sensors

MDPI

Communication

Real-Time 2-D Lidar Odometry Based on ICP

Fuxing Li, Shenglan Liu *, Xuedong Zhao and Liyan Zhang

College of Mechanical & Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; lfx_mail@163.com (F.L.); cnc@nuaa.edu.cn (X.Z.); zhangly@nuaa.edu.cn (L.Z.)
* Correspondence: meeslliu@nuaa.edu.cn

Abstract: This study presents a 2-D lidar odometry based on an ICP (iterative closest point) variant used in a simple and straightforward platform that achieves real-time and low-drift performance. With a designated multi-scale feature extraction procedure, the lidar cloud information can be utilized at multiple levels and the speed of data association can be accelerated according to the multi-scale data structure, thereby achieving robust feature extraction and fast scan-matching algorithms. First, on a large scale, the lidar point cloud data are classified according to the curvature into two parts: smooth collection and rough collection. Then, on a small scale, noise and unstable points in the smooth or rough collection are filtered, and edge points and corner points are extracted. Then, the proposed tangent-vector-pairs based on edge and corner points are applied to evaluate the rotation term, which is significant for producing a stable solution in motion estimation. We compare our performance with two excellent open-source SLAM algorithms, Cartographer and Hector SLAM, using collected and open-access datasets in structured indoor environments. The results indicate that our method can achieve better accuracy.

Keywords: 2-D lidar; multi-scale; feature extraction; motion estimation

Citation: Li, F.; Liu, S.; Zhao, X.; Zhang, L. Real-Time 2-D Lidar Odometry Based on ICP. *Sensors* **2021**, *21*, 7162. https://doi.org/10.3390/s21217162

Academic Editor: Antonia Spano'

Received: 13 September 2021
Accepted: 25 October 2021
Published: 29 October 2021

1. Introduction

Simultaneous localization and mapping (SLAM) technology has developed rapidly and is widely used. Whether in automated driving [1–3] or unmanned industrial transportation, or service robots, SLAM covers most scenarios from high-speed to low-speed motion. The field of service robots mainly includes security robots, guide robots, sweeping household robots, etc. The application scenarios range from a broad outdoor environment to a limited indoor environment. Localization is the essential task to be completed in the motion estimation module in SLAM [4]. Although various sensors are used for motion estimation, such as IMU, GNSS, and camera [5], lidar is still highly competitive in SLAM applications, thanks to its high measurement accuracy, high anti-interference ability, and stability to light illumination.

This paper studies the localization of service robots using 2-D lidar in indoor environments. For lidar SLAM, filtering-based algorithms were first widely used and applied to 2-D lidar [6]. The most likely pose of a robot is estimated with the distribution of sequential sensor data. Such techniques are usually sensitive to computing resources. With the progress of nonlinear solvers, lidar SLAM gradually adopts optimization methods to solve motion estimation, such as KartoSLAM [7], HectorSLAM [8], LagoSLAM [9], and Cartography [10]. These methods do not deliberately extract features when performing data association but use raw data to perform motion estimation directly. In 3-D lidar SLAM, downsampling and feature extraction are performed in the data preprocessing stage to obtain less noisy and more stable data. This difference in data processing is also due to the fact that 3-D lidar produces more information than 2-D lidar, so it is objectively more difficult to extract effective and rich features from 2-D lidar data. Moreover, a general single-scale feature extraction procedure can easily lead to information loss.

For localization tasks in indoor structured environments, 2-D lidar SLAM is sufficient to solve most problems; however, minimal attention has been given to multi-scale feature extraction of the front-end of 2-D lidar SLAM. Therefore, this paper focuses on a 2-D lidar-based approach.

2. Related Work

The essence of the localization task is to match the sequential sensor data, known as scan-matching, which can be solved by the optimization-based method. The method usually forms a graph model with vertices and edges, in which the vertices are the variables to be optimized, and the edges are the constraint relationship between the vertices [11], and then the model can be transformed into a least-squares problem.

ICP [12] and its variants [13–16] can be used to match lidar point clouds between different frames. These techniques vary from point-to-point to point-to-plane in data correspondence, and some remove obvious outliers with geometric features, such as normal vector and curvature. Some of them are specially designed for 3-D data. For example, GICP [14] uses the correspondence between points and planes, and NICP [15] applies the 3-D structure around the points to form data association. In 2-D lidar data, it is difficult to find and utilize geometric features, such as a plane or a normal vector to a plane. Moreover, it is time-consuming for the process of finding the closest points. Tian et al. [17] proposed a method for the acceleration of data correspondence with an assisted intensity to reduce the computational cost and avoid divergences.

Apart from scan-matching, some methods focus on the distribution of lidar point clouds in the environment to estimate the pose of a robot. For example, Gmapping [18] and CoreSLAM [19] both utilize improved particle filters for localization, but they are sensitive to computing resources or have poor positioning performance. Ren et al. [20] proposed an improved correlative scan matching in a probabilistic framework, which can evaluate a robot's pose with the distribution of lidar data.

LagoSLAM [9] and KartoSLAM [7] construct an SLAM problem with a pose graph model, which finds what pose state the robot is in that is most likely to obtain the current observations. HectorSLAM [8] and Cartographer [10] match the point cloud with the map. The scan matcher also needs to find the most likely position of the laser point on the map to retrieve the pose of a robot. The improved HectorSLAM and a control network constraint-based method [21] focus on reducing the cumulative error in the backend [22].

These scan-matching methods or pose estimation techniques designed for 2-D SLAM usually do not specially extract features for 2-D lidar data; some non-essential data are retained when processing lidar data. While feature extraction is common in 3-D lidar SLAM, it is a crucial part of the front end in LOAM [23] and LeGO-LOAM [24]. LOAM extracts edge points and plane points and uses the distance from the point to line and point to plane for data association. LeGO-LOAM segments the point cloud in advance before feature extraction, which greatly reduces noise and improves the efficiency of feature extraction. Grant et al. [25] proposed a new planar feature extraction method for fast point cloud registration. When faced with a large number of dense clouds, one can apply bundle adjustment to lidar SLAM to handle large-scale edge and plane features [26] or use sliding windows to increase the speed of the system [27].

These feature extraction procedures only extract features on a single scale, and the information in the point cloud is not taken into account sufficiently, such as PCA-based line detection [28]. One of the advantages of multi-scale feature extraction is that various levels of information can be retrieved. The construction of geometric features is usually very sensitive to the size of the neighborhood, and the size of the scale is not easy to control. Conversely, the data structure of multi-scale features can be used to construct an effective search space or to omit the secondary search process, which greatly accelerates the speed of data association. Pfister et al. [29] used 2-D lidar data to construct multi-scale point and line features and adopted a coarse-to-fine strategy to traverse matching points, significantly reducing the search space for fine-scale features and improving the robustness

of scan-matching. Wu et al. [30] extracted the multi-scale corner features with curvature detector in 2-D lidar data. To better classify the laser point cloud, Hang et al. [31] extracted multi-scale features and embedded them in a low-dimensional and robust subspace to obtain a more compact geometric feature representation and achieve a better classification performance. Bosse et al. [32] designed a feature descriptor that captured the local structure, and the descriptor is robust to measurement noise.

This paper considers the influence of noise on sensor data, applies multi-scale metrics to achieve multi-level and robust feature extraction, and then combines with the ICP and the proposed tangent-vector pair to obtain robust motion estimation. Due to the multi-scale data structure, searching and relating data can be carried out quickly, which can save time in the iterative solution process.

Naturally, during the movement of lidar, one inherent problem is point cloud distortion. It takes time for lidar to transmit and receive signals, which results in the beams inevitably being received at different positions during the lidar movement. Moreover, the more intense the movement, the more pronounced the change. This phenomenon is known as motion distortion. Generally, auxiliary sensors are required to calculate the distorted movement of each point by linear interpolation and remove the distortion. Another common way is to apply the extended Kalman filter (EKF) to integrate lidar, IMU, and other information [33,34] to the de-skew cloud. However, when the sensor movement is not so intense, it is acceptable to ignore this movement distortion. Or, it can be assumed that the sensor moves at a constant speed to calculate the movement of each point. Specifically, in our case, the lidar moves at a low speed of 0.3–0.5 m/s, so we ignore the distortion.

3. System Overview

3.1. Lidar Hardware

As shown in Figure 1, this paper utilizes an RPLIDAR S1 manufactured by SLAMTEC. The lidar has a horizontal field of view of 360 degrees, a measurement distance of 0.1–40 m, a scanning frequency of 10 Hz, a measurement resolution of 3 cm, and a measurement accuracy of ± 5 cm. All obtained point clouds are in the lidar coordinate $\{L\}$ of the left hand.

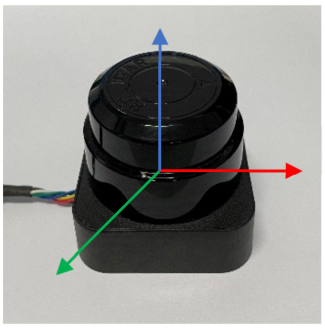


Figure 1. Single-line mechanical rotating lidar.

3.2. Software System Overview

Downsampling raw data and eliminating outliers can reduce the burden of computational resources and improve the real-time performance of the system; therefore, this paper designs a multi-scale feature extraction procedure that can reserve the appropriate data. In Figure 2, we first classify the lidar data into two parts, smooth collection $\mathcal{P}_s$ and rough collection $\mathcal{P}_r$, according to their curvature defined in Formula (1). Then, the corresponding point pairs $< \boldsymbol{p}_s, \boldsymbol{p}'_s >$ and $< \boldsymbol{p}_r, \boldsymbol{p}'_r >$ are obtained between the sequential sensor data. Respectively, the edge point $\boldsymbol{p}_e$ and corner point $\boldsymbol{p}_c$ are extracted from the smooth collection and rough collection according to the constraint conditions we proposed in Formulas (3) and (4). In addition, the point pairs $< \boldsymbol{p}_e, \boldsymbol{p}'_e >$ and $< \boldsymbol{p}_c, \boldsymbol{p}'_c >$ are obtained, and they yield the tangent-vector pairs $< \boldsymbol{\tau}, \boldsymbol{\tau}' >$. The motion estimation problem is solved by our ICP variant algorithm proposed in Formula (6). Then, the transformation between two lidar frames can be obtained, and the current lidar pose-state in the world coordinate is established.

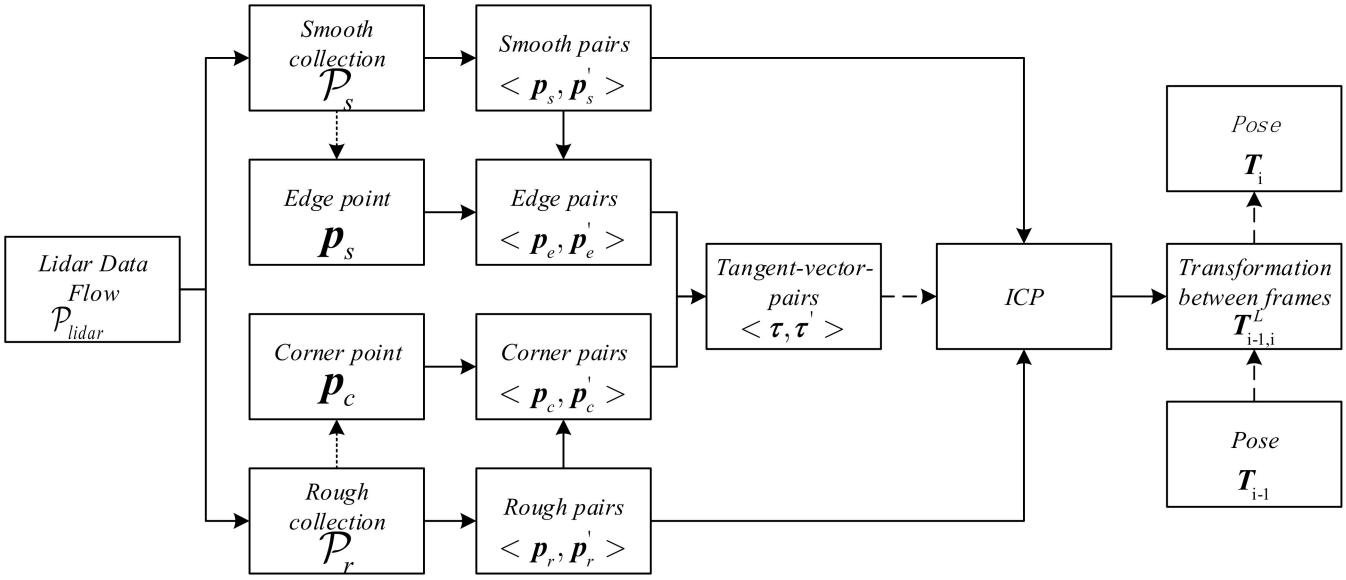


Figure 2. i represents the current frame, and $i-1$ represents the previous frame. $T^L_{i-1,i}$ is the transformation matrix between frame i and $i-1$ in the coordinate system $\{L\}$.

4. Lidar Odometry

4.1. Feature Extraction

Central to our method is a multi-scale metric procedure that allows feature analysis at multiple scales, using the local neighborhood's size as a scale parameter.

On a large scale, we evaluate the smoothness of a point using a neighborhood composed of more points. The calculation process similar to the method in [20] is Formula (1) and we set $|\mathcal{C}|$ to 10. When $\kappa_1 > 0.1$, a point is classified to rough collection; otherwise, it is classified to smooth collection. However, the neighborhood's rough points or sensor noise can easily affect the actual smooth point, which is detected as a rough point. Although, when having a large sample size in the calculation, a few misclassified points will not significantly affect the final result. For example, when we use the ICP method, we will not deliberately exclude such points because enough correct correspondence pair samples are involved in the evaluation:

$$\kappa_1 = \frac{1}{|\mathcal{C}|} \left\| \sum_{j \in \mathcal{C}, j \neq k} \left(p_j - p_k \right) \right\| \tag{1}$$

When the total sample size becomes smaller, such misclassified points have a more significant impact. We must consider the data on a smaller scale to utilize two adjacent points to compute the tangent vector in Formula (2) and then define the straightness similar to the method of [35] in Formula (3) to evaluate the smoothness of a point in a small area. Moreover, we use the tangent vector difference in Formula (4) to evaluate the extent of noise influence on points because the sensor has the measurement accuracy, for example, the lidar measurement accuracy we use is ± 5 cm. When the lidar scans a plane, we clearly notice that the measuring point is throbbing in a specific range, and this type of point is easily recognized as a rough point, but it belongs to a point on a straight line. Therefore, we implement the small-scale extraction for the points classified by the large-scale feature extraction, and the filtered points are used for rotation estimation in Section 4.2.

In Formula (2), τ_k is the tangent vector of the k-th point p_k. In Formula (3), l_k is the straightness of the k-th point p_k. $\mathcal{C}_k$ is a collection of the neighborhood and $|\mathcal{C}_k|$ is set to two. In Formula (4), κ_2 represents the difference between two sequential tangent vectors:

$$\tau_k = \frac{p_{k+1} - p_k}{\| p_{k+1} - p_k \|} \tag{2}$$

$$l_k = \boldsymbol{\tau}_k \cdot \frac{1}{|\mathcal{C}_k|} \sum_{j \in \mathcal{C}_k, j \neq k} \left(\boldsymbol{p}_j - \boldsymbol{p}_k \right) \tag{3}$$

$$\kappa_2 = \frac{\angle \boldsymbol{\tau}_{k-1} \boldsymbol{\tau}_k}{\left\| \frac{\boldsymbol{p}_{k+1} - \boldsymbol{p}_k}{2} \right\| + \left\| \frac{\boldsymbol{p}_k - \boldsymbol{p}_{k-1}}{2} \right\|} \tag{4}$$

Generally, in the smooth collection, when the straightness of a point is very large ($l_k > 1.5$), the k-th point and its neighbors should be on a straight line. If the influence of the noise is small ($\kappa_2 < 0.5$), the point should be classified as an edge point; in the rough collection, when the straightness of a point is very small ($l_k < 1$), it may be at corners. If the "noise" is considerable ($\kappa_2 > 1$), it means that the tangent vector changes significantly. The point should be classified as corner points. In this way, we have a small number of more stable point sets by small-scale classification. Figure 3 illustrates a schematic diagram of the different results produced by small-scale and large-scale neighborhoods. Multi-scale classification can exclude some single-scale misclassification points. A specific example is shown in Figure 4.

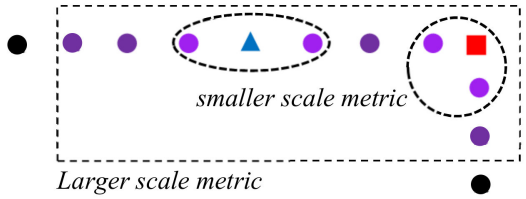


Figure 3. The blue triangle is an edge point, and the red square is a corner point. When classified at a large scale, the edge point is likely to be classified into corners.

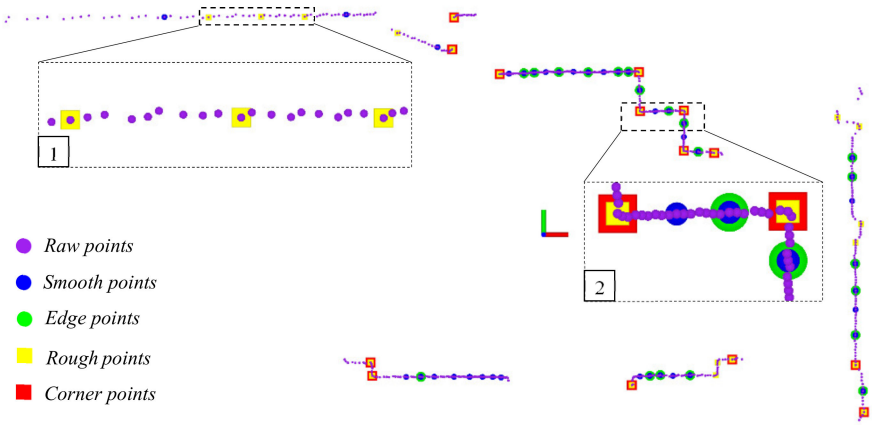


Figure 4. The result of the multi-scale metric procedure. In dotted box 1, the raw points are not strictly distributed in a straight line due to the influence of noise, and we find large-scale metric results in misclassified points. The yellow rectangle should not belong to rough points. In dotted box 2, the red corner points and green edge points are in the correct category.

4.2. Motion Estimation

The raw data are classified into four categories: smooth collection, rough collection, edge point, and corner point. The large-scale metric classified data, smooth collection and rough collection, are used as the input of the ICP method. In ICP, the KD-tree is used to construct a search space to find the relevant point, that is, the nearest point, and then iterated until convergence or the maximum number of iterations is reached. During the iteration, the method first finds the closest point and then generates a preliminary related point pair, and Formula (5) must be met; otherwise, the point pair is discarded:

$$\left\| \kappa_1 - \kappa_1' \right\| < \varepsilon_{\kappa_1} \tag{5}$$

In our experimental setup, we set ε_{κ_1} to 0.005. Then, all the point pairs $< \boldsymbol{p}_s, \boldsymbol{p}'_s >$ and $< \boldsymbol{p}_r, \boldsymbol{p}'_r >$ build up the set $\mathcal{P}$. We use $\boldsymbol{M}_1 \in SE(3)$ to represent the transformation matrix and the corresponding Lie algebra is $\boldsymbol{\xi} \in se(3)$. The formula of ICP is written in the form:

$$r_1(\mathcal{P}, \boldsymbol{\xi}) = \boldsymbol{p}_n - exp\left(\hat{\boldsymbol{\xi}}\right)\boldsymbol{p}'_n \tag{6}$$

where $exp(\hat{\boldsymbol{\xi}}) \rightarrow \boldsymbol{M}_1$ is the mapping from Lie algebra to Lie group, and $< \boldsymbol{p}_n, \boldsymbol{p}'_n >$ is the pair in $\mathcal{P}$.

For small-scale classified point cloud data, edge points and corner points are collected from the points extracted from large-scale features, and the relationship between points is built. However, all point pairs must meet the conditions of the following Formulas (3) and (4) or are discarded:

$$\left\|\kappa_2 - \kappa'_2\right\| < \varepsilon_{\kappa_2} \tag{7}$$

$$\left\|l_k - l'_k\right\| < \varepsilon_{l_k} \tag{8}$$

In practice, ε_{κ_2} and ε_{l_k} are set to 0.05. However, this paper does not directly use edge points and corner points but uses these points to calculate the tangent-vector pair to estimate the degree of freedom of rotation. First, we explain how to use the tangent-vector pair to calculate the rotation angle. In the left of Figure 5a, there are several feature points, which can be edge points or corner points, and then their tangent vectors are calculated. In this paper, these tangent vectors are connected end to end. These feature points are stable, so the shape of the formed polygon is also stable, which is why the points generated by the large-scale classification method are not used. It can be assumed that a polygon rotates slightly on the plane. If we know the rotation angle of each side, then the rotation angle of the polygon can be easily calculated. Here, the sides of the polygon are the tangent vectors. For the convenience of presentation, all unit tangent vectors are moved to the origin of the coordinate system to form a tangent indicatrix, which can abstractly represent the distribution state of polygon sides, and further omit line segments and simplify into an indicatrix. In practice, the shape of the polygon represented by the indicatrix extracted from the previous frame and the indicatrix of the current frame should be basically the same, but the rotation angle is different, which is why the tangent-vector pair can estimate the degree of freedom of rotation. A specific example is shown in Figure 5b,c.

For the rotation matrix $\boldsymbol{R}_2 \in SO(3), exp(\hat{\boldsymbol{\phi}}) \rightarrow \boldsymbol{R}_2$, and the relative transformation matrix $\boldsymbol{M}_2 \in SE(3)$ in Formula (9). The corresponding tangent vector pair is $< \boldsymbol{\tau}_k, \boldsymbol{\tau}'_k >$. The solution for rotation of tangent-vector pairs is in Formula (10) and $< \boldsymbol{\tau}_m, \boldsymbol{\tau}'_m > \in \mathcal{T}$:

$$\boldsymbol{M}_2 = \begin{bmatrix} \boldsymbol{R}_2 & \boldsymbol{t}_2 \\ 0^\top & 1 \end{bmatrix}, t_2 = 0, \boldsymbol{M}_2 \in SE(3) \tag{9}$$

$$r_2(\mathcal{T}, \phi) = \boldsymbol{\tau}_m - exp(\hat{\boldsymbol{\phi}})\boldsymbol{\tau}_m{}' \tag{10}$$

The full motion estimation solution is in Formula (11), where ρ is the Cauchy kernel function. The Lie algebra perturbation model is used to solve the Jacobian matrix of two error terms Formulas (6) and (10):

$$\min_{\xi,\phi}\left\{\sum^{N}\|r_1(\mathcal{P}, \boldsymbol{\xi})\|^2 + \sum^{M}\rho\|r_2(\mathcal{T}, \phi)\|^2\right\} \tag{11}$$

$$\boldsymbol{J}_1 = \frac{\partial r_1}{\partial \delta\boldsymbol{\xi}} = -\left(exp(\hat{\boldsymbol{\xi}})\boldsymbol{p}'_n\right)^{\odot} \tag{12}$$

$$\boldsymbol{J}_2 = \frac{\partial r_2}{\partial \boldsymbol{\varphi}} = \left(exp(\hat{\boldsymbol{\phi}})\boldsymbol{\tau}'_m\right)^{\wedge} \tag{13}$$

where $\delta\boldsymbol{\xi}$ and $\boldsymbol{\varphi}$ are small perturbations of $\boldsymbol{\xi}$ and ϕ, respectively. We obtain the transformation between frames in Formula (14). s is the number of edges and corner pairs. ε is a threshold and set to 50:

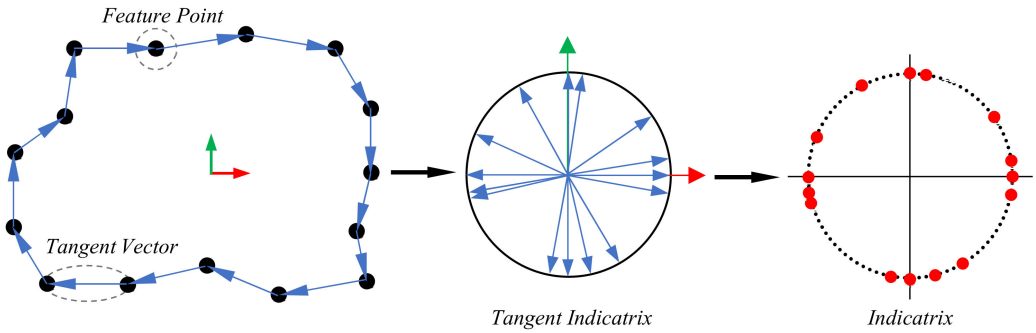
$$\boldsymbol{T}^L = \boldsymbol{M}_1 \otimes \boldsymbol{M}_2 \triangleq \begin{bmatrix} \boldsymbol{R}_1 & \boldsymbol{t}_1 \\ 0^\top & 1 \end{bmatrix} \otimes \begin{bmatrix} \boldsymbol{R}_2 & \boldsymbol{t}_2 \\ 0^\top & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t}_1 \\ 0^\top & 1 \end{bmatrix}, \boldsymbol{R} = \begin{cases} \boldsymbol{R}_1, s < \varepsilon \\ \boldsymbol{R}_2, s \geq \varepsilon \end{cases} \tag{14}$$

Therefore, the problem of motion estimation can be solved by the L-M method in Formula (15):

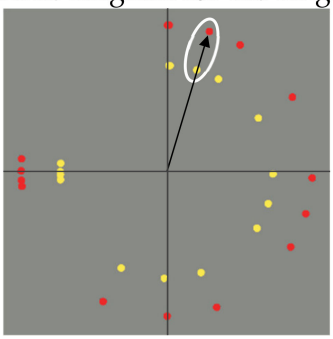
$$\boldsymbol{T}^L \leftarrow \boldsymbol{T}^L - \left(\boldsymbol{J}^\top \boldsymbol{J} + \lambda \boldsymbol{D}^\top \boldsymbol{D}\right)^{-1} \boldsymbol{J}^\top r(\cdot) \tag{15}$$

where $\boldsymbol{D}$ is a non-negative diagonal matrix, which is taken as the square root of the diagonal elements of $\boldsymbol{J}^\top \boldsymbol{J}$. λ is a parameter defined in the L-M method. After evaluating the motion between frames in the lidar coordinate, we obtain the relationship of the sensor's ego-motion in the world coordinate in Formula (16). The motion estimation algorithm is shown in Algorithm 1. Finally, continuous motion estimation constitutes a 2-D lidar odometry.

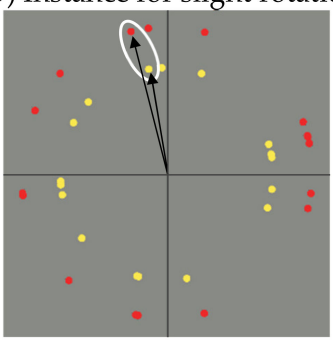
$$\boldsymbol{T}_i = \boldsymbol{T}^L_{i-1,i} \boldsymbol{T}_{i-1} \tag{16}$$



(**a**) The schematic diagram for the tangent indicatrix



(**b**) Instance for slight rotation



(**c**) Instance for large rotation

Figure 5. The illustration of the tangent-vector pair for steadily solving rotation DoF. (**a**) Left shows an example when the tangent vector is computed by two points, so the tangent vector is end-to-end. (**b**) The tangent-vector pair is in the white ellipse, the angular deviation of the vector pair is 0.63 degrees, and the average angle deviation of all the vector pairs is 0.98 degrees. The sensor is almost not rotating. (**c**) In the white ellipse, the angular deviation of the vector pair is 4.52 degrees, and the average angle deviation of all vector pairs is 3.80 degrees. The sensor is rotating.

Algorithm 1: Motion Estimation

Input: smooth collection $\mathcal{P}_s$, rough collection $\mathcal{P}_r$
Output: transformation matrix $\boldsymbol{T}^L$

set $\boldsymbol{T}^L$ to the identity matrix
for $i = 0$ to *max iteration* **do**
  for each smooth point $\boldsymbol{p}_s$ in $\mathcal{P}_s$ or rough point $\boldsymbol{p}_r$ in $\mathcal{P}_r$ **do**
    Find the closest point $\boldsymbol{p}'_s$ and $\boldsymbol{p}'_r$ for $\boldsymbol{p}_s$ and $\boldsymbol{p}_r$ in last scan regarding to smooth collection and rough collection, respectively.
    All the point pairs $< \boldsymbol{p}_s, \boldsymbol{p}'_s >$ and $< \boldsymbol{p}_r, \boldsymbol{p}'_r >$ yield to Formula (5)
  end for
  Compute τ_k, l_k, κ_2 for all points in $< \boldsymbol{p}_s, \boldsymbol{p}'_s >$ and $< \boldsymbol{p}_r, \boldsymbol{p}'_r >$.
  for each point $\boldsymbol{p}_s$ in $< \boldsymbol{p}_s, \boldsymbol{p}'_s >$ and $\boldsymbol{p}_r$ in $< \boldsymbol{p}_r, \boldsymbol{p}'_r >$ **do**
    Classify point as edge $\boldsymbol{p}_e$ when $l_k > 1.5$ and $\kappa_2 < 0.5$,
    Classify point as corner $\boldsymbol{p}_c$ when $l_k < 1$ and $\kappa_2 > 1$.
    All the point pairs $< \boldsymbol{p}_e, \boldsymbol{p}'_e >$ and $< \boldsymbol{p}_c, \boldsymbol{p}'_c >$ yield to Formulas (7) and (8).
  end for
  Compute new τ_k for all the points in $< \boldsymbol{p}_e, \boldsymbol{p}'_e >$ and $< \boldsymbol{p}_c, \boldsymbol{p}'_c >$, then obtain $< \boldsymbol{\tau}, \boldsymbol{\tau}' >$.
  Use $< \boldsymbol{p}_s, \boldsymbol{p}'_s >$ and $< \boldsymbol{p}_r, \boldsymbol{p}'_r >$ as input of Formula (6).
  Use $< \boldsymbol{\tau}, \boldsymbol{\tau}' >$ as input of Formula (10).
  Update $\boldsymbol{T}^L$ for next iteration.
  if the convergence is satisfied **then**
    Return $\boldsymbol{T}^L$
  end if
end for

5. Experiment

We performed algorithm verification in an Ubuntu system (18.04 LTS) running Robot Operation System (ROS in Melodic version) on an Intel 3.7 GHz, 6-core CPU, 16 GiB memory computer. The algorithm uses no more than two cores and no more than 2 GiB memory.

5.1. Our Datasets

We collected three datasets in different areas of the underground garage and one dataset in our lab. We used tape to plan the trajectory points every 2.5 m and then held the lidar to walk clockwise along with the trajectory points in straight lines. In the underground garage, the lidar was above the car's roof all the way to ensure that the collected data were all relative to walls and pillars. In the lab, people were walking around. The starting point and ending point were at the same location. The trajectories of the four groups were 65, 92.5, 9, and 25 m, and the walking speed was about 0.3–0.5 m/s.

We recorded the starting and ending positions and calculated the distance between the two positions, representing the displacement's total drift. Chart 1 shows the four algorithm results, including the ICP, Cartographer, HectorSLAM, and our method. In general, the smaller the loop, the smaller the accumulated error. Our method outperformed the others. We omitted some data with an error of more than 10%. As shown in the trajectory in Figure 6 drawn by EVO [36], ICP and HectorSLAM did not perform very well in some datasets B, C, D and A, B, C, respectively. The HectorSLAM usually failed in rotation while our method performed well. Specifically, in scene D of Figure 6d, we collected the data when people were walking around in the lab and found that in general, the algorithms performed better in the static environment than in the dynamic environment but the error of our method was about 2%, and our method performed better than others.

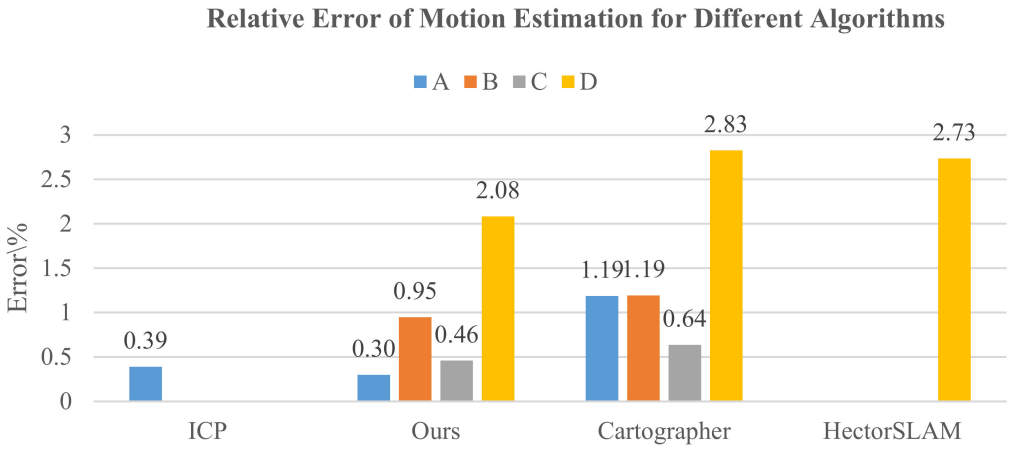
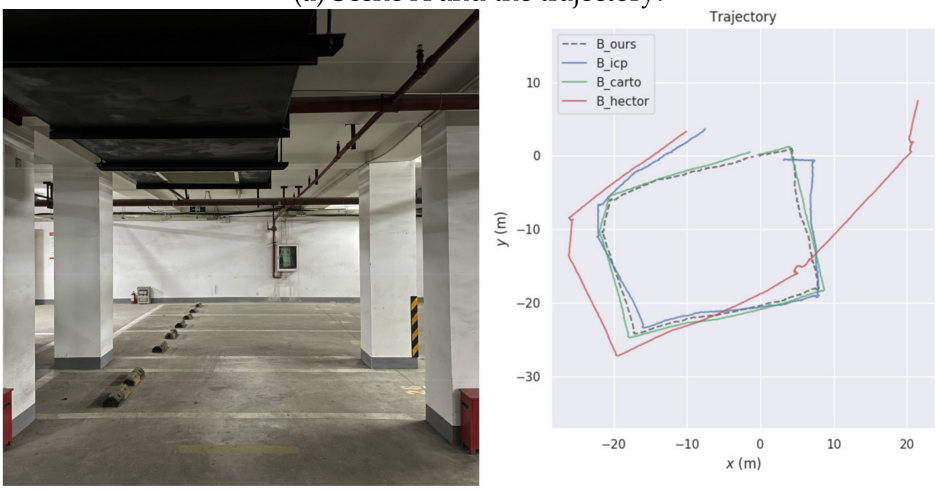


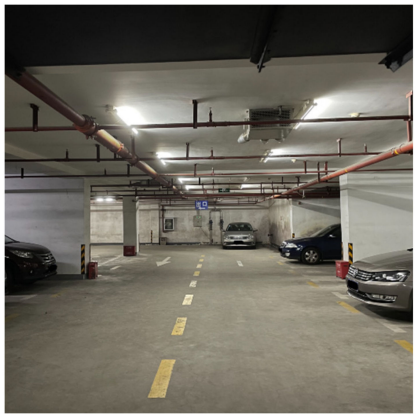
Chart 1. Relative errors of motion estimation drift for different algorithms. A, B, C represent the datasets of different scenes in the garage. D is the dataset of lab.

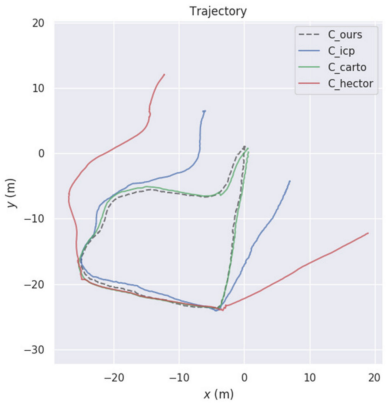
(**a**) Scene A and the trajectory.



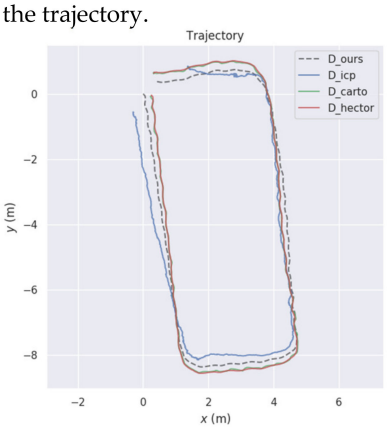
(**b**) Scene B and the trajectory.

Figure 6. *Cont*.





(**c**) Scene C and the trajectory.



(**d**) Scene D and the trajectory.

Figure 6. (**a**–**c**) Static scenes from the underground garage. (**d**) The dynamic scene from the lab.

To verify the real-time performance, the wall clock time of our method is recorded in Table 1, and the algorithm can achieve up to 2.9 times the real-time performance, and the minimum is 1.1 times. It can be noticed that the real-time performance drops in datasets E and F, mainly because the scanning frequency of the lidar used in these two data sets was 40 Hz, which is four times that of our lidar. This means that more data needs to be processed in the same period of time. In Table 2, we compute the runtime of the two main modules of our method in dataset B. For the large-scale feature extraction and data association, the average time per frame is about 56 ms; for the small-scale feature extraction and data association, the average time is about 26 ms. As one scan of RPLIDAR S1 takes 100 ms, our algorithm achieves real-time performance. We can also figure out that a small-scale feature extraction module takes less time than a large-scale feature extraction module due to the multi-scale data structure.

Table 1. Real-time performance of our method.

Dataset	Time of Bag/s	Wall Clock Time/s	Real-Time
A	192.5	67.2	2.9
B	245.3	145.4	1.7
C	134.3	75.1	1.8
D	81.9	47.9	1.7
E	145.4	129.2	1.1
F	105.2	89.6	1.2

Table 2. Runtime of the two main modules.

Module	Max (ms)	Min (ms)	Mean (ms)
Large-scale feature and data association	89.3	10.7	56.1
Small-scale feature and data association	64.9	5.1	25.9

5.2. Open-Access Datasets

We used the open-access datasets in [8]. To verify the low drift of the algorithm in this paper, we selected a large loop dataset located in the building of Schloss Dagstuhl and another with a small loop in a long and narrow corridor. The displacement's total drift is shown in Chart 2, and our method outperformed Cartographer and HectorSLAM in dataset F. We omitted the data with an error of more than 10%.

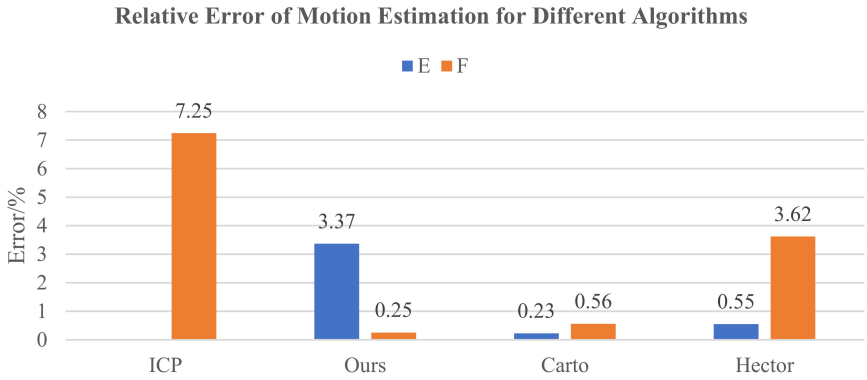
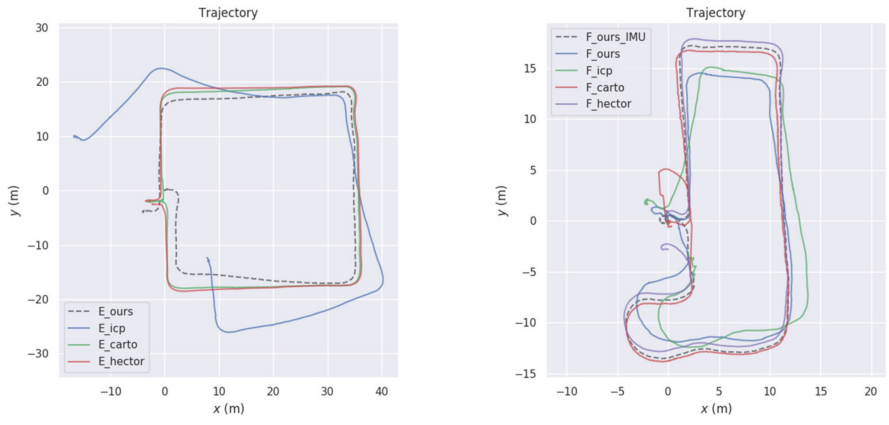


Chart 2. Relative errors of motion estimation drift for different algorithms. E and F are different open-access datasets.

In dataset F, we regarded the wheel odometry data as ground truth and used EVO to calculate the absolute pose error (APE) and relative pose error (RPE). In Table 3, we learn that our method's maximum absolute error of trajectory is 3.27 m, and the root mean square error (RMSE) is 1.71 m, the maximum relative error of trajectory is 0.19 m, and RMSE is 0.03. Few features can be extracted from the lidar data for the long and narrow corridor in dataset F. Therefore, we could find more displacement drift in the y-direction of Figure 7b. Since dataset F has IMU information, we first tried to apply IMU to remove the distortion of the point cloud with the reading of the accelerometer and gyroscope. Then, we used translation provided by IMU to assist our method to overcome the displacement drift in the long and narrow corridor. In Figure 7b, our method with IMU performed better than that without IMU.

Table 3. Absolute pose error and relative pose error.

Dataset	Error	Max/m	Min/m	Mean/m	RMSE/m
F	APE	3.27	0.01	1.42	1.71
	RPE	0.19	<0.01	0.02	0.03



(**a**) (**b**)

Figure 7. Trajectories for datasets E and F. (**a**) The trajectory of scene E, (**b**) The trajectory of scene F.

6. Conclusions

This paper presents a real-time and low-drift 2-D lidar odometry algorithm for the indoor environment. We proposed a multi-scale metric procedure to extract robust features to utilize the 2-D lidar cloud information at multiple levels. Furthermore, the small-scale feature extraction component is less time-consuming due to omitting the secondary searching process. The proposed tangent-vector pair achieves robust performance when evaluating the rotation of the motion. The algorithm was validated on different datasets and compared with outstanding open-source algorithms. The results showed that our method can achieve better accuracy.

In future work, we intend to carry out research on multi-sensor fusion localization and mapping based on IMU and add a loop closure module to further improve the accuracy of the system.

Author Contributions: The work described in this article is the collaborative development of all authors. Data curation, F.L.; Formal analysis, F.L.; Funding acquisition, X.Z.; Methodology, F.L. and S.L.; Project administration, L.Z.; Software, F.L.; Supervision, S.L.; Validation, S.L.; Visualization, F.L.; Writing—original draft, F.L.; Writing—review and editing, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China(NSFC) Project under Grants No. 52075260.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, Q.; Liang, P.; Xia, J.; Wang, T.; Song, M.; Xu, X.; Zhang, J.; Fan, Y.; Liu, L. A Highly Accurate Positioning Solution for C-V2X Systems. *Sensors* **2021**, *21*, 1175. [CrossRef] [PubMed]
2. Ilci, V.; Toth, C. High Definition 3D Map Creation Using GNSS/IMU/LiDAR Sensor Integration to Support Autonomous Vehicle Navigation. *Sensors* **2020**, *20*, 899. [CrossRef] [PubMed]
3. Chiang, K.-W.; Tsai, G.-J.; Li, Y.-H.; Li, Y.; El-Sheimy, N. Navigation Engine Design for Automated Driving Using INS/GNSS/3D LiDAR-SLAM and Integrity Assessment. *Remote Sens.* **2020**, *12*, 1564. [CrossRef]
4. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
5. Su, T.; Zhu, H.; Zhao, P.; Li, Z.; Zhang, S.; Liang, H. A Robust LiDAR-based SLAM for Autonomous Vehicles aided by GPS/INS Integrated Navigation System. In Proceedings of the 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE), Dalian, China, 15–17 July 2021; pp. 351–358. [CrossRef]
6. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; The MIT Press: Cambridge, MA, USA, 2005.
7. Konolige, K.; Grisetti, G.; Kümmerle, R.; Burgard, W.; Limketkai, B.; Vincent, R. Efficient Sparse Pose Adjustment for 2D mapping. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Japan, 1–5 November 2010; pp. 22–29. [CrossRef]
8. Kohlbrecher, S.; Von Stryk, O.; Meyer, J.; Klingauf, U. A flexible and scalable SLAM system with full 3D motion estimation. In Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, Japan, 1–5 November 2011; pp. 155–160. [CrossRef]
9. Durrant-Whyte, H.; Roy, N.; Pieter Abbeel, P. A Linear Approximation for Graph-Based Simultaneous Localization and Mapping. In *Robotics: Science and Systems VII*; MIT Press: Cambridge, MA, USA, 2012; pp. 41–48.
10. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278. [CrossRef]
11. Kümmerle, R.; Grisetti, G.; Strasdat, H.; Konolige, K.; Burgard, W. G^2o: A general framework for graph optimization. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3607–3613. [CrossRef]
12. Besl, P.J.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
13. Censi, A. An ICP variant using a point-to-line metric. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 19–25. [CrossRef]
14. Segal, A.; Haehnel, D.; Thrun, S. Generalized-ICP. *Proc. Robot. Sci. Syst.* **2009**, *2*, 435.

15. Serafin, J.; Grisetti, G. NICP: Dense normal based point cloud registration. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 742–749. [CrossRef]
16. Deschaud, J.-E. IMLS-SLAM: Scan-to-model matching based on 3D data. *Proc. IEEE Int. Conf. Robot. Automat.* **2018**, 2480–2485. [CrossRef]
17. Tian, Y.; Liu, X.; Li, L.; Wang, W. Intensity-Assisted ICP for Fast Registration of 2D-LIDAR. *Sensors* **2019**, *19*, 2124. [CrossRef] [PubMed]
18. Grisetti, G.; Stachniss, C.; Burgard, W. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Trans. Robot.* **2007**, *23*, 34–46. [CrossRef]
19. Steux, B.; El Hamzaoui, O. tinySLAM: A SLAM algorithm in less than 200 lines C-language program. In Proceedings of the 2010 11th International Conference on Control Automation Robotics & Vision, Singapore, 7–12 December 2010; pp. 1975–1979. [CrossRef]
20. Ren, R.; Fu, H.; Wu, M. Large-Scale Outdoor SLAM Based on 2D Lidar. *Electronics* **2019**, *8*, 613. [CrossRef]
21. Wen, J.; Qian, C.; Tang, J.; Liu, H.; Ye, W.; Fan, X. 2D LiDAR SLAM Back-End Optimization with Control Network Constraint for Mobile Mapping. *Sensors* **2018**, *18*, 3668. [CrossRef] [PubMed]
22. Weichen, W.E.I.; Shirinzadeh, B.; Ghafarian, M.; Esakkiappan, S.; Shen, T. Hector SLAM with ICP Trajectory Matching. In Proceedings of the 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Boston, MA, USA, 6–10 July 2020; pp. 1971–1976. [CrossRef]
23. Zhang, J.; Singh, S. Low-drift and real-time lidar odometry and mapping. *Auton. Robot.* **2017**, *41*, 401–416. [CrossRef]
24. Shan, T.; Englot, B. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4758–4765. [CrossRef]
25. Grant, W.S.; Voorhies, R.C.; Itti, L. Finding planes in LiDAR point clouds for real-time registration. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 4347–4354. [CrossRef]
26. Liu, Z.; Zhang, F. BALM: Bundle Adjustment for Lidar Mapping. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3184–3191. [CrossRef]
27. Zhou, L.; Koppel, D.; Kaess, M. LiDAR SLAM with Plane Adjustment for Indoor Environment. *IEEE Robot. Autom. Lett.* **2021**, *6*, 7073–7080. [CrossRef]
28. Opromolla, R.; Fasano, G.; Grassi, M.; Savvaris, A.; Moccia, A. PCA-Based Line Detection from Range Data for Mapping and Localization-Aiding of UAVs. *Int. J. Aerosp. Eng.* **2017**, *14*. [CrossRef]
29. Pfister, S.T.; Burdick, J.W. Multi-scale point and line range data algorithms for mapping and localization. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation. ICRA 2006, Orlando, FL, USA, 15–19 May 2006; pp. 1159–1166. [CrossRef]
30. Wu, D.; Meng, Y.; Zhan, K.; Ma, F. A LIDAR SLAM Based on Point-Line Features for Underground Mining Vehicle. In Proceedings of the 2018 Chinese Automation Congress (CAC), Xi'an, China, 30 November–2 December 2018; pp. 2879–2883. [CrossRef]
31. Huang, R.; Hong, D.; Xu, Y.; Yao, W.; Stilla, U. Multi-Scale Local Context Embedding for LiDAR Point Cloud Classification. *IEEE Geosci. Remote. Sens. Lett.* **2020**, *17*, 721–725. [CrossRef]
32. Bosse, M.; Zlot, R. Keypoint design and evaluation for place recognition in 2d lidar maps. *Robot. Auton. Syst.* **2009**, *57*, 1211–1224. [CrossRef]
33. Lynen, S.; Achtelik, M.W.; Weiss, S.; Chli, M.; Siegwart, R. A robust and modular multi-sensor fusion approach applied to MAV navigation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 3923–3929. [CrossRef]
34. Gao, Y.; Liu, S.; Atia, M.M.; Noureldin, A. INS/GPS/LiDAR Integrated Navigation System for Urban and Indoor Environments using Hybrid Scan Matching Algorithm. *Sensors* **2015**, *15*, 23286–23302. [CrossRef] [PubMed]
35. Liu, S.; Martin, R.R.; Langbein, F.C.; Rosin, P.L. Segmenting Geometric Reliefs from Textured Background Surfaces. *Comput.-Aided Des. Appl.* **2007**, *4*, 565–583. [CrossRef]
36. Grupp, M. Evo: Python Package for the Evaluation of Odometry and SLAM. 13 September 2017. Available online: https://github.com/MichaelGrupp/evo (accessed on 24 October 2021).