Hindawi
Journal of Sensors
Volume 2018, Article ID 2937694, 14 pages
https://doi.org/10.1155/2018/2937694

Research Article

A Real-Time 3D Perception and Reconstruction System Based on a 2D Laser Scanner

Zheng Fang,[1] **Shibo Zhao**,[1] **Shiguang Wen**,[1] **and Yu Zhang**[2]

[1]*Faculty of Robot Science and Engineering, Northeastern University, Shenyang, China*
[2]*State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310058, China*

Correspondence should be addressed to Zheng Fang; fangzheng@mail.neu.edu.cn

Received 5 August 2017; Revised 7 February 2018; Accepted 28 February 2018; Published 16 May 2018

Academic Editor: Paolo Bruschi

This paper presents a real-time and low-cost 3D perception and reconstruction system which is suitable for autonomous navigation and large-scale environment reconstruction. The 3D mapping system is based on a rotating 2D planar laser scanner driven by a step motor, which is suitable for continuous mapping. However, for such a continuous mapping system, the challenge is that the range measurements are received at different times when the 3D LiDAR is moving, which will result in big distortion of the local 3D point cloud. As a result, the errors in motion estimation can cause misregistration of the resulting point cloud. In order to continuously estimate the trajectory of the sensor, we first extract feature points from the local point cloud and then estimate the transformation between current frame to local map to get the LiDAR odometry. After that, we use the estimated motion to remove the distortion of the local point cloud and then register the undistorted local point cloud to the global point cloud to get accurate global map. Finally, we propose a coarse-to-fine graph optimization method to minimize the global drift. The proposed 3D sensor system is advantageous due to its mechanical simplicity, mobility, low weight, low cost, and real-time estimation. To validate the performance of the proposed system, we carried out several experiments to verify its accuracy, robustness, and efficiency. The experimental results show that our system can accurately estimate the trajectory of the sensor and build a quality 3D point cloud map simultaneously.

1. Introduction

Three-dimensional perception is very important to many mobile robotic systems, since the robot is increasingly required to operate in dynamic, unstructured environments and interact safely and effectively with humans or other robots [1–5]. Beside robotics domain, three-dimensional perception and reconstruction are also a key technology for many nonrobotic applications such as surveying, object scanning, 3D manufacturing, and entertainments [6–10].

Nowadays, there are many sensors that can be used to acquire 3D data, for example, scanning LiDAR, stereo vision, and structure light sensors. Among those sensors, scanning LiDAR is mostly used for acquiring 3D data due to its accuracy and long measurement range [11]. In geographic areas, there are several commercial products already available in the market for 3D laser mapping, for example, scanning LiDAR from Faro (http://www.faro.com/) and Leica (https://lasers.leica-geosystems.com/) company. However, those sensors are usually very expensive, and the sizes of those sensors are usually very big. Besides, most of them are only suitable for "stop-and-scan" applications, which means you must put the sensor statically to acquire data and process the data offline to get 3D model of the environments. Therefore, those sensors usually are not suitable for robotics application. In robotics area, the most widely used commercial 3D range finders are provided by Velodyne (http://velodynelidar.com/) with multibeam models HDL-64E, HDL-32E, and VLP-16. These models differ mainly in the number of laser/detectors located in the rotating head. In recent years, there are also some new 3D range finders based on solid-state LiDAR technology, for example, S3 LiDAR sensor from Quanergy (http://quanergy.com/). The main advantage of these sensors is the high acquisition rate

that makes them suitable for vehicles moving at high speeds. However, they have a limited vertical resolution and are more expensive than 2D laser scanners. There are also several hand-held 3D scanners that are commercially available (e.g., Leica's T-scan). Unfortunately, those sensors are primarily intended for object scanning applications. Therefore, they often require modification to the environment and have limited working volume that is not appropriate for large-scale mobile applications. In recent years, RGB-D cameras [9], which are based on structured lighting or TOF (time of flight) technology, become very popular due to its low price and depth measurement ability. However, those sensors are limited in their sensing range, precision, and the lighting conditions in which they can operate. For example, the Asus Xtion has an effective range from 0.5 to 4 meters with precision of around 12 cm at 4 m range and does not work in bright sunlight. Stereo cameras are also widely studied for real-time 3D reconstruction [12]. However, stereo cameras have similar precision characteristics, and their performance is dependent on lighting conditions and textural appearance of the environments.

Besides those commercial 3D scanning sensors, researchers have also developed many customized 3D sensors for their specific applications. Up to now, several customized 3D laser scanners have been developed for autonomous robots by means of a rotating laser scanner with a 360° horizontal FoV, allowing for detection of obstacles in all directions. For example, Morales et al. [13] designed a low-cost 3D laser rangefinder based on pitching a commercial 2D rangefinder around its optical center. They also developed a 3D laser scanner by continuously rotating a 2D laser scanner [14]. However, in their work, they only describe the design and development of the hardware of the sensor and do not propose efficient algorithms for processing the received 3D point cloud. Therefore, those sensors are generally suitable for "stop-and-scan" applications. Zhang and Singh [15] proposed a continuously scanning system LOAM which could be a back-and-forth or continuous rotation configuration. The mapping results of their system are very impressive. However, they do not provide detailed description of their hardware system. Besides, the author only proposed LiDAR odometry and mapping algorithms. Though the local drift is very low, it is not suitable for large-scale mapping since accumulative error is inevitable. Bosse et al. [16] also proposed a sensor called *Zebedee*, which is constructed from a 2D range scanner coupled with a passive linkage mechanism, such as a spring. By shaking the sensor, the device's field of view could be extended outside of its standard scanning plane. However, this system needs to collect the data first and then process the data offline. Therefore, it is not suitable for real-time applications, such as autonomous navigation of mobile robots.

In this paper, we present a low-cost customized 3D laser scanner based on a rotating 2D laser scanner which is suitable for both autonomous navigation and large-scale environment reconstruction. We not only describe the details of the hardware design but also present a real-time motion estimation algorithm that can continuously estimate the trajectory of the sensor and build the 3D point cloud of the environment simultaneously. The proposed 3D sensor system is advantageous due to its mechanical simplicity, mobility, low weight, low cost, and real-time estimation. Therefore, our sensor can not only be used for autonomous navigation but also be suitable for surveying and reconstruction of areas. To validate the performance of the proposed system, we carried out several experiments to verify its accuracy, robustness, and efficiency. The experimental results show that our system can accurately estimate the trajectory of the sensor and build a quality 3D point cloud map simultaneously.

The rest of the paper is organized as follows. In Section 2, a brief discussion of related work is presented. Section 3 describes the hardware and software system of our sensor. Then, our real-time pose estimation, 3D mapping, and global map optimization algorithms are detailed in Section 4. In Section 5, we validate our system in different environments. Section 6 concludes the paper.

2. Related Work

In the recent year, several different kinds of customized 3D laser scanner system built from a 2D laser scanner have been proposed in the robotics community. Those systems mainly differ in the driving mechanism [13, 16, 17] and data processing methods [18–20]. In the following, we will have a brief review of the existing systems and the reconstruction methods they used.

2.1. Driving Mechanism. Currently, there are several ways to make 3D scanners from a 2D laser scanner. According to its driving mechanism, they can be divided into two groups, namely, *passive driving mechanism* and *active driving mechanism*.

The first type is to use passive mechanisms. For example, Bosse et al. [16] proposed a sensor called Zebedee, which is constructed from a 2D range scanner coupled with a passive linkage mechanism, such as a spring. By mounting the other end of the passive linkage mechanism to a moving body, disturbances resulting from accelerations and vibrations of the body propel the 2D scanner in an irregular fashion, thereby extending the device's field of view outside of its standard scanning plane. By combining the information from the 2D laser scanner and a rigid-mounted industrial-grade IMU, this system can accurately recover the trajectory and create the 3D map. However, this sensor is not very good for mobile navigation since if the robot runs on a flat floor, there would be no enough vibrations to actuate the sensor. Kaul et al. [21] also proposed a passively actuated rotating laser scanner for aerial 3D mapping. A key feature of the system is a novel passively driven mechanism to rotate a lightweight 2D laser scanner using the rotor downdraft from a quadcopter. The data generated from the spinning laser is input into a continuous-time simultaneous localization and mapping solution to produce an accurate 6DoF trajectory estimate and a 3D point cloud map.

The second type is to use active mechanisms. For this type, usually there is a motor actively driving the sensor around one axis. When rotating a 2D unit, two basic 3D

scanning configurations are possible: pitching (x-axis rotation) and rolling (y-axis rotation) scans. For example, Morales et al. [13] designed a low-cost 3D laser rangefinder based on pitching a commercial 2D rangefinder around its optical center. The pitching scan is preferable for mobile robotics and area monitoring because it obtains information of the region in front of the 3D rangefinder faster than the rolling scan. This is because the rolling scan always requires a complete 180° rotation to avoid dead zones. However, compared to pitching scanner, the advantage of a rolling scanner is that its field of view can be widened to 360°. Therefore, they also developed a 3D laser scanner by continuously rotating a 2D laser scanner [14].

2.2. Motion Estimation and Registration Methods. For the customized 3D LiDAR sensor which is based on a 2D laser scanner, there are different ways to estimate the motion of sensor and reconstruct the environment using the perceived 3D point cloud. The existing method could be divided into two types: stop-and-scan methods and continuous methods.

Previously, most existing 3D mapping solutions either eliminate sensor motion by taking a stop-and-scan approach or attempt to correct the motion using odometric sensors, such as wheel or visual odometry. Many researchers take the approach of frequently stopping the platform in order to take a stationary scan [19]. However, for mobile robotics system, this type of behavior is undesirable as it limits the efficiency of the task. Since there is no movement when taking a scan, the advantage of this kind of scanning is that there is no distortion in the perceived 3d point cloud. Therefore, it is easier to register the point cloud. As to the registration methods, several approaches have been proposed to estimate the motion by means of 3D scan registration [19, 22, 23]. Most of these approaches are derived from the iterative closest Point (ICP) algorithm [18, 24]. For example, generalized ICP (GICP) [22] unifies the ICP formulation for various error metrics such as point-to-point, point-to-plane, and plane-to-plane. Beside ICP methods, the 3DNDT [20] method discretizes point clouds in 3D grids and aligns Gaussian statistics within grid cells to perform scan registration.

Nowadays, people prefer to develop continuous estimation methods for a rotating 2D laser scanner. However, collecting consistent 3D laser data when the sensor is moving is often difficult. The trajectory of the sensor during the scan must be considered when constructing point clouds, otherwise the structure of the cloud will be severely distorted and potentially unrecognizable. To solve this problem, several methods have been proposed. For example, a two-step method is proposed to remove the distortion [25]: an ICP-based velocity estimation step is followed by a distortion compensation step, using the computed velocity. A similar technique is also used to compensate for the distortion introduced by a single-axis 3D LiDAR [26]. However, if the scanning motion is relatively slow, motion distortion can be severe. This is especially the case when a 2-axis LiDAR is used since one axis is typically much slower than the other. Therefore, many researchers try to use other sensors to provide velocity measurements to remove the distortion. For example, the LiDAR cloud can be registered by state estimation from visual odometry integrated with an IMU [5, 27].

3. System Overview

In this section, we will give a detailed description of the hardware and software system of our sensor. We first describe the hardware system, then present the architecture of the software system.

3.1. Hardware System. The 3D laser scanner is based on spinning a 2D laser scanner. The mechanical system of the 3D reconstruction system is composed of two parts, namely, the rotating head and driving body as shown in Figure 1. The head is mainly composed of a continuously rotating 2D laser scanner. To make the 3D point cloud registration easy, we align the rotating axis y along with the scanning plane x_1y_1. Axis y_1 of the LiDAR sensor is motored by a step motor with a coupling. Since the scanner will rotate continuously, we connect its signal and power lines to a slip ring to solve revolving problem. The step motor is equipped with an encoder to record the rotating angle of the 2D scanner.

The whole 3D sensor system consists of a 2D laser scanner, a step motor driving system, and a PC for real-time registration as shown in Figure 2. The PC is a Surface Pro3 tablet from Microsoft, which has an i7 processor, 256 Gb solid hardware and 12-inch touch screen for displaying. The PC is responsible for receiving scans from the 2D laser scanner and the encoder data from the step motor, then converting the 2D scans into 3D point cloud for LiDAR odometry estimation and mapping, and finally displaying the reconstructed 3D point cloud map in real time. The 2D laser scanner used in our system is a Hokuyo UTM-30LX, which is a 2D time-of-flight laser with a 270 deg field of view, 30 m maximum measurement range, and 40 Hz scanning rate. The dimension of the UTM-30LX is 60×60 mm, and its mass is 210 g, which makes it ideal for low-weight requirements. The motion controller is responsible for controlling the rotating speed of the 2D laser scanner, which is composed of an ARM STM32 board, a step motor, an encoder, and a communication board. The whole system is powered by a 12 V lithium battery.

3.2. Software Systems. The software of the whole 3D reconstruction system is composed of two parts, namely, low-level motion control and high-level reconstruction. The low-level motion control module is responsible for controlling the 2D laser scanner rotating at a constant speed and reading and sending the encoder data to the PC via USB-RS232 data line. The high-level reconstruction module is responsible for receiving the scanning data and encoder data and then estimating the motion of the sensor and registering the received point cloud into a 3D map.

3.2.1. Low-Level Motion Control. The 2D scanner is controlled to rotate at a speed of 180°/s by the STM32 embedded controller. The overall precision of the LiDAR

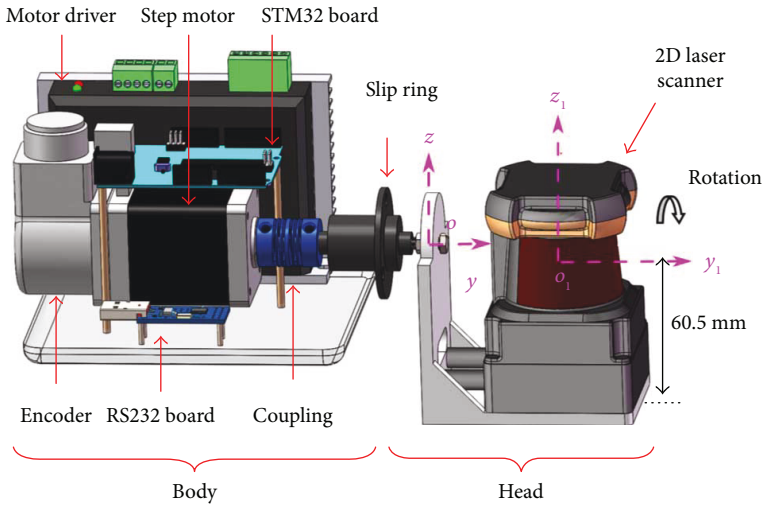


Figure 1: Mechanical structure of the 3D reconstruction system.

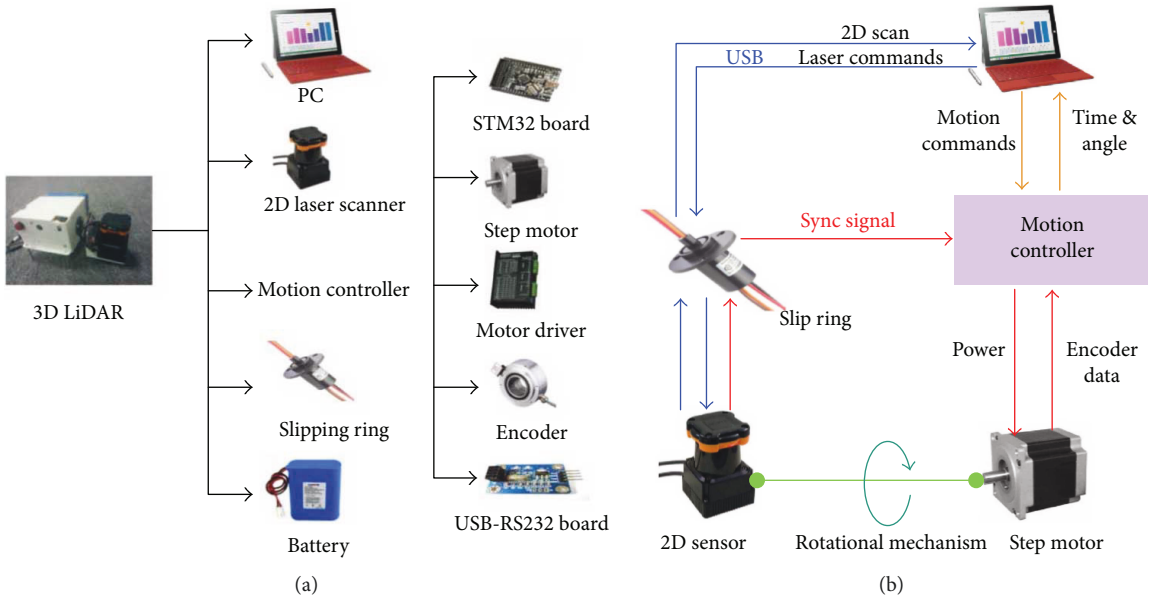


Figure 2: Hardware and functional diagram of the 3D reconstruction system: on the (a) is the hardware system and on the (b) is the functional diagram.

sensor strongly depends on the measurement precision of the instantaneous angle of axis y (as shown in Figure 1), henceforth denoted as β. To determine that angle, a high-precision absolute rotary encoder is used. The encoder's resolution is 10 bits. In general, each scan point must be related to the very value of β (continuously provided by the encoder) at the very time when this point is measured. However, the UTM-30LX does not output point after point in real time. Fortunately, the UTM-30LX triggers a synchronization impulse after it finishes a complete scanning. The embedded controller reads the encoder angle data at the exact time it receives a trigger signal. And finally, the embedded controller sends the raw angle and scans to Surface Pro3 via USB to serial converter. We assume the rotation speed of 2D laser scanner during a scan is constant, then we can calculate the exact angle when each point is measured.

3.2.2. High-Level Reconstruction Software. After receiving the 2D scan and angle data from the 2D laser scanner and the motion controller, the high-level reconstruction algorithm needs to estimate the motion of sensor based on the received data and construct the 3D map in real time. The whole architecture of the motion estimation and reconstruction software is shown in Figure 3. We first need to convert the 2D scans into 3D point cloud by computing the Cartesian coordinates of each point. Then, we need to calibrate the sensor since small errors in the attachment of the 2D device to the rotating mechanism provoke a big distortion in the point cloud. Finally, we need to estimate the motion of the sensor and

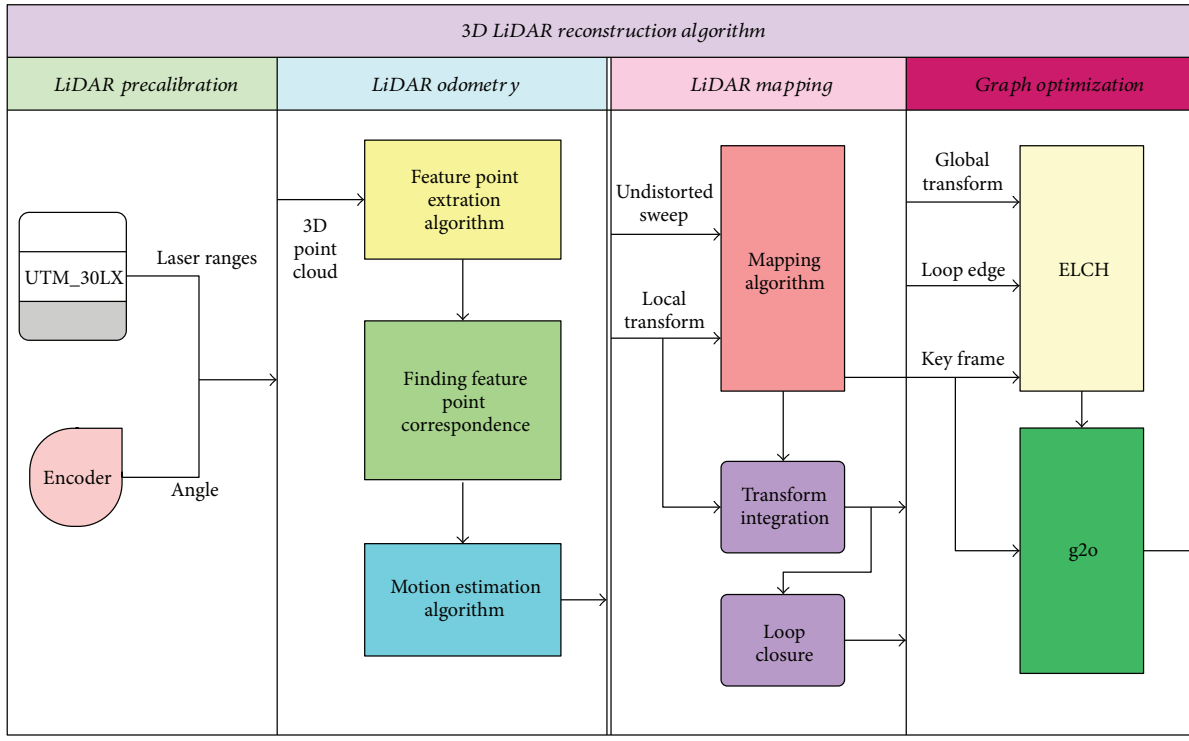


Figure 3: Software architecture of the 3D reconstruction system.

register the increasingly perceived 3D point cloud into a 3D world map.

(1) Generate 3D Point Cloud. The Microsoft Surface PC receives scan data from the 2D laser scanner which is a vector composed of angle θ and range measurement ρ. At the same time, the computer receives the rotating angle φ. The whole coordinate systems are shown in Figure 4. A point of the 2D device is given by its polar coordinates: angle θ and range ρ. If we assume the reference frame $oxyz$ of the system is defined as coincident with that of the 2D device, then, the Cartesian coordinates of the point cloud can be computed from following equation:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 & \cos(\phi) \\ 1 & 0 \\ 0 & \sin(\phi) \end{pmatrix} \begin{pmatrix} \rho \sin(\theta) \\ \rho \cos(\theta) \end{pmatrix}. \quad (1)$$

(2) Geometric Calibration. Practically, due to the installation error, small errors in the attachment of the 2D device to the rotating mechanism results in y_1 not perfectly aligned with y as shown in Figure 5. This error will result in a distortion in the point cloud computed with 1. To remove the distortion, we need to calibrate the sensor. For our customized 3D laser scanner which is a 3D scanner with a 2D laser rangefinder rotating on its optical center, the calibration method proposed in [17] can be used to obtain mechanical misalignments. After calibration, 2 can be employed to obtain 3D Cartesian coordinates of a point in the 3D frame instead of 1.

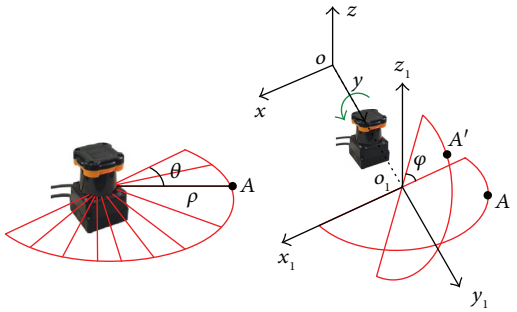


Figure 4: (a) Point A in 2D scan plane; (b) after rotating around y-axis to get A'.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} C(\alpha_0)C(\beta_0) & -C(\alpha_0)S(\beta_0) \\ C(\Theta)S(\beta_0) + C(\beta_0)S(\alpha_0)S(\Theta) & C(\Theta)C(\beta_0) - S(\alpha_0)S(\Theta)S(\beta_0) \\ S(\Theta)S(\beta_0) - C(\Theta)C(\beta_0)S(\alpha_0) & C(\beta_0)S(\Theta) + C(\Theta)S(\alpha_0)S(\beta_0) \end{pmatrix} \begin{pmatrix} \rho C(\theta) \\ \rho S(\theta) \end{pmatrix}, \quad (2)$$

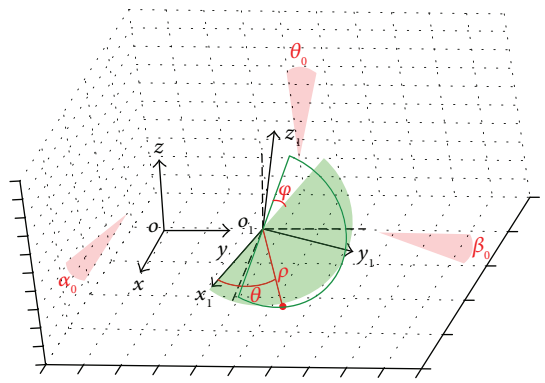


Figure 5: Misalignments between the 2D sensor frame and the 3D sensor frame.

where $\Theta = \theta_0 + \theta$ and θ_0 represent the zero angle of the rotation mechanism. θ_0 is an extrinsic parameter that depends on the installation of the sensor, which will not provoke distortion.

(3) Motion Estimation and Reconstruction. After calibration, if we place the sensor at a place statically, we can get a locally undistorted point cloud. However, in our system, we want to carry the sensor and move around. The movement of the sensor will cause the distortion of point cloud; therefore, we need to continuously estimate the trajectory (laser odometry) of the sensor and remove the distortion of the point cloud. After that, we can register the undistorted point cloud into a global map (mapping). To reduce the local drift of laser odometry estimation, we build a local map and calculate the laser odometry based on local map instead of last sweep. To reduce the global drift for large-scale reconstruction, we first use ELCH [28] algorithm to coarsely distribute the loop-closure error. Then, we use g2o [29] framework to finely optimize the pose graph to get globally consistent map (pose graph optimization). The details of motion estimation and graph optimization algorithms will be described in the following section.

4. Motion Estimation and Reconstruction Algorithm

Large-scale environment reconstruction by using 3D laser scanners has been widely used [26, 30]. The challenge for our system is that our 3D reconstruction system is based on a rotating 2D laser scanner; therefore, there are big distortions as our sensor moves, which pose big challenges for correct and accurate registration. Currently, most 3D reconstruction systems estimate the motion and reconstruct the environment through offline processing. They usually collect the data using the sensor and then process those data on a powerful computer to get accurate model of the environment. In contrast to them, our algorithm runs in real time. We continuously estimate the motion of the sensor, remove the distortion of the point cloud, and register the local point cloud to get the whole map. In the following, we will detail the process of motion estimation, mapping, and global optimization algorithms.

4.1. Odometry Estimation Algorithm. First, we assume that the movement of the sensor is continuous and smooth over time, without abrupt changes. As a convention, we use right uppercase superscription to indicate the coordinate systems. We define a sweep as the sensor complete one-scan coverage. We use right subscription k to indicate the sweeps and P_k to indicate the point cloud perceived during sweep k. The function of odometry estimation is to estimate the relative motion of the sensor during a sweeping time and use the estimated motion to remove the distortion of sweep P_k to get undistorted point cloud. We use similar idea as described in [15] to estimate the odometry. The difference is that they use scan-to-sweep strategy for relative pose estimation. But in our work, in order to improve the robustness, we use scan-to-local map strategy instead of scan-to-sweep strategy. The detail of the odometry estimation is as follows.

4.1.1. Feature Extraction. First, we extract feature points from the LiDAR cloud P_k. For each scan, it returns range values on a scan plane with 0.25° resolution. However, as the laser scanner rotates at an angular velocity of 180°/s and generate scans at 40 Hz, the resolution in the perpendicular direction to the scan planes is 180°/40 = 4.5°. We select two kinds of features which are edge points and planar points. Let $X^L_{(k,i)}$ be a point in P_k (the superscript L means in laser coordinate system), and let S be the set of consecutive points of i returned by the laser scanner in the same scan. We use the following equation to determine whether a point belonged to edge or planar points:

$$c = \frac{1}{|S| \cdot \left\| X^L_{(k,i)} \right\|} \sum_{j \in S, j \neq i} \left\| \left(X^L_{(k,i)} - X^L_{(k,j)} \right) \right\|. \tag{3}$$

According to the c value, the points in a scan can be classified into edge points and planar points. The edge points usually have big c values while the planar points have small c values. We select a feature as planar feature if its c value is smaller than a threshold and a feature as edge feature if its c value is bigger than a threshold. In our implementation, the threshold is determined by experiments. In order to evenly distribute the feature points, we also divided a scan into four identical subregions. In each subregion, there are 2 edge points and 4 planar points in maximum.

4.1.2. Feature Matching

(1) Feature Projecting. Since we assume that the angular and linear velocities of the sensor during a sweep are constant, therefore this allows us to linearly interpolate the pose transform within a sweep for the points that are received at different times. Given a point i, $i \in P_{K+1}$, and its corresponding timestamp t_i and let $T^L_{(k+1,i)}$ be the pose transform between $[t_{k+1}, t_i]$, then $T^L_{(k+1,i)}$ can be computed by linear interpolation of T^L_{k+1}:

$$T^L_{(k+1,i)} = \frac{t_i - t_{k+1}}{t - t_{k+1}} T^L_{k+1}. \tag{4}$$

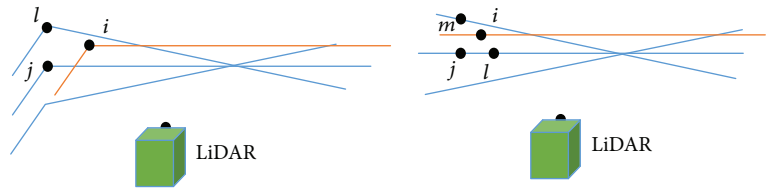


Figure 6: (a) Finding corresponding edge point and edge line; (b) finding corresponding planar point and planar patch.

Assuming that we have point cloud P_{k+1}, then we can extract edge feature points ε_{k+1} and planar feature points μ_{k+1} using 3. We transform features ε_{k+1} and μ_{k+1} to the starting time of sweep $k+1$ and denote them as $\tilde{\varepsilon}_{k+1}$ and $\tilde{\mu}_{k+1}$, then we can get the following equation according to 4:

$$\begin{pmatrix} X^L_{(k+1,i)} \\ 1 \end{pmatrix} = T^L_{(k+1,i)} \begin{pmatrix} \tilde{X}^L_{(k+1,i)} \\ 1 \end{pmatrix}, \tag{5}$$

where $X^L_{(k+1,i)}$ is a point i in ε_{k+1} or μ_{k+1} and $\tilde{X}^L_{(k+1,i)}$ is the corresponding point in $\tilde{\varepsilon}_{k+1,i}$ or $\tilde{\mu}_{k+1}$.

(2) Feature to Local Map Matching. For relative motion estimation, a common strategy is to compute the transform between the current sweep P_{k+1} to the last sweep P_k. This is a kind of strategy similar to frame-to-frame matching strategy which is widely used in visual odometry. However, this strategy is not very robust. This is because the time duration of one sweep is very short; if there is a sharp turn around a corner, the scanning scene will change dramatically. If we just register current scan to the last sweep, the registration may fail due to insufficient overlap. Here, we use a local map strategy. Instead of matching to the last sweep, we create a local map by accumulating the last 3 sweeps and then match current scan to the created local map, which improves the robustness of the whole algorithm.

At the beginning of running our system, we assume the sensor is static; therefore, we concatenate the first 3 full sweeps as a local map M^L_k, $(k \geq 3)$. We transform the local map M^L_k to time stamp t_{k+1} and denote it as $\bar{M}^L_k$ and store it in a 3D kd tree. For ε_{k+1} and μ_{k+1}, we need to find the corresponding edges and planes in $\bar{M}^L_k$. The feature matching process is described in Figure 6. Let i be a point in $\tilde{\varepsilon}_{k+1}$, $i \in \tilde{\varepsilon}_{k+1}$. The edge line is represented by two points. Let j be the closest neighbor of i in M_k, and let l be the closest neighbor of i in the two consecutive scans to the scan of j as shown in Figure 6(a). Therefore, (j, l) forms the correspondence of i. To verify if both j and l are edge points, we check the smoothness of the local surface based on 3. Figure 6(b) shows the procedure of finding a planar patch as the correspondence of a planar point, where (j, l, m) forms the correspondence of point i, $i \in \tilde{\mu}_{k+1}$.

(3) Error Metric Computation. After finding the correspondences, we can calculate the distance from a feature point to its correspondence. We will recover the motion by minimizing the overall distances of the features.

For an edge point $\tilde{X}^L_{(k+1,i)} = (x_0, y_0, z_0)$, $i \in \tilde{\varepsilon}_{k+1}$, if (j, l) is its corresponding edge line $(\bar{X}^L_{(k,j)} = (x_1, y_1, z_1), \bar{X}^L_{(k,l)} = (x_2, y_2, z_2), j, l \in \bar{M}^L_k)$, then the point to line distance can be computed as

$$d_\varepsilon = \frac{\left| \left(\tilde{X}^L_{(k+1,i)} - \bar{X}^L_{(k,j)} \right) \times \left(\tilde{X}^L_{(k+1,i)} - \bar{X}^L_{(k,l)} \right) \right|}{\left| \bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)} \right|}. \tag{6}$$

For a planar point $\tilde{X}^L_{(k+1,i)} = (x_0, y_0, z_0)$, $i \in \tilde{\mu}_{k+1}$, if (j, l, m) is the corresponding planar patch $(X^L_{(k,j)} = (x_1, y_1, z_1), X^L_{(k,l)} = (x_2, y_2, z_2)$, and $X^L_{(k,m)} = (x_3, y_3, z_3)$, $j, l, m \in M^L_k)$, then the point to plane distance is

$$d_\mu = \frac{\left| \left(\tilde{X}^L_{(k+1,i)} - \bar{X}^L_{(k,j)} \right) \left[\left(\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)} \right) \times \left(\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,m)} \right) \right] \right|}{\left| \left(\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)} \right) \times \left(\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,m)} \right) \right|}. \tag{7}$$

(4) Motion Estimation. Recall that 6 and 7 compute the distances between feature points and their correspondences. By combining 4, 5, 6, and 7, we can get the geometric relationship between the feature point and their correspondences:

$$f_\varepsilon \left(X^L_{(k+1,i)}, T^L_{k+1} \right) = d_\varepsilon, \quad i \in \varepsilon_{k+1}, \tag{8}$$

$$f_\mu \left(X^L_{(k+1,i)}, T^L_{k+1} \right) = d_\mu, \quad i \in \mu_{k+1}. \tag{9}$$

For each feature point in ε_{k+1} and μ_{k+1}, we can get one of the above equation. Stack all the equations and we can get

$$f \left(T^L_{(k+1)} \right) = d, \tag{10}$$

where each row corresponds to one feature point and d is its corresponding distance. Our object is to minimize d towards zero through nonlinear iterative optimization to estimate the transform $T^L_{(k+1)}$. Taking the first-order Taylor expansion at $T^L_{(k+1)}$, we get

$$f \approx f \left(T^L_{(k+1)} \right) + \mathbf{A} \left(T^L_{(k+1)} \right) \left(T^L - T^L_{(k+1)} \right), \tag{11}$$

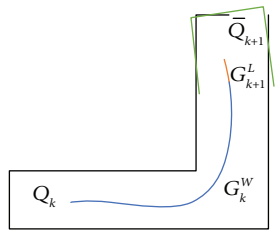


Figure 7: Mapping process.

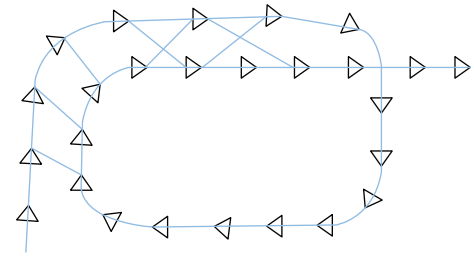


Figure 8: g2o pose optimization: each triangle represents laser pose and the blue edge represents the constraint relationship between the pose obtained by the front-end algorithm. By using these constraints, each pose is optimized so as to satisfy the constraints.

where $\mathbf{A}(T^L_{(k+1)})$ is the Jacobian matrix of f at $T^L_{(k+1)}$. Therefore, we convert the nonlinear optimization problem into a linear least square problem:

$$\min \quad \left\| \mathbf{A}\left(T^L_{(k+1)}\right)\left(T^L - T^L_{(k+1)}\right) + f\left(T^L_{(k+1)}\right) \right\|^2. \tag{12}$$

Finally, we can get

$$T^L_{(k+1)} \leftarrow T^L_{(k+1)} + \Delta T, \tag{13}$$

where $\Delta T = -(A(T^L_{(k+1)})^T A(T^L_{(k+1)}))^{-1} A(T^L_{(k+1)})^T f(T^L_{(k+1)})$.

4.2. Mapping Algorithm. At the end of sweep $k+1$, the odometry estimates an undistorted point cloud $\bar{P}^L_{k+1}$ and simultaneously a pose transform T^L_{k+1} which is the sensor motion during the sweep between time $[t_{k+1}, t_{k+2}]$. The mapping algorithm matches and registers the undistorted point cloud $\bar{P}^L_{k+1}$ to the global map in the world coordinate, as shown in Figure 7. In the figure, the blue curve denotes the global pose G^W_k of the LiDAR sensor at the end time of sweep k, and the orange curve denotes the pose estimation G^L_{k+1} for sweep $k+1$ from the LiDAR odometry estimation algorithm. Combining G^W_k and G^L_{k+1}, we can get an initial guess of transform G^W_{k+1}. Then, we can transform the local undistorted point cloud $\bar{P}^L_{k+1}$ to the global world coordinate, denoted as $\bar{Q}_{k+1}$ and shown as green cloud in Figure 7. Next, the mapping algorithm matches $\bar{Q}_{k+1}$ to Q_k to get a refined pose estimation G^W_{k+1} and register $\bar{Q}_{k+1}$ onto Q_k to get a new map Q_{k+1}.

4.3. Pose Graph Optimization. In the previous section, we obtain the refined pose transformation by the motion estimation in the mapping algorithm. However, since the accumulative error is inevitable in the calculation process, we cannot construct globally consistent map for large-scale environments. Here, we first utilize ELCH [28] as the coarse graph optimization method to solve the global map consistency problem to a certain extent by finding the minimum accumulated error in the map and trajectory estimation. Then, we use g2o [29] for fine graph optimization which considers the relationship among global constraints.

4.4. Coarse Graph Optimization. The coarse graph optimization method is mainly based on geometric relationship of point cloud. First, we save the point cloud data every 3 s as a keyframe, which is stored in a vector containing keyframes and pose information. Then, the Euclidean distance between the current frame and all the historical keyframes is calculated. When the distance is less than the threshold of 1.5 m, it is considered that there potentially exists a closed loop. Since the previous frames adjacent to the current frame are also easily able to meet the above requirement, therefore these data should be excluded. Here, we use ELCH [28], a heuristic algorithm, to do coarse graph optimization. The approach's key idea is that it will spread the final accumulated error to every frame according to the value of uncertainty.

When loop closure is detected, we use ICP algorithm to calculate the accumulated error transformation matrix ΔT between the start keyframe and end keyframe in a closed loop. This ΔT is then assigned to each node in the SLAM graph. Here, the SLAM graph is denoted as $G = (V, E)$, where the vertex V represents the pose of the scanned keyframes and E is the uncertainty between the vertices. Suppose that an undirected graph $G = (V, E)$ has been constructed, which contains two special nodes v_f and v_l, denoting the starting and ending nodes of the closed loop in the graph, with corresponding weights of 0 and 1, respectively.

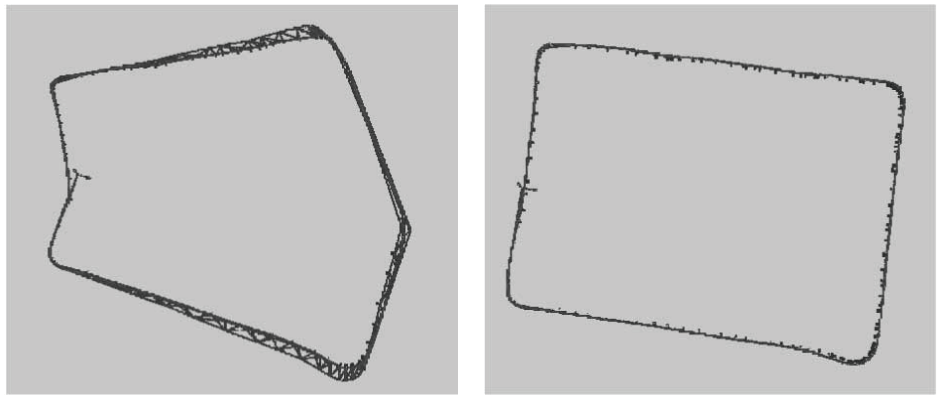
Then, according to the Dijkstra algorithm, we obtain the uncertainty value equivalent to weight w_i. The Dijkstra algorithm is used to calculate the shortest path from each node to other nodes in the set S. Since the initial set contains only closed-loop nodes v_f and v_f, the first iteration only calculates the shortest path from the beginning to the end, and the cost of the path is $d(v_l, v_f)$:

$$d(v_l, v_f) \coloneqq \sum_{\text{edge}<i,j>\in\text{path}} c_{i.j}. \tag{14}$$

$c_{i,j}$ in 14 represents the edge of the connected nodes v_f and v_f, indicating the uncertainty of the edge, so the cost of a path represents the sum of the uncertainties of all the edges in the path. Based on this path cost value, the weight of each node in the path is updated according to following equation:

$$w_i = w_s + \frac{d(v_s, v_i)}{d(v_s, v_e)}(w_e - w_s). \tag{15}$$

w_e and w_s in 15 denote the last and first nodes of the current closed loop.



(a) (b)

Figure 9: g2o result: (a) the hypothetical pose map optimization; (b) the optimized pose diagram.

Figure 10: Reconstructed map: the green trajectory is the ground truth from OptiTrack system, and the red trajectory is the estimated trajectory of our SLAM system. The left figure is top view and the right figure is side view.

While updating the weight of each node in the path, it is also necessary to detect the connectivity between the nodes. For example, when the degree of a node exceeds 2, the node is added to the set S. When a closed-loop path is computed, all edges in the path are removed from the graph G, and then the degree of the nodes v_s and v_e is determined. If the node degree is zero, the node is also removed from the graph. After that, the coarse graph optimization algorithm continues iteratively to calculate the remaining points of the set. The processing flow is the same as the above process until all the closed loops are processed; thus, the weight value of each node in the graph is obtained.

4.5. Fine Graph Optimization. In the previous section, we introduce the coarse graph optimization. However, it only considers the local constraints in the optimization process and does not consider the relationship among the global constraints. In this paper, a general framework for optimization (g2o) [29] is also used for fine graph optimization. g2o is an extensible template library written in C++ that provides an easy-to-use framework for handling SLAM-related issues. Here, for our laser three-dimensional reconstruction system, we only use the pose optimization, as shown in Figure 8.

In our 3D laser SLAM system, we send the edges after the coarse graph optimization to g2o. At the same time, we continue to create the local constraints by connecting each key frame to the first three keyframes through ICP algorithm to establish a constraint relationship. If the matching in the ICP is successful, the matching constraint is taken as a new edge in g2o. We set the number of optimization iterations to 100. We use Levenberg-Marquardt algorithm as the nonlinear optimization algorithm and Huber kernel as the edge kernel function. An illustrative example is shown in Figure 9:

5. Experiments and Analyses

The customized 3D reconstruction sensor has been tested in variant practical indoor environments. Since the used Hokuyo UTM-30LX laser scanner is an indoor laser scanner, we did not carry out experiments in outdoor environments. We carried out 3 different indoor experiments to test the performance of our system. We evaluate the quality of our system by considering both the map and trajectory accuracies. In the first experiment, we quantitatively analyze the performance of our system using a motion capture system. We compared the estimated trajectory of the sensor to the ground-truth data provided by the motion capture system to show the estimation accuracy of our system. In the second experiment, we put our mapping sensor on a ground mobile robot. On the ground mobile robot, we also have a 2D laser scanner, which performs a 2D SLAM algorithm by using wheel odometry and laser data. We compare the trajectory between 2D SLAM and 3D mapping to show the accuracy of the estimation in large indoor environment. We also compared the 2D map projected from 3D global map to the 2D map generated by the 2D SLAM to show the accuracy of the mapping. In the third experiment, we test our sensor in indoor long corridors by manually carrying the sensor. In this experiment, we show the importance of loop-closure detection and global pose optimization for reducing global drift in challenging environments.

During experiments, the system processing the LiDAR data using a Microsoft Surface Pro 3 tablet with Intel i7 4650U CPU and 8 Gb memory running Ubuntu 14.04 and ROS Indigo. We developed our algorithms using C++, PCL

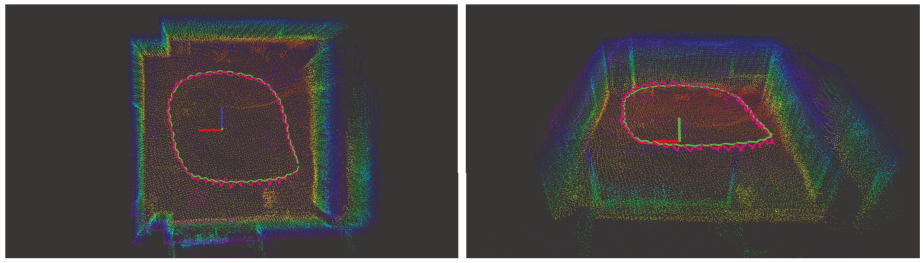


Figure 11: Absolute trajectory error of our system. Poses of the trajectory are projected on the XY-plane.

1.7, Eigen. The experiment video can be found at https://youtu.be/qXESSWWK1rQ or in the Supplementary Material (available here).

5.1. Motion Capture Experiments. Quantitative analysis can be performed on the trajectory provided a ground-truth trajectory estimate is available. We therefore evaluate the accuracy of the trajectory in a motion capture environment in which we can track the position of our sensor with high precision. An OptiTrack motion capture system consisting of 8 Prime 13W cameras is capable of tracking the positions of tags containing multiple reflective targets to millimeter precision at more than 200 Hz, which is good enough as ground truth.

In this experiment, the sensor body follows an irregular path with the sensor held at varying heights. The sensor moves at a speed about 0.5 m/s. The environment containing the OptiTrack system is a 7×6 m room as shown in Figure 10, with few computer workstations and furnitures. We mounted several reflective tags on top of our LiDAR sensor to provide 6DoF tracking as it is moved around in the room. The estimated trajectory and ground-truth trajectory from the OptiTrack system are plotted in Figure 10.

We quantify the accuracy of the estimated trajectories using the measure of the absolute trajectory error (ATE) proposed by [31]. It is based on determining relative and absolute differences between estimated and ground-truth poses. Global consistency is measured by first aligning and then directly comparing absolute pose estimates (and trajectories):

$$\mathrm{ATE}(F_{i:n}) \coloneqq \left(\frac{1}{m}\sum_{i+1}^{m} \left\|\mathrm{trans}(F_i(\Delta))\right\|^2\right)^{1/2} \quad (16)$$

with $F_i(\Delta) \coloneqq Q_i^{-1} S P_i$, where S is the rigid body transformation mapping the estimated trajectory $P_{i:n}$ to the ground-truth trajectory $Q_{i:n}$. In Figure 11, we show the trajectory estimates obtained from our system as well as the deviation from the ground-truth trajectory. The mean translational and rotational errors are 0.049 m and 0.536°, respectively. From Figures 10 and 12, you can see that the experimental environment is very clear. Most times, there are only smooth walls and glass windows. These are very challenging for laser-based reconstruction system, since there are only few geometry features and laser sensor does not work well with glass. In the future, we will incorporate visual information to further improve the robustness and accuracy of the system in geometry feature-less environments.

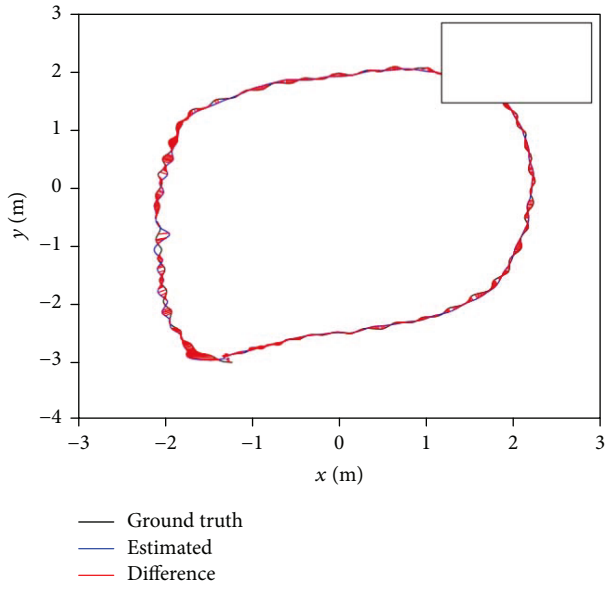


Figure 12: Motion capture room.

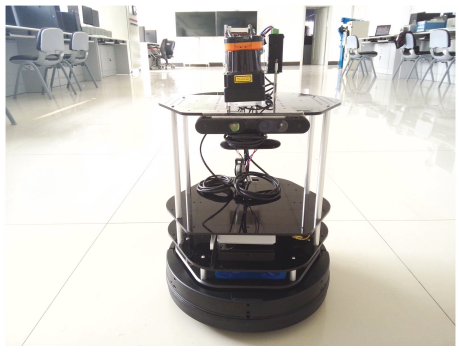


Figure 13: 3D laser scanner on a turtlebot.

5.2. Mobile Mapping Experiments. Though in motion capture room we can get accurate ground truth, the environment is simple and small. In this experiment, we put our sensor on a ground mobile robot as shown in Figure 13. The robot is driven through a big office environment with tables, chairs, and computers. This environment was chosen to test the performance of our system in big and complex indoor environments. Since it is difficult to get ground truth in big

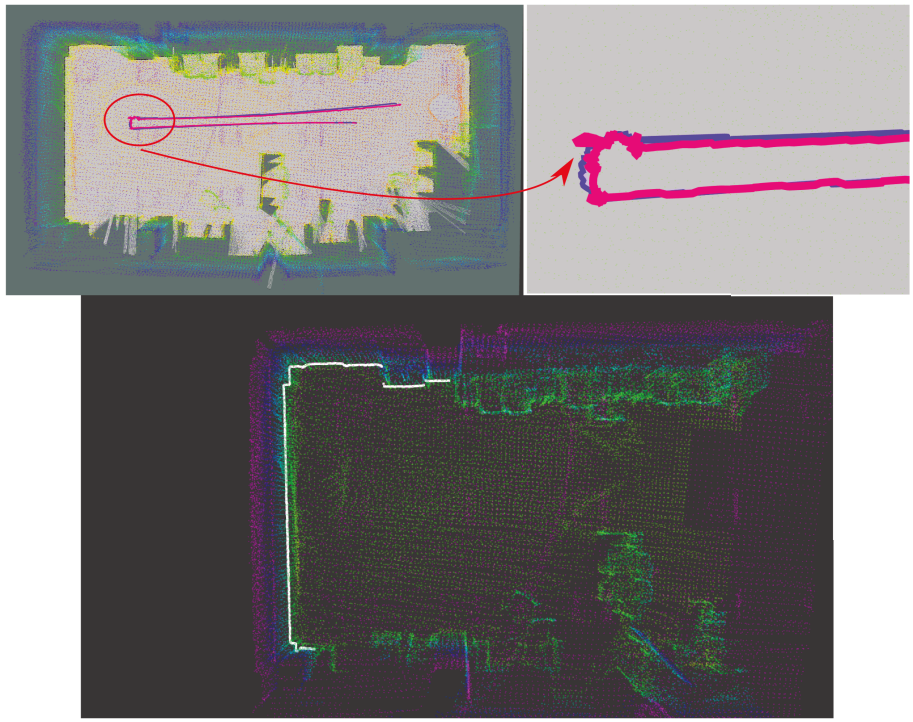


Figure 14: A comparison between the trajectory estimated by the 3D reconstruction system and the trajectory estimate from the 2D laser SLAM system as ground truth in office environment.

indoor environment, we compare the estimated 6DoF trajectory with the ground-truth trajectory computed from the robot's 2D laser SLAM system to show the performance of our system.

The top figures of Figure 14 depict the 2D view of the estimated trajectory of our sensor and the trajectory computed from the 2D laser SLAM. The total length is about 60 m. As you can see, our estimated trajectory can accurately align with the 2D SLAM trajectory in most times. The difficult motion for accurate recovering is fast spot turns, when portions of the sweeps might not be sufficiently constrained. For example, when the sensor is turning around a corner where there is only one smooth wall, it is difficult to recover the translation since the problem becomes a degeneration problem. The bottom figure of Figure 14 shows an oblique view of the 3D point cloud generated from an experiment and overlaid with the points from the horizontal laser. In the zoomed-in region, we can see the undistorted shapes of the environments. Besides, from top left figure of Figure 14, we can see that our 3D point cloud model aligned with the 2D occupancy grid map very accurately; therefore, our system works very well in complex indoor environments. The mean translational and rotational errors in these experiments are 0.023 m and 0.373°, respectively.

5.3. Hand-Held Experiments. We also tested our sensor and algorithms by carrying our system and walking around in indoor environments. We conducted tests to measure accumulated drift of the motion estimate. Since it is difficult to get 6DoF ground-truth data in large indoor environments, we choose a 30 m×20 m rectangular hallway which contains a closed loop. This environment is very challenging for our system since the long corridor contains little geometry features that our algorithms can use. We started our sensor from one place, then walked around, and finally went back to the same place. The motion estimates generate a gap between the starting and finishing positions, which indicates the amount of drift. In practice, there is an accumulative error. The experimental results show that the growth of translational errors is about to be less than 2% of distance traveled and rotational errors less than 0.3° per meter. If we do not have loop-closure detection and global optimization, the map will become inconsistent after a big loop traveling. However, since our system has loop-closure detection and coarse-to-fine graph optimization, the error can be minimized to make the map consistent as shown in Figure 15. The left figure of Figure 15 shows the map before optimization and the right figure shows the map after optimization. From the reconstructed map in the red box, it can be seen that the map shows obvious inconsistency before using the closed-loop correction algorithm. After optimization, the inconsistency is minimized. The green line is the precorrection trajectory and the red line is the closed-loop corrected trajectory. This experiment shows the importance of loop-closure detection and global optimization for long-distance movement.

5.4. Odometry Estimation Comparison. In this part, we did an experiment to show the improvement of our scan-to-local map method compared to original scan-to-sweep method. We carried out the experiment in a long-corridor environment of one building in our university. The long corridor is around 200 meters and it has two closed loops. There

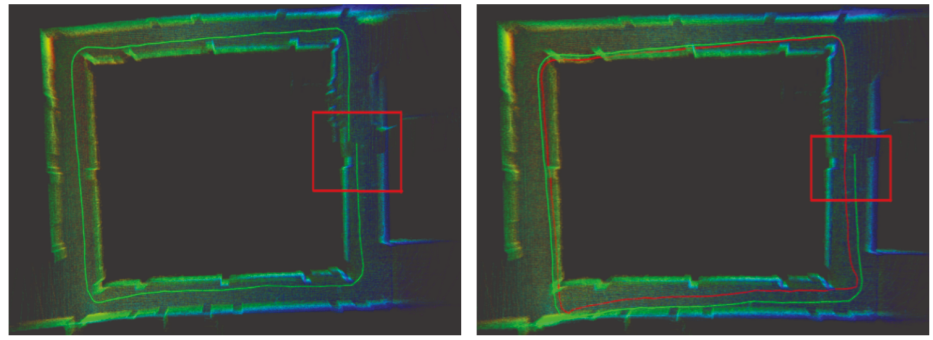


Figure 15: Indoor loop-closure test: left is before optimization and right is after optimization.

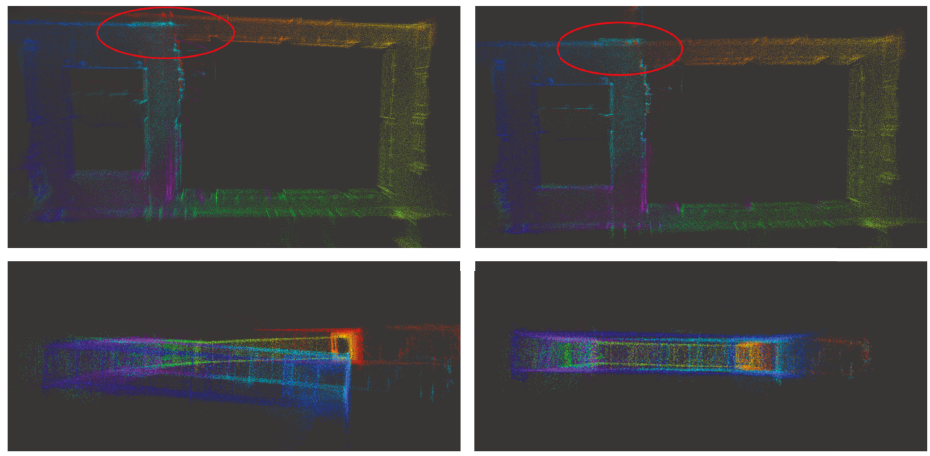


Figure 16: Comparison of odometry estimation by using scan-to-sweep method with scan-to-local map method: left figures are the results of scan-to-sweep method, which have bigger accumulative drift. Right figures are the results of our method, which have much smaller drift.

are some geometry features in the corridor. LOAM uses scan-to-sweep method to match the feature points which however leads to a big accumulative error and makes the map distorted. We use scan-to-local map matching method, which in a way can reduce the local accumulative error and distortion by using more information in matching corresponding points.

The experiment results are shown in Figure 16. The left column figures are the results of LOAM method, while the right figures are ours. The first row shows the map in top view, while the second row shows the map in side view. For fairness, we disabled the loop-closure function in our system and run both algorithms using the same dataset. From the results, you can see that there is a big drift of LOAM algorithm in the map as denoted by the red circle. For the same dataset, however, our method has smaller drift. We can also find that our method has much smaller accumulative error in the z-axis direction from the bottom figures in Figure 16. The experimental results demonstrate that our scan-to-local map method has a lower drift.

6. Conclusions

This paper has described a 3D laser scanner with 360° field of view. Our low-cost 3D laser rangefinder consisted of a 2D LiDAR scanner continuously rotating around its optical center which is suitable for 3D perception in robotics and reconstruction in other applications. We also described a parallel motion estimation, mapping, and global pose optimization algorithm that enable our system to output the 6DoF sensor pose and reconstruct consistent 3D map in real time. In the future, we will continue to improve the performance of our system from hardware and software aspects. For the hardware, we will make the sensor more compact and integrate with other sensors like IMU or camera. For the software, we will consider to use sensor fusion methods to further reduce the drift of the estimation algorithm.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61573091 and no. 61673341), Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (no.ICT170302), Fundamental Research Funds for the Central Universities (N172608005) and Doctoral Scientific Research Foundation of Liaoning Province (no.20170520244).

Supplementary Materials

The supplementary material is a video clip of all the experiments. The experiment video can also be found at https://youtu.be/qXESSWWK1rQ. *(Supplementary Materials)*

References

[1] A. Hornung, M. Phillips, E. Gil Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in three-dimensional cluttered environments for mobile manipulation," in *2012 IEEE International Conference on Robotics and Automation*, pp. 423–429, Saint Paul, MN, USA, May 2012.

[2] M. Nieuwenhuisen and S. Behnke, "Hierarchical planning with 3D local multiresolution obstacle avoidance for micro aerial vehicles," in *ISR/Robotik 2014; 41st International Symposium on Robotics*, pp. 1–7, Munich, Germany, June 2014.

[3] M. Schadler, J. Stückler, and S. Behnke, "Rough terrain 3D mapping and navigation using a continuously rotating 2D laser scanner," *KI - Künstliche Intelligenz*, vol. 28, no. 2, pp. 93–99, 2014.

[4] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 692–697, November-December 2012.

[5] D. Droeschel, Holz, and S. Behnke, "Omnidirectional perception for lightweight MAVs using a continuously rotating 3D laser
Omnidirektionale wahrnehmung für leichte MAVs mittels eines kontinuierlich rotierenden 3D-laserscanners," *Photogrammetrie - Fernerkundung - Geoinformation*, vol. 2014, no. 5, pp. 451–464, 2014.

[6] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: simultaneous localisation and mapping at the level of objects," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1352–1359, Portland, OR, USA, June 2013.

[7] R. Zlot and M. Bosse, "Efficient large-scale three-dimensional mobile mapping for underground mines," *Journal of Field Robotics*, vol. 31, no. 5, pp. 758–779, 2014.

[8] S. Vidas, P. Moghadam, and M. Bosse, "3D thermal mapping of building interiors using an RGB-D and thermal camera," in *2013 IEEE International Conference on Robotics and Automation*, pp. 2311–2318, Karlsruhe, Germany, May 2013.

[9] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: using Kinect-style depth cameras for dense 3D modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[10] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM—3D mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[11] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[12] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: dense 3d reconstruction in real-time," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 963–968, Baden-Baden, Germany, June 2011.

[13] J. Morales, J. L. Martínez, A. Mandow, A. Pequeño-Boter, and A. García-Cerezo, "Design and development of a fast and precise low-cost 3D laser rangefinder," in *2011 IEEE International Conference on Mechatronics*, pp. 621–626, Istanbul, Turkey, April 2011.

[14] J. L. Martínez, J. Morales, A. J. Reina, A. Mandow, A. Pequeño-Boter, and A. García-Cerezo, "Construction and calibration of a low-cost 3D laser scanner with 360° field of view for mobile robots," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, pp. 149–154, Seville, Spain, March 2015.

[15] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, 2017.

[16] M. Bosse, R. Zlot, and P. Flick, "Zebedee: design of a spring-mounted 3-D range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[17] J. Morales, J. Martínez, A. Mandow, A. Reina, A. Pequeño-Boter, and A. García-Cerezo, "Boresight calibration of construction misalignments for 3D scanners built with a 2D laser rangefinder rotating on its optical center," *Sensors*, vol. 14, no. 11, pp. 20025–20040, 2014.

[18] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[19] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM with approximate data association," in *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005*, pp. 242–249, Seattle, WA, USA, July 2005.

[20] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1377–1393, 2012.

[21] L. Kaul, R. Zlot, and M. Bosse, "Continuous-time three-dimensional mapping for micro aerial vehicles with a passively actuated rotating laser scanner," *Journal of Field Robotics*, vol. 33, no. 1, pp. 103–132, 2016.

[22] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems V*, Seattle, WA, USA, June 2009.

[23] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[24] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets: open-source library and experimental protocol," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[25] S. Hong, H. Ko, and J. Kim, "VICP: velocity updating iterative closest point algorithm," in *2010 IEEE International Conference on Robotics and Automation*, pp. 1893–1898, Anchorage, AK, USA, May 2010.

[26] F. Moosmann and C. Stiller, "Velodyne SLAM," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 393–398, Baden-Baden, Germany, June 2011.

[27] S. Scherer, J. Rehder, S. Achar et al., "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, no. 1-2, pp. 189–214, 2012.

[28] J. Sprickerhof, A. Nüchter, K. Lingemann, and J. Hertzberg, "An explicit loop closing technique for 6D SLAM," in *Proceedings of the 4th European Conference on Mobile Robots, ECMR 09*, I. Petrovic and A. J. Lilien-thal, Eds., pp. 229–234, KoREMA, Mlini/Dubrovnik, Croatia, 2009.

[29] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: a general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, Shanghai, China, May 2011.

[30] P. Kr, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3D map maintenance in dynamic environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3712–3719, Hong Kong, China, June 2014.

[31] J. Sturm, W. Burgard, and D. Cremers, "Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark," in *IEEE/RJS International Conference on Intelligent Robot*, Vilamoura, Algarve, Portugal, 2012.