L_k}\mathbf{\Pi}/||{}^{L_k}\mathbf{\Pi}|| \\ ||{}^{L_k}\mathbf{\Pi}|| \end{bmatrix} \tag{32} $$

The relation between $\mathfrak{pp}$ and $\mathfrak{cp}$ can be described as:

$$ \mathbf{\Pi} = \mathbf{n}_{\mathfrak{pp}}(\mathbf{n}_{\mathfrak{pp}}^{\top}\mathbf{p}_{\mathfrak{pp}}) \tag{33} $$

By using $\mathfrak{cp}$, we can represent the associated plane patches found from IV-C as a measurement function:

$$ \mathbf{z}_j := {}^{L_{k+1}}\mathbf{\Pi}_j = {}^{L_{k+1}}_{L_k}\mathbf{R}\, {}^{L_k}\mathbf{n}_i({}^{L_k}d_i - {}^{L_k}\mathbf{p}_{L_{k+1}}^{\top}\, {}^{L_k}\mathbf{n}_i) + \mathbf{w}_j \tag{34} $$

where $\mathbf{w}_j$ is the noise of ${}^{L_{k+1}}\mathbf{\Pi}_j$ computed from ${}^{L_{k+1}}\mathfrak{pp}_j$. Note that the LiDAR-IMU extrinsic and the time offset involves in frame transformation as Eq. (26)-(30) shows. Therefore we expand our state (1) to include the extrinsic $\{{}^{L}_{I}\mathbf{R}, {}^{I}\mathbf{p}_L\}$ and the time offset ${}^{I}t_L$, and perform online calibration at the same time. Now we linearize the above function to get residual as:

$$ \tilde{\mathbf{z}}_j = \mathbf{H}_{\mathbf{\Pi}}\, {}^{L_k}\tilde{\mathbf{\Pi}}_i + \mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} + \mathbf{w}_j \tag{35} $$

We apply the MSCKF feature marginalization strategy to our plane features; after whitening the noise $\mathbf{w}_j$, stacking all the measurements corresponding to ${}^{L_k}\mathbf{\Pi}_i$ and multiplying the left nullspace of $\mathbf{H}_{\mathbf{\Pi}}$ on the left side results in a residual:

$$ \tilde{\mathbf{z}}_i = \bar{\mathbf{H}}_{\mathbf{x}}\tilde{\mathbf{x}}_k + \mathbf{w}_i \tag{36} $$

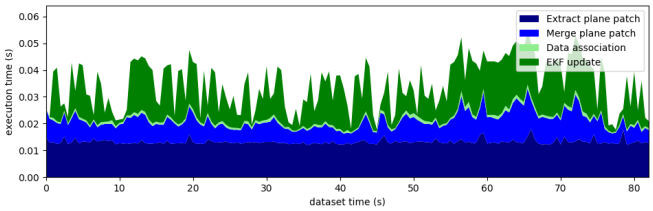

Fig. 4: Timing of different system components reported in milliseconds for the simulated dataset. Recorded on an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz processor in single threaded execution.

TABLE II: Simulation parameters and prior single standard deviations that perturbations of measurements and initial states were drawn from.

Parameter	Value	Parameter	Value
Cam Freq. (hz)	20	IMU Freq. (hz)	400
LiDAR Freq. (hz)	20	Num. Clones	10
Max. Feats	200	Pixel Proj. (px)	1
Gyro. White Noise	1.7e-04	Gyro. Rand. Walk	1.9e-05
Accel. White Noise	2.0e-03	Accel. Rand. Walk	3.0e-03
LiDAR Channel	64	LiDAR H. Resolution	0.5
LiDAR Point Noise	2.0e-02	LiDAR Merge Itr.	3
LiDAR Neighbors	15	LiDAR Sample Interval	15
LiDAR Extrinsic Ptrb.	5.0e-02	LiDAR Toff. Ptrb.	1.0e-02

As discussed in Section IV-C, there may be false associations, and thus we perform the following Mahalanobis-distance test with the residual (36) as:

$$ \gamma = \tilde{\mathbf{z}}_i(\bar{\mathbf{H}}_{\mathbf{x}}\mathbf{P}\bar{\mathbf{H}}_{\mathbf{x}}^{\top} + \mathbf{Q}_i)^{-1}\tilde{\mathbf{z}}_i^{\top} \tag{37} $$

where $\mathbf{P}$ and $\mathbf{Q}_i$ are the covariance matrix of the state and $\mathbf{w}_i$. If Mahalanobis-distance is smaller than the threshold, we consider the associations are valid, and Eq. (36) can be directly used in the MSCKF update without storing features in the state, leading to substantial computational savings. The used plane patches are removed from the cloud to avoid the reuse of information.

V. Simulation Results

MINS is implemented within the *OpenVINS* [3] framework which provides both simulation and evaluation utilities. In order to validate the proposed LiDAR integration method, we fuse LiDAR with VIO in the simulated environment shown in Fig 1. We leverage the LiDAR point cloud simulator from [15] to simulate HDL-64E, and have listed the key simulation parameters in Table II. Note that, here we only present the LiDAR evaluation results, as the other sensor integration methods and their calibrations are already validated in our prior work [1]–[3].

A. Timing Analysis

As the simulated HDL-64E works at 20 Hz, the maximum available processing time is 50 ms in order to be real-time. As shown in Fig. 4, it is clear that the system is able to incorporate all the measurements at a very high speed. It took averagely 12 ms for plane patch extraction, 8 ms for merging the planes, 1 ms for data association, 25 ms for EKF update, 47 ms for the total process.

It is also important to report the number of measurements processed, as we do not want to speed up the system just by discarding the measurements. On average, the number of planes extracted from a single LiDAR point cloud was 1494, we get 321 planes after the merger, found 1777 associations

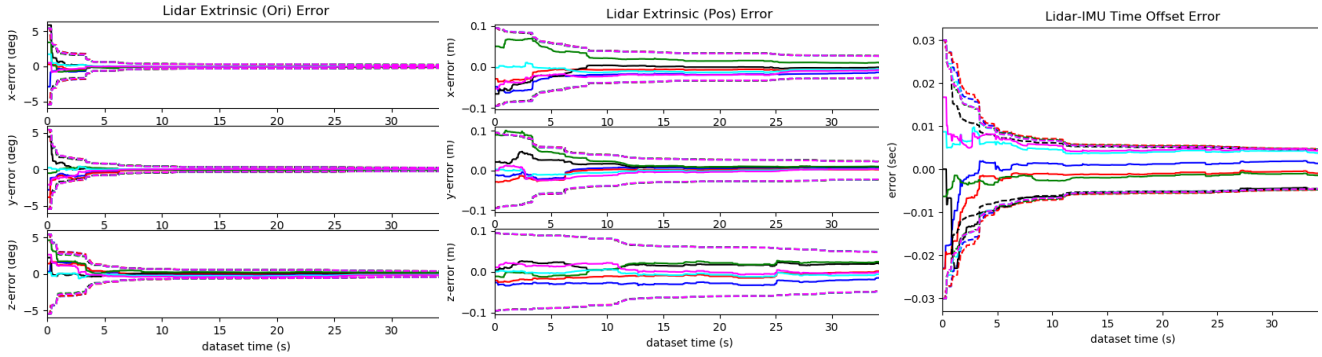

Fig. 5: Calibration errors of each parameter (solid) and 3σ bound (dotted) for six different runs under random motion. Each colors denote runs with different realization of the measurement noise and the initial values. Both the estimation errors and 3σ bounds can converge in about 10 seconds.

TABLE III: Root mean squared error (RMSE) of each algorithm (degree/meter).

Alogorithm	VIO	Wheel-VIO	GPS-VIO	LiDAR-VIO	MINS
RMSE	5.27/66.31	1.71/42.74	1.25/2.65	1.92/26.30	0.94/2.10

among 9 ppps (240 plane feature candidates), and used 109 planes to update 10 IMU poses (60-dimensional vector), showing the system was fully functional.

B. Extrinsic and Time offset Calibration

To validate that all LiDAR-IMU calibration parameters are able to converge, we simulated a trajectory which excited all-axes motions. Shown in Fig. 5, we performed online extrinsic and time offset calibration six times with different initial perturbations. We additionally plot the 3σ bounds, which should bound the error in the case that our estimator is consistent. It is clear that all calibration parameters are able to quickly converge to near their true values and remain within 3σ bounds.

VI. Real-world Experimental Results

We further evaluate *MINS* in a real-world dataset, KAIST *urban*39 [36], which is collected in urban area with 11.06 km long trajectory, and used the stereo camera, IMU, wheel encoder, GPS, and 16 channel LiDAR for the estimation. Based on VIO, different combination of sensors are tested along with calibration and the resulting trajectories of each algorithm are shown in Fig. 6. The root mean squared error (RMSE) of orientation and position of each algorithm compared to the ground truth the dataset provides are summarized in Table III.

Overall, the VIO showed scale issue and combinating additional sensor was able to solve the problem. The GPS-VIO showed the best result among three sensor combinations, taking advantage of its global measurements from GPS. The LiDAR-VIO was able to run around 2 times faster than the real-time and especially showed a good z-axis estimation results. This is because the LiDAR was mounted at an angle of 45 degrees downward providing most of the scans from the surface of the road, thus the extracted planes on the road could prevent the z-directional drift. The proposed *MINS* fuses all the sensors and records the most accurate results with all calibration parameters' convergence while running in real-time, showing globally accurate and locally precise localization performance.

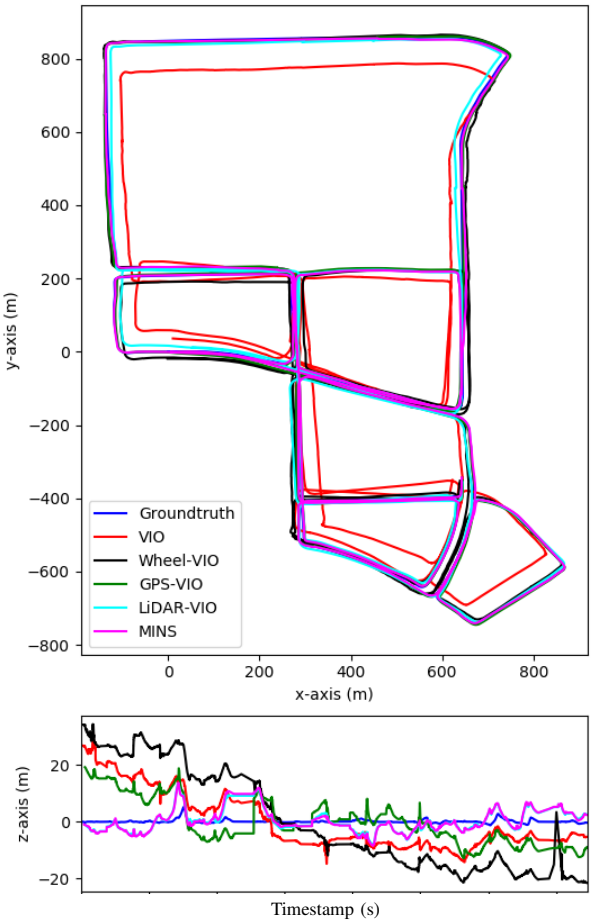

Fig. 6: Trajectories of each algorithm on *Urban39* dataset. Top view (top) and vertical absolute trajectory error (bottom)

VII. Conclusions and Future Work

In this paper, we have developed an efficient and consistent MSCKF-based multi-sensor aided INS, *MINS*, that fuses IMU, camera, wheel, GPS, and LiDAR measurements and performs online spatiotemporal calibration of all sensors. In particular, we have been primarily focusing on efficiently integrating the LiDAR measurements, which was a bottleneck issue for real-time multi-sensor localization due to its large volume of data, and proposed to extract plane patches from the point clouds and track them over scans to form motion constraints for MSCKF update. Simulation results

show our method's ability to integrate 64 channel 20 Hz LiDAR in real-time, along with calibration convergence. The proposed *MINS* was also validated in the real datasets, showing its globally accurate and locally precise real-time localization performance. In the future, we will investigate how to efficiently include loop closures into the system.

REFERENCES

[1] W. Lee, K. Eckenhoff, P. Geneva, and G. Huang, "Intermittent gps-aided vio: Online initialization and calibration," in *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020.

[2] W. Lee, K. Eckenhoff, Y. Yang, P. Geneva, and G. Huang, "Visual-inertial-wheel odometry with online calibration," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2020.

[3] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020. [Online]. Available: https://github.com/rpng/open_vins

[4] V. Lidar, "Hdl-64e: High definition real-time 3d lidar." [Online]. Available: https://www.ilikebigbits.com/2017_09_25_plane_from_points_2.html

[5] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[6] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *IEEE/RSJ international conference on intelligent robots and systems*, 2013, pp. 3923–3929.

[7] K. Hausman, S. Weiss, R. Brockers, L. Matthies, and G. S. Sukhatme, "Self-calibrating multi-sensor fusion with probabilistic measurement validation for seamless sensor switching on a uav," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4289–4296.

[8] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft mav," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4974–4981.

[9] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *arXiv preprint arXiv:1901.03642*, 2019.

[10] J. K. Suhr, J. Jang, D. Min, and H. G. Jung, "Sensor fusion-based low-cost vehicle localization system for complex urban environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1078–1086, 2016.

[11] X. Meng, H. Wang, and B. Liu, "A robust vehicle localization approach based on gnss/imu/dmi/lidar sensor fusion for autonomous vehicles," *Sensors*, vol. 17, no. 9, p. 2140, 2017.

[12] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "LIC-Fusion: Lidar-inertial-camera odometry," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Macau, China, Nov. 2019, (accepted).

[13] X. Zuo, Y. Yang, P. Geneva, J. Lv, Y. Liu, G. Huang, and M. Pollefeys, "Lic-fusion 2.0: Lidar-inertial-camera odometry with sliding-window plane-feature tracking," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2020.

[14] X. Zuo, M. Zhang, Y. Chen, Y. liu, G. Huang, and M. Li., "Visual-inertial localization for skid-steering robots with kinematic constraints," in *Proc. of the International Symposium on Robotics Research (ISRR)*, Hanoi, Vietnam, Oct. 2019.

[15] P. Geneva, N. Merrill, Y. Yang, C. Chen, W. Lee, and G. Huang, "Versatile 3d multi-sensor fusion for lightweight 2d localization," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2020.

[16] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *arXiv preprint arXiv:2010.08196*, 2020.

[17] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "Lins: A lidar-inertial state estimator for robust and efficient navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8899–8906.

[18] P. Alliez, F. Bonardi, S. Bouchafa, J.-Y. Didier, H. Hadj-Abdelkader, F. I. I. Muñoz, V. Kachurka, B. Rault, M. Robin, and D. Roussel, "Real-time multi-slam system for agent localization and 3d mapping in dynamic scenarios," in *International Confererence on Intelligent Robots and Systems (IROS 2020)*, 2020.

[19] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[20] ——, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9, 2014.

[21] L. Zhang, D. Chen, and W. Liu, "Point-plane slam based on line-based plane segmentation approach," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2016, pp. 1287–1292.

[22] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3-d mapping," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 424–441, 2010.

[23] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[24] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3d lidar and dense stereo," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2181–2188.

[25] M. Velas, M. Spanel, M. Hradis, and A. Herout, "Cnn for imu assisted odometry estimation using velodyne lidar," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2018, pp. 71–77.

[26] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6243–6252.

[27] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam—learning a compact, optimisable representation for dense visual slam," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2560–2568.

[28] X. Ju, D. Xu, and H. Zhao, "Scene-aware error modeling of lidar/visual odometry for fusion-based vehicle localization," *arXiv preprint arXiv:2003.13109*, 2020.

[29] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep., Mar. 2005.

[30] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[31] G. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011, vol. 2.

[32] M. Li, "Visual-inertial odometry on resource-constrained systems," Ph.D. dissertation, UC Riverside, 2014.

[33] Y. Yang, P. Geneva, K. Eckenhoff, and G. Huang, "Degenerate motion analysis for aided INS with online spatial and temporal calibration," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 2070–2077, 2019.

[34] E. Ernerfeldt, "Fitting a plane to noisy points in 3d." [Online]. Available: https://www.ilikebigbits.com/2017_09_25_plane_from_points_2.html

[35] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: Lidar-inertial 3d plane slam," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, Oct. 1-5, 2018.

[36] 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.