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

Robust SRIF-based LiDAR-IMU Localization for Autonomous Vehicles

Kun Li[*,+], Zhanpeng Ouyang[*,×], Lan Hu[×], Dayang Hao[+] and Laurent Kneip[×]

Abstract— **We present a tightly-coupled multi-sensor fusion architecture for autonomous vehicle applications, which achieves centimetre-level accuracy and high robustness in various scenarios. In order to realize robust and accurate point-cloud feature matching we propose a novel method for extracting structural, highly discriminative features from LiDAR point clouds. For high frequency motion prediction and noise propagation, we use incremental on-manifold IMU pre-integration. We also adopt a multi-frame sliding window square root inverse filter, so that the system maintains numerically stable results under the premise of limited power consumption. To verify our methodology, we test the fusion algorithm in multiple applications and platforms equipped with a LiDAR-IMU system. Our results demonstrate that our fusion framework attains state-of-the-art localization accuracy, high robustness and a good generalization ability.**

I. Introduction

In recent years, the requirement for autonomous transportation and replacement of human labour has continuously pushed autonomous, Unmanned Ground Vehicles (UGVs) as an important future technology with a wide range of applications. A crucial ingredient for such systems is given by robust localization in large-scale urban environments providing precise, centimetre-level global position estimation. A common solution to this problem is given by Inertial Navigation Systems (INS) employing a Real-Time Kinetic (RTK) [16] Global Positioning System (GPS) and an Inertial Measurement Unit (IMU) [1], [23], [29]. However, Global Navigation Satellite Systems (GNSS) suffer from disruptions or signal loss due to obstruction of the sky caused for example by *urban canyons* [7]. One solution to this problem is given by using odometry information to compensate for sudden jumps in the GNSS measurements [7], [20], [24]. However, the performance often remains insufficient for autonomous driving related applications.

Light detection and ranging (LiDAR) [14], [26] devices come to the scope providing accurate 3 dimensional (3D) point-cloud measurements with multi-layer laser beams and fast scanning speed. The combination of LiDARs and other sensors [4], [6], [11], [12], [15], [22] is a popular strategy in odometry and mapping [21], [31]–[33]. Among those sensors, the IMU is most commonly used for LiDAR point-cloud undistortion [15], [21], [31].

[+]Damo Academy, Alibaba Group.
{lk158400, qingquan.qing}@alibaba-inc.com
[×]Mobile Perception Lab, SIST, ShanghaiTech
{ouyangzhp,hulan,lkneip}@shanghaitech.edu.cn
L. Kneip is also with the Shanghai Engineering Research Center of Intelligent Vision and Imaging. The authors would like to thank the funding sponsored by Natural Science Foundation of Shanghai (grant number: 19ZR1434000).
[*] equal contribution

An increasingly popular solution to the urban canyon problem is given by dropping GNSS measurements altogether, and by relying on a LiDAR that measures 3D scans of the environment and aligns them with a prior, geo-registered 3D map. The present paper proposes such a framework. However, most LiDAR-based localization solutions with prior point-cloud maps [8], [17], [25] assume that the road scenes are relatively constant, while new constructions, roadside vegetation, partial occlusions by dynamic objects, and adverse weather conditions may severly compromise robustness.

We present a tightly-coupled LiDAR-IMU fusion architecture for robust and efficient localization given a prior point-cloud feature map. Our method describes point-cloud types in a more detailed way, which greatly reduces the problem of mismatching. Different fusion algorithms exist. For example, learning-based methods [19], [30] have become popular in recent years. Other algorithms [2], [18], [27] use classical filters in the back end, which ensures accuracy and real-time performance. Our fusion method is an extension of SRIF [27] to LiDAR-IMU systems. SRIF permits the use of single precision representation, so it achieves a considerable improvement in computational efficiency compared against the double precision scheme on resource constrained mobile platforms.

An overview of the framework is shown in Fig.1. Our contributions are as follows:

- We propose a novel feature extraction method for LiDAR point-clouds. The utilization of these novel features enables precise registration and operation in dynamic or partially changed environments. It compares favourably against classical Iterative Closest Point (ICP) [5], [31] or Normal Distribution Transform (NDT) [3], [25] based alternatives.
- Our localization framework is built upon the Square Root Inverse Filter (SRIF) [27]. It gives our work the characteristics of numerical stability and low computational power consumption enabling real-time operation even on ARM-based platforms.
- As proven by our experimental results, our framework performs well in challenging scenarios given by a highly dynamic, drifting vehicle or localization in feature-poor scenarios. Our system also achieves better accuracy and lower computation load than the localization module of the state-of-the art LOAM [31] framework.

Our paper is structured as follows. Section II introduces the usage of the IMU sensor as well as the details of the extraction, classification and aggregation of multiple different

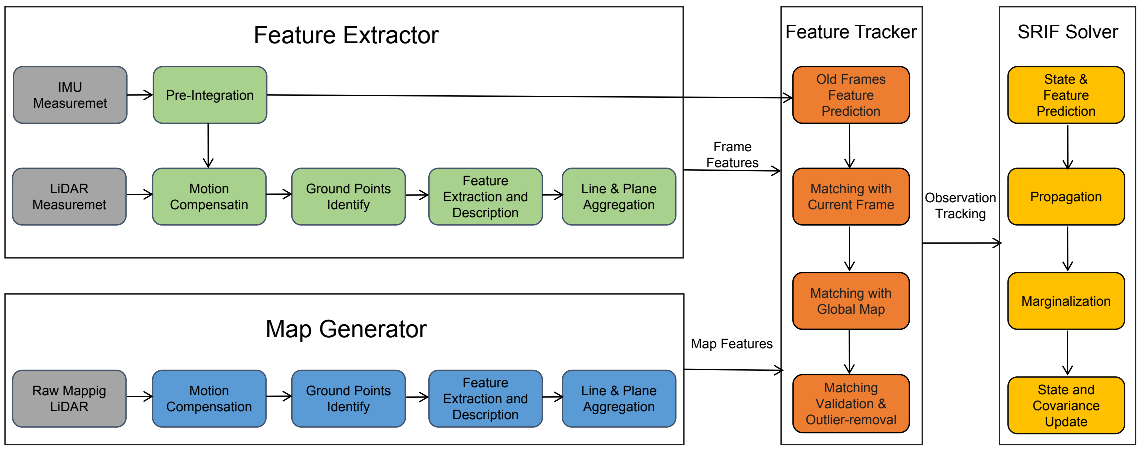


Fig. 1. Overview of the proposed multi-sensor fusion framework: grey blocks indicate raw sensor inputs; green blocks represent feature extraction, classification and aggregation modules; the blue blocks are the corresponding counterparts for the off-line generation of the global map prior; and the core parts of the SRIF solver are in yellow blocks.

types of features. Section III describes our feature map generator. Section IV introduces the maintenance of constraints within a sliding window including the establishment of correspondences between observations and the global map. Section V presents how we use the SRIF solver to obtain precise global vehicle pose estimates. Section VI finally concludes with our experimental results on multiple, real-world cases, proving the validity of our method.

II. Feature Extractor

The discriminative power of point-level features in LiDAR data is weak owing to the fact that LiDAR points are sparse and lack textural information. We propose the extraction of multiple line and surfel features, which are the green blocks in Fig. 1. Compared against ICP-based tracking, our higher-level feature-based tracker can alleviate computational load while improving robustness.

The 3D point-cloud is formatted as a matrix where the column index denotes the horizontal angle and the row index the vertical beam layer. The relative time Δt of point $\mathbf{p}_{i,j}$ (column index i and row index j) indicates the time elapsed since the scan's first point has been sampled, and we define the depth of $\mathbf{p}_{i,j}$ by $\lambda_{i,j}$. With all points registered in this lattice matrix, we are ready to perform motion compensation and high-level feature extraction.

A. Motion Compensation

The LiDAR obtains point-clouds by periodic scanning of the environment. However, the LiDAR moves while capturing one scan, which distorts the captured point cloud. Undistortion consists of registering the point-cloud with respect to the same time instant. In this work, we use the IMU as an auxiliary device for relative motion estimation and motion compensation. We use a low-cost MTI-3 MEMS IMU which outputs 200 Hz dynamic measurements that are hardware-synchronized with the LiDAR via bus communication. The IMU pre-integration and LiDAR motion compensation follows the standard, pre-integration approach on the SO(3) *manifold* as proposed in [10] and [9].

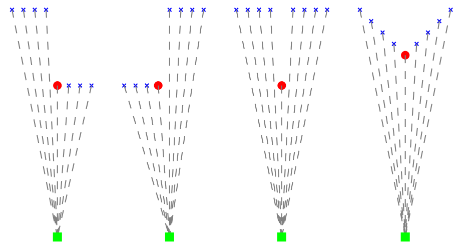

(a) Left (b) Right (c) Single (d) Normal

Fig. 2. The four LiDAR corner line types: *left* and *right corner* represent partially viewed objects, *single* denotes a pole-like object, and *normal* represents a common edge.

B. Ground Plane Identification

For autonomous vehicles, typically about one third of the LiDAR's scans are ground points, which do not provide distinctive information for the extraction, description, and tracking of features. To alleviate the complexity of the later feature extraction, we first identify the ground points with the assistance of the IMU. We get the gravity direction $\mathbf{n}_g$ of the current frame from the IMU pre-integration. We furthermore know the height d of the LiDAR. With $\mathbf{n}_g$ and the height d given, we identify the points on the ground plane if $d-\tau_1 < |[\mathbf{p}_{i,j}\ \lambda_{i,j}]^T\mathbf{n}_g| < d+\tau_1$ where τ_1 is a predefined threshold. We set it to 0.15m in our system. The ground plane is furthermore added to the pose estimation as an additional surfel constraint.

C. LiDAR Feature Extraction and Description

In order to make the system more robust in dynamic scenarios, it's essential to extract point-cloud features and put them into correctly identified classes. We first divide the points into two classes: corner points and normal points, for which we rely on geometric properties. Instead of using curvature to define the smoothness of a point, we simply consider the *arccos* of the angle formed by the line segments between the target point and the two adjacent points. It is given by

$$\psi_s(\mathbf{p}_{i,j}, o=1) = 57.3^\circ \cdot \arccos\left(\frac{(\mathbf{p}_{i,j-o}-\mathbf{p}_{i,j})^T(\mathbf{p}_{i,j+o}-\mathbf{p}_{i,j})}{\|\mathbf{p}_{i,j-o}-\mathbf{p}_{i,j}\|\cdot\|\mathbf{p}_{i,j+o}-\mathbf{p}_{i,j}\|}\right) \quad (1)$$

As shown in Fig. 2, the corner points are then further sub-classified into one of 4 categories: left corners, right corners, single corners and normal corners. The four corner types are defined and is covered as follows:

1) **Left corner points** and **Right corner points** (cf. Fig.2(a) and Fig.2(b), typically found on buildings or large shelters). For a left corner point, the laser beam for example scans a building to the right of the vehicle, and only a part of the LiDAR scan will cover this building. The left-most point in the *i*th horizontal layer of the scan that is still located on the building is defined as a left corner point, denoted $\mathbf{p}_{i,j}$. Let $\mathbf{p}_{i,j-m}$ be the first valid[1] point on the left of $\mathbf{p}_{i,j}$. A left corner point should satisfy either

$$m > \tau_2 \text{ or } \lambda_{i,j-m} - \lambda_{i,j} > \tau_3. \quad (2)$$

τ_2 and τ_3 are two pre-set thresholds. Similarly, a right corner point needs to satisfy either

$$n > \tau_2 \text{ or } \lambda_{i,j} - \lambda_{i,j+n} > \tau_3, \quad (3)$$

where n is defined such $\mathbf{p}_{i,j+n}$ is the first valid point to the right in scan row i. In our system, $\tau_2 = 5, \tau_3 = 0.5m$.

2) **Single corner points** (cf. Fig. 2(c)): Describes a pole-like object like a trunk or a telegraph pole. Single corner points must satisfy both

$$\lambda_{i,j} - \lambda_{i,j-m} > \tau_3 \text{ and } \lambda_{i,j} - \lambda_{i,j+n} > \tau_3. \quad (4)$$

The adjacent points are ignored and will not be used to compose the feature.

3) **Normal corner points** (cf. Fig. 2(d)): Normal corner points are the most common corner points. When a laser beam scans a building, it may scan two adjacent, non-parallel surfaces. The points at the intersection of the two surfaces would be identified as normal corner points. They satisfy the constraint

$$\psi_s(\mathbf{p}_{i,j}, o=2) < \tau_4. \quad (5)$$

We additionally request at least three of the five points to the left and the right of the candidate normal corner point to be sufficiently smooth. This can be verified using the condition

$$\psi_s(\mathbf{p}_{i,j+k}, o=1) > \tau_5,\ k = \pm\{1,2,\cdots,5\}. \quad (6)$$

In our system, $\tau_4 = 134^\circ$ and $\tau_5 = 143^\circ$.

Remaining points that have not been classified as corner or ground points are marked as normal surfel points. We call such points *surfels* and aggregate them by means of a clustering strategy.

[1]valid means the depth of the point is within a tolerated minimum and maximum depth.

D. Line and Surfel Aggregation

We again use the lattice grids to further divide 3D points into cells for fast and precise aggregation of points into higher-level features. This is done sequentially for lines and surfels. Every layer in the point cloud is divided into x angular bins (referred to here as cells), which are marked as empty or occupied depending on whether or not they contain a point of a particular type. Occupied cells at the same angular index are then aggregated from adjacent layers if certain compatibility conditions are met (cf. Algorithm 2 and 1). For lines and normal as well as ground surfel points, we use 180, 72, and 36 cells in each scan row. Define $\mathbf{G}_{i,j}$ to be the j_{th} bin in the i_{th} layer.

Algorithm 1 Surfel Aggregation

```
Input: marked surfel points
Output: surfel feature
  step1: For the cell $\mathbf{G}_{i,j}$ that contains the corner point and its adjacent cell $\mathbf{G}_{i+1,j}$, verify that the total number of points in these two cells is greater than 4.
  step2: Randomly select a point $\mathbf{p}_1$ from $\mathbf{G}_{i,j}$. Select the first point $\mathbf{p}_2$ and the last point $\mathbf{p}_3$ in $\mathbf{G}_{i+1,j}$. Calculate the unit normal vector $\mathbf{n}$ of the plane composed of these three points.
  step3: Select the first point $\mathbf{p}_4$ and the last point $\mathbf{p}_5$ in $\mathbf{G}_{i,j}$. $\mathbf{v}_1$ is the unit vector from $\mathbf{p}_2$ to $\mathbf{p}_4$. $\mathbf{v}_2$ is the unit vector from $\mathbf{p}_3$ to $\mathbf{p}_5$.
  step4: If $|\cos(\mathbf{n}^T\mathbf{v}_1)| < \tau_1$ and $|\cos(\mathbf{n}^T\mathbf{v}_2)| < \tau_1$. The points in $\mathbf{G}_{i,j}$ $\mathbf{G}_{i+1,j}$ can form a surfel.
  step5: The points are fitted into a surfel by [28].
```

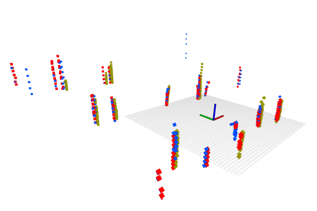
Algorithm 2 Line Aggregation

```
Input: marked line corner points
Output: line feature
  step1: Find the cell $\mathbf{G}_{i,j}$ where the corner point $\mathbf{p}_1$ is located and the cells $\mathbf{G}_{i+m,j+n}$ near it (m = -1, 0, 1, n = -1, 0, 1). Save $\mathbf{p}_1$ into S.
  step2: Select two same type points $\mathbf{p}_1$ and $\mathbf{p}_2$ from $\mathbf{G}_{i,j}$ and $\mathbf{G}_{i+m,j+n}$, respectively, and calculate the depth difference between $\mathbf{p}_1$ and $\mathbf{p}_2$. If the depth difference is small enough, add $\mathbf{p}_2$ to the set S.
  step3: Continue searching for points of the same type that meet the depth difference requirement and each time add those points to S.
  step4: The points in S are fitted into a line by [13].
```

We cluster marked points into four types of lines (left line, right line, single line and normal line) and two types of surfels (normal surfel and ground surfel) according to the original point corner types.

III. Map Generator

As we perform a map-based localization, the quality of the global map is crucial for the localization accuracy. We therefore use a dedicated vehicle which equipped with


(a) Tracking lines

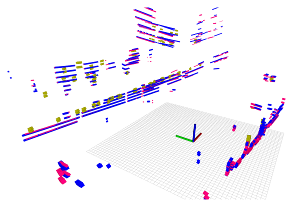

(b) Tracking surfels

Fig. 3. Feature tracking performance: lines and surfels extracted in the current frame (red), features projected from the last frame by prediction (blue) and features with global correspondences (yellow). Left: line matches. Right: surfel matches.

a high-class XW-GI5651 inertial navigation system (INS), an RTK system, a 32-layer and two 16-layer Robosense LiDARs, and a set of encoders. Such a well-equipped vehicle can provide a highly accurate point-cloud map. We use the same method as in Section II to get the line and surfel features in the global map.

IV. Feature Tracker

To fuse a new frame $\mathbb{F}_k$ into the sliding window $\mathbb{W}_{k-1}$, we need to associate same-type features according their spatial distance and type. Let $l_i^w = (\mathbf{c}_i^w, \mathbf{n}_i^w, s_i^w)$ be the i_{th} features of the sliding window, where $\mathbf{c}_i^w$ is the center point of feature l_i^w, $\mathbf{n}_i^w$ is the corresponding normal, and s_i^w is the feature type. Accordingly, the j_{th} features of the current frame $\mathbb{F}_k$ are represented by l_j^f. Let the sliding window frame $\mathbb{W}_{k-1}$ coincide with the frame $\mathbb{F}_{k-1}$. For a new frame $\mathbb{F}_k$, the IMU provides the relative pose from $\mathbb{F}_k$ to $\mathbb{F}_{k-1}$ (and thus $\mathbb{W}_{k-1}$). Let $\mathbf{R}_{k,k-1}$ and $\mathbf{t}_{k,k-1}$ denote the rotation and translation from $\mathbb{W}_{k-1}$ to $\mathbb{F}_k$, respectively.

For each feature l_i^f in the current frame, we first find its 10 nearest neighbors $N(l_i^f)$ within the sliding window and according to their spatial distance. The matching score to the j_{th} neighbor feature within $N(l_i^f)$ is given by:

$$m_{i,j} = I(s_j^w, s_i^f) g(\mathbf{R}_{k,k-1}\mathbf{n}_j^w, \mathbf{n}_i^f) \exp^{-\|\mathbf{R}_{k,k-1}\mathbf{c}_j^w + \mathbf{t}_{k,k-1} - \mathbf{c}_i^f\|^2} \quad (7)$$

where $I()$ is an indicator function for same feature type (if $s_j^w == s_i^f$, then $I(s_j^w, s_i^f) = 1$, else $I(s_j^w, s_i^f) = 0$), $g()$ expresses normal vector deviations. In our system, $g(\mathbf{n}_1, \mathbf{n}_2) = |\mathbf{n}_1^T \mathbf{n}_2|$. We find feature matches by maximizing

$$j^* = \arg \max_{j \in N(l_j^f)} \{m_{i,j}\}, \quad (8)$$

and checking that $m_{i,j^*} > \tau_6$. $\tau_6 = 0.9$ in our system.

Further matches between the current frame and the global map are found by a similar strategy. An example matching result is shown in Fig. 3(a) and Fig. 3(b).

V. SRIF Solver

The pose optimizer is implemented as a sliding window filter. The filter's state is defined as:

$$\mathbf{x}_\text{k} = \left[\mathbf{x}_\text{F}^1, \cdots, \mathbf{x}_\text{F}^n\right]$$

where n is the length of the window, and $\mathbf{x}_F^i$ is the i-th frame's status in the sliding window given by

$$\mathbf{x}_\text{F}^i = \left[\mathbf{q}_\text{WI}^i \quad \mathbf{p}_\text{WI}^i \quad \mathbf{v}_\text{WI}^i \quad \mathbf{b}_\text{g}^i \quad \mathbf{b}_\text{a}^i\right]. \quad (9)$$

$\mathbf{q}_\text{WI}^i$ and $\mathbf{p}_\text{WI}^i$ are the absolute pose in the world frame predicted by the IMU; $\mathbf{v}_\text{WI}^i$ is the frame's velocity vector; and $\mathbf{b}_\text{g}^i$ and $\mathbf{b}_\text{a}^i$ represent the gyroscope and accelerometer biases.

Upon each newly added keyframe k, the objective of the solver is to minimize the cost function $\mathscr{C}_k$:

$$\mathscr{C}_k = \mathscr{C}_{k-1} + \mathscr{C}_u + \mathscr{C}_f, \quad (10)$$

where $\mathscr{C}_{k-1}$ is the cost term of the prior frame, $\mathscr{C}_u$ represents the loss with respect to the IMU pre-integration term, and $\mathscr{C}_f$ represents the registration error of the LiDAR features in the slide window $\mathbb{W}_k$. The prior cost term is formulated as $\mathscr{C}_{k-1}(\tilde{\mathbf{x}}_{k-1}) = \|\mathbf{I}_{k-1}\tilde{\mathbf{x}}_{k-1} - \mathbf{r}_{k-1}\|^2$, where $\mathbf{I}_{k-1}$ and $\mathbf{r}_{k-1}$ are the prior upper-triangular information filter matrices and the corresponding residuals, respectively. $\tilde{\mathbf{x}}_{k-1}$ in turn represents the error representation of the last state. Our solver is an extension of SRIF. For details one the propagation and marginalization steps, the reader is kindly referred to [27].

The update is completed by stacking all Jacobians and residuals from different sensor sources. We thereby construct an overdetermined constraint function $\mathscr{C}_k$ of all states within the sliding window.

Residuals: As outlined in Sec II, we extract both line and surfel features from LiDAR point clouds, which hence serve to construct $\mathscr{C}_f$. For two corresponding line features, let $\mathbf{p}_{obs}^j$ be the j_{th} point measured on the detected line in the laser scan. Let $\mathbf{p}_m^0$ and $\mathbf{p}_m^1$ furthermore be any two points on the corresponding line from the global map. $\mathbf{d}_l$ represents the distance error between the observation and the landmark, and it is given by

$$\mathbf{d}_l = \sum_{j=1}^{N} h_0\left(\frac{\|\mathbf{e}_0^j \times \mathbf{e}_1^j)\|}{\|\boldsymbol{\iota}\|}\right), \text{ where} \quad (11)$$

$$\mathbf{e}_0^j = \mathbf{p}_m^0 - \mathbf{p}_{obs}^j, \quad \mathbf{e}_1^j = \mathbf{p}_m^1 - \mathbf{p}_{obs}^j, \quad \boldsymbol{\iota} = \mathbf{p}_m^0 - \mathbf{p}_m^1,$$

and N is the number of points on the line.

The derivation for corresponding plane residuals is similar. Let $\mathbf{p}_{obs}^j$ be the j_{th} point observation on the corresponding plane in the sliding window. Let $\mathbf{p}_m^0$, $\mathbf{p}_m^1$ and $\mathbf{p}_m^2$ be three points on the plane in the map. The distance $\mathbf{d}_p$ between the two planes is then given as

$$\mathbf{d}_p = \sum_{j=1}^{N} h_1(\boldsymbol{\iota}^j \cdot \mathbf{e}_n), \text{ where } \mathbf{e}_n = \frac{\mathbf{e}_0 \times \mathbf{e}_1}{\|\mathbf{e}_0 \times \mathbf{e}_1\|}, \quad (12)$$

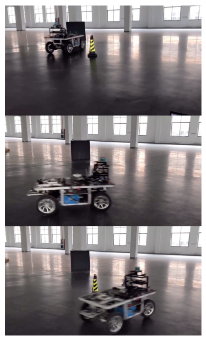
$$\mathbf{e}_0 = \mathbf{p}_m^0 - \mathbf{p}_m^1, \quad \mathbf{e}_1 = \mathbf{p}_m^2 - \mathbf{p}_{obs}^j, \quad \boldsymbol{\iota}^j = \mathbf{p}_{obs}^j - \mathbf{p}_m^0.$$

Functions $h_x(\cdot)$ denote robust Huber norms replacing the $L2$-norm.

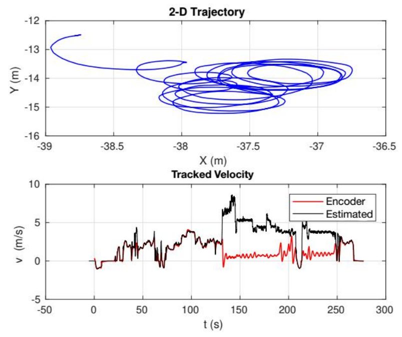
Fig. 4. Last-mile delivery vehicle used in our experiments.

VI. EXPERIMENTS

Our test platform is the delivery vehicle in Fig. 4 equipped with a low-cost MTI-3 MEMS IMU, and a 16-layer Robosense LiDAR. All experiments are conducted on this platform, which has already evolved into a commercial product. The average speed of the vehicle is about 2.5m/s. We compare our solution against the state-of-the-art LOAM [31] algorithm, for which the implementation is open source. For the sake of a fair comparison, we also give LOAM a high-quality prior global map, which is built by the LOAM mapping back-end with fixed ground truth poses. During the localization, both our method and LOAM perform pure localization. Our experiments are divided into three parts:



(a) Drift Test



(b) Performance

Fig. 5. (a) Drift test (i.e. fast *donut-circle* trajectories). (b) Velocity estimation by our fusion framework compared against IMU-encoder fusion.

1) Drift Test: In order to test the robustness of our LiDAR-IMU fusion framework, we conduct an indoor drift test on a special chassis-only car equipped with a 16-layer LiDAR, an IMU, wheel odometers, and a drift controller device (Fig. 5(a)). We choose an indoor environment with a polished ground surface that easily causes drift. We test several controlled behaviours including a *donut-circle* trajectory, an *8-figure* path, and a *side-ways drift* upon a sudden brake. The top-right sub-figure in Fig. 5(b) shows the car's complete trajectory in the horizontal plane containing more than 20 continuous circles and multiple side-drift manoeuvres. The bottom right figure in Fig. 5(b) shows the estimated linear velocity of the wheel odometry senor as well as our method. Initially, our estimated velocity is similar to the one returned by the encoder. However, after about 135s, the car starts to drift and linear velocities generally become much larger. The encoder however fails and returns very small velocities. The favourable behavior of our algorithm under drift is also

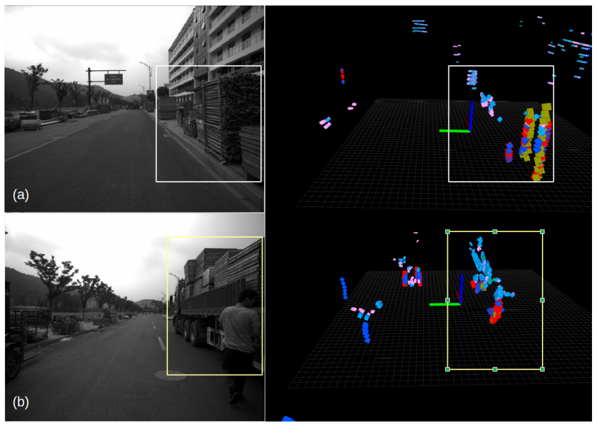


Fig. 6. Successfully tracked global map despite partial occlusions by a temporary construction site (a) and a parked truck (b) (affected regions are highlighted by bounding boxes).

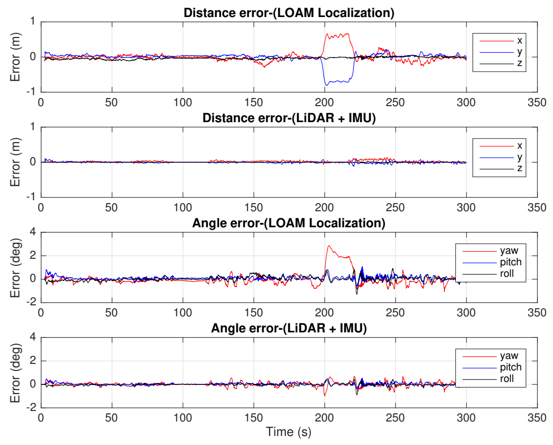


Fig. 7. Pose error comparison of our LiDAR-IMU method and LOAM [31] for testing robustness against global map occlusions. LiDAR scan obstructions occur between 200 - 220s.

introduced in our supplementary video file. Note that we do not compare against LOAM here, because the quality of the IMU priors under high-speed rotational motion degrades, thus affecting the effectiveness of the motion compensation. As a result, the point-level feature associations in LOAM become unstable, and tracking fails. As indicated by the smoothness of our result, our algorithm in turn maintains stable and accurate tracking throughout the entire sequence.

2) Occlusion Test: In our next experiment, we validate the robustness of our localization framework against occlusions. Note that this and further experiments use the UGV in Fig. 4. The occlusion test analyses several different situations, one example scenario being illustrated in Fig. 6. The left sub-figure shows an image captured during navigation. The right sub-figure depicts the occluded scene features. The first and second row show the original and current observations. In the latter view, a parked vehicle occludes the original features and produces new features that cannot be registered to the map. Point-level associations as

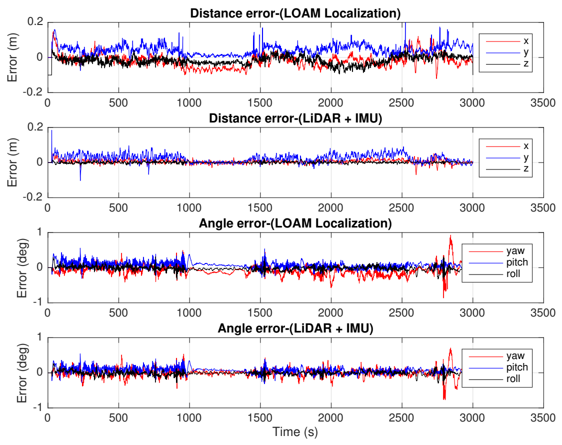


Fig. 8. Pose error comparison of our LiDAR-IMU solution and LOAM [31].

TABLE I

AVERAGE ABSOLUTE ERRORS.

mean/std	x(m)	y(m)	z(m)	yaw(deg)	pitch(deg)	roll(deg)
LOAM	0.0315	0.0480	0.0328	0.1665	0.1126	0.0618
	0.0270	0.0299	0.0228	0.1172	0.0768	0.0500
Our	**0.0085**	**0.0329**	**0.0049**	**0.1233**	**0.1054**	**0.0551**
	0.0073	**0.0125**	**0.0038**	**0.0908**	**0.0582**	**0.0473**

used in LOAM may easily generate wrong correspondences in such situations, which will impact on the localization accuracy. The proposed method however stays robust as it may for example not detect any features in the bounding boxes, and thus simply omit the addition of constraints for this part of the observations. Fig 7 shows the trajectory errors of both LOAM and our method. The scene obstructions occur between 200s -220s, where LOAM starts to produce a lot of drift. The impact of occlusions onto our method is much smaller. As can be observed in Fig. 7, the translation error only presents small fluctuations.

3) Trajectory Accuracy and Smoothness: In this experiment, the UGV runs in an industrial park. Fig. 8 shows the vehicle pose errors compared against ground truth. The latter is generated by a well-equipped vehicle as mentioned in Sec III. As can be observed, our algorithm produces better and more stable performance than LOAM, especially in terms of the vehicle position. Table I shows the corresponding absolute mean errors and their standard deviation, illustrating the substantial improvement over the reference implementation.

Besides pose accuracy, we also evaluate the smoothness of the estimated trajectories, which is an important factor for the planning, navigation and control (PNC) modules further down-stream. To evaluate trajectory smoothness, we fit third-order curves $\mathbf{C}$ to trajectory segments and calculate the root mean square trajectory error (RMSE) $\delta \mathrm{p}(t) = \|\bar{\mathbf{p}}(t) - \mathbf{C}(t)\|$ between estimated positions and the positions along the fitted curves. The detailed statistical analysis of our LiDAR-IMU fusion algorithm and LOAM is shown in Table. II. The best results are each time presented in bold. The smoothness is

TABLE II

STATISTICAL COMPARISON AGAINST THE MODIFIED LOAM SYSTEM. THE TABLE ILLUSTRATES THE ACCURACY AND SMOOTHNESS OF ESTIMATES IN DIFFERENT SCENES AS WELL AS COMPUTATIONAL RESOURCE CONSUMPTION.

Datasets	Indexes (m)	LOAM [31]	LiDAR+IMU
Road	Accuracy	0.1371	**0.1035**
	Max deviation	0.2399	**0.1524**
	Smoothness	0.0095	**0.0084**
	Max jerk	0.1352	**0.0667**
Campus	Accuracy	0.1226	**0.1031**
	Max deviation	0.2685	**0.1841**
	Smoothness	0.0125	**0.0058**
	Max jerk	0.0529	**0.0396**
Warehouse	Accuracy	0.1581	**0.0923**
	Max deviation	0.3625	**0.2137**
	Smoothness	0.0232	**0.0148**
	Max jerk	0.0982	**0.0798**
Perf.	Average (core)	170%	**70%**
	Peak (core)	220%	**95%**

evaluated on datasets captured in three different environments: an outdoor road, a campus, and a warehouse. In each environment, the vehicle navigates for more than 12h and 100km. On each dataset, *accuracy* denotes the RMSE between estimated and ground truth trajectories, *max deviation* denotes the maximum position error, and *smoothness* and *max jerk* are the above-mentioned errors between our estimated trajectories and the fitted, third-order curves $\mathbf{C}$. As can be can observed, our algorithm not only provides higher accuracy but also produces smoother trajectories. All results including further tests on a large passenger vehicle exerting high-speed forward motion can be found in the supplementary video file.

4) Computational efficiency: It is worth highlighting that our algorithm is a relatively light-weight framework. We only retain the stably tracked features from LiDAR point-clouds (normally less than 20 line features and 30 plane feature per scan) within the sliding window rather than running ICP over entire feature point clouds with thousands of points. Table II shows a performance analysis on an i7 CPU. The average CPU load for our multi-sensor fusion framework is only 70%, compared to 170% for LOAM.

VII. CONCLUSION

We have proposed a systematic solution to tightly-coupled fusion of LiDAR and IMU signals obtaining robust and precise localization performance for autonomous vehicles. The fusion framework performs well in various challenging cases, including urban canyons, indoor or GPS-denied areas. Accuracy and robustness are generally very good, and our solution has a good ability to handle partially changed environments. The performance of our localization framework is put to an ultimate test by applying it to aggressive and highly dynamic indoor trajectories captured by a drifting vehicle. Our method has been deployed on a real autonomous delivery vehicle for package delivery services, where it meets application requirements despite a low-cost sensor suite and challenging real-world environments.

References

[1] F. Aghili and A. Salerno. Driftless 3-d attitude determination and positioning of mobile robots by integration of imu with two rtk gpss. *IEEE/ASME Transactions on Mechatronics*, 18(1):21–31, 2013.

[2] N. Akai, L. Y. Morales, T. Yamaguchi, E. Takeuchi, Y. Yoshihara, H. Okuda, T. Suzuki, and Y. Ninomiya. Autonomous driving based on accurate localization using multilayer lidar and dead reckoning. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.

[3] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.

[4] X. Chen, H. Zhang, H. Lu, J. Xiao, Q. Qiu, and Y. Li. Robust slam system based on monocular vision and lidar for robotic urban search and rescue. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 41–47. IEEE, 2017.

[5] D. Chetverikov, D. Stepanov, and P. Krsek. Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *Image and vision computing*, 23(3):299–309, 2005.

[6] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus. Synthetic 2d lidar for precise vehicle localization in 3d urban environment. In *2013 IEEE International Conference on Robotics and Automation*, pages 1554–1559. IEEE, 2013.

[7] Y. Cui and S. S. Ge. Autonomous vehicle positioning with gps in urban canyon environments. *IEEE transactions on robotics and automation*, 19(1):15–25, 2003.

[8] D. Filliat and J.-A. Meyer. Map-based navigation in mobile robots:: I. a review of localization strategies. *Cognitive Systems Research*, 4(4):243–282, 2003.

[9] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. 2015.

[10] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

[11] T. Gee, J. James, W. Van Der Mark, P. Delmas, and G. Gimel'farb. Lidar guided stereo simultaneous localization and mapping (slam) for uav outdoor 3-d scene reconstruction. In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. IEEE, 2016.

[12] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang. Lips: Lidar-inertial 3d plane slam. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 123–130. IEEE, 2018.

[13] B. Guo, Q. Li, X. Huang, and C. Wang. An improved method for power-line reconstruction from point cloud data. *Remote sensing*, 8(1):36, 2016.

[14] R. Halterman and M. Bruch. Velodyne hdl-64e lidar for unmanned surface vehicle obstacle detection. In *Unmanned Systems Technology XII*, volume 7692, page 76920D. International Society for Optics and Photonics, 2010.

[15] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016.

[22] A. Sujiwo, T. Ando, E. Takeuchi, Y. Ninomiya, and M. Edahiro. Monocular vision-based localization using orb-slam with lidar-aided

[16] R. B. Langley. Rtk gps. *GPS World*, 9(9):70–76, 1998.

[17] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems*, volume 4, page 1. Citeseer, 2007.

[18] H. Liu, Q. Ye, H. Wang, L. Chen, and J. Yang. A precise and robust segmentation-based lidar localization system for automated urban driving. *Remote Sensing*, 11(11):1348, 2019.

[19] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song. L3-net: Towards learning based lidar localization for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6389–6398, 2019.

[20] M. Schreiber, H. Königshof, A.-M. Hellmund, and C. Stiller. Vehicle localization with tightly coupled gnss and visual odometry. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 858–863. IEEE, 2016.

[21] T. Shan and B. Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

mapping in real-world robot challenge. *Journal of robotics and mechatronics*, 28(4):479–490, 2016.

[23] R. Takai, O. BARAWID Jr, K. Ishii, and N. Noguchi. Development of crawler-type robot tractor based on gps and imu. *IFAC Proceedings Volumes*, 43(26):151–156, 2010.

[24] C. Vicek, P. McLain, and M. Murphy. Gps/dead reckoning for vehicle tracking in the" urban canyon" environment. In *Vehicle Navigation and Information Systems Conference, 1993., Proceedings of the IEEE-IEE*, pages 461–34. IEEE, 1993.

[25] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, and S. Song. Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4670–4677. IEEE, 2018.

[26] Z. Wang, Y. Liu, Q. Liao, H. Ye, M. Liu, and L. Wang. Characterization of a rs-lidar for 3d perception. In *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 564–569. IEEE, 2018.

[27] K. Wu, A. Ahmed, G. A. Georgiou, and S. I. Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Robotics: Science and Systems*, volume 2, 2015.

[28] M. Y. Yang and W. Förstner. Plane detection in point cloud data. In *Proceedings of the 2nd int conf on machine control guidance, Bonn*, volume 1, pages 95–104, 2010.

[29] J. Yi, J. Zhang, D. Song, and S. Jayasuriya. Imu-based localization and slip estimation for skid-steered mobile robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2845–2850. IEEE, 2007.

[30] H. Yin, Y. Wang, X. Ding, L. Tang, S. Huang, and R. Xiong. 3d lidar-based global localization using siamese neural network. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1380–1392, 2019.

[31] J. Zhang and S. Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.

[32] 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)*, pages 2174–2181. IEEE, 2015.

[33] J. Zhang and S. Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, 2017.