2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

# Low-drift LiDAR-only Odometry and Mapping for UGVs in Environments with Non-level Roads

Xiangyu Chen[1], Yinchuan Wang[1], Chaoqun Wang*[1], Rui Song*[1,2], and Yibin Li[1]

***Abstract*— This study focuses on localization and mapping for UGVs when they are deployed in environments with non-level roads. In these scenarios, the vehicles need to travel through flat but not necessarily level grounds, i.e., ascent or descent, which may cause drifts of the robot pose and distortion of the map. We develop a low-drift LiDAR odometry and mapping approach for the UGV with LiDAR as the only exteroceptive sensor. A factor-graph based pose optimization method is developed with a specifically designed factor named *slope factor*. This factor includes the slope information that is estimated from a real-time LiDAR data stream. The slope information is also used to enhance the loop-closure detection procedure. Moreover, an incremental pitch estimation mechanism is designed to achieve further pose estimation refinement. We demonstrate the effectiveness of the developed framework in real-world environments. The odometry drift is lower and the map is more precise than experiments with the state-of-the-arts. Notably, on the Kitti dataset, our method also exhibits convincing performance, demonstrating its strength in more general application scenarios.**

## I. INTRODUCTION

In recent years, unmanned ground vehicles (UGVs) have received considerable attention and been widely adopted in various fields, such as autonomous driving and logistics. Oftentimes they are deployed in environments with non-level roads, which are common in urban scenes. Typically, there is no previous knowledge for robots when they are first deployed in these applications, thus they need to build a map for localizing themselves with onboard sensors. This is known as simultaneous localization and mapping (SLAM), which is the prerequisite for robots to achieve various tasks.

The research focusing on SLAM is booming and can be classified into several categories according to the employed sensors. One branch drawing much research attention is the vision-based approach [1]. These methods feature richer information in the environment and make rapid progress owing to the advancement of image-processing techniques. However, they are easily affected by the illumination variations, which limits their applications in complex scenarios. Comparatively, the LiDAR is more robust to environmental changes and can provide more accurate distance information, promoting their wide deployment in various applications [2].

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62103237.

[1] School of Control Science and Engineering, Shandong University, Jinan, China. Email: {xychensdu, yinchuanvvang}@163.com, {chaoqunwang, rsong, liyb}@sdu.edu.cn.

[2] Shandong Research Institute of Industrial Technology, Shandong University, Jinan, China. Email: rsong@sdu.edu.cn.

* Corresponding Author.

Fig. 1: LiDAR-only odometry and mapping in environments with non-level roads. The bottom-right figure indicates the mapping building performance.

This study utilizes the LiDAR as the only sensor for UGV's localization and mapping in outdoor environments.

LiDAR SLAM is a rapidly developed research field over the past few years [3]. The pioneer work uses the Particle Filter [4] or Extended Kalman Filter [5] to estimate the pose transformation. As the scene grows in scale and complexity, traditional methods are unable to meet the requirements of real-time operation. Thus, the feature-based methods [6] are soaring in popularity. Moreover, in recent years, graph theory is specifically introduced to establish the relationship between robot pose and sensor data to further save computational resources [7]–[9] in complex environments. However, few researchers focus on localization and mapping with UGV in the environments with slope way. Current LiDAR-based SLAM approaches are of unsatisfactory performance when deployed in these scenarios. They often have unacceptable elevation error even without slope way. Localization and mapping in these environments still remain a challenging problem.

In this study, we present a LiDAR-based SLAM framework towards low-drift localization and mapping in the environment with the non-level roads. The pose feature of the UGV when it encounters slope in this environment is analyzed and extracted. By employing the factor-graph based method, we further develop a slope factor to integrate the slope feature into the SLAM framework. Additionally, we integrate the slope information into loop closure detection module. Of note is a special vehicle pitch estimation mechanism utilized to further improve the SLAM performance. Overall, the proposed framework can handle the elevation drift and achieve better localization performance and less map distortion without additional sensors, as shown in Fig. 1. The main contributions lie in the following aspects:

- We propose a slope factor that provides a constraint for robot rotation by estimating the slope from LiDAR data.

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9982264

It is merged into the factor graph to reduce the drift in localization and mapping.
- We present a loop closure detection approach that considers both the position and orientation error, which decreases the drift when returning to the ever place.
- An incremental angle approximation mechanism is designed to calculate the pitch variations when the vehicle climbs the slope, which further improves the pose estimation accuracy.

The designed framework is evaluated in dataset and real-world outdoor environments, which demonstrates its effectiveness and accuracy[1].

The remainder of this paper is organized as follows: Sec. II reviews related LiDAR methods. Then, Sec. III presents the proposed LiDAR SLAM framework in detail. Sec. IV showcases experiments and our performance in outdoor environments. We conclude this study and introduce future directions in Sec. V.

## II. RELATED WORK

LiDAR SLAM has been a hot research topic over the past few decades. In principle, the transformation between consecutive frames of LiDAR is obtained by the point cloud registration methods, such as ICP [10], NDT [11], and GICP [12]. To achieve real-time performance, it is preferred to use feature points rather than the whole points to perform registration. LOAM [6] extracts the Edge and Surf points as the feature points to get the optimal transformation matrix with the Euclidean distance cost. F-LOAM [13] uses the same feature extraction method and replaces the iteration calculation with a nun-iterative two-stage distortion compensation method to achieve better performance. However, simply relying on feature matching may lead to drift of LiDAR odometry and map distortion due to the continuously accumulated error.

To cope with the odometry drift and map distortion, the loop closure detection has been integrated into the LiDAR SLAM framework, which has already been widely used in visual SLAM [1] [14]. In LeGO-LOAM [15], the loop closure is detected by the difference between the current robot pose and the poses in the pose graph. Then, the pose graph is optimized to refine the estimated poses. Lin *et al.* [16] propose a 2D histogram based loop closure detection method. Ramezani *et al.* [17] utilize the deep-learning based loop closure detection method to solve the odometry drift for the floating robot platform.

Besides, to improve the SLAM performance, another important branch of research introduces additional sensors. Shan *et al.* present the LIO-SAM [18] which is a sensor-fusion based SLAM method with Inertial Measurement Unit (IMU) and GPS. This method showcases more accurate localization performance in outdoor environments. The LVI-SAM [19] combines the visual sensor, IMU, and LiDAR. This fusion-based method improves the localization performance in both texture-less and feature-less environments. Nevertheless, introducing additional sensors may cause extra effort such as calibration.

There are also attempts that develop new frameworks to cope with the odometry and map drift. Behley *et al.* [20] propose a novel surfel-based map structure to store data in the mapping procedure. This method shows better performance when utilizing dense LiDAR data. LOL [21] uses an offline map to correct the robot pose for estimating the drift error. Additionally, some deep-learning based semantic methods [22]–[24] has been integrated into navigation tasks. Semantic LiDAR SLAM is also a popular research topic in them. Chen *et al.* [25] extract semantic labels from LiDAR data by a segmentation network. Then, they integrate the network output with LiDAR SLAM to label the map with semantic information.

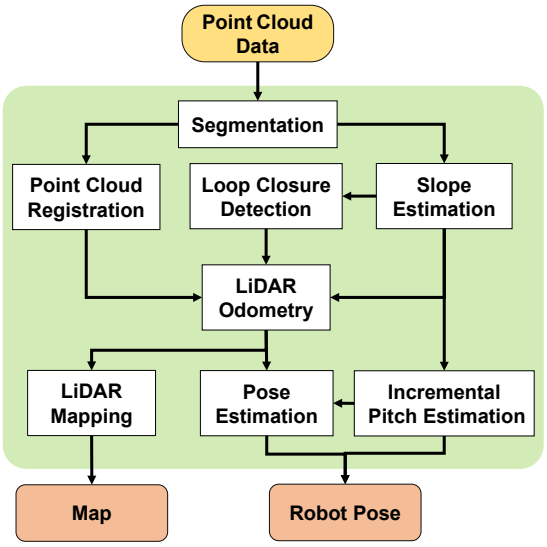



Fig. 2: The systematic diagram of the developed framework.

## III. METHODOLOGY

### A. System Overview

The overview of the our LiDAR SLAM pipeline is shown in Fig. 2. The system takes the point cloud data from LiDAR as input and outputs the map and the robot pose. Firstly, the input point cloud is segmented into the ground points $P_g$ and other points $P_c$ using the algorithm presented in [15]. The points in $P_c$ are used for point cloud registration in the developed framework. In particular, we use the points in $P_g$ to detect the slope in real-time. Then, a factor graph is established and updated by using $Gtsam$ [26] to create a *LiDAR Odometry*. The factor graph is specially designed to incorporate a slope factor taking into account the slope around the robot. The *Loop Closure Detection* monitors whether the robot revisits the passed area to enrich the factor graph. The *LiDAR Mapping* and *Pose Estimation* are used for generating the point cloud map and the pose of the robot, respectively. Additionally, we develop an *Incremental Pitch Estimation* module, which serves as a reference to further refine the estimated robot pose.

[1]Video: https://www.youtube.com/watch?v=VWPcX7ydmU4

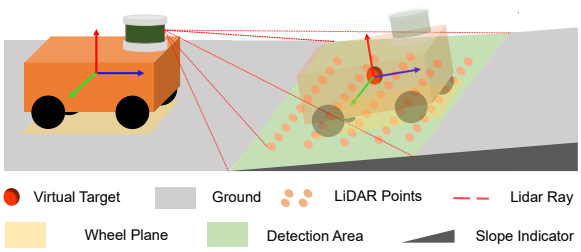



Fig. 3: The formation of a slope factor as a reference in pose estimation.

### B. Slope Factor

In view of the 2.5D environment with slopes that the robot operates in, we consider six-degree robot pose $\mathbf{x} = [x, y, z, roll, pitch, yaw]$. To cope with this non-level terrain in robot localization and mapping, we propose to firstly detect the slope during the robot operation. To that end, a slope estimation method based on sparse point cloud is developed. Define the wheel plane as the one spanned by the polygon connecting the supporting wheels of the robot, as shown in Fig. 3. In the robot frame, a polygon area $\Xi$ that is the same as the robot footprint is set ahead of the robot on the wheel plane. Without loss of generality, assume the robot footprint is a rectangle and $\Xi = [x_{min}, x_{max}, y_{min}, y_{max}]$. At a navigation moment $t$, given the point cloud set $P_c$, we obtain a point set $\tilde{P}$ that satisfies $\tilde{P} = \{p^i | p^i \in P_c, p_x^i \in [x_{min}, x_{max}], p_y^i \in [y_{min}, y_{max}]\}$. This set $\tilde{P}$ is utilized to estimate whether there exists a slope way ahead of the robot. We use the Principal Component Analysis (PCA) to get the normal vector of the estimated slope. Firstly, denote a matrix with the point set:

$$\hat{P} = \begin{bmatrix} p_x^1 - \bar{p}_x & p_y^1 - \bar{p}_y & p_z^1 - \bar{p}_z \\ p_x^2 - \bar{p}_x & p_y^2 - \bar{p}_y & p_z^2 - \bar{p}_z \\ \vdots & \vdots & \vdots \\ p_x^n - \bar{p}_x & p_y^n - \bar{p}_y & p_z^n - \bar{p}_z \end{bmatrix}, p_i^j \in \tilde{P}, \quad (1)$$

where $\bar{p}_x, \bar{p}_y, \bar{p}_z$ are the mean coordinates in three directions,

$$\bar{p}_x = \frac{\Sigma p_x^j}{n}, \bar{p}_y = \frac{\Sigma p_y^j}{n}, \bar{p}_z = \frac{\Sigma p_z^j}{n}, p_i \in \tilde{P}. \quad (2)$$

Then, we make the covariance matrix $\mathcal{P} = \hat{P}\hat{P}^T$ and use Singular Value Decomposition (SVD) to convert it to $\mathcal{P} = U\Sigma V^T$. The normal vector $\vec{n}_s$ of $\hat{P}$ is obtained as the first eigen vector in $U$ corresponding to the largest eigen value in $\mathcal{V}$.

In the navigation process, suppose the robot pose is $\mathbf{x}_t$ at time $t$ and the $\mathbf{x}_{t+\Delta}$ is the robot pose at $t + \Delta$. As shown in Fig. 3, assume at time $t + \Delta$, the robot is on a slope that the robot detects at time $t$, then the normal vector of robot wheel plane at time $t + \Delta$ is $\vec{n}_s$. Denote the normal vector of the robot wheel plane at time $t$ as $\vec{n}_t$, then the angle $\theta$ between these two normal vectors can be calculated by

$$\theta = \arccos \frac{\vec{n}_s \vec{n}_t}{|\vec{n}_s||\vec{n}_t|}. \quad (3)$$

This angle also indicates the robot orientation disparity between time $t$ and $t + \Delta$. Therefore, it can be used as a reference in localization if the robot passes through these two-wheel planes in the navigation.

However, the robot may violate the assumption and does not necessarily travel through the wheel plane at time $t + \Delta$. In this case, the angle $\theta$ should not be used as a reference in the SLAM system. That is to say, the robot should firstly determine whether the robot will pass the slope at a time $t + \Delta$. In the world frame, the position of the virtual target simulating the robot at time $t + \Delta$ is determined by

$$\mathbf{x}_t^v = T_R^W T_s^R \bar{P} = \left[\bar{p_x}, \bar{p_y}, \bar{p_z}\right]^T. \quad (4)$$

Here, the $T_S^R$ is the translation matrix between the sensor and the robot frame. The $T_R^W$ is the translation matrix between the robot pose $\mathbf{x}_t$ and the world frame, provided by the LiDAR odometry at time $t$ (Sec. III-C). Then, one can evaluate whether the robot will locate at the sloped area at time $t + \Delta$. A slope factor is formed by ($\mathbf{x}_t$, $\mathbf{x}_t^v$, $\theta$). Once the robot reaches the virtual target region, the slope factor will be added to the factor graph to improve the accuracy of *LiDAR odometry*.

### C. LiDAR Odometry

The *LiDAR Odometry* module maintains a factor graph to estimate the robot pose. The keyframes are extracted in the developed system to release the computational burden and memory space, which is a typical routine in the computer vision community for generating the nodes of a factor graph. Denote $F_i$, $\mathbf{x}_i^k$, and $\mathbf{x}_t$ as a keyframe, the estimated pose of the keyframe, and the robot pose at time $t$. The selection of the keyframes is based on the distance changes of robot pose to achieve the even distribution of keyframes on robot trajectory. At a moment $t$ when the robot travels through the environment, we monitor the distance between the robot pose $\mathbf{x}_t$ and the pose of the last keyframe $\mathbf{x}_i^k$:

$$d_k = \|(\mathbf{x}_t - \mathbf{x}_i^k)\|. \quad (5)$$

A distance threshold $\mathcal{T}_k$ is predefined. Once $d_k > \mathcal{T}_k$, we add a new keyframe $F$ and update the factor graph. The factor graph employs the $\mathbf{x}_i^k$ as nodes and includes three factors, the `Successive Transformation Factor`, the `Slope Factor`, and the `Loop Closure Factor`, as indicated in Fig. 4.

The `Successive Transformation Factor` refers to the transformations among adjacent frames in the interval between two keyframes. The *Point Cloud Registration* module is utilized to find the transformation matrix from the adjacent LiDAR data. Firstly, the surf points and edge points are extracted from the $P_c$ according to their *smoothness* at their located area. The *smoothness* $\zeta$ of point $p^i$ is calculated by

$$\zeta = \frac{\| \sum_{j \in S, j \neq i} (p^j - p^i) \|}{|S| \cdot \|p^i\|}, \quad (6)$$

where $S$ includes the continuous points in the same scan of $p^i$ in LiDAR. A threshold $\mathcal{T}_c$ is set to distinguish the surf

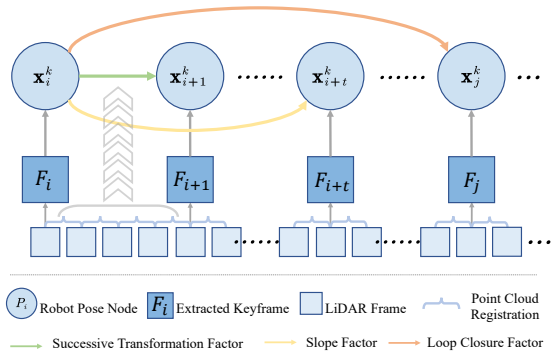



Fig. 4: The structure of the factor graph in the developed localization and mapping framework.

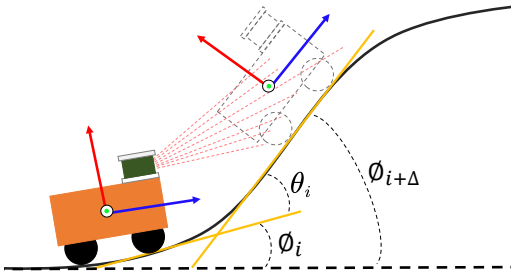



Fig. 5: Incremental pitch angle estimation when the robot climbing the slope.

points ($\zeta > \mathcal{T}_c$) and edge points ($\zeta < \mathcal{T}_c$). Then, an ICP-based scan-matching method uses the two groups of points to get the transformation matrix $T_r$ of the robot between two frames [6]. The transformation provided by the *Point Cloud Registration* module acts as a factor in the factor graph to improve the accuracy of pose estimation.

Meanwhile, we detect the distance $d_v$ between the keyframe pose $\mathbf{x}_{i+t}^k$ and the virtual target determined at a previous moment $\mathbf{x}_{t-\Delta}^v$

$$d_v = \|(\mathbf{x}_{i+t}^k - \mathbf{x}_{t-\Delta}^v)\|. \tag{7}$$

We also set a threshold $\mathcal{T}_v$ to detect whether the robot arrives at the area where one virtual target locates. Once the $d_v$ is smaller than $\mathcal{T}_v$, the robot will be deemed to be on a slope detected previously. The pose $\mathbf{x}_{t-\Delta}^v$ is associate with the node $\mathbf{x}_{i+t}^k$. Then, the triple ($\mathbf{x}_{i-\Delta}$, $\mathbf{x}_{i-\Delta}^v$, $\theta$) is added to the factor graph, where the pose $x_{i-\Delta}$ will find the nearest node on the factor graph to form the constraint.

In terms of the `Loop Closure Factor`, we especially consider the robot pose information when it sees the same scene. When a new node $\mathbf{x}_i^k$ is added, we search the previous nodes on the factor graph to find the close node $\mathbf{x}_j^k$ in Euclidean space. Once the distance between $\mathbf{x}_i^k$ and $\mathbf{x}_j^k$ is smaller than a threshold $\mathcal{T}_c^p$, we will calculate the orientation difference between $\mathbf{x}_i^k$ and $\mathbf{x}_j^k$. Let $R_i$ and $R_j$ be rotation matrices of $\mathbf{x}_i^k$ and $\mathbf{x}_j^k$ with respect to the world frame, we calculate $\mathcal{R} = R_i * R_j$. Then, the angle of the difference rotations is obtained

$$\sigma = arccos(\frac{trace(\mathcal{R}) - 1}{2}). \tag{8}$$

Consider the moving direction of the vehicle, we further refine the angle difference as

$$\hat{\sigma} = \begin{cases} \sigma & if \ \ |\sigma| \leq \pi \\ |\sigma| - \pi & if \ \ |\sigma| > \pi \end{cases}, \tag{9}$$

In this regard, only if the angle $\hat{\sigma}$ is smaller than a threshold $\mathcal{T}_c^o$ will this factor be taken into consideration.

After adding the factors into the factor graph, we use the iSAM2 [9] to optimize the factor graph. Then the optimized graph is able to output the robot pose. Besides, we design an incremental pitch estimation method to correct the robot pitch angle by the extracted slope information.

### D. Incremental Pitch Correction

In typically structured environments with slopes, the vehicle is prone to climb the slope up and down following the straight slope ascending or descending direction, without considering the aggressive vehicle maneuver in the SLAM. Besides, the operator may drive the vehicle to climb the slope as desired in order to build a precise map. Based on these observations, we design a special mechanism that utilizes the slope information to partially refine the robot pose. When the robot climbs the slope as expected, the pitch angle $\phi$ of the robot frame can reflect the difference between two successive robot positions, as indicated in Fig. 5. In this case, the pitch angle between two successive vehicle poses on the slope satisfies $\phi_{i+1} = \phi_i + \theta_i$. We normally set the initial pitch $\phi_0$ as zero before climbing. Thus, the pitch angle of the robot pose at the time $i$ can be regarded as the accumulation of slope angle $\theta$ as

$$\begin{aligned} \phi_i &= \phi_{i-1} + \theta_{i-1} \\ &= \sum_{j=0}^{i-1} \theta_j. \end{aligned} \tag{10}$$

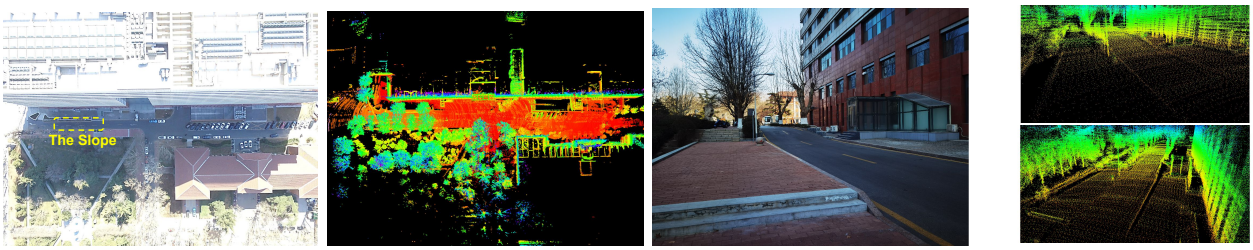
Here, we have two estimations of pitch angle by the factor graph and the incremental pitch estimation using Eq. 10. To get the precise pitch angle of robot $\hat{\phi}$, we combine the two estimations as

$$\hat{\phi}_i = \begin{cases} (1-\eta)\tilde{\phi}_i + \eta\phi_i & if \ \ |\tilde{\phi}_i - \phi| > \mathcal{T}_\phi \\ \phi_i & if \ \ |\tilde{\phi}_i - \phi| \leq \mathcal{T}_\phi \end{cases}, \tag{11}$$

where $\tilde{\phi}_i$ is the pitch angle estimation by LiDAR Odometry and $\hat{\phi}_i$ is the final output of the pitch. $\eta$ is a tunable parameter to achieve the trade-off between these two results.

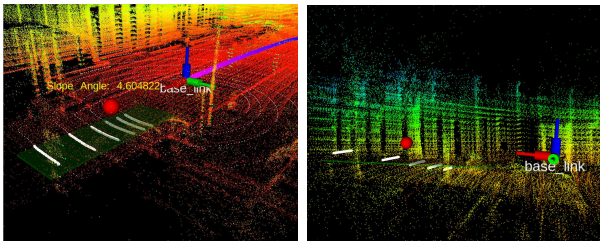
## IV. Experiments and Results

The developed framework is evaluated in various environments. Currently, there are few off-the-shelf datasets that meet the evaluation requirements. We hence conduct experiments in the real-world environment including a parking lot, the alike boulevard road, and the slope way, as shown in Fig. 6(a). The slope way inside the environment connects two flat roads with 1.12 m elevation difference. In the experiments,



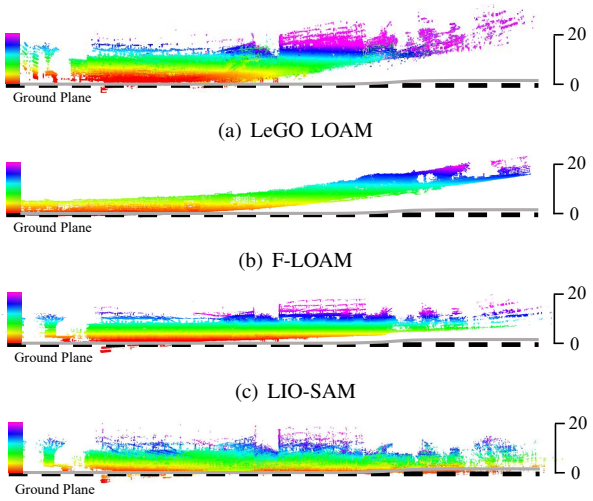

(a) (b) (c) (d)

Fig. 6: The experimental environment settings and mapping results, including (a)-(b) the the testing environment and its corresponding point cloud map in bird-view, and (b)-(c) the slope and its detailed point cloud map.




(a) (b)

Fig. 7: Slope estimation and factor generation in localization and mapping. The green region depicts the area $\Xi$ ahead of the robot.




(a) LeGO LOAM

(b) F-LOAM

(c) LIO-SAM

(d) Ours

Fig. 8: The point cloud map generated by different SLAM methods in the side view. The gray line indicates the ground plane with a slope.

the data is collected using a All-terrain Vehicle (ATV)(see Fig. 1). A Velodyne Puck LiDAR is mounted on top of the UGV. We thoroughly evaluate the developed framework with various metrics. The evaluation tool *EVO* [27] is utilized in the experiments. The code of the framework and the tested data is organizing and accessible[2].

### A. Overall Performance

The snapshots in the vehicle map building are shown in Fig. 7. The slope detecting mechanism is working effectively, as indicated by the rectangle region in this figure. In these two figures, the slopes can be successfully detected and their angles can also be obtained, which verifies the effectiveness of the proposed method. The snapshot of the built point cloud map in the real-world environment is shown in Fig. 6(b). The robot is able to build the map of the environment and the map shows a good match with the real-world environment. Particularly, we show the details of the slope map, as shown in Fig. 6(d), which indicates a convincing map building performance in coping with the slope way.

### B. Map Distortion

To further evaluate the map building performance, we compare the developed method with the state-of-the-art LiDAR SLAM methods, including the LeGO-LOAM [15], the F-LOAM [13], and the LIO-SAM [18]. The experiments are conducted in the aforementioned real-world environment. The robot firstly moves along the flat level ground and then climbs a slope to reach the higher level ground. In environmental mapping with slopes, we focus more on the elevation error that is significant with the state-of-the-art methods. Therefore, the performance of the map building using different methods is shown in the side view along the road, as shown in Fig. 8. As indicated by the ground truth, one can observe that the developed framework showcases better performance than the other methods that exhibit significant altitude drift. The developed framework provides various constraints in terms of the orientation, thus being more accurate in pose estimation and providing more precise environmental map.

TABLE I: The elevation error of different methods in different scenarios.

| | LeGO | F-LOAM | LIO-SAM | Ours |
|---|---|---|---|---|
| Elevation Err(G)(m) | 12.59 | 9.16 | 4.45 | **0.18** |
| Elevation Err(L)(m) | 3.41 | 1.87 | 2.93 | **0.31** |

In the above experiments, we specifically record the altitude differences between the two level grounds connected by the slope, i.e., the slope elevation, in the building map. The results are shown in Table I. The ground truth slope elevation is 1.12 m. We specify the slope elevation error (Elevation Error (G)) in the global map built using different methods. As Table I indicates, the developed method shows

[2]https://github.com/SDURobotNav/slopeLO

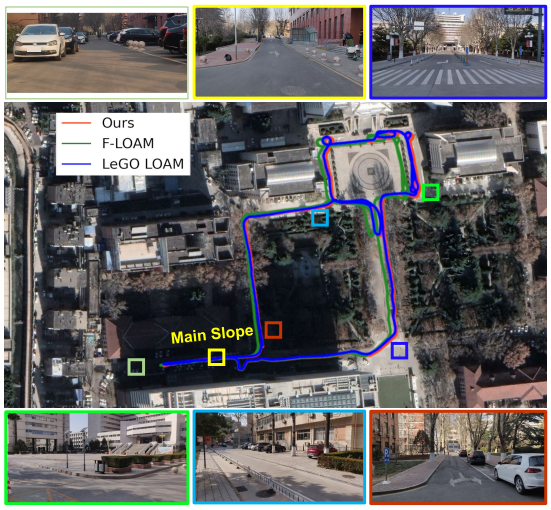



Fig. 9: The trajectories using different SLAM methods in the environment with non-level road.

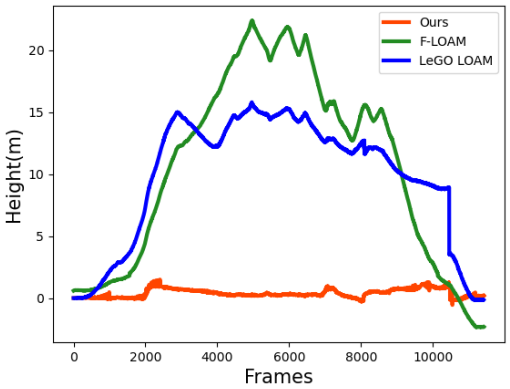



Fig. 10: The elevation change of the robot in traveling through the testing environments.

TABLE II: Statistics of metrics when the robot revisits the start place.

| | LeGO | F-LOAM | Ours |
|---|---|---|---|
| Path Length(m) | 1021.29 | 980.33 | 1039.52 |
| Elevation Err(m) | 15.99 | 24.74 | **2.05** |
| Position Err(m) | 1.61 | 9.12 | **1.48** |

the smallest elevation error compared with the other methods. These methods except ours show critical elevation drift when building the map.

To further demonstrate the effectiveness of the developed method, we conduct experiments at the place where the robot start building the map near the slope. The elevation error (Elevation Err(L)) is also recorded in Table I. Obviously, the developed method is better than the others in reducing the map elevation error. Notably, the slope elevation error in this local test is smaller compared with the comparative methods in global test, indicating there are accumulated errors with the other methods even the robot travels through the flat level ground. Compared with the LeGO-LOAM and the F-LOAM, the LIO-SAM shows better performance via introducing extra IMU in the SLAM process. Our developed method showcases similar performance in both local and global tests. It indicates that the introduction of the slope factor and the incremental pitch estimation method is effective in improving the map accuracy in environments with non-level roads.

### C. Region Revisiting Analysis

To further demonstrate the strength of the developed method, we conduct experiments in a relatively large-scale scenario in our campus. There is only one obvious slope there with 1.12 m height. The UGV runs alongside the road and finally revisits the starting position. It runs with different methods and the travel trajectories are depicted in Fig. 9. These trajectories are similar except that the trajectory using F-LOAM has a little drift. We depict the real-time elevation value of the vehicle when it runs in the campus, as shown in Fig. 10. One can observe that the altitude of the robot is almost the same when it returns the original start location with both our method and the LeGO-LOAM method. The F-LOAM shows relatively significant error in elevation.

Furthermore, to quantitatively evaluate the developed method, we record the elevation error, the position error, and the traveled path distance in Table II. It is straightforward to see that the developed method exhibits smaller elevation error and position error when the vehicle returns to its starting place, which demonstrates the strength of our method. The elevation error using LeGO-LOAM is similar while the position error with that method is larger. Besides, it is of note that the distance our vehicle traveled is longer. It means that the accumulated error along the road using our method is the least compared with the other methods.

### D. Evaluation with Kitti Data-set

To verify the robustness of the developed framework in different application scenarios, we use the Kitti dataset that is widely used for the evaluation of a SLAM method. This dataset records the sensor (e.g., LiDAR, camera, and GPS) information equipped on a car. We use the typical *sequence 00* with 3724.187 meters route for comparison with other methods. The generated trajectories with different methods are shown in Fig. 11. The three trajectories all have slight drifts compared with the ground truth. The trajectories of LeGO-LOAM and our method successfully close the loop at the starting region while the trajectory of F-LOAM fails. It shows the effectiveness of the developed method in different application scenarios even without non-level road, demonstrating its potential in other navigation cases.

More specifically, the difference between the length of the generated trajectory and the ground truth is utilized as a evaluation metric *Length Error*. We also calculate the *Relative Pose Error (RPE)* and the *Absolute Trajectory Error (ATE)* that are defined in [27]. The evaluation metrics are presented in Table III. Although the *ATE* of our method is a little larger than that of the F-LOAM, the *Length Error* and the *RPE* of ours are the least, therefore the proposed method achieves relatively better performance than the others. Hence, our method is applicable and also able to provide convincing performance in more general application cases.

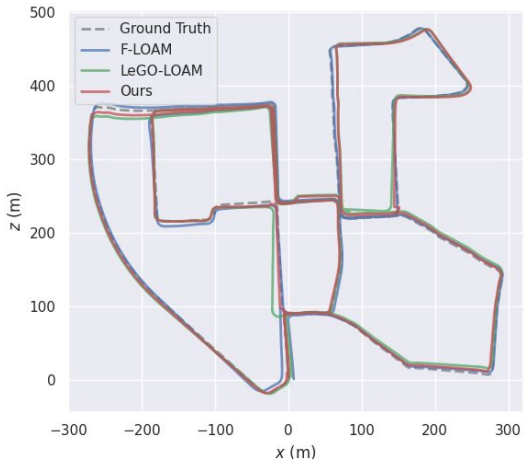



Fig. 11: The trajectories of different methods and the ground truth using the Kitti dataset.

TABLE III: Comparison evaluation with different metrics using Kitti dataset

| Methods | LeGO | F-LOAM | Ours |
|---|---|---|---|
| Length Error(m) | 16.165 | 22.900 | **2.501** |
| RPE(m) | 0.1085395 | 1.1520 | **0.071299** |
| ATE(m) | 8.312153 | **4.043993** | 5.082818 |

## V. Conclusions and Future Work

In this paper, we present a LiDAR-based localization and mapping approach for robot navigation in outside environments with non-level road. In contrast to the other methods, the developed method integrates the slope information into the SLAM framework, which forms useful constraints and references for the LiDAR odometry and the map building. The experimental results demonstrate that our framework can lead to low-drift pose estimation and build a more precise map compared with state of the arts. Notably, it can also achieve similar or even better performance in environments without non-level road.

Currently, the developed method is evaluated on structured slope way in urban environments. In the future, we will further develop our framework to make it useful in environments with various uneven terrains.

## References


[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] H. Zhao, M. Chiba, R. Shibasaki, X. Shao, J. Cui, and H. Zha, "Slam in a dynamic large outdoor environment using a laser scanner," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1455–1462.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[5] J. Nieto, T. Bailey, and E. Nebot, "Recursive scan-matching slam," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 39–49, 2007, simultaneous Localisation and Map Building. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889006001461

[6] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

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

[8] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[9] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

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

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

[12] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[13] H. Wang, C. Wang, C. Chen, and L. Xie, "F-loam : Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[14] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

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

[16] J. Lin and F. Zhang, "A fast, complete, point cloud based loop closure for lidar odometry and mapping," *arXiv preprint arXiv:1909.11811*, 2019.

[17] M. Ramezani, G. Tinchev, E. Iuganov, and M. Fallon, "Online lidar-slam for legged robots with robust registration and deep-learned loop closure," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4158–4164.

[18] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[19] T. Shan, B. Englot, C. Ratti, and D. Rus, "Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5692–5698.

[20] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments," in *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[21] D. Rozenberszki and A. Majdik, "LOL: Lidar-only Odometry and Localization in 3D point cloud maps," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[22] L. Zhao, M. Wang, and Y. Yue, "Sem-aug: Improving camera-lidar feature fusion with semantic augmentation for 3d vehicle detection," *IEEE Robotics and Automation Letters*, pp. 1–8, 2022.

[23] Q. Zhang, M. Wang, Y. Yue, and T. Liu, "Lcr-smm: Large convergence region semantic map matching through expectation maximization," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2021.

[24] Y. Yue, M. Wen, C. Zhao, Y. Wang, and D. Wang, "Cosem: Collaborative semantic map matching framework for autonomous robots," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 4, pp. 3843–3853, 2022.

[25] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based Semantic SLAM," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[26] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[27] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.