electronics

MDPI

Article

LiDAR SLAM with a Wheel Encoder in a Featureless Tunnel Environment

Iulian Filip [1], Juhyun Pyo [2], Meungsuk Lee [2] and Hangil Joe [1,*]

[1] Department of Robot and Smart System Engineering, Kyungpook National University, Daegu 41566, Republic of Korea
[2] Korea Institue of Robotics and Technology Convergence, Pohang 37666, Republic of Korea
* Correspondence: hgjoe@knu.ac.kr

Abstract: Simultaneous localization and mapping (SLAM) represents a crucial algorithm in the autonomous navigation of ground vehicles. Several studies were conducted to improve the SLAM algorithm using various sensors and robot platforms. However, only a few works have focused on applications inside low-illuminated featureless tunnel environments. In this work, we present an improved SLAM algorithm using wheel encoder data from an autonomous ground vehicle (AGV) to obtain robust performance in a featureless tunnel environment. The improved SLAM system uses FAST-LIO2 LiDAR SLAM as the baseline algorithm, and the additional wheel encoder sensor data are integrated into the baseline SLAM structure using the extended Kalman filter (EKF) algorithm. The EKF algorithm is used after the LiDAR odometry estimation and before the mapping process of FAST-LIO2. The prediction step uses the wheel encoder and inertial measurement unit (IMU) data, while the correction step uses the FAST-LIO2 LiDAR state estimation. We used an AGV to conduct experiments in flat and inclined terrain sections in a tunnel environment. The results showed that the mapping and the localization process in the SLAM algorithm was greatly improved in a featureless tunnel environment considering both inclined and flat terrains.

Keywords: LiDAR SLAM; wheel encoder; tunnel

Citation: Filip, I.; Pyo, J.; Lee, M.; Joe, H. LiDAR SLAM with a Wheel Encoder in a Featureless Tunnel Environment. *Electronics* **2023**, *12*, 1002. https://doi.org/10.3390/electronics12041002

Academic Editors: Peter Sarcevic, Sašo Tomažič, Akos Odry, Sara Stančin and Gábor Kertész

Received: 31 December 2022
Revised: 20 January 2023
Accepted: 24 January 2023
Published: 17 February 2023

1. Introduction

Mobile robots are beginning to replace human tasks such as inspections [1,2]. Detailed inspection tasks require a significant amount of time and high costs. The accurate 3D mapping of the environment can be conducted with the availability of high-precision and accurate sensors such as cameras and LiDAR. These maps can help identify specific areas that require maintenance and reduce the overall inspection time [3]. The robot must localize itself on the map to perform 3D mapping of the environment; the corresponding algorithm is called simultaneous localization and mapping (SLAM). Much research has been conducted to create SLAM algorithms using various sensors or a fusion of multiple sensors.

For autonomous ground vehicles (AGVs), cameras are one of the most widely used sensors in the state-of-the-art SLAM algorithms. Notably, visual SLAM [4,5] algorithms work by extracting information from the camera images. Although visual SLAM algorithms offer good results, they are strongly dependent on the environment illumination [6,7]; they do not perform well in low-light environments. Moreover, LiDAR sensors are also widely used in SLAM algorithms; they are very precise and have high resolution. Additionally, LiDAR sensors are not affected by the illumination variations, causing them to be suitable for operation in low-light environments such as tunnels.

Feature-based LiDAR SLAM algorithms [8–11] work by extracting key features, usually planar and edge features, from the point cloud. These key features are then used to perform LiDAR odometry and scan matching. Thus, as not all the LiDAR points are utilized, the algorithm's computational complexity is reduced. However, the preprocessing of

the LiDAR data is first required to extract the features. Notably, direct-based LiDAR SLAM algorithms methods [12,13] use all the LiDAR data points. Performing LiDAR odometry and scan matching with all the LiDAR data points usually requires a large amount of computational resources; to address this concern, different methods for storing and matching the data points have been explored to reduce the computational complexity. However, most LiDAR SLAM algorithms are of a feature-based nature. As the LiDAR data are collected sequentially, we first need to correct the distorted points to use the data. For this, LiDAR-only SLAM algorithms use the constant velocity model for the LiDAR to correct the distortion. These algorithms [8,11] can perform fairly well for LiDAR with a constant motion; however, if the motion is aggressive, their performance decreases. LiDAR-inertial SLAM algorithms [10,13] use IMU data for LiDAR distortion correction. Moreover, some algorithms [10,12] incorporate additional sensors such as an IMU and a GPS to reduce the accumulated LiDAR odometry drift; other algorithms [9,10] reduce this accumulated drift by performing "loop closure," which recognizes the already visited place and readjusts the obtained map.

LOAM [8] extracts features from the LiDAR data and achieves accurate results with low errors for drift when using only the LiDAR sensor. Many LiDAR SLAM algorithms [9–11] employ a feature extraction method similar to that in LOAM. In [9], a lightweight, ground-optimized LOAM (LeGO-LOAM) method was proposed that employed the LOAM algorithm, modifying and optimizing it on an AGV. The research conducted in [10] presented a factor-based tightly coupled LiDAR-inertial odometry via smoothing and mapping (LIO-SAM), which allowed for the addition of other sensor measurements, such as an IMU to predict the LiDAR motion during the scanning process, a GPS, and a loop closure feature for drift error elimination. Another factor-based graph algorithm, HDL-Graph-SLAM, was presented in [12], which implemented the IMU sensor measurements and used the normal distributions transform (NDT) [14] scan matching algorithm instead of the traditionally used iterative closest points (ICP) [15] method. FAST-LIO2 [13] is a tightly coupled LiDAR-inertial framework that directly uses raw points instead of extracting the features from the dataset when updating the map using an incremental kd-tree [16] structure. Wang et al. developed an F-LOAM [11] that used LOAM as a base platform and improved it by considering the shape of the extracted features and replacing the traditional iterative distortion compensation with a two-part noniterative compensation method. Other works have focused on surveying LiDAR SLAM algorithms. In [17], Akpınar et al. presented a LiDAR SLAM comparison between the feature-rich outdoor and indoor environments. Filipenko et al. compared the performances of visual and LiDAR SLAMs [18].

The performances of the briefly described SLAM algorithms have been tested in a feature-rich environment. Most tests were performed outside, where the environment is rich in distinct features. However, if we consider implementing the discussed LiDAR SLAM algorithms in a tunnel environment, the results might not be optimal, because tunnel environments usually lack strong features, the walls are smooth, and the LiDAR scans are very similar. As the LiDAR moves through the tunnel, accurate scan matching becomes difficult, leading to increasing drift errors. Additionally, the loop closure algorithm might detect incorrect loops owing to the small distinction between the LiDAR scans. Absolute measurement sensors such as GPS, which can help localize a robot, cannot be used because GPS sensors do not work in a tunnel environment. Much LiDAR SLAM research has been conducted; however, few studies have focused on LiDAR SLAM performance in a featureless tunnel environment. This was the motivation behind the work presented in this paper. We focused on obtaining a robust LiDAR SLAM algorithm for a featureless tunnel environment that can, in turn, be applied in tunnel–wall inspection by employing an AGV. The contributions of this study are summarized as follows:

- We modify a state-of-the-art LiDAR SLAM algorithm by incorporating additional wheel encoder sensor data from an unmanned ground vehicle by employing an EKF

to improve the localization and mapping accuracy of the SLAM framework in a dark, featureless tunnel environment.
- We improve the LiDAR SLAM algorithm localization and mapping accuracy by implementing additional sensor data from the ground vehicle's wheel encoders.
- We performed extensive experiments in flat and inclined terrain sections of the tunnel environment to evaluate the LiDAR SLAM performance.

The paper is organized as follows. Section 2 describes the related work. Section 3 presents the proposed method, followed by the experiment and results in Section 4. Finally, the discussion and conclusions are presented in Sections 5 and 6, respectively.

2. Related Work

Before conducting the work described in this paper, we used the same robot platform to compare seven state-of-the-art LiDAR SLAM algorithms in the tunnel environment [19]. The compared LiDAR SLAM algorithms were as follows: LeGO-LOAM [9], LIO-SAM [10], F-LOAM [11], HDL-SLAM [12], FAST-LIO2 [13], SC-LeGO-LOAM [20], and SC-LIO-SAM [21]. We considered increasing the number of features by adding artificial landmarks to improve the performance of the LiDAR SLAM frameworks in a featureless tunnel environment. Feature-based LiDAR SLAM algorithms usually extract two kinds of features, namely, planar and edge. As planar features are primarily present in the tunnel environment (edge features being scarce), we designed our artificial landmarks to have strong edge features. The experiment results show that the addition of artificial landmarks improves the performance of the LiDAR SLAM algorithms. However, considering the tunnel length and the manufacturing and installation time and costs, the use of artificial landmarks to enhance the LiDAR SLAM performance cannot be considered as an optimal solution. For this reason, we explored the incorporation of the AGV wheel encoder data into the SLAM framework to improve its performance.

When the LiDAR SLAM algorithm runs using an AGV platform, it becomes useful if the wheel encoder information can be incorporated into the algorithm to help reduce the drift errors. Depending on the robot platform, the wheel encoder data can be available at different rates. Yun Su et al. developed GR-LOAM [22], which is a LiDAR SLAM algorithm optimized for ground vehicles that incorporates wheel encoder sensor data that are published at a rate of 100 Hz. The fast data publishing rate of the wheel odometry allows it to be used for the pre-integration step to correct the LiDAR distortion during the LiDAR scan. GR-LOAM uses a wheel encoder and IMU-fused odometry model for LiDAR pose estimation, where the weight of each sensor is adjusted dynamically. The features are extracted from the LiDAR data, and the robot pose is estimated. The ground point constraint is used to further optimize the pose estimates. The two-wheel kinematic model is employed to obtain the wheel odometry. The intrinsic parameters of the wheel odometry model are experimentally calculated.

EKF-LOAM [23] is another work that incorporates wheel odometry information, which employs LeGO-LOAM as the base algorithm and improves its odometry estimation by fusing LiDAR, IMU, and wheel encoder data into an EKF. As the wheel encoder information is obtained at a lower rate of 20 Hz, only IMU information is used for the pre-integration process. LiDAR odometry is estimated after the features are extracted from the LiDAR information. Usually, in the LiDAR SLAM pipeline, the LiDAR odometry information is sent to the optimization part of the algorithm for the mapping and pose estimation processes. However, in EKF-LOAM, the LiDAR odometry information is sent to an extended Kalman filter that fuses it with IMU and encoder data; the skid steering kinematic model is employed to obtain the wheel odometry information [24].

Both GR-LOAM and EKF-LOAM algorithms incorporate wheel encoder data into the SLAM framework but they do it differently. GR-LOAM uses it in the pre-integration step, while EKF-LOAM uses it after the LiDAR odometry is obtained. Considering our preliminary LiDAR comparison work and the featureless tunnel environment, we selected a direct-based LiDAR SLAM algorithm as the base algorithm. Selecting a feature-based

LiDAR SLAM as the base algorithm is not a favorable option owing to the lack of edge features in the tunnel environment. After conducting the preliminary LiDAR SLAM comparison work, we selected FAST-LIO2 as our base algorithm. FAST-LIO2 is a tightly coupled LiDAR-inertial direct-based method that achieves better results compared to the feature-based methods in the featureless tunnel environment. As we used a fast-rate, accurate IMU sensor in our experiment, the pre-integration step of FAST-LIO2 does not produce large errors. Instead, the LiDAR odometry estimation produces large errors due to the presence of low features in the tunnel environment. Therefore, we proposed incorporating the wheel encoder data into the FAST-LIO2-based algorithm. The proposed method has several differences from the related works [22,23]. Although LeGO-LOAM is a lightweight, optimized LiDAR SLAM for the AGV, it performs state estimation using the extracted planar and edge features. The tunnel environment lacks edge features due to the long, continuous smooth walls. As a result, the state estimation of LeGO-LOAM in tunnel environment has large errors. In contrast, FAST-LIO2 is a more suitable base algorithm because it directly uses the LiDAR raw data without any need for extracting the features. Similar to LeGO-LOAM, FAST-LIO2 is a computationally inexpensive algorithm, causing it to be suitable for usage on single-board computers. Additionally, by using the wheel encoder data from AGV, we plan to improve the performance of the FAST-LIO2 on the inclined terrain section of the tunnel environment because FAST-LIO2 outputs high error results in our preliminary experiment on the inclined terrain [19].

3. Proposed Method

This section starts by describing the original FAST-LIO2 framework overview, proceeding further with the motion model definition and EKF equations. Finally, the algorithm flow overview of the improved FAST-LIO2 LiDAR SLAM is presented.

3.1. FAST-LIO2 Overview

FAST-LIO2 [13] is a tightly coupled direct-based LiDAR-inertial SLAM algorithm. Unlike feature-based LiDAR SLAM algorithms, which extract edge and planar features from the LiDAR data, FAST-LIO2 directly uses the LiDAR point-cloud data. This facilitates less computational power requirements and time consumption owing to the exclusion of the LiDAR data preprocessing step. However, the direct use of raw data causes the scan matching process to be computationally heavy. To solve this issue, Wei Xu et al. created a new lightweight Kalman gain calculation formula that allows for performing a scan matching function using the raw LiDAR points directly while consuming low computational power. Additionally, an incremental k-d tree structure is employed to perform the mapping process. Moreover, the use of two parallel nodes allows for the incremental updates and dynamic rebalancing of the k-d tree structure to simultaneously occur. Figure 1 shows a simplified system overview diagram.

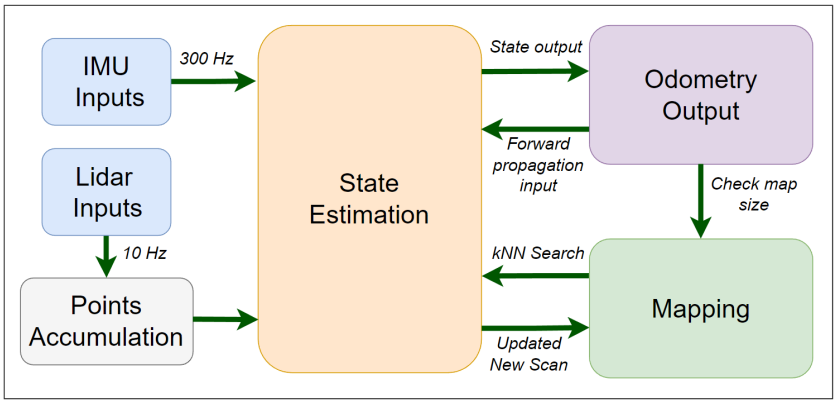

Figure 1. Simplified system overview of the original FAST-LIO2 SLAM algorithm.

The FAST-LIO2 algorithm performs accurate mapping and localization processes while using low computational power. In [13], Wei Xu et al. conducted a comparison of the average processing time per laser scan among several state-of-the-art LiDAR SLAM algorithms on different datasets. The results show that FAST-LIO2 spends less processing time per laser scan compared to other SLAM frameworks. This and the ability to use the raw LiDAR points directly cause FAST-LIO2 to be a suitable SLAM for use in a featureless tunnel environment by employing an onboard computer with limited resources on a ground vehicle.

3.2. Motion Model

We use a four-wheeled ground vehicle in our experiments. The vehicle employs a skid steering motion model, as shown in Figure 2a. Considering the robot's constraints, let v and ω, the linear and angular velocity vectors in the robot frame, respectively, be defined as $v = \begin{pmatrix} v_x & 0 & 0 \end{pmatrix}^T$ and $\omega = \begin{pmatrix} 0 & 0 & \omega \end{pmatrix}^T$. Let ω_{FL}, ω_{BL}, ω_{FR}, and ω_{BR} be the angular velocities of the front-left, back-left, front-right, and back-right wheels, respectively. Moreover, let v_{FL}, v_{BL}, v_{FR}, and v_{BR} be the linear velocities of the front-left, back-left, front-right, and back-right wheels, respectively. During our experiment, the robot follows a rectangular trajectory path on the ground. The robot performs a straight motion and it rotates at the corners of the trajectory while being stationary. Considering this motion, we simplify our motion model to the two-wheel differential drive motion model represented in Figure 2b.

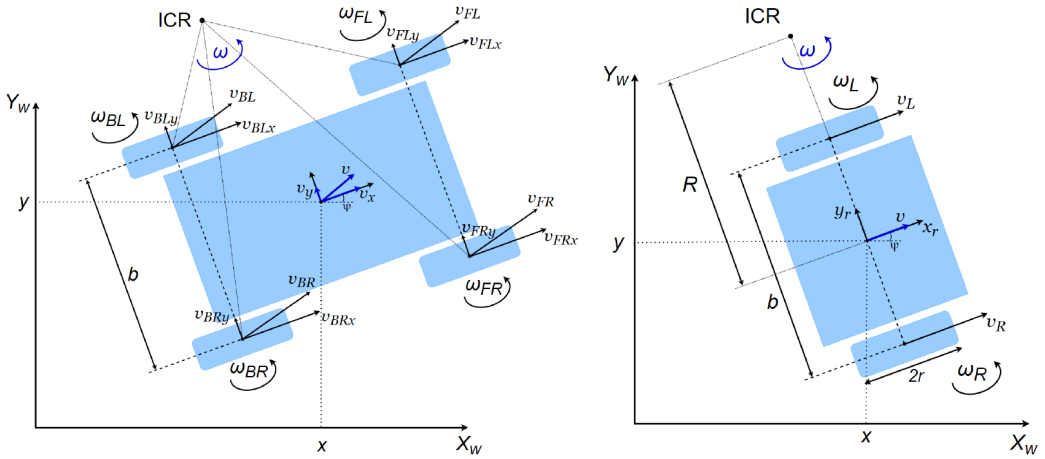

Figure 2. Motion model: (**a**) skid steering, and (**b**) two-wheel differential drive.

Let v_L and v_R be the sum of linear velocities of the wheels from the left and right side, respectively. Then, we have the following relation:

$$\begin{pmatrix} v_R \\ v_L \end{pmatrix} = r \begin{pmatrix} \omega_R \\ \omega_L \end{pmatrix} = 0.5r \begin{pmatrix} \omega_{FR} + \omega_{BR} \\ \omega_{FL} + \omega_{BL} \end{pmatrix}, \tag{1}$$

where ω_L and ω_R are the sum of angular velocities of the wheels on the left and right side, respectively. Expressing the robot's linear and angular velocity in terms of the wheel's linear velocity, we have

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} 0.5(v_R + v_L) \\ (1/b)(v_R - v_L) \end{pmatrix}. \tag{2}$$

Writing Equation (2) in terms of the robot wheel's angular velocity, we have

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = 0.5r \begin{pmatrix} 0.5(\omega_{FR} + \omega_{BR} + \omega_{FL} + \omega_{BL}) \\ (1/b)(\omega_{FR} + \omega_{BR} - \omega_{FL} - \omega_{BL}) \end{pmatrix}, \tag{3}$$

where r is the radius of the wheel and b is the distance between the right and left wheels.

Let v_G be the vector of the robot's velocity in the global coordinate frame. Considering the nonholonomic constraints, the rotation of the robot around the roll axis does not affect its motion; therefore, we consider the rotation matrix around the roll axis, R_x, to be the identity matrix. Thus, using the information from the motion model described in the previous section, we have the global velocity vector equal to

$$v_G = R_x \cdot R_y \cdot R_z \cdot \begin{pmatrix} v_x \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} v_x \cos(\theta) \cos(\psi) \\ v_x \cos(\theta) \sin(\psi) \\ -v_x \sin(\theta) \end{pmatrix}. \tag{4}$$

3.3. EKF

We employ an extended Kalman filter (EKF) algorithm to incorporate the wheel encoder data into the SLAM odometry estimation structure. The EKF is a widely used algorithm for combining multiple sensor measurements. It comprises three stages: the prediction step, Kalman gain calculation, and the correction step. The FAST-LIO2 SLAM framework estimates the odometry at the same frequency as the LiDAR sensor (i.e., 10 Hz). Our robot platform provides angular velocity and linear velocity of the robot at a frequency of 50 Hz. As the robot velocity information is provided at a much higher frequency, it is used in the prediction step. Moreover, the LiDAR odometry estimated by the SLAM algorithm is used in the correction step. As the robot moves with a relatively constant speed during the experiment run, we consider the motion with zero acceleration for estimating the robot displacement during the prediction step. We define the robot state with six components. Let $s = \begin{pmatrix} x_G & y_G & z_G & \phi_G & \theta_G & \psi_G \end{pmatrix}^T$, where x_G, y_G, and z_G are the global position coordinates and ϕ_G, θ_G, and ψ_G are the orientation roll, pitch, and yaw angles of the robot, respectively. We define the robot state update in Equation (5), where $\triangle s$ is the state change in the time $\triangle t$ between two consecutive robot states:

$$s_{t+1} = s_t + \triangle s_t. \tag{5}$$

Let the state change be $\triangle s = \begin{pmatrix} \triangle x_G & \triangle y_G & \triangle z_G & \triangle \phi_G & \triangle \theta_G & \triangle \psi_G \end{pmatrix}^T$. Our IMU sensor publishes accurate orientation values at a fast rate. We simplify our state update and use the orientation values of roll, pitch, and yaw directly from the sensor. By considering the robot velocity in the global frame defined in Equation (4), the state update then becomes

$$s_{t+1} = \begin{pmatrix} x_{G,t} \\ y_{G,t} \\ z_{G,t} \\ \phi_{G,t+1} \\ \theta_{G,t+1} \\ \psi_{G,t+1} \end{pmatrix} + \begin{pmatrix} v_x \cos(\theta_{G,t+1}) \cos(\psi_{G,t+1}) \\ v_x \cos(\theta_{G,t+1}) \sin(\psi_{G,t+1}) \\ -v_x \sin(\theta_{G,t+1}) \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot \triangle t. \tag{6}$$

Only the position updates incrementally. The orientation values are collected at each timestamp from the sensor. Let $\hat{s}$ be the EKF prediction step state and P be the prediction step covariance matrix. The prediction step can then be defined as follows:

$$\hat{s} = s_{t+1} = \begin{pmatrix} x_{G,t} + v_{x,t+1} \cos(\theta_{G,t+1}) \cos(\psi_{G,t+1}) \triangle t \\ y_{G,t} + v_{x,t+1} \cos(\theta_{G,t+1}) \sin(\psi_{G,t+1}) \triangle t \\ z_{G,t} - v_{x,t+1} \sin(\theta_{G,t+1}) \triangle t \\ \phi_{G,t+1} \\ \theta_{G,t+1} \\ \psi_{G,t+1} \end{pmatrix} \tag{7}$$

$$\hat{P} = F \cdot P_0 \cdot F^T + Q, \tag{8}$$

where F is the Jacobian matrix of the predicted state $\hat{s}$ and P_0 and Q are covariance matrices; these matrices are defined as follows:

$$P_0 = \begin{pmatrix} \sigma_{p,x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{p,y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{p,z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{p,\phi}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{p,\theta}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{p,\psi}^2 \end{pmatrix} \tag{9}$$

$$Q = \begin{pmatrix} \sigma_{q,x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{q,y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{q,z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{q,\phi}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{q,\theta}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{q,\psi}^2 \end{pmatrix} \tag{10}$$

$$F = J(\hat{s}) = \begin{pmatrix} 1 & 0 & 0 & 0 & -v_x \sin(\theta_G)\cos(\psi_G) \triangle t & -v_x \cos(\theta_G)\sin(\psi_G) \triangle t \\ 0 & 1 & 0 & 0 & -v_x \sin(\theta_G)\sin(\psi_G) \triangle t & v_x \cos(\theta_G)\cos(\psi_G) \triangle t \\ 0 & 0 & 1 & 0 & -v_x \cos(\theta_G) \triangle t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{11}$$

with the covariance values being $\sigma_{p,x}^2 = \sigma_{p,y}^2 = \sigma_{p,z}^2 = 0.1$, $\sigma_{p,\phi}^2 = \sigma_{p,\theta}^2 = \sigma_{p,\psi}^2 = 0.01$, $\sigma_{q,x}^2 = \sigma_{q,y}^2 = \sigma_{q,z}^2 = 0.01$, and $\sigma_{q,\phi}^2 = \sigma_{q,\theta}^2 = \sigma_{q,\psi}^2 = 0.001$. Let $s_L = \begin{pmatrix} x_L & y_L & z_L & \phi_L & \theta_L & \psi_L \end{pmatrix}^T$ be the measurement robot state that is provided by the FAST-LIO2 LiDAR odometry. The Kalman gain is defined as follows:

$$K = \hat{P} \cdot H^T (H \cdot \hat{P} \cdot H^T + E)^{-1}, \tag{12}$$

where H is the Jacobian of the measurement state and E is the measurement covariance matrix; these matrices are defined as follows:

$$H = J(s_L) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{13}$$

$$E = \begin{pmatrix} \sigma_{e,x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{e,y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{e,z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{e,\phi}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{e,\theta}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{e,\psi}^2 \end{pmatrix}, \tag{14}$$

with the covariance values being $\sigma_{e,x}^2 = \sigma_{e,y}^2 = \sigma_{e,z}^2 = 1$ and $\sigma_{e,\phi}^2 = \sigma_{e,\theta}^2 = \sigma_{e,\psi}^2 = 5$. As the Jacobian of the measurement state is the identity matrix, the Kalman gain and correction step are as follows:

$$K = \hat{P} \cdot (\hat{P} + E)^{-1} \tag{15}$$

$$s = \hat{s} + K \cdot (s_L - \hat{s}) \tag{16}$$

$$P = \hat{P} - \hat{P} \cdot K, \tag{17}$$

where s and P are the updated robot state and covariance matrix, respectively. As the prediction and correction steps run at 50 and 10 Hz, respectively, five prediction steps occur before the correction step happens. Notably, the covariance matrices values were experimentally tuned to obtain the best results.

3.4. Wheel Encoder Aided FAST-LIO2

As per the results of the previous experiments conducted in the featureless tunnel, FAST-LIO2 does not always output accurate results. To improve its performance in the tunnel environment, we propose incorporating the additional ground vehicle's encoder data into the SLAM odometry estimation process by using the motion model and EKF algorithm described in the Sections 3.2 and 3.3, respectively. Figure 3 depicts the new SLAM system overview.

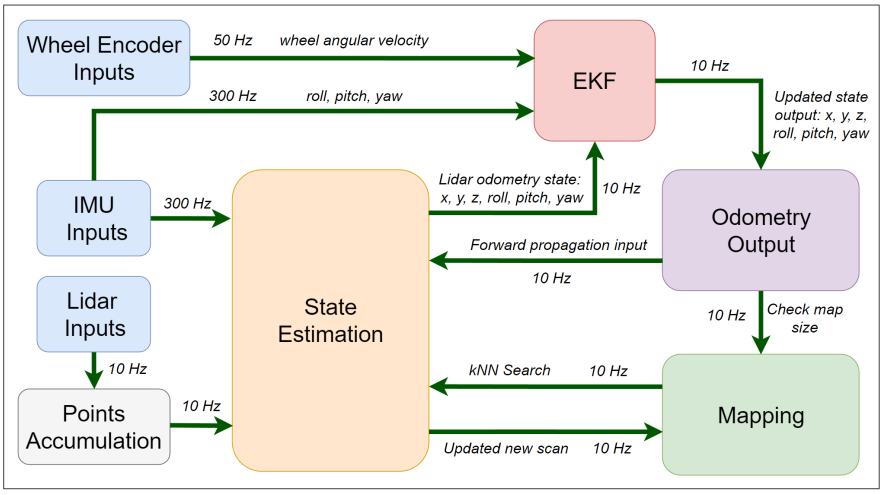

Figure 3. Overview of the modified FAST-LIO2 SLAM algorithm.

The original FAST-LIO2 algorithm starts with the IMU measurements by performing forward and backward propagation steps to correct the LiDAR distortion motion. The state estimation is then obtained using the iterated Kalman filter on the manifold [25] technique until convergence is achieved. Our developed EKF algorithm is then applied to further improve the state estimation. The wheel encoder and IMU data are used for the EKF prediction step that runs at 50 Hz. The IMU data are used for the orientation estimation, while the IMU and wheel encoder data are used for the displacement estimation. The EKF correction step running at 10 Hz uses the estimated state from the original SLAM structure as the measurement state input. After the state is updated, the new odometry information is sent to the mapping process. Thus, in summary, our EKF algorithm improves the LiDAR position and orientation estimation during the LiDAR scan, thereby improving the scan matching and mapping process in the featureless tunnel environment.

4. Experiment and Results

This section starts with a description of the setup used for the experiments conducted in the featureless tunnel environment and error metrics used for the performance evaluation. It is then followed by the experimental results obtained. As we conducted experiments in the flat (horizontal) and inclined terrain sections of the tunnel, we present their results separately.

4.1. Experiment Setup

For our experiments in the tunnel environment, we utilized a four-wheeled skid steering AGV. Figure 4 shows the vehicle's configuration. The robot provides odometry data from wheel encoders at a frequency of 50 Hz; it is additionally equipped with 3D LiDAR and IMU sensors running at 10 and 300 Hz, respectively. The ground vehicle runs with a maximum speed of 1.5 m/s and has a maximum payload of 50 kg. Being powered by a 24 V 30 Ah battery, it can run for 4 h. Additionally, an onboard Jetson single-board computer is installed on the vehicle, which communicates using the controller area network interface.

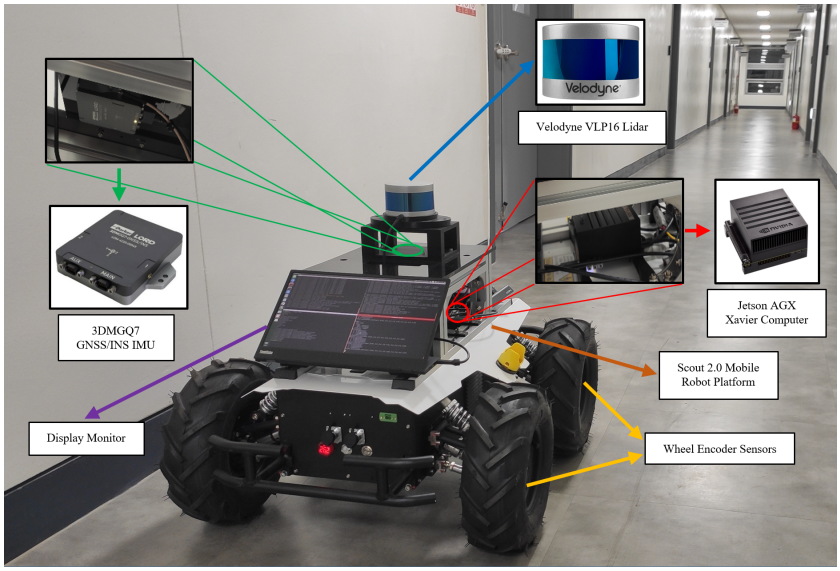

Figure 4. AGV configuration.

The robot, which is installed on the ground using ropes, is manually operated during the experiments to follow a preset trajectory. The robot follows the trajectory such that the LiDAR-sensor center always remains above the trajectory. The onboard computer is used to collect the sensor data that are later used for the performance evaluation. Ubuntu 18.04 with ROS Melodic runs on the onboard computer. The LiDAR sensor, with a horizontal field of view (FOV) of 360° and vertical FOV of 30°, communicates with the computer using the Ethernet interface. The LiDAR accuracy is up to a few centimeters; the measurement range reaches up to 100 m. The IMU establishes a connection with the onboard computer using the USB connector. It outputs accurate data, with a 2-cm positional, a 0.2° heading accuracy, and a variable sampling rate of up to 1 kHz. The LiDAR sensor is fixed on top of the robot's center of rotation, while the IMU sensor is placed under the LiDAR sensor so that the vertical fixed frame axes of both sensors coincide. The fixed frame coordinates of the LiDAR and IMU sensors also coincide; thus, in the tested SLAM algorithm (FAST-LIO2 in our case) configuration settings, the LiDAR–IMU extrinsic matrix is set to the identity matrix.

To evaluate the performances of the original and modified FAST-LIO2 algorithms, the experiment is performed in a nuclear facility underground the featureless tunnel flat and inclined terrain sections shown in Figure 5a and Figure 5b, respectively. On the ground, a rectangular trajectory with a size of $35.0 \cdot 4.5$ m is set. The ground vehicle is then manually operated to follow this set trajectory. In both the terrain sections of the tunnel, we considered the same trajectory sizes and same experiment runs. We performed two types of experiment runs. The first one consists of following the rectangular trajectory and completing one loop by returning to the initial position. For the second type, we perform two loops of the same trajectory before returning to the initial position. At the trajectory corners, the robot is turning while not performing any forward or backward motion. During the experiment runs, we recorded a dataset from three sensors, namely, LiDAR, IMU, and wheel encoders. In each of

the tunnel sections, we conducted four different experiment runs and two different runs for the one- and two-loop trajectory paths, respectively.

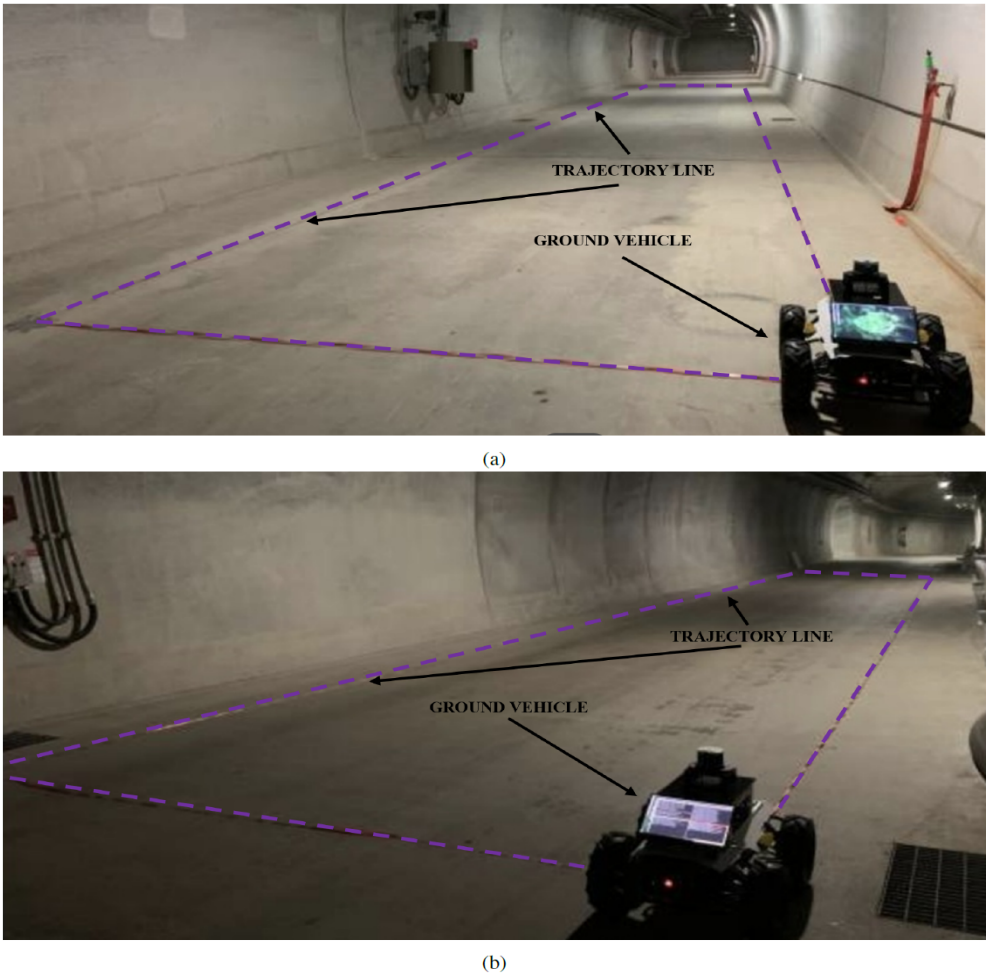

Figure 5. Experiment setup in the tunnel environment: (**a**) flat (horizontal) terrain section and (**b**) inclined terrain section.

4.2. Error Metrics

We employ the absolute trajectory error (ATE) and relative trajectory error (RTE) methods to evaluate the LiDAR SLAM algorithm performance. These error metrics are widely used for the trajectory evaluation of SLAM algorithms. Zhang et al. explain how to obtain these errors in detail in [26]. For obtaining ATE and RTE, we used the same methodology as described in [26]. ATE is obtained by calculating the root mean squared error (RMSE) of all trajectory points after aligning the estimated and ground truth trajectories. During our experiment, we follow a rectangular trajectory; therefore, the aligning process becomes simple. Thus, the equation for computing ATE becomes

$$ATE = \sqrt{\frac{1}{m}\Sigma_{j=1}^{m}(\hat{T}_j - T_j)^2}, \tag{18}$$

where $\hat{T}_j$ and T_j represent all the ground truth and estimated trajectory points, respectively.

When calculating RTE, we do not use the whole trajectory path but only a part of it. First, we select a number of sub-trajectories of length d from the whole path. After aligning the first point of the estimated trajectory with the ground truth, we then compute the end pose RMSE. Next, we repeat the same procedure but for different sub-trajectory lengths.

All the errors are collected, and the result is visualized using a boxplot. If n is the number of selected sub-trajectories with the length d, then RTE is represented as follows:

$$RTE_i = \sqrt{\frac{1}{m}\Sigma_{j=1}^{m}(\hat{S}_j - S_j)^2} \tag{19}$$

$$RTE = \begin{pmatrix} RTE_1 & RTE_2 & \cdots & RTE_n \end{pmatrix}, \tag{20}$$

where $i \in \begin{pmatrix} 1 & 2 & \cdots & n \end{pmatrix}$ and $\hat{S}_j$ and S_j represent all the ground truth and estimated sub-trajectory points, respectively. ATE is a single value error metric, while RTE is a collection of multiple values that provide a better understanding of how the error is changing in different parts of the estimated trajectory.

4.3. Flat Section Results

This section presents the results obtained from the experiment in the flat terrain section of the tunnel. Figure 6 depicts the estimated single-loop trajectories from the original and modified FAST-LIO2, where trajectories a–d and e–f represent the single-loop and double-loop experiment runs, respectively. Table 1 presents the ATE results; Figure 7 depicts the RTE results as quartiles. The RTE shown in Figure 7a corresponds with the path displayed in Figure 6a and so on.

Table 1. ATE (in meters) for one- and two-loop paths in the flat section in the tunnel environment. L1–L4 represent the ATE for the one-loop paths, while L5.1–L6.2 represent that for the first and second loops of the two-loop paths.

SLAM Algorithm	L1	L2	L3	L4	L5.1	L5.2	L6.1	L6.2
FAST-LIO2	11.92	12.11	1.19	12.21	11.62	23.81	1.13	0.97
Proposed Method	1.32	1.23	1.11	1.17	1.18	0.93	1.09	1.02

As shown in Figure 6, in the single-loop runs for three out of four cases, the original FAST-LIO2 has similar path outputs, with the exception of the path in Figure 6c, where the estimated path is close to the ground truth. We can obtain completely different results in some cases despite conducting the experiments in the same environment and following the same trajectory. The same situation is manifested for the double-loop paths. In contrast, our modified FAST-LIO2 algorithm estimates the trajectory that is close to the ground truth in all the cases. As per the ATE results presented in Table 1, our modified FAST-LIO2 has reduced the errors significantly compared to the original algorithm; however, we cannot deduce much information from the ATE results, unlike the RTE results. Each estimated trajectory path has the RTE represented as four quartiles, signifying the trajectory error at a specific trajectory path and noted by the distance traveled by the robot platform. Thus, we can find the parts of the trajectory that display the highest and lowest errors. Three out of the four trajectories estimated by the original FAST-LIO2 for single-loop runs display similar RTE results. Specifically, we notice that the trajectory error is lower for the first half of the run, where the estimated path is close to the ground truth; when the robot is returning, the error is increased.

4.4. Inclined Section Results

This section presents the results obtained from the experiments conducted in the inclined terrain section of the tunnel, where the elevation information is also included in the estimated trajectories. Table 2 presents the ATE results for the inclined terrain section. Figure 8a–d and Figure 8e,f show the generated 3D paths for the single- and double-loop runs, respectively. Figure 9 shows the RTE values as quartiles. Similarly, the RTE results in Figure 9a correspond to the obtained path in Figure 8a and so on.

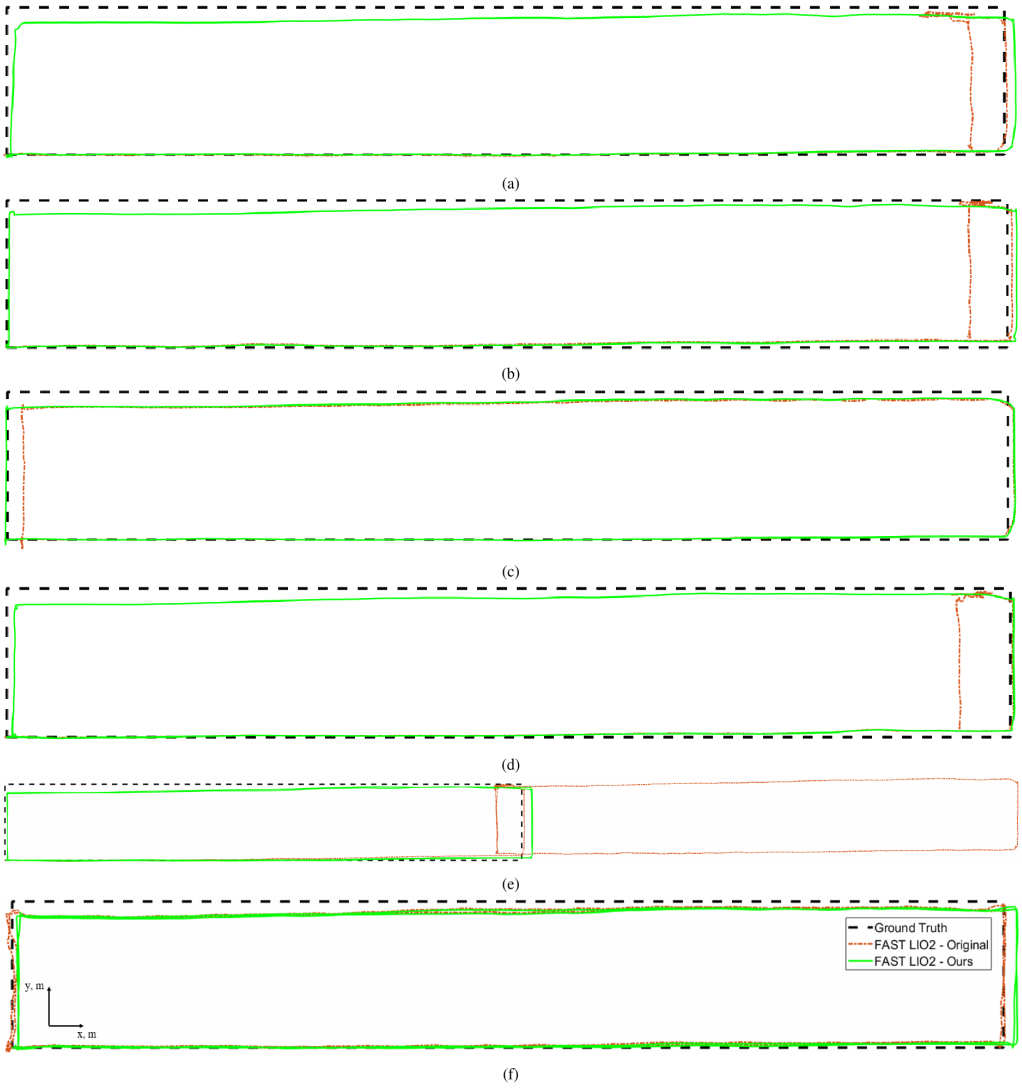

Figure 6. Estimated trajectory path comparison between the original and improved FAST-LIO2 in a flat terrain section of the tunnel environment: (**a**–**d**) single-loop runs, and (**e**,**f**) double-loop runs.

When comparing the generated paths in the inclined section with those in the flat section, we can observe that the performance of the original FAST-LIO2 is much worse in the inclined section (no generated path is close to the ground truth). The estimated trajectory is relatively good only in the y global direction. In contrast, our modified FAST-LIO2 algorithm generates paths that are close to the ground truth in all six experiment runs. Moreover, the elevation information of the trajectory is estimated well. Compared to the ATE results, the RTE results show a better understanding of the trajectory error throughout the whole paths. Specifically, from the estimated trajectories in Figure 8d,e we can see that the original FAST-LIO2 algorithm outputs a lower trajectory error in the latter part of the trajectory run compared to that of the other part.

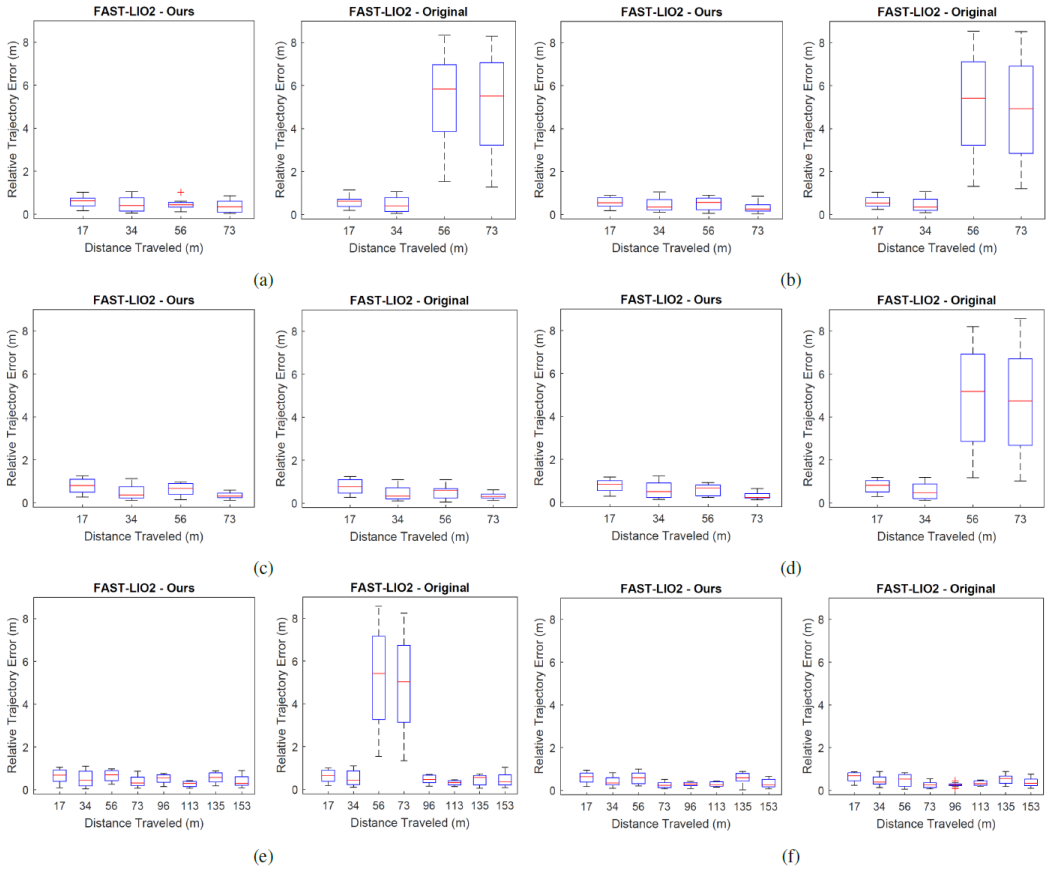

Figure 7. *RTE* results of the original and improved FAST-LIO2 in the flat terrain section of the tunnel environment: (**a**–**d**) single-loop runs, and (**e**,**f**) double-loop runs.

Table 2. *ATE* (in meters) for one- and two-loop paths in the inclined section in the tunnel environment. L1–L4 represent the *ATE* for the one-loop paths, while L5.1–L6.2 represent that for the first and second loops of the two-loop paths.

SLAM Algorithm	L1	L2	L3	L4	L5.1	L5.2	L6.1	L6.2
FAST-LIO2	12.43	15.42	11.94	11.24	12.06	19.54	11.63	17.06
Proposed Method	1.11	1.22	1.25	1.85	1.19	1.30	1.13	1.25

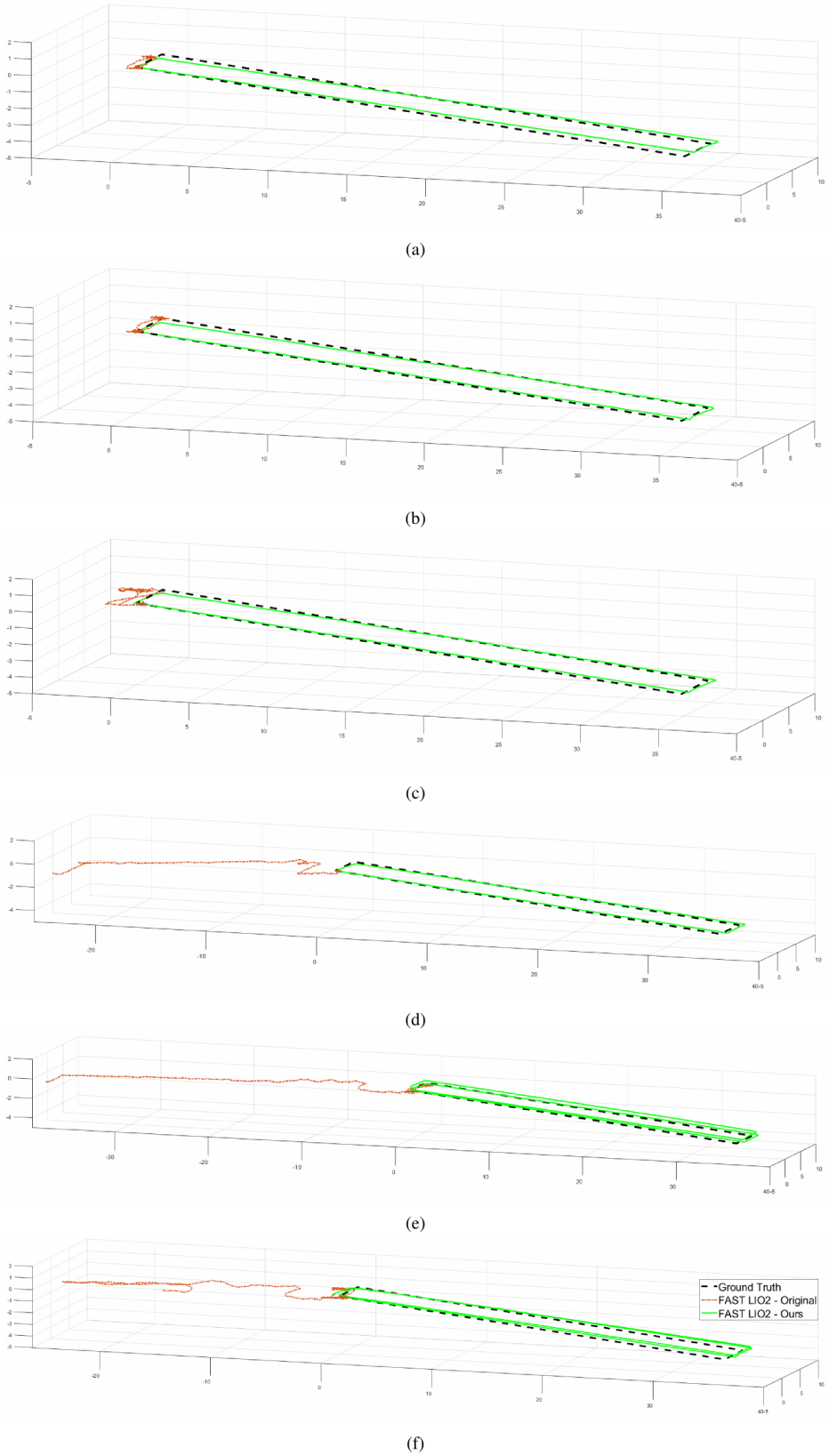

Figure 8. Estimated trajectory path comparison between the original and improved FAST-LIO2 in the inclined terrain section of the tunnel environment: (**a**–**d**) single-loop runs, and (**e**,**f**) double-loop runs.

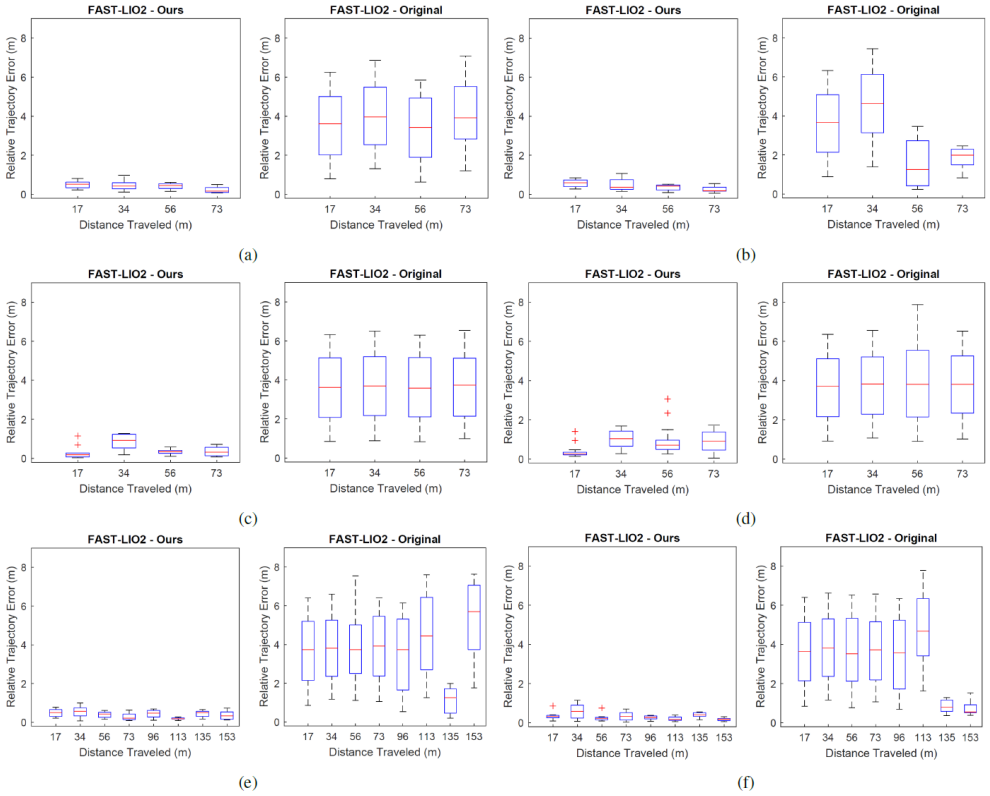

Figure 9. *RTE* results of the original and improved FAST-LIO2 in the inclined terrain section of the tunnel environment: (**a**–**d**) single-loop runs, and (**e**,**f**) double-loop runs.

5. Discussion

The results obtained in this paper showed that the modified FAST-LIO2 estimates better trajectory and has reduced *ATE* and *RTE* results compared to the original FAST-LIO2 algorithm in flat and inclined terrain sections of the tunnel environment. Despite having multiple similar trajectory runs, we observe that the output of the performance of the original FAST-LIO2 differs in some cases. In the flat section of the single-loop trajectory runs, one result has better trajectory estimation compared to the remaining three. We suspect that more features were present during the third single-loop run; thus, the scan matching process has better results. The increased number of features may be due to the presence of workers at the experiment site during the experiment run. When we look at the remaining trajectories estimated by the single-loop runs, they have the same *RTE* results for the second part of the loop. As stated earlier, the increase in the trajectory error in the second part of the loop may be due to the reduced feature points at the end of the trajectory compared to the beginning. Another reason for having different results may be due to the decreased repeatability performance of the FAST-LIO2 algorithm in a featureless environment. The modified FAST-LIO2 algorithm solves these issues, even in the case of a faulty scan matching process, due to reliable state prediction from the wheel encoders. These results are possible when the AGV moves at a constant speed. Our EKF prediction equations consider motion with zero acceleration. Similar results may not be obtained if we conduct experiments with aggressive AGV motion.

The results for the original FAST-LIO2 in the inclined terrain section of the tunnel are worse compared to those in the flat terrain section. Similar to the flat section experiments, we can observe that the repeatability of the SLAM results is not constant in similar experiment runs. We suspect a drop in the original FAST-LIO2 performance in the inclined section of the tunnel owing to a lesser feature presence compared to that in the flat section. Moreover, we observe that the original FAST-LIO2 incorrectly estimates the elevation information. Similar to the experiments in the flat section, our modified FAST-LIO2 algorithm

shows improved results due to the reliable displacement and orientation predictions from the wheel encoder and IMU data, respectively.

As the results in the flat and inclined terrain sections of the tunnel differ, using the same EKF parameter values for all the experiments does not always produce the best results. Depending on the original FAST-LIO2 algorithm performance, we adjust the covariance matrix values to improve the trajectory errors. In future work, we plan to employ adaptable covariance matrices that use the optimal values based on the feature richness of the environment.

6. Conclusions

This paper presented a modified state-of-the-art LiDAR SLAM algorithm that incorporates wheel encoder data from an AGV by employing an EKF. A direct-based LiDAR SLAM algorithm (namely, FAST-LIO2) was chosen as the base SLAM algorithm. We conducted experiments in flat and inclined terrain sections of a featureless tunnel environment. The experiment results show an improved localization and mapping for both tunnel sections. In future work, we consider employing an adaptable covariance matrix in the extended Kalman filter structure based on the number of distinct features present in the environment.

Author Contributions: Conceptualization, H.J.; methodology, I.F.; hardware configuration, J.P. and M.L.; writing—original draft preparation, I.F.; visualization, I.F.; supervision, H.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Kyungpook National University Research Fund, 2020.

Data Availability Statement: Not applicable.

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

References

1. Jang, H.; Kim, T.Y.; Lee, Y.C.; Song, Y.H.; Choi, H.R. Autonomous Navigation of In-Pipe Inspection Robot Using Contact Sensor Modules. *IEEE/ASME Trans. Mechatron.* **2022**, *27*, 4665–4674. [CrossRef]
2. Sahbel, A.; Abbas, A.; Sattar, T. System Design and Implementation of Wall Climbing Robot for Wind Turbine Blade Inspection. In Proceedings of the 2019 International Conference on Innovative Trends in Computer Engineering (ITCE), Aswan, Egypt, 2–4 February 2019; pp. 242–247. [CrossRef]
3. Yang, L.; Li, B.; Li, W.; Brand, H.; Jiang, B.; Xiao, J. Concrete defects inspection and 3D mapping using CityFlyer quadrotor robot. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 991–1002. [CrossRef]
4. Campos, C.; Elvira, R.; Gomez, J.J.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
5. Ferrera, M.; Eudes, A.; Moras, J.; Sanfourche, M.; Le Besnerais, G. OV^{2}SLAM: A Fully Online and Versatile Visual SLAM for Real-Time Applications. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1399–1406. [CrossRef]
6. Jiang, J.; Chen, X.; Dai, W.; Gao, Z.; Zhang, Y. Thermal-Inertial SLAM for the Environments With Challenging Illumination. *IEEE Robot. Autom. Lett.* **2022**, *7*, 8767–8774. [CrossRef]
7. Piao, J.C.; Kim, S.D. Real-Time Visual-Inertial SLAM Based on Adaptive Keyframe Selection for Mobile AR Applications. *IEEE Trans. Multimed.* **2019**, *21*, 2827–2836. [CrossRef]
8. Zhang, J.; Singh, S. LOAM: Lidar Odometry and Mapping in real-time. In Proceedings of the Robotics: Science and Systems Conference (RSS), Berkeley, CA, USA, 12–16 July 2014; pp. 109–111.
9. Shan, T.; Englot, B. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4758–4765.
10. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Daniela, R. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 5135–5142.
11. Wang, H.; Wang, C.; Chen, C.; Xie, L. F-LOAM: Fast LiDAR Odometry and Mapping. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021.
12. Koide, K.; Miura, J.; Menegatti, E. A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 1729881419841532. [CrossRef]
13. Xu, W.; Cai, Y.; He, D.; Lin, J.; Zhang, F. FAST-LIO2: Fast Direct LiDAR-inertial Odometry. *arXiv* **2021**, arXiv:2107.06829.
14. Magnusson, M.; Lilienthal, A.; Duckett, T. Scan Registration for Autonomous Mining Vehicles Using 3D-NDT. *J. Field Robot.* **2007**, *24*, 803–827. [CrossRef]

15. Besl, P.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
16. Cai, Y.; Xu, W.; Zhang, F. ikd-Tree: An Incremental KD Tree for Robotic Applications. *arXiv* **2021**, arXiv:2102.10808.
17. Akpınar, B. Performance of Different SLAM Algorithms for Indoor and Outdoor Mapping Applications. *Appl. Syst. Innov.* **2021**, *4*, 101. [CrossRef]
18. Filipenko, M.; Afanasyev, I. Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment. In Proceedings of the 2018 International Conference on Intelligent Systems (IS), Funchal, Portugal, 25–27 September 2018; pp. 400–407. [CrossRef]
19. Filip, I.; Pyo, J.; Lee, M.; Joe, H. Lidar SLAM Comparison in a Featureless Tunnel Environment. In Proceedings of the 2022 22nd International Conference on Control, Automation and Systems (ICCAS), Jeju, Republic of Korea, 27 November–1 December 2022; pp. 1648–1653.
20. Kim, G. SC-LeGO-LOAM. 2020. Available online: https://github.com/irapkaist/SC-LeGO-LOAM (accessed on 8 May 2022).
21. Kim, G. SC-LIO-SAM. 2021. Available online: https://github.com/gisbi-kim/SC-LIO-SAM (accessed on 5 May 2022).
22. Su, Y.; Wang, T.; Shao, S.; Yao, C.; Wang, Z. GR-LOAM: LiDAR-based sensor fusion SLAM for ground robots on complex terrain. *Robot. Auton. Syst.* **2021**, *140*, 103759. [CrossRef]
23. Júnior, G.P.C.; Rezende, A.M.C.; Miranda, V.R.F.; Fernandes, R.; Azpúrua, H.; Neto, A.A.; Pessin, G.; Freitas, G.M. EKF-LOAM: An Adaptive Fusion of LiDAR SLAM With Wheel Odometry and Inertial Data for Confined Spaces With Few Geometric Features. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 1458–1471. [CrossRef]
24. Mandow, A.; Martinez, J.L.; Morales, J.; Blanco, J.L.; Garcia-Cerezo, A.; Gonzalez, J. Experimental kinematics for wheeled skid-steer mobile robots. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 1222–1227. [CrossRef]
25. He, D.; Xu, W.; Zhang, F. Kalman Filters on Differentiable Manifolds. *arXiv* **2021**, arXiv:2102.03804.
26. Zhang, Z.; Scaramuzza, D. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.