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

Lightweight 3-D Localization and Mapping for Solid-State LiDAR

Han Wang[1], Chen Wang[2], and Lihua Xie[1]

Abstract— **The LIght Detection And Ranging (LiDAR) sensor has become one of the most important perceptual devices due to its important role in simultaneous localization and mapping (SLAM). Existing SLAM methods are mainly developed for mechanical LiDAR sensors, which are often adopted by large scale robots. Recently, the solid-state LiDAR is introduced and becomes popular since it provides a cost-effective and lightweight solution for small scale robots. Compared to mechanical LiDAR, solid-state LiDAR sensors have higher update frequency and angular resolution, but also have smaller field of view (FoV), which is very challenging for existing LiDAR SLAM algorithms. Therefore, it is necessary to have a more robust and computationally efficient SLAM method for this new sensing device. To this end, we propose a new SLAM framework for solid-state LiDAR sensors, which involves feature extraction, odometry estimation, and probability map building. The proposed method is evaluated on a warehouse robot and a hand-held device. In the experiments, we demonstrate both the accuracy and efficiency of our method using an Intel L515 solid-state LiDAR. The results show that our method is able to provide precise localization and high quality mapping. We made the source codes public at `https://github.com/wh200720041/SSL_SLAM`.**

I. INTRODUCTION

The LIght Detection And Ranging (LiDAR) is a kind of Time of Flight (ToF) sensor which measures the object distance by emitting laser to the object and capturing the travelling time. It is one of the most popular perceptual systems in robotic applications due to its high precision, long cover range, and high reliability. Traditional localization approaches often leverage on external setup and hence lack of robustness in different scenarios. For example, UWB localization relies on the pre-installation and calibration of multiple anchors, and WiFi localization [1] requires multiple routers in order to achieve high accuracy. In comparison, LiDAR SLAM provides an external device/landmark-free localization for mobile robots in both indoor and outdoor environments [2]–[4]. Moreover, LiDAR SLAM is less affected by environment uncertainties, *e.g.*, weather change or illumination change, compared to other SLAM system such as visual SLAM [5] and visual inertial SLAM [6]–[7]. Those properties have made LiDAR SLAM widely applied to various robotic applications such as autonomous driving [8], drone inspection [9], and intelligent manufacturing [10].

This work was supported by Delta-NTU Corporate Laboratory for Cyber-Physical Systems under the National Research Foundation Corporate Lab @ University Scheme.

[1]Han Wang and Lihua Xie are with the School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798. e-mail: {wang.han,elhxie}@ntu.edu.sg

[2]Chen Wang is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. e-mail: chenwang@dr.com

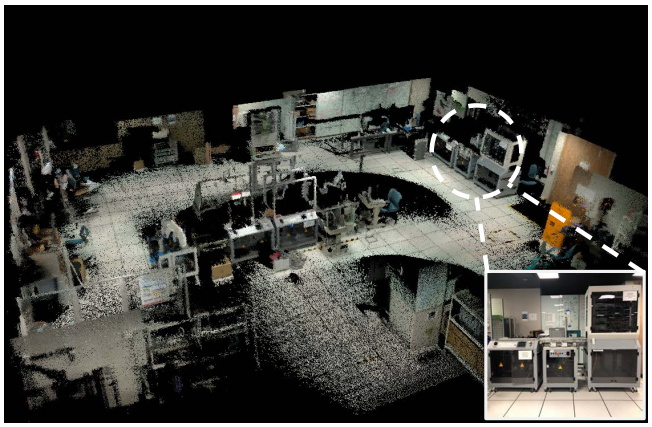

Fig. 1: Example of indoor localization and mapping in the warehouse environment. The proposed method is integrated to an AGV used for warehouse manipulation. Our method is able to provide real-time localization and dense mapping on an embedded mini computer.

Existing approaches were mainly developed for mechanical LiDAR sensors which collect the surrounding information by spinning a high frequency laser array. Although they have achieved impressive experimental results on large scale mapping [11]–[12], it is not popularly used due to the high cost. Moreover, the mechanical LiDAR is difficult to be implemented on small scale system due to its size and weight [13]. For example, the durability of flight of Unmanned Aerial Vehicles (UAVs) is significantly reduced by carrying a mechanical LiDAR for building inspection [14]. In addition, it is also not possible to integrate mechanical LiDAR into hand-held device due to its large size.

Recently, the introduction of solid-state LiDAR provides a cost-effective and lightweight solution for LiDAR SLAM system. The solid-state LiDAR is a system that built entirely on a silicon chip with no moving parts involved [15]. Hence both the size and the weight can be made much smaller than mechanical LiDAR. Moreover, the solid-state LiDAR is resistant to vibrations by removing rotating mechanical structure. The latest solid-state LiDAR only costs 10% price of a mechanical LiDAR and is as small as a smart phone, which has a great potential to become a dominant perception system for small scale application such as Virtual Reality (VR) [16], drone inspection, and indoor navigation. In terms of performance, solid-state LiDAR is able to achieve an accuracy up to 1-2 *cm* and a detectable range up to a few hundreds of meters.

Sensor	Type	Frequency	FoV	Horizontal Resolution	Vertical Resolution	Detection Range	Accuracy	Dimension	Weight
Velodyne VLP-16	Mechanical	5-20Hz	$360^\circ \times 32^\circ$	0.1-0.4°	2.0°	0.5-100 m	3 cm	103 × 72 mm	860 g
Realsense L515	Solid-state	30Hz	$70^\circ \times 55^\circ$	0.07°	0.07°	0.25-9 m	1.4 cm	61× 26 mm	95 g

TABLE I: Difference of Mechanical and Solid-state LiDAR.

Although the performance of mechanical LiDAR and solid-state LiDAR is similar, the implementation or challenge of LiDAR SLAM is different. To illustrate the difference between two LiDARs, we use Velodyne VLP-16 and Realsense L515 for example. The specifications can be found at Table I. Solid-state LiDAR has higher angular resolution, implying that the points are more dense within the same scanning area. Therefore, traditional LiDAR odometry methods such as Iterative Closest Point (ICP) [17] can be computationally inefficient since there are more points involved for calculation. Secondly, the update frequency of solid-state LiDAR is higher, while traditional LiDAR SLAM such as LOAM [18] is not computationally efficient enough to reach real-time performance. Another challenge is the pyramid-like coverage view that can cause severe tracking loss during large rotation. Therefore, a more computationally efficient and robust algorithm has to be designed in order to achieve real-time performance for solid-state LiDAR-based SLAM.

In this work, we propose a novel and lightweight SLAM framework for solid-state LiDAR, consisting of feature extraction, odometry estimation and probability map construction. Inspired by existing LiDAR SLAM methods such as LOAM [18] and LeGO-LOAM [19], we propose a new rotation-invariant feature extraction method that exploits both horizontal and vertical curvature. The proposed method provides real-time localization on mobile platforms. Thorough experiments have been conducted to evaluate its performance. The main contributions of this paper are listed as follows:

- We propose a novel SLAM framework for solid-state LiDAR, which targets to tackle perception system with small FoV and high updating frequency. We make the proposed method open sourced.
- We introduce an improved feature extraction strategy which is able to search for consistent features under significant rotations. Moreover, left Lie derivative is used for iterative pose estimation so that the pose is stored in a singularity-free format.
- A thorough evaluation of the proposed method is presented. More specifically, we integrate an Intel L515 solid state LiDAR to AGVs and test the proposed method in a complex warehouse environment. The proposed method can provide real time localization and is robust under rotations.

This paper is organized as follows: Section II reviews the related works on existing LiDAR SLAM approaches. Section III describes the details of the proposed approach, including feature point selection, laser odometry estimation, and probability map construction. Section IV shows experimental results and comparison with existing works, followed by conclusion in Section V.

II. RELATED WORK

Existing works on LiDAR SLAM mainly estimate the location by either scan-to-scan matching or feature-based matching. To match between two scan inputs, Iterative Closest Point (ICP) [20] is one of the most classic methods used. Starting with an initial pose guess, the ICP algorithm determines the point correspondence by finding the closest point in target frames. A transformation matrix between the current frame and target frame can be estimated by minimizing the Euclidean distance between point pairs. Subsequently, the derived transformation matrix leads to new correspondence relationship and new transformation matrix. And the transformation matrix converges through an iterative manner. A LiDAR scan input consists of tens of thousands of points and direct point cloud matching is computationally inefficient. Hence it is mainly implemented to LiDAR SLAM with high computational resources and low update frequency, *e.g.*, HDL-Graph-SLAM [21], and LiDAR-Only Odometry and Localization (LOL) [22]. In the meantime, raw point cloud matching is also sensitive to noise in practice. For example, in autonomous driving, the measurements from trees alongside the road are often noisy. Such noise greatly reduces the matching accuracy and causes localization drift.

A more computationally efficient way is to match in feature space where less points are involved to find the pose transformation. Zhang *et al.* [18] propose a new framework via matching on the edge features and planar features to solve the LiDAR SLAM problem for mobile robots. The edge features and planar features are collected based on local smoothness and are maintained in separate maps. And the feature correspondence is picked by finding the nearest global edges and planes respectively. The localization is estimated by minimizing the point-to-edge and point-to-plane distance. The experimental results also achieve impressive performance on public dataset evaluation such as KITTI dataset [23]. An improved version is LeGO-LOAM [19], which targets to solve the LiDAR SLAM for Unmanned Ground Vehicles (UGVs). The surrounding points from the current frame are segmented into ground points and non-ground points. For UGVs, the ground points are often planar points and those points are used to identify roll, pitch angle and z translation. The results are subsequently used as prior knowledge for edge feature matching, which targets to calculate x, y translation and yaw angle. By taking segmentation and two-stage matching, the feature points selected are

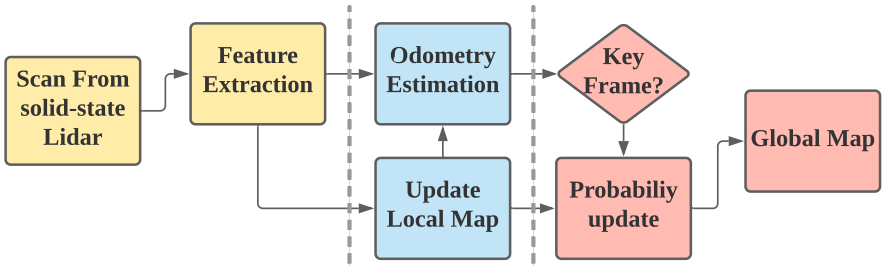

Fig. 2: System overview of the proposed method. It consists of three main modules: feature extraction, odometry estimation and probability map construction, which are highlighted in yellow, blue and red, respectively.

reduced by 40%. The experimental results show that LeGO-LOAM is able to run on small computing platform at real-time performance.

Recently some works aim to use some specially designed LiDAR with small FoV. In [15] the author proposes a new LiDAR SLAM framework for small Fov LiDAR, *i.e.*, the DJI Livox. DJI Livox spins a laser beam in cone shape direction and provides smaller FoV but denser points. To tackle the localization problem for small FoV sensor, the author introduces a new feature extraction method with the analysis of local LiDAR intensity. Compared to traditional feature extraction, the proposed feature extraction is more consistent over nearby frames which can subsequently reduce the tracking loss. However, it is mainly designed for large scale outdoor environment, while the performance in the indoor complex environment is not demonstrated.

III. METHODOLOGY

In this section, the proposed method is described in details. As shown Fig. 2, the system consists of three main modules, namely feature extraction, odometry estimation, and probability map construction. We firstly present the rotation invariant feature extraction method and then introduce odometry estimation via local feature matching. Finally we present the probability map building and scene reconstruction.

A. Feature Extraction

A solid-state LiDAR integrates all sensors on a single silicon chip without moving parts. It often has higher resolution and higher update frequency compared to a mechanical LiDAR. Hence it might be too computational heavy to match the raw point clouds. Inspired by LOAM [18], we leverage the edge and planar matching which is much more computationally efficient. Before processing the data, we remove the noisy points based on the measured distance. It is observed that readings near the maximum detection range are often less accurate due to the low reflected energy thus we pre-filter those noisy points.

The point cloud from a LiDAR scan is unordered. To compute the edge and planar features, We firstly project the point cloud into a 2-D point matrix. For the k^{th} LiDAR scan input $\mathcal{P}_k$, it is segmented based on the vertical angle and horizontal angle of each point. Given a point $\mathbf{p}_i = \{x_i, y_i, z_i\} \in \mathcal{P}_k$, the vertical angle α_i and horizontal angle θ_i are calculated by:

$$\begin{aligned} \alpha_i &= \arctan\left(\frac{y_i}{x_i}\right), \\ \theta_i &= \arctan\left(\frac{z_i}{x_i}\right). \end{aligned} \tag{1}$$

The point cloud is then segmented by equally diving vertical detection range $[\alpha_{min}, \alpha_{max}]$ and horizontal detection range $[\theta_{min}, \theta_{max}]$ into M sectors and N sectors, where $[\alpha_{min}$, α_{max}, θ_{min} and θ_{max} are the minimum vertical angle, maximum vertical angle, minimum horizontal angle, maximum horizontal angle according to the sensor specifications. Hence, the point cloud is segmented into $M \times N$ cells. For a solid-state LiDAR with vertical angular resolution of α_r and horizontal resolution of θ_r, M and N is selected as half of total points in a single direction, i.e., $M = \frac{\alpha_{max}-\alpha_{min}}{2\times\alpha_r}$ and $N = \frac{\theta_{max}-\theta_{min}}{2\times\theta_r}$. In general, larger M and N will take more computational resources. In each cell (m, n), where $m \in [1, M]$, $n \in [1, N]$ and the symbol $[1, N]$ represents $\{1, 2, \cdots, N\}$, we calculate the mean measurement $\mathbf{p}_k^{(m,n)}$ by finding the geometry center of all points in the cell.

To extract the edge and planar features, we search for its nearby points and define the local smoothness by:

$$\sigma_k^{(m,n)} = \frac{1}{\lambda^2} \cdot \sum_{\mathbf{p}_k^{(i,j)} \in \mathcal{S}_k^{(m,n)}} (||\mathbf{p}_k^{(i,j)}|| - ||\mathbf{p}_k^{(m,n)}||), \tag{2}$$

with

$$\mathcal{S}_k^{(m,n)} = \{\mathbf{p}_k^{(i,j)} | i \in [m-\lambda, m+\lambda], j \in [n-\lambda, n+\lambda]\}, \tag{3}$$

where λ is a pre-defined searching radius. A larger λ means searching for more surrounding points and requires more computational resource. The local smoothness indicates the sharpness of surrounding information. A higher $\sigma_k^{(m,n)}$ indicates that the local surface is sharp, thus the corresponding point $\mathbf{p}_k^{(m,n)}$ is taken as edge feature. A smaller $\sigma_k^{(m,n)}$ indicates that the local surface is flat, thus the corresponding point $\mathbf{p}_k^{(m,n)}$ is taken as plane feature.

B. Odometry Estimation

The odometry estimation is a task to estimate the robot current pose $\mathbf{T}_k \in SE(3)$ in global coordinate based on the historical laser scan $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_{k-1}$. Traditionally the

trajectory is estimated by either scan-to-scan match or scan-to-map match. The scan-to-scan match aligns the current frame with last frame that is faster. However, a single laser scan contains less surrounding information compared to local map and causes drift in the long run. Therefore, we use the scan-to-map match in order to improve the performance. To reduce the computational cost, sliding window approach is used. The edge features and planar features from the nearby frames are used to build the local feature maps. For the current input P_k, we define the local map $M_k = \{P_{k-1}, P_{k-2}, \cdots, P_{k-q}\}$, where q is the number of frames used to build local map.

As mentioned above, matching on raw point clouds is less efficient and sensitive to noise. Thus, we leverage in matching edge points and planar points in feature space. For each edge point $\mathbf{p}_k \in P_k$ and its transform in the local map coordinate $\hat{\mathbf{p}}_k = \mathbf{T}_k \cdot \mathbf{p}_k$, we search for the nearest edge from the local map. Note that the local map is divided into edge map and planar map, and each map is built by K-D tree in order to increase searching efficiency. Thus we can select two nearest edge feature points $\mathbf{p}_1^{\mathcal{E}}$ and $\mathbf{p}_2^{\mathcal{E}}$ from the local map for each edge point. The point-to-edge residual $f_{\mathcal{E}}(\hat{\mathbf{p}}_k)$ is defined as the distance between $\hat{\mathbf{p}}_k$ and the edge crossing $\mathbf{p}_1^{\mathcal{E}}$ and $\mathbf{p}_2^{\mathcal{E}}$:

$$f_{\mathcal{E}}(\hat{\mathbf{p}}_k) = \frac{|(\hat{\mathbf{p}}_k - \mathbf{p}_2^{\mathcal{E}}) \times (\hat{\mathbf{p}}_k - \mathbf{p}_1^{\mathcal{E}})|}{|\mathbf{p}_1^{\mathcal{E}} - \mathbf{p}_2^{\mathcal{E}}|}, \tag{4}$$

where symbol $\times$ is the cross product of two vectors. Note that if the number of nearby points is less than 2, the point-to-edge residual is not taken into accounted by the final cost. Similarly, for each planar point $\mathbf{p}_k \in P_k$, we search for the nearest planar features from the local map. To estimate a plane in 3D space, it is necessary to have 3 points. Hence for a given planar feature point $\mathbf{p}_k$ and its transform in local map coordinate $\hat{\mathbf{p}}_k = \mathbf{T}_k \cdot \mathbf{p}_k$, we can find 3 nearest points $\mathbf{p}_{\mathcal{S}}^1$, $\mathbf{p}_{\mathcal{S}}^2$, and $\mathbf{p}_{\mathcal{S}}^3$ from the local map. The point-to-plane residual $f_{\mathcal{S}}(\hat{\mathbf{p}}_k)$ is defined as the distance between $\hat{\mathbf{p}}_k$ and the plane crossing $\mathbf{p}_{\mathcal{S}}^1$, $\mathbf{p}_{\mathcal{S}}^2$, and $\mathbf{p}_{\mathcal{S}}^3$:

$$f_{\mathcal{S}}(\hat{\mathbf{p}}_k) = (\hat{\mathbf{p}}_k - \mathbf{p}_1^{\mathcal{S}})^T \cdot \frac{(\mathbf{p}_1^{\mathcal{S}} - \mathbf{p}_2^{\mathcal{S}}) \times (\mathbf{p}_1^{\mathcal{S}} - \mathbf{p}_3^{\mathcal{S}})}{|(\mathbf{p}_1^{\mathcal{S}} - \mathbf{p}_2^{\mathcal{S}}) \times (\mathbf{p}_1^{\mathcal{S}} - \mathbf{p}_3^{\mathcal{S}})|}. \tag{5}$$

Similar to the edge residual, the point-to-plane residual is not taken into account when the number of nearby points is less than 3. The final odometry is estimated by minimizing the point-to-plane residual and point-to-edge residual:

$$\arg\min_{\mathbf{T}_k} \sum f_{\mathcal{E}}(\hat{\mathbf{p}}_k) + \sum f_{\mathcal{S}}(\hat{\mathbf{p}}_k). \tag{6}$$

This non-linear optimization problem can be solved by the Gauss–Newton optimization method. We use left perturbation scheme and apply increment on Lie Group. Compared to the differentiation model in LOAM [18], there are several advantages: (i) the rotation or pose is stored in a singularity-free format; (ii) at each iteration unconstrained optimization is performed; (iii) the manipulations occur at the matrix level so that there is no need to worry about taking the derivatives of a bunch of scalar trigonometric functions, which can easily lead to errors [24]. Define $\xi_k = [\rho, \phi] \in \mathfrak{se}(3)$ and the transformation matrix:

$$\mathbf{T}_k = \exp(\xi_k^{\wedge}), \tag{7}$$

where the notation $\xi^{\wedge}$ converts a 6D vector into a 4×4 matrix by:

$$\xi^{\wedge} = \begin{bmatrix} [\rho]_{\times} & \phi \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix}, \tag{8}$$

where $[\cdot]_{\times}$ is the skew matrix of a 3D vector. The left perturbation model can be calculated by:

$$\begin{aligned} \mathbf{J}_p = \frac{\partial \mathbf{T}_k \mathbf{p}_k}{\partial \delta\xi} &= \lim_{\delta\xi \to \mathbf{0}} \frac{(\exp(\delta\xi^{\wedge}) \cdot \mathbf{T}_k \mathbf{p}_k - \mathbf{T}_k \mathbf{p}_k)}{\delta\xi} \\ &= \begin{bmatrix} \mathbf{I}_{3\times 3} & -[\mathbf{T}_k \mathbf{p}_k]_{\times} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \end{bmatrix}. \end{aligned} \tag{9}$$

Note that here $[\mathbf{T}_k \mathbf{p}_k]_{\times}$ transforms 4D point expression $\{x, y, z, 1\}$ into 3D point expression $\{x, y, z\}$ before calculating the skew matrix. The Jacobian matrix of point-to-edge residual is defined by

$$\begin{aligned} \mathbf{J}_{\mathcal{E}} &= \frac{\partial f_{\mathcal{E}}(\hat{\mathbf{p}}_k)}{\partial \hat{\mathbf{p}}_k} \cdot \mathbf{J}_p \\ &= \mathbf{p}_n^T \cdot \frac{\mathbf{J}_p \times (\hat{\mathbf{p}}_k - \mathbf{p}_1^{\mathcal{E}}) + (\hat{\mathbf{p}}_k - \mathbf{p}_2^{\mathcal{E}}) \times \mathbf{J}_p}{|\mathbf{p}_1^{\mathcal{E}} - \mathbf{p}_2^{\mathcal{E}}|} \\ &= \mathbf{p}_n^T \cdot \frac{(\mathbf{p}_1^{\mathcal{E}} - \mathbf{p}_2^{\mathcal{E}})}{|\mathbf{p}_1^{\mathcal{E}} - \mathbf{p}_2^{\mathcal{E}}|} \times \mathbf{J}_p, \end{aligned} \tag{10}$$

where

$$\mathbf{p}_n = \frac{(\hat{\mathbf{p}}_k - \mathbf{p}_2^{\mathcal{E}}) \times (\hat{\mathbf{p}}_k - \mathbf{p}_1^{\mathcal{E}})}{|(\hat{\mathbf{p}}_k - \mathbf{p}_2^{\mathcal{E}}) \times (\hat{\mathbf{p}}_k - \mathbf{p}_1^{\mathcal{E}})|}. \tag{11}$$

Similarly, we can derive the Jacobian matrix of point-to-plane residual:

$$\begin{aligned} \mathbf{J}_{\mathcal{S}} &= \frac{\partial f_{\mathcal{S}}(\hat{\mathbf{p}}_k)}{\partial \hat{\mathbf{p}}_k} \cdot \mathbf{J}_p \\ &= \frac{[(\mathbf{p}_1^{\mathcal{S}} - \mathbf{p}_2^{\mathcal{S}}) \times (\mathbf{p}_1^{\mathcal{S}} - \mathbf{p}_3^{\mathcal{S}})]^T}{|(\mathbf{p}_1^{\mathcal{S}} - \mathbf{p}_2^{\mathcal{S}}]) \times (\mathbf{p}_1^{\mathcal{S}} - \mathbf{p}_3^{\mathcal{S}})|} \cdot \mathbf{J}_p. \end{aligned} \tag{12}$$

The alignment of current scan and local map may not be ideal at the beginning. Therefore, it is necessary to search for better feature correspondences. The optimal feature correspondence can be found through an iterative manner, *i.e.*, with an initial pose $\mathbf{T}_k^0$ and initial correspondence based on $\mathbf{T}_k^0$, we can derive the odometry estimation $\mathbf{T}_k^1$ then $\mathbf{T}_k^2$ and $\mathbf{T}_k^3$, *etc*. This finally converges to the optimal estimation of current pose. Although iterative calculation is computationally inefficient, a good estimation of initial pose alignment is able to speed up the convergence. To find an better initial alignment, we assume constant angular velocity and linear translation:

$$\begin{aligned} \mathbf{T}_k^0 &= \mathbf{T}_{k-1} \cdot \mathbf{T}_{k-2}^{k-1} \\ &= \mathbf{T}_{k-1} \cdot \mathbf{T}_{k-2}^{-1} \cdot \mathbf{T}_{k-1}. \end{aligned} \tag{13}$$

The process of iterative odometry estimation is listed in Algorithm 1.

Algorithm 1: Pose estimation for Solid-state LiDAR

Input : Current scan $\mathcal{P}_k$, the robot trajectory $\mathbf{T}_{1:k-1}$
Output: Current pose $\mathbf{T}_k$

begin
  if *Not Initialized* **then** $\mathbf{T}_0 \leftarrow \mathbf{0}$;
  Calculate the local smoothness and extract edge features and planar features ;
  Compute initial alignment $\mathbf{T}_k^0 \leftarrow \mathbf{T}_{k-1}\mathbf{T}_{k-2}^{-1}\mathbf{T}_{k-1}$;
  for *Pose optimization is not converged* **do**
    for *each point* $\boldsymbol{p}_k \in \mathcal{P}_k$ **do**
      Transform to map coordinate $\hat{\mathbf{p}}_k \leftarrow \mathbf{T}_k^{i-1}\mathbf{p}_k$;
      if p_k *is an edge feature* **then**
        Compute edge residual $f_{\mathcal{E}}$ and Jacobian matrix $J_{\mathcal{E}}$;
      end
      if $\mathbf{p}_k$ *is a planar feature* **then**
        compute planar residual $f_{\mathcal{S}}$ and Jacobian matrix $J_{\mathcal{S}}$;
      end
    end
    if *nonlinear optimization converges* **then**
      Compute $\mathbf{T}_k^i$;
    end
  end
  Update current scan into local map and remove oldest scan from local map;
  Return current pose $\mathbf{T}_k$;
end

C. Probability Map Construction

The global map is often of large size and it is not computationally efficient to update it with each frame. Therefore, we only use key frames to update and reconstruct map. The key frames are selected based on the following criterion: (1) If the displacement of robot is significant enough (*i.e.*, greater than a pre-defined threshold); (2) If change of rotation angle (including roll, pitch, yaw angle change) is large; (3) If the time elapsed is more than a certain period. In practice, the rotation and translation thresholds are defined based on the FoV of the sensor, while the minimum update rate is defined based on the computational power of the processor. To increase the searching efficiency, an octree is used to construct the global map. It only takes computational complexity of $\mathcal{O}(\log n)$ to search for a specific node from an octree of depth n, which can significantly reduce mapping cost [25]. For each cell in octree, we use $P(n|z_{1:t})$ to present the probability of the existence of an object [26]:

$$P(n|z_{1:t}) = \left[1 + \frac{1-P(n|z_t)}{P(n|z_t)} \cdot \frac{1-P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \cdot \frac{P(n)}{1-P(n)}\right]^{-1}, \tag{14}$$

where z_t is the current measurement, $z_{1:t-1}$ is the historical measurements from key frames, $P(n)$ is the prior probability, which is set to 0.5 for unknown area.

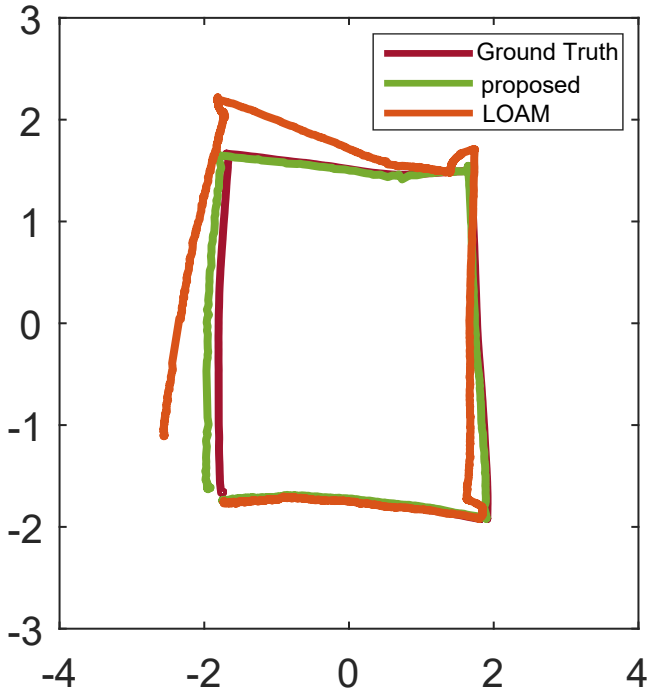

Fig. 3: Comparison among the proposed method, LOAM and ground truth. LOAM loses tracking when the rotation speed is high while the proposed method is able to track accurately. The units are in meters.

IV. EXPERIMENT EVALUATION

A. Experiment Setup

In this section we present experimental results of the proposed method. Our method is firstly evaluated in a room equipped with a VICON system. Then, it is implemented on an Automated Guided Vehicle (AGV) that is used for warehouse manipulation. We analyze the performance of the proposed method and compare it with existing LiDAR SLAM. To further illustrate the robustness, the proposed method is also integrated into a hand-held device used for 3D scanning. In our experiment, an Intel Realsense L515 is used for demonstration. It is a small FoV solid-state LiDAR with $70^\circ \times 55^\circ$ viewing angle and 30 Hz update frequency. It is smaller and lighter than a smart phone so that it can be used in many mobile robotic platforms. The algorithm is coded in C++ and is implemented on Ubuntu 18.04 and ROS Melodic [27]. In the first experiment, the proposed method is tested on a desktop PC with an Intel 6-core i7-8700 processor CPU. For the experiment on the AGV and hand-held device, an Intel NUC mini computing platform is used which has an i5-10210U processor.

B. Performance Evaluation and Comparison

To evaluate the localization results, our method is compared with the ground truth, which is provided by a VICON

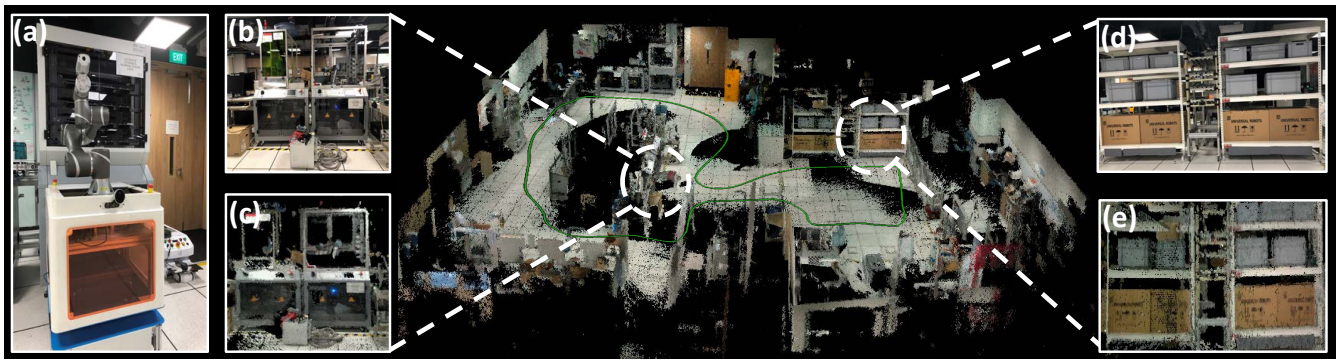

Fig. 4: Example of indoor localization and mapping in warehouse environment. (a) The AGV platform used for warehouse manipulation with a solid-state LiDAR mounted in the front. And the reconstructed map is shown in the middle. We randomly picked two places for illustration. (b) and (d) are the original camera view. (c) and (e) are reconstructed scene based on the proposed approach. The trajectory is plotted in green

system. A robot is manually controlled to move around the VICON room of size 4m $\times$ 4m. The result is shown in Fig. 3, where the trajectories of ground truth and our method are plotted in red and green, respectively. The average computing time is 31 ms per frame. Our method achieves a translational error of 5*cm*. We also compared our method with LOAM [18] that is widely used for LiDAR SLAM. The vertical angle and horizontal angle inputs in LOAM are changed according to the sensor properties of L515, while we keep the number of edge and planar features unchanged. The result of LOAM is plotted in orange. It is obvious that tracking loss happens to LOAM when there is large rotation, while our method is still able to track accurately.

C. Performance on Warehouse Robot

The algorithm is then evaluated on an AGV running in a warehouse environment. Smart manufacturing is one of the main applications for SLAM. In an advanced factory, the robot is supposed to transport, process, and assemble products automatically. This requires the robot to efficiently localize itself in complex and highly dynamic environments with moving operators and other robots. In this experiment, the proposed method is integrated into an industrial AGV that is shown in Fig. 4 (a). The robot is used for griping and transporting the materials with a robot arm on the top. A solid-state LiDAR is mounted in the front to provide localization. In this task, the AGV platform automatically explores the warehouse and reconstructs the environment at the maximum speed of 0.8 m/s. The localization and mapping results are shown in Fig. 4 (b) with the robot trajectory plotted in green. Our method achieves an average computing time of 42 *ms* on an Intel NUC mini computing platform. It can be seen that our method is able to provide reliable localization in complex environments. We compare the 3D mapping results with the images. The image view of the original warehouse is shown in Fig. 4 (b), (d) and the built 3D map is shown in Fig. 4 (c), (e). To evaluate the quality of mapping, we compare the built objects with

Fig. 5: Integration of the proposed method on a hand-held scanner. (a) Light weight hand-held scanner using solid-state LiDAR as perception system. (b) Localization and mapping result, the trajectory is plotted in green.

actual objects. Specifically, we measure the size of operating machines which are shown in Fig.4 (b) and Fig.4 (d). The edge points from built object are picked and the Euclidean distances between edge points are measured. The results are calculated by taking the average of the measured distances. The measured dimension and actual size of the operation machine in Fig. 4 (b) are $1.17m \times 1.88m$ and $1.15m \times 1.85m$, while the measured dimension and actual size of operation machine in Fig. 4 (d) are $1.16m \times 1.94m$ and $1.16m \times 1.95m$. It can be seen that our approach is also able to build a high resolution 3D map that is close to real environment.

Method	Success	Tracking loss
The proposed method	6	0
A-LOAM	1	5

TABLE II: Results of rotation test.

D. Performance on Hand-held Device

1) 3D Mapping: To further demonstrate the robustness, the proposed method is also evaluated on a hand-held device. With the growing of Virtual Reality (VR), Augmented Reality (AR), and gaming industry, SLAM has been implemented on various mobile devices such as smart phones and VR glasses. However, most mobile platforms have limited computational resources. Compared to implementation on warehouse robot, the hand-held device suffers from vibration and large viewing angle change which can cause tracking loss and localization failure. Our system is built as a mobile scanner which is shown in Fig. 5(a). The hand-held scanner consists of three modules: perception module, rotation platform, and computing module. It is less than 500 grams and can be implemented on many applications such as drone navigation. In the experiment, we hold the 3D scanner and scan the indoor environment at normal walking speed. The localization and mapping result is shown in Fig. 5(b), with the trajectory plotted in green. Our method can accurately localize itself and perform mapping in real time.

2) Rotation Test: The hand-held device often has higher rotation changes which can result in tracking loss. In order to demonstrate the performance of the proposed method under large rotation, we put the solid-state LiDAR in horizontal orientation and randomly rotate the solid-state LiDAR with maximum rotation speed of 1.57 rad/s. The solid-state LiDAR returns to horizontal orientation and we record the angle deviation. Tracking loss is considered when the final angle deviation is greater than 10 degrees. We compare our method with A-LOAM and the results are shown in Table II. It can be seen that our method has a higher success rate compared to A-LOAM.

V. CONCLUSION

In this paper, we present a full SLAM framework for solid-state LiDAR which is an emerging LiDAR system with higher update frequency and smaller FoV compared to traditional mechanical LiDAR. Our system mainly consists of rotation invariant feature extraction, odometry estimation, and probability map construction. The propose method is able to support real time localization and densed mapping on an embedded mini PC. Thorough experiments have been performed to evaluate the proposed method, including experiments on a warehouse AGV and a hand-held mobile device. The results show that the proposed method is able to provide reliable and accurate localization and mapping at high frequency. It can be implemented on most of mobile platform such as UAV and hand-held scanner. We have made the source codes publicly available.

REFERENCES

[1] H. Zou, M. Jin, H. Jiang, L. Xie, and C. J. Spanos, "WinIPS: WiFi-based non-intrusive indoor positioning system with online radio map construction and adaptation," *IEEE Transactions on Wireless Communications*, vol. 16, no. 12, pp. 8118–8130, 2017.

[2] H. Hong and B. H. Lee, "Probabilistic normal distributions transform representation for accurate 3d point cloud registration," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3333–3338.

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

[4] J.-E. Deschaud, "IMLS-SLAM: scan-to-model matching based on 3D data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2480–2485.

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

[6] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[7] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[8] S. Yuan, H. Wang, and L. Xie, "Survey on localization systems and algorithms for unmanned systems," *Unmanned Systems*, vol. 09, no. 02, pp. 129–163, 2021.

[9] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6423–6430.

[10] H. Wang, C. Wang, and L. Xie, "Intensity scan context: Coding intensity and geometry relations for loop closure detection," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2095–2101.

[11] 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)*, 2020.

[12] 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)*, 2015, pp. 2174–2181.

[13] H. Wang, M. Cao, H. Jiang, and L. Xie, "Feasible computationally efficient path planning for uav collision avoidance," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, 2018, pp. 576–581.

[14] L. Wallace, A. Lucieer, C. Watson, and D. Turner, "Development of a UAV-LiDAR system with application to forest inventory," *Remote sensing*, vol. 4, no. 6, pp. 1519–1543, 2012.

[15] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3126–3131.

[16] G. C. Burdea and P. Coiffet, *Virtual reality technology*. John Wiley & Sons, 2003.

[17] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[18] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in real-time." in *Robotics: Science and Systems*, vol. 2, 2014, p. 9.

[19] 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)*, 2018, pp. 4758–4765.

[20] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Object recognition supported by user interaction for service robots*, vol. 3. IEEE, 2002, pp. 545–548.

[21] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LiDAR-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, pp. 1–16, 2019.

[22] D. Rozenberszki and A. Majdik, "LOL: Lidar-only Odometry and Localization in 3d point cloud maps," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
[24] T. Barfoot, *State Estimation for Robotics: A Matrix Lie Group Approach*. Cambridge University Press, 2017.
[25] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.
[26] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 116–121.
[27] M. Quigley, B. Gerkey, and W. Smart, *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O'Reilly Media, 2015.