공학석사학위논문

도심도로에서 자율주행차량의 라이다 기반 강건한 위치 및 자세 추정

Robust Pose Estimation using LiDAR for Autonomous Vehicles on Urban Roads

2023년 2월

서울대학교 대학원

기계공학부

권 우 진

도심도로에서 자율주행차량의 라이다 기반 강건한 위치 및 자세 추정

Robust Pose Estimation using LiDAR for Autonomous Vehicles on Urban Roads

지도교수 이 경 수

이 논문을 공학석사 학위논문으로 제출함

2022년 10월

서울대학교 대학원

기계공학부

권 우 진

권 우 진의 공학석사 학위논문을 인준함

2022년 12월

위 원 장 : 조 규 진 (인)

부위원장 : 이 경 수 (인)

위 원 : 이 호 원 (인)

Abstract

Robust Pose Estimation using LiDAR for Autonomous Vehicles on Urban Roads

Woojin Kwon
Department of Mechanical Engineering
The Graduate School
Seoul National University

This paper presents a method for tackling erroneous odometry estimation results from LiDAR-based simultaneous localization and mapping (SLAM) techniques on complex urban roads. Most SLAM techniques estimate sensor odometry through a comparison between measurements from the current and the previous step. As such, a static environment is generally more advantageous for SLAM systems. However, urban environments contain a significant number of dynamic objects, the point clouds of which can noticeably hinder the performance of SLAM systems. As a countermeasure, this paper proposes a 3D LiDAR SLAM system based on static LiDAR point clouds for use in dynamic outdoor urban environments. The proposed method is primarily composed of two parts, moving object detection and pose estimation through 3D LiDAR SLAM. First, moving objects in the vicinity of the ego-vehicle are detected from a referred algorithm based on a geometric model-free approach (GMFA) and a static obstacle map (STOM). GMFA works in conjunction with STOM to estimate the state of moving objects in

real-time. The bounding boxes occupied by these moving objects are utilized to remove points corresponding to dynamic objects in the raw LiDAR point clouds. The remaining static points are applied to LiDAR SLAM. The second part of the proposed method describes odometry estimation through referred LiDAR SLAM, LeGO-LOAM. The LeGO-LOAM, a feature-based LiDAR SLAM framework, converts LiDAR point clouds into range images, from which edge and planar points are extracted as features. The range images are further utilized in a preprocessing stage to improve the computation efficiency of the overall algorithm. Additionally, a 6-DOF transformation is utilized, the model equation of which can be obtained by setting a residual to be the distance between an extracted feature of the current step and the corresponding feature geometry of the previous step. The equation is optimized through the Levenberg-Marquardt method. Furthermore, GMFA and LeGO-LOAM operate in parallel to resolve computational delays associated with GMFA. Actual vehicle tests were conducted on urban roads through a test vehicle equipped with a 32-channel 3D LiDAR and a real-time kinematics GPS (RTK GPS). Validations results have shown the proposed method to significantly decrease estimation errors related to moving feature points while securing target output frequency.

Keywords : Localization, Autonomous vehicle, Moving object, Odometry, Static environment, Parallel operation, LiDAR
Student Number : 2021-28230

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Research Motivation

The rapid development of autonomous driving technology has allowed its application to complex environments over the years. Following the progress in this field, standards for autonomous vehicles from renowned programs, such as the DARPA Urban Challenge, are becoming increasingly meticulous, requiring more complex motion planning algorithms for the vehicles. In order to develop advanced motion planning algorithms for autonomous vehicles, one can say that accurate localization of the ego-vehicle is of the utmost importance. So far, the GPS has stood as the simplest solution to acquire the pose of the ego-vehicle. However, GPS has shown degradation in localization performance depending on the surrounding environment, particularly in urban environments such as tunnels and near high-rise buildings. The reduction of GPS signals in such areas has proven to severely affect the accuracy of the estimated pose, resulting in significant safety concerns.

The simultaneous localization and mapping (SLAM) technique has been proposed to assist localization capabilities and reduce dependence on GPS. So far, SLAM has shown numerous use cases on various platforms such as indoor robots and autonomous vehicles. SLAM estimates odometry by observing changes in sensor measurements, forming a map of the global coordinates. The map is then used to correct the estimated pose obtained through accumulated odometry when the sensors detect a point that has been traversed before, a process known as loop-closure. Most implementations of SLAM form a system that accumulates past states for this

loop-closure process, and hence a preference for graph-based SLAM exists for its efficient state management [1].

Furthermore, most SLAM algorithms carry the assumption that the sensors move around in environments without dynamic objects as SLAM works by matching measurements from the surrounding environment. As such, SLAM shows the best performance in static environments, where dynamic objects are treated as noise and outliers that negatively affect its performance [2, 3]. This is further intensified in feature-based SLAM where the odometry is estimated through the extraction of specific features from a LiDAR point cloud. Feature-based SLAM shows improved computation efficiency due to the reduced number of processed points but is also susceptible to faults in estimation if a large amount of these extracted features belongs to moving objects.

This research has the primary purpose of reducing the effect of dynamic objects in feature-based SLAM, ultimately preventing disruptions in estimations. The proposed method forms a static point cloud by referring to an algorithm meant for detecting and tracking dynamic objects in the LiDAR point cloud, which acts as an input to LiDAR SLAM to better estimate the pose of the ego-vehicle.

1.2. Previous Research

1.2.1. Moving Object Detection

To date, a wide variety of methods have been utilized to detect moving objects with LiDAR. Of these, the pointwise tracking method is one of the most intuitive methods which works by finding the corresponding points in successive LiDAR scans. Kaestner et al. [4] utilized an unsupervised approach to estimate the states of each point. However, pointwise tracking has its issues with regard to real-time operations. Pomerleau, Francois, et al. [5] have made claims that the method shows ideal performance for stationary sensors only. Next, geometric model-based approach models have also been suggested, which model the motion and shape of target objects through a probabilistic method. A single Bayes filter is applied to estimate each dynamic state of a vehicle along with a simultaneous update of the vehicle's geometric shape through an efficient Rao-Blackwellized particle filter (RBPF) [6]. He, Mengwen, et al. [7] have shown similar usage of RBPF to estimate the geometry of a vehicle with the addition of a scaling series particle filter (SSPF) to ensure efficient motion estimation and geometry fitting. Another method for detecting moving objects is through the usage of a map developed with a LiDAR point cloud [8, 9]. Pagad. S [9] proposed an algorithm that showed a simultaneous utilization of a model-based and a map-based method. The algorithm transforms input point clouds into voxels, volumes of space, which are then structured as octrees [10, 11]. Additionally, a neural network model has been implemented to initialize object and non-object points in the point cloud at a preprocessing stage. The occupancy probability of each grid is then updated to form an occupancy map of the

static environment. Finally, this map is utilized as a filter to detect moving objects within the LiDAR point cloud. Again, this method carries the same issue with regard to real-time operations as occupancy maps must be pre-built with repeatedly sampled data to increase its accuracy.

The aforementioned geometric model approaches provide a good platform for classifying dynamic objects but hold reduced weight when it comes to removing moving features in SLAM. As long as accurate information regarding the location, state, and the location of the corresponding points is given, the present study can fulfill its primary goal. As a method to track moving objects by forming real-time static maps without the use of geometric models, literature regarding geometric model-free approaches were reviewed [12]. Simply put, the algorithm is comprised of a complimentary operation between STOM, a map consisting of static obstacles, and the tracking of moving objects. This method guarantees real-time operations while providing performance that matches that of pointwise tracking. Furthermore, sets of rigid bodies (SPRB) [13] are used to track a variety of objects without categorizing the objects' geometric characteristics. For this reason, GMFA was selected as the main method for obtaining static LiDAR point clouds in this paper. Further details are given in the methodology section.

1.2.2. SLAM

1.2.2.1. Visual SLAM

Visual SLAM can be generally classified into 3 types: Feature-base, Direct-based. And RGB-D camera-based SLAM. Firstly, feature-based SLAM is used to

estimate odometry through feature matching with vision camera data from the current and the previous step. This method has been extensively researched, giving rise to multiple feature extraction methods for the image plane, such as the Harris corner, SIFT, SURF, FAST, and ORB [14-17]. The first feature-based visual SLAM, MonoSLAM, was developed by Davison et al. [18] with a monocular camera. MonoSLAM utilizes an extended Kalman filter (EKF) to estimate 6-DOF camera odometry and the 3D positions of feature points within an image. On a similar note, Chiuso et al. [19] utilized the EKF to estimate a sensor motion propagating map and state uncertainties This method, however, is inapplicable to long-distance motion and impractical for handling multiple features simultaneously as a simple gradient descent optimization is used to match features. Hence, the main contribution of MonoSLAM comes from its active search [20] for features and the simplification of propagation for real-time operations. Here, the linear and angular velocities are estimated to be elements of a state to predict and model motion. In this process, constant velocity between steps is assumed, and updates are performed through a point mass model. Measurement updates are then made by matching features from a new image with those of the previous step. As is with monocular cameras, depth information regarding the features is unavailable. To cope with this missing information, a map and feature initialization process is first carried out with an object of known size to estimate the depth of the features. MonoSLAM has shown real-time operation capabilities for environments with a small number of features, but beyond a certain increase in the environment and the size of the state vector, the real-time operation cannot be guaranteed.

To reduce the computational cost of MonoSLAM, Klein Georg and Murray David [21] developed an algorithm known as PTAM using a multi-thread to perform feature tracking and mapping in parallel. PTAM also introduced a keyframe structure; an input image is designated as a keyframe if there is a large disparity between the input image and the previous keyframe. Another advantage PTAM shows over MonoSLAM is bundle adjustment (BA for odometry estimation which allows PTAM to handle multiple feature points [22]. Overall, the above methods allow PTAM to guarantee real-time BA optimization. Upon the introduction of PTAM, most visual SLAM methods nowadays utilize multi-threads.

As an extension of PTAM, ORB SLAM [23] was developed, which is currently one of the most prominent feature-based visual SLAM methodologies. Similar to that of PTAM, this method also utilizes BA, multi-threads, and keyframe selection for odometry estimation through a monocular camera. However, ORB SLAM brings additional methods that contribute to this algorithm:

1) Selection of an ORB feature [17] for all system tasks. In contrast to the patch search of PTAM for finding a feature point in a map, an ORB feature is selected instead as a matching target which is used for all system tasks.
2) The usage of a covisibility graph and an essential graph for real-time tracking, mapping, and loop-closing. In the covisibility graph, keyframes act as a node while the edge between two nodes is denoted as covisibility information, which indicates the number of observations shared in the same environment. The managed connections within a covisibility graph are local, implying that tracking and mapping processes are independent of the global map size. On the other hand, an essential graph corresponds to the global

map and contains all keyframes and subgraphs from the covisibility graph. Hence, a minimal number of essential connections are used to span a tree structure allowing for an effective and efficient operation of loop closure and pose graph optimization.

3) Automatic map initialization that does not require human intervention. Two models are utilized for the automatic initialization: Homography for a planar scene and a fundamental matrix for a non-planar scene. These two models allow for efficient odometry computation regardless of scenes. A heuristic approach is adopted to obtain scores for the two models, of which the one with a high score is utilized for motion recovery and map construction through BA.

Contrary to the PTAM, ORB SLAM is capable of achieving accurate and real-time results in a wide variety of environments. Furthermore, follow-up research for the extension of ORB SLAM to stereo and RGB-D cameras is currently available [24].

Moving on, direct-based SLAM uses all image pixels available to optimize odometry and is commonly referred to as a feature-less approach. LSD-SLAM [25] currently stands as one of the representative direct-based SLAM methods for monocular cameras. First, input images and a 7-DOF pose are tracked as pose graphs and keyframes in a 3D global environment. This environment is then reconstructed semi-dense maps [26], formed through a probabilistic method. Hence, two keyframes with the closest match to that of the reconstructed 3D map are used to estimate odometry considering a scale drift of the input images [27]. Even with the excessive computational load arising from the calculation of photometric errors for

all pixels within an image, LSD-SLAM is still capable of real-time operation through a semi-dense approach.

RGB-D SLAM, on the other hand, can utilize a 3D environment as an RGB-D camera is capable of providing not only a 2D image but also depth information for each pixel, contrary to that of a monocular camera which presents issues regarding scale Depth information is typically collected through a light time-of-flight or a structured light approach [28]. The extensive progress made in developing RGB-D cameras over the years has drastically reduced its cost, along with its portability allowing for further research to be conducted with regards to RGB-D SLAM. Since 3D information for pixels are available, methods such as iterative closest point (ICP) are used in matching processes to estimate odometry. Coupling these methods with the RGB values has resulted in noticeable performance improvements for SLAM.

In this regard, Newcombe et al. [29] developed KinectFusion, a method that utilizes voxelization to construct a 3D environment with a Kinect camera [30], the first RGB-D camera developed by Microsoft. In this research, the 6-DOF sensor pose is estimated by matching consecutive voxel spaces using the ICP method. In this research, the 6-DOF sensor pose is estimated by matching consecutive voxel spaces using the ICP method. However, KinectFusion was developed for not only tracking the location and motion of robots and autonomous vehicles, as is the case for SLAM, but also for user-interfacing through 3D modeling. As such, the KinectFusion algorithm also contains methods for object segmentation and rendering. Similar algorithms that conduct 3D object segmentation were developed by Salas-Moreno et al. [31] and Tateno et al. [32]. These algorithms utilize 3D objects recognized within the environment to refine the map and sensor odometry. Alejo Concha and Javier

Civera [33] developed RGBDTAM, a fusion of RGB-D SLAM with direct-based SLAM. RGBDTAM optimizes sensor odometry by minimizing a robust loss function which is a weighted sum of photometric error and inverse depth error. Qinxuan Sun et al. [34] proposed a seamless fusion of plane and edge features, aptly named Plane-Edge-SLAM. In this algorithm, constraints for camera motion arising from the plane and edge features are first defined. Next, these constraints are used for a seamless fusion process, achieving robust and accurate motion estimation. Unlike point features such as SIFT, SURF, and FAST, plane features have demonstrated high tolerance of changes in illumination and better performance in areas lacking in texture [34, 35].

So far, a variety of visual SLAM methods have been discussed, which are cost-effective approaches to utilizing SLAM. However, limitations exist when it comes to estimating exact 3D motion with 2D images. While RGB-D cameras have allowed for 3D measurements, they are still susceptible to illumination and perspective changes, with limited detection ranges when compared to that of LiDAR sensors.

1.2.2.2. LiDAR SLAM

LiDAR sensors boast a long detection range, wider field of view, dense information, and are robust to environmental factors when compared to cameras. As such, LiDAR-based SLAM has been actively researched over the years. However, due to the large number of points scanned, distortion can occur from a single scan of a LiDAR if the sensor is moving, unless the sampling rate is overwhelmingly higher than its motion. Therefore, most LiDAR SLAM algorithms incorporate motion compensation for more accurate results, particularly for algorithms designed to work

on high-speed platforms such as autonomous vehicles. This compensation is typically done through a motion model with measurements from the IMU or a GPS/INS [36-38]. Barfoot et al. [37] simply approached this problem with a constant velocity model and a gaussian process. Furgale et al. [38] took a different approach by using B-spline functions for modeling sensor motions. Bosse et al. [39] designed *Zebedee*, an implementation of SLAM using a 2D LiDAR and the IMU, to be mounted on various platforms or even be handheld. In Zebedee, a surface element (surfel), containing a position and a normal vector, is utilized to match scans and estimate odometry. Raw point cloud data is clustered into the form of a multi-resolution voxel grid, from which a surfel is computed for clusters with sufficient points. 6-DOF odometry is initialized with IMU measurements before estimation is carried out through a surfel-to-surfel matching process, coupled with IMU deviation and an M-estimator based on a Lorentzian function [40]. Additionally, Zebedee utilizes measurement latency information to synchronize LiDAR and IMU measurements to obtain more accurate estimation results. However, their paper still suggests that improvements in accuracy are required and real-time implementation is currently difficult. Currently, one of the most well-known LiDAR SLAM methods is LOAM, developed by Zhang and Singh [41]. LOAM utilizes raw LiDAR point clouds in the form of a range image [42] that shares similarities with a projected 2D image. Within the range image, the distance between continuous points within the same row is used as a metric to extract edge and planar points. These points are then used as feature points for sequential matching. Next, the distance between the geometries (edge lines and planar patches), formed by the feature points in the previous step, and the feature points of the current step undergo a Levenberg-

Marquardt (L-M) optimization process [43] to estimate LiDAR odometry. The odometry and feature points are stored in a KD-tree for later use in a map optimization sequence which includes loop-closure. To date, LOAM has inspired many and birthed multiple variations of itself. Shan and Englot [44] considered the fact that platforms on which LiDARs are installed typically move on the ground. With this information, further developments were made on LOAM through a ground-optimized approach, known as LeGO-LOAM. LeGO-LOAM was built on a similar platform to LOAM, with the introduction of several methods that improve computational efficiency and accuracy:

1) Segmentation of ground and non-ground points. Within the developed LiDAR range image, a column-wise ground labeling process [45] is carried out to segment ground points and non-ground points. From the ground points, only planar features are extracted whereas only edge features are extracted from the non-ground points. This process significantly reduced the overall number of calculations required.
2) Object-wise segmentation. A range image-based segmentation approach [46] was adopted to segment each object within the point cloud of a range image. For each segment, viable objects contain sufficient points for feature extraction, whereas small objects and sensor noise only contain a small number of points. Using this information, only feasible features are utilized, improving the estimation accuracy for sensor motion.
3) 2-step optimization for 6-DOF sensor motion. A 2-step Levenberg-Marquardt optimization is carried out to obtain 6-DOF sensor motion in 3D coordinates. Optimization is first done using planar features, followed by

optimization of edge features. The planar features and edge features each contribute to separate components of the transformation.

This method utilizes a reduced number of features to achieve improved accuracy within 60% of the computation time required for LOAM. Other variations of LOAM include the LiO-SAM [47] which uses the IMU to correct distortions in LiDAR data while estimating and rejecting IMU bias to improve performance. Additionally, GPS measurements were adopted into the factor graph [48], a back-end of the estimation process, to optimize sensor motion. A common feature for the methods listed so far was all SLAM techniques based on a single LiDAR. Consequently, it follows that multiple LiDARs can be used to improve measurement density over a wider area to tackle the issue of point sparsity. However, the issue with such a method arises from LiDAR calibrations since the exact position of points is required for the effective implementation of SLAM. Jiao et al. [49] proposed M-LOAM, a method for the robust implementation of LiDAR SLAM, to handle the aforementioned issues with multi-LiDAR setups. Their research included online and self-calibration of multiple LiDARs into a single coordinate system by checking for calibration convergence during the odometry estimation process. However, LiDAR SLAM ultimately required feature points, which are all negatively affected by moving features from dynamic objects. In developing a solution to this issue, LeGO-LOAM was selected among the various LiDAR SLAM algorithms for its simplicity and performance.

1.3. Thesis Objective and Outline

The primary objective of this research is to develop a LiDAR SLAM system that counteracts the performance degradation caused by dynamic objects in feature-based LiDAR SLAM through a real-time removal of moving objects in LiDAR point clouds. From the perspective of a sensor, it is difficult to distinguish between scenarios where a LiDAR is moving forwards in a static environment and scenarios where an object is moving toward a stationary LiDAR. In such scenarios, LiDAR odometry estimation techniques that assume static environments are prone to failure cases. For such cases, it is advantageous to first remove moving objects from the raw LiDAR point cloud to construct a static point cloud for use in LiDAR SLAM. Figure 1 depicts the overall structure of the proposed algorithm.

The current paper is structured as follows. Chapter 2 introduces the overall structure and methodology adopted in the proposed algorithm. The perception modules used for removing dynamic objects and LeGO-LOAM, a practical LiDAR SLAM method, are explained in detail. Furthermore, the construction process for a static LiDAR point cloud through the time compensation of outputs from GMFA is included. Chapter 3 details the environment and results of validation tests conducted for the proposed method. Manual driving data obtained with an actual vehicle equipped with sensors on urban roads were used. Failure cases for conventional LiDAR SLAM methods arising from moving objects within the dataset are compared qualitatively with the results of the proposed method. 6-DOF pose errors referred to from an RTK GPS and computational time were selected as primary metrics for comparison. Finally, chapter 4 provides a conclusion and future prospects of this study.

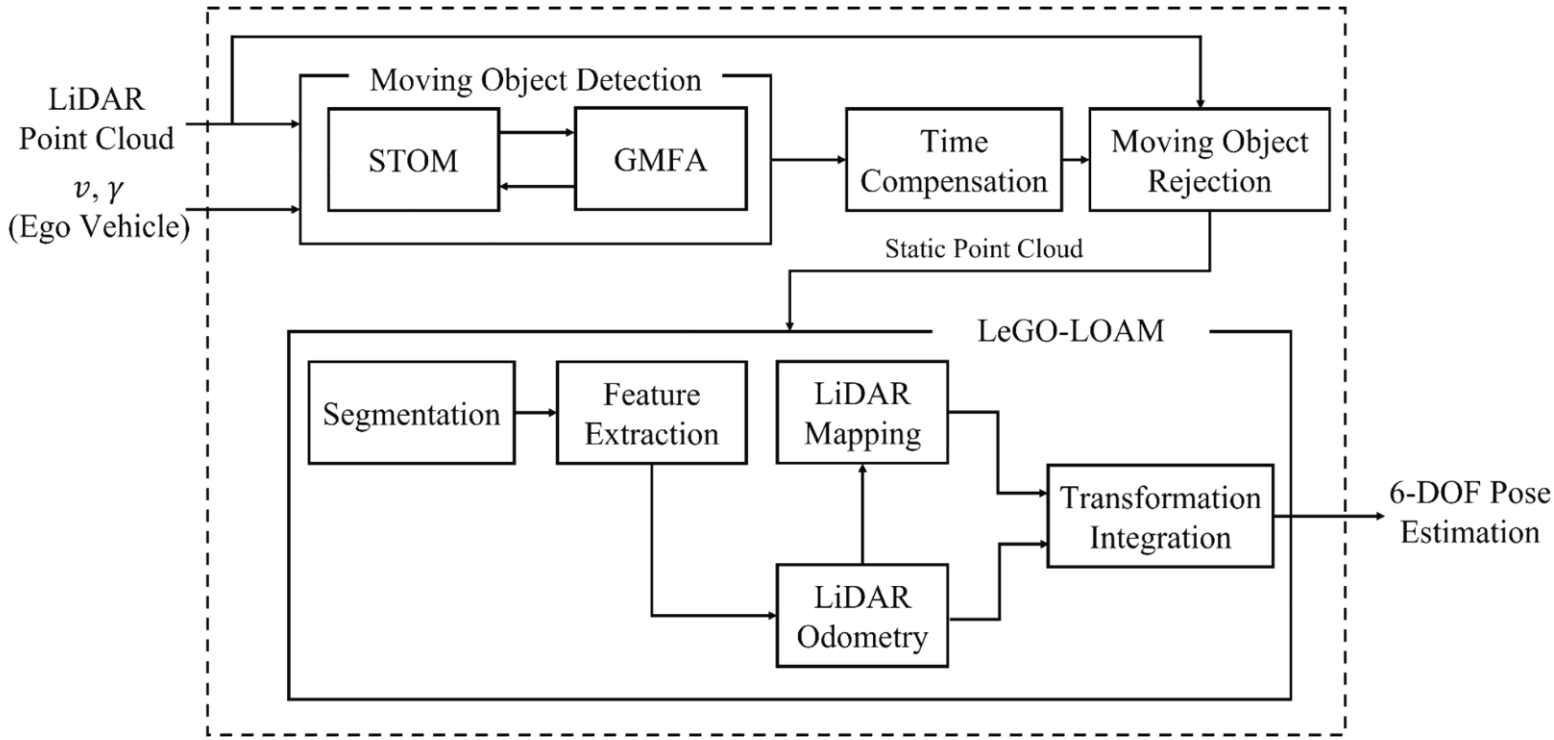


Figure 1. Overview of proposed method.

Chapter 2. Methodology

2.1. Moving Object Detection & Rejection

A referral was made to a method implementing GMFA and STOM [12] in this paper in order to achieve real-time removal of moving objects in a LiDAR point cloud. This section describes the detailed process of the referred method and sections that were utilized in the proposed algorithm.

2.1.1. Static Obstacle Map

STOM denotes a map comprised of 2 grids, each of which can have one of two states: Static and Unknown. First, each grid is initialized with identical static probabilities. Next, the STOM of the previous step ($\widehat{M}_{k-1}$) is transformed into the frame of the current step using a point mass model that incorporates the velocity and yaw rate of the ego-vehicle. Figure 2 visualizes the prediction for the current step's STOM ($\widehat{M}_{k|k-1}$) based off the transformed STOM ($\overline{M}_k$) of the previous step. Grids $\widehat{m}_{k-1}^{i}$ $[i = 1,2,3,4]$ of $\widehat{M}_{k|k-1}$ are selected if the distance with the center of $\overline{m}_k^j$ is less than $\sqrt{2}d_{grid}$. Then the prediction probability of the j-th grid can be expressed as the weighted average shown in equation (1).

$$L = \sum_{i=1}^{4} l_i^{-1}$$

$$p(\overline{m}_k^j) = \begin{cases} \sum_{i=1}^{4} \frac{l_i^{-1}}{L} p(\widehat{m}_{k-1}^i) & (l_i \neq 0) \\ p(\widehat{m}_{k-1}^i) & (l_i = 0) \end{cases} \tag{1}$$

Using the states of LiDAR points obtained from GMFA, the measurement for each grid can be classified into one of four types: Free, Unclassified, Moving, and Static. Following this, a measurement update is carried out using the 1st order Markov assumption. The update equation is represented as the product of the prediction STOM probability and the experimentally predetermined likelihood.

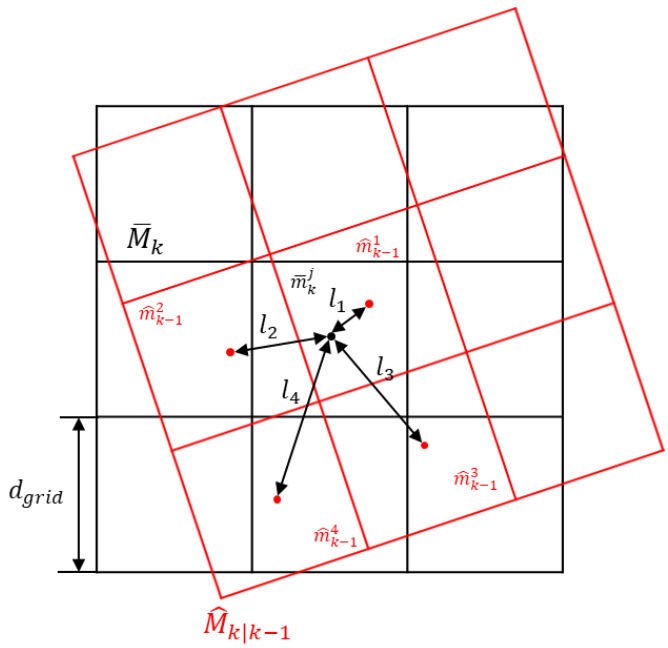


Figure 2. Current STOM prediction based off previous step. [12]

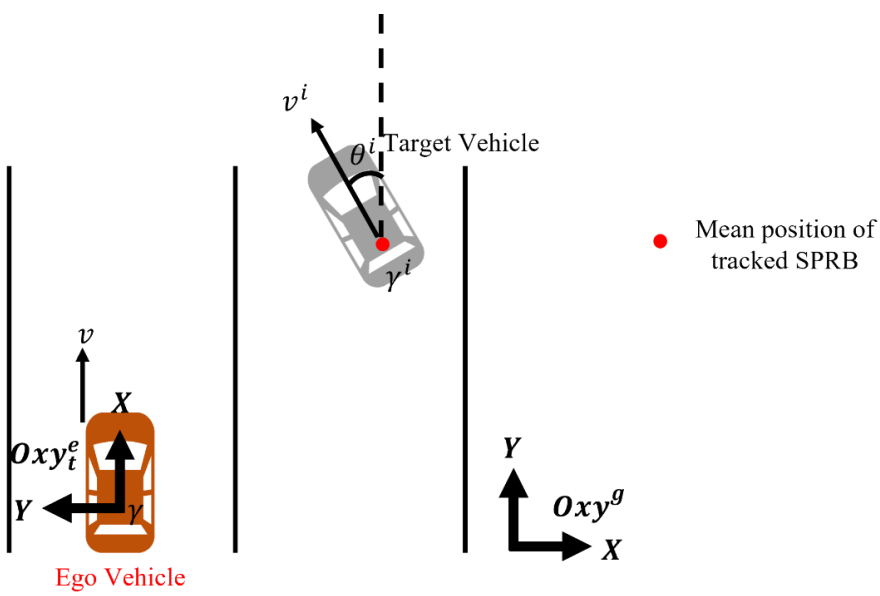


Figure 3. States of target in GMFA.

2.1.2. Geometric Model-Free Approach

GMFA detects moving objects and tracks their states using candidates of moving points acquired from STOM. Figure 3 depicts the target states of GMFA. The state estimation of GMFA is conducted in two coordinates: the global coordinate frame (Oxy^g) and the local coordinate frame of the ego-vehicle at time t (Oxy_t^e). The state of the i-th track in the k step (o_k^i) is comprised of 8 components, detailed in equation (2). The SPRB (S^i) is the accumulated LiDAR point cloud of the corresponding track for 4 steps, where the mean position of S^i is denoted by $p^{x,i}$ and $p^{y,i}$ in Oxy_t^e. θ^i represents the yaw angle of the track in Oxy_t^e frame. v^i, γ^i, a^i, and $\dot{\gamma}^i$ indicate the longitudinal velocity, yaw rate, acceleration in the direction of v^i, and the angular acceleration at Oxy^g, respectively.

$$o_k^i = [S^i,\ p^{x,i},\ p^{y,i},\ \theta^i,\ v^i,\ \gamma^i,\ a^i,\ \dot{\gamma}^i] \tag{2}$$

GMFA clusters candidates of moving points through a distance-based clustering method [50]. Each cluster contains a 4-dimensional feature vector where each dimension corresponds to the mean positions of the comprised points, the maximum, and the minimum eigenvalue of the covariance. Here, a correspondence between the cluster of the previous step and the current step is observed, followed by a compensation of the previous step's cluster (Z_{k-1}) into the current coordinate frame of the ego-vehicle. Next, the difference between the feature vectors of the current step's cluster (Z_k) and the compensated cluster of the previous state ($\bar{Z}_{k-1}$) is computed, the norm of which is compared against a specific value. If the norm of the

difference is smaller than a set value, Z_k is initialized as a new target. The tracks states are then initialized through changes in position and orientation calculated by an ICP matching process with Z_k and $\bar{Z}_{k-1}$. If the resulting track remains indefinitely or discontinuity arises within the correspondence of the tracks, the overall process may become increasingly inefficient. Hence, a confidence index is assigned to each track in GMFA. The confidence index of all tracks is initialized as 2, which increases by 1 every time the track is associated with a new measurement, up to a maximum of 50. If clusters that correspond to existing tracks are not found within 2 consecutive LiDAR scans, the value of the index is reduced by 30%. As this process is repeated, if the confidence index for any track is smaller than, or equal to, 8, the index is assigned a new value of 3. Finally, any tracks with a confidence index smaller than 2 are removed.

If a correspondence between the tracks of the previous and current step can be formed, a state update is conducted for each track that has correspondence. As for the update method, an EKF and particle filter-based method was suggested by H. Lee et al. [12], of which the EKF was adopted for this paper. The adopted EKF method assumes gaussian distribution and utilizes comparatively simple calculations which is advantageous for the LiDAR odometry estimation at a later stage. The state of the track is divided into the S^i and the dynamic state (x^i) as follows:

$$\begin{gathered} o_k^i = [S^i,\ x^i] \\ x^i = [p^{x,i},\ p^{y,i},\ \theta^i,\ v^i,\ \gamma^i,\ a^i,\ \dot{\gamma}^i] \end{gathered} \tag{3}$$

Next, the prediction update for the dynamic states and covariance is conducted through a constant acceleration model and method proposed by Simon [51], detailed in equation (4) and (5), where the dynamic states of the ego-vehicle, v_k and γ_k, make up the input, u_k. The prediction update for S^i is conducted through a 2D linear transformation method that utilizes the state of the target vehicle, x_{k-1}^i and u_k. Following this, the assigned n-th cluster in the k-th step (Z_k^n) is matched with the predicted cluster in the prior step. Finally, the measurement update for the dynamic state of each track is done using the mean point and yaw angle values given by Z_k^n.

$$
\begin{aligned}
\bar{x}_{k+1}^{i} &= \boldsymbol{f}_k\left(x_k^i, u_k, w_k\right) \\
&= [f_1\, f_2\, f_3\, f_4\, f_5\, f_6\, f_7]^{\mathrm{T}} \\
f_1 &= p_k^{x,i} + v_k^i \cos\left(\theta_k^i\right) \Delta t - v_k \Delta t + p_k^{y,i} \cdot \gamma_k \Delta t \\
f_2 &= p_k^{y,i} + v_k^n \sin(\theta_k^n) dt - p_k^{x,i} \cdot \gamma_k dt \\
f_3 &= \theta_k^i + \gamma_k^i dt - \gamma_k dt \\
f_4 &= v_k^i + a_k^i \Delta t \\
f_5 &= \gamma_k^i + \dot{\gamma}_k^i \Delta t \\
f_6 &= a_k^i + w_k^1, \qquad f_7 = \dot{\gamma}_k^n + w_k^2 \\
w_k &\sim (0, Q_k)
\end{aligned}
\tag{4}
$$

$$
F_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial x^n}\right|_{\hat{x}_k^i}
$$

$$
= \begin{bmatrix}
1 & \gamma_k \Delta t & -v_k^i \sin\left(\theta_k^i\right)\Delta t & \cos\left(\theta_k^i\right)\Delta t & 0 & 0 & 0 \\
-\gamma_k \Delta t & 1 & v_k^i \cos\left(\theta_k^i\right)\Delta t & \sin\left(\theta_k^i\right)\Delta t & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & \Delta t & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \Delta t & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & \Delta t \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\tag{5}
$$

$$
L_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial w}\right|_{\hat{x}_k^i}
$$

$$
\overline{\mathrm{P}}_{k+1}^{i} = F_k \widehat{\mathrm{P}}_k^i F_k^T + L_k Q_k L_k^T
$$

2.2. LiDAR SLAM

For this paper, LeGO-LOAM [44] was selected as the method for estimating LiDAR odometry and map construction. LeGO-LOAM is a method specialized for ground vehicles. LeGO-LOAM is comprised of the following processes: feature extraction and segmentation, LiDAR odometry, LiDAR mapping, and transformation integration. First, in feature extraction and segmentation, the LiDAR point cloud is preprocessed. Next, the LiDAR odometry process performs an optimization through the feature mating of two consecutive LiDAR scans. A local map is then formed in the LiDAR mapping module and finally, the 6-DOF pose is calculated through a correlation with map correction and LiDAR odometry from successive scans. The overview for LeGO-LOAM is visualized in Figure 4.

2.2.1. Segmentation

In this section, the methods used for improving computational efficiency by preprocessing LiDAR point clouds are introduced. An input point cloud is first projected on a range image of preset resolutions. For this paper, the projected range image has a resolution of 1800x32 pixels determined based on the specifications and data obtained from one full scan of a Velodyne Ultra Puck LiDAR [52]. Once the image projection is complete, column-wise ground labeling [45] is carried out, visualized in Figure 5. In Figure 5, the blue box indicates the column comprised of a set of points that share the same azimuth among different LiDAR channels. If the angle denoted by successive points within a column is smaller than a certain threshold, denoted by case 2, the points are considered to be part of the ground.

Conversely, if the angle is larger than the threshold, denoted by case 1, the points are considered to be part of a standing object. This method is applied to all successive points in a column for ground labeling. Here, the computational load for feature extraction is reduced as only planar features need to be found for ground-labeled points.

Next, range image-based segmentation [46] is carried out to obtain robustness in noisy environments. The segmentation process clusters and labels points from the LiDAR point cloud according to the object that they belong to. By limiting the minimum number of points in a cluster to 30, sufficient for extracting valid features, noisy point clouds can be rejected. The segmentation method is visualized in Figure 6. Here, $\overline{OA}$ indicates the straight-line path connecting the sensor (O) with point A and $\overline{AB}$ indicates the line connecting point A with a nearby point B. If the angle α formed by $\overline{OA}$ and $\overline{AB}$ is below a certain threshold, the two points are considered to be from the same object. The segmentation method is applied recursively for adjacent points over all available points.

2.2.2. Feature Extraction

The features, edge and planar, used in estimating LiDAR odometry for LeGO-LOAM are identical to that of LOAM [41]. However, as previously mentioned in sections 1.2.2.2 and 2.2.1, edge and planar features are extracted from segmented points and ground-labeled points respectively. As a metric for determining features from a raw point cloud, a measure for smoothness, c, is used. The smoothness for a center point, p_i, of a set of neighboring points, S, within a single row of a range

image is calculated as equation (6).

$$c = \frac{1}{|S| \cdot \|r_i\|} \left\| \sum_{j \in S, j \neq i} (r_j - r_i) \right\| \tag{6}$$

In equation (6), $|S|$ indicates the total number of points neighboring p_i, while r refers to the distance between the point and the sensor. If the smoothness of point p_i is beneath a certain threshold, p_i is taken to be a planar point. Conversely, if the smoothness is higher than the threshold, the point is considered to be an edge feature. LeGO-LOAM defines a unit for feature extraction to be a row of a sub-image. Here, a sub-image refers to horizontally split portions of the full range image. Additionally, there is a limit set for the number of features that can be extracted from one unit. For a given unit, let F_e and F_p represent edge features with maximum c and planar features with minimum c respectively. Let $\mathbb{F}_e$ and $\mathbb{F}_p$ represent a set of features, containing F_e and F_p, with reduced sharpness and reduced flatness respectively. In this paper, values for F_e, F_p, $\mathbb{F}_e$, and $\mathbb{F}_p$ has been set to 2, 4, 20, and 40. As a sidenote, $\mathbb{F}_e$ and $\mathbb{F}_p$ are also stored within a KD-tree to be used in scan-matching.

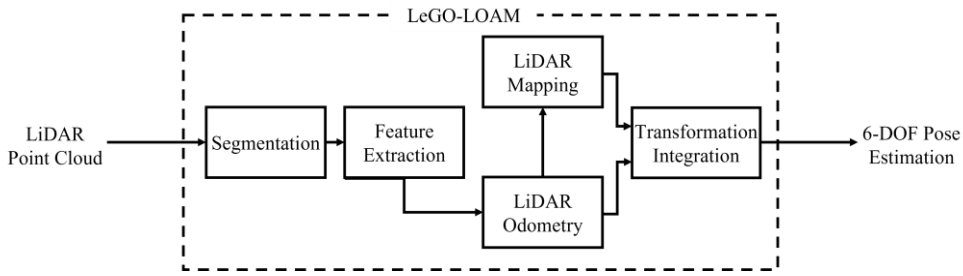
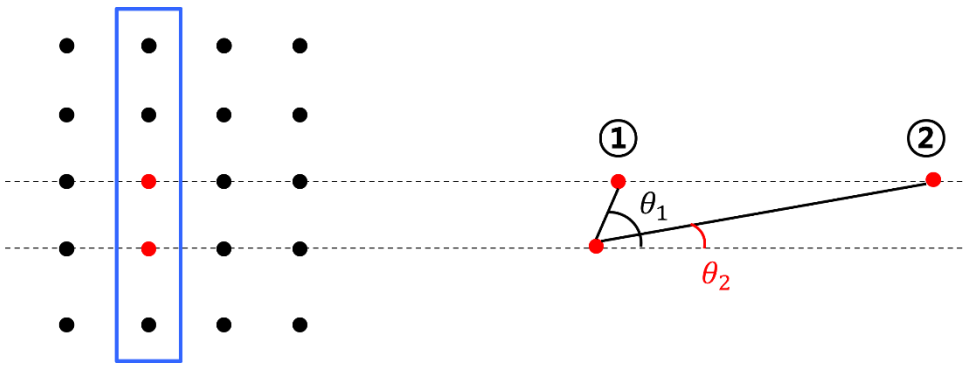


Figure 4. Total algorithm overview of LeGO-LOAM.



(a) Front view of range image (b) Side view of range image

Figure 5. Column-wise ground labeling.

(a) Front view of range image (b) Top view of range image

Figure 6. Range image-based segmentation.

2.2.3. LiDAR Odometry and Mapping

LeGO-LOAM is a feature-based LiDAR SLAM that matches features extracted from two consecutive LiDAR scans to estimate odometry. For scan matching, features that correspond to the two consecutive scans need to be found. Features in the current step that are included in F_e^t and F_p^t are matched with the closest features included in $\mathbb{F}_e^{t-1}$ and $\mathbb{F}_p^{t-1}$. Here, points contained in all 4 sets are compensated through linear interpolation to the timestep corresponding to the instant where the input scan ends. Then, the 3 features in $\mathbb{F}_p^{t-1}$ closest to a planar point in F_p^t are found, following which the distance from a patch, formed by these 3 points, to the current planar point is calculated. The calculation method is adopted from [41]. On the other hand, for edge features, the 2 closest features are found. Again, the distance between the prior edge line and the current edge point in F_e^t is calculated. The distances obtained through the calculations can be expressed as a non-linear function based on the 6-DOF transformation up to the current time t. Since all feature points are compensated to a certain timestep, the transformation can be estimated through an optimization process that minimizes the calculated distances toward zero. LeGO-LOAM utilizes the L-M method for this process, which has been altered into a two-step process to improve estimation performance; residuals acquired from planar features are optimized for the estimation of t_z, θ_{roll}, and θ_{pitch} in the local coordinates, whereas the edge features are utilized to estimate t_x, t_y, and θ_{yaw}. The local coordinates for LeGO-LOAM are depicted in Figure 7.

After LiDAR odometry, LiDAR mapping is conducted to refine transformation at a lower operating rate in multi-thread. LeGO-LOAM saves only extracted features,

not entire point clouds. In LeGO-LOAM, only extracted features, not entire point clouds, are saved and managed with a KD-tree. A global LiDAR map is obtained by choosing appropriate feature sets within a specified range from the current position of the LiDAR. Finally, the transformation refinement is done by carrying out another L-M optimization over the computed distances between corresponding points of the obtained LiDAR map and the current feature set $\{\mathbb{F}_p^t, \mathbb{F}_e^t\}$. For scenarios where locations that have been traversed once are revisited, loop-closure is conducted and new constraints are defined through the ICP method.

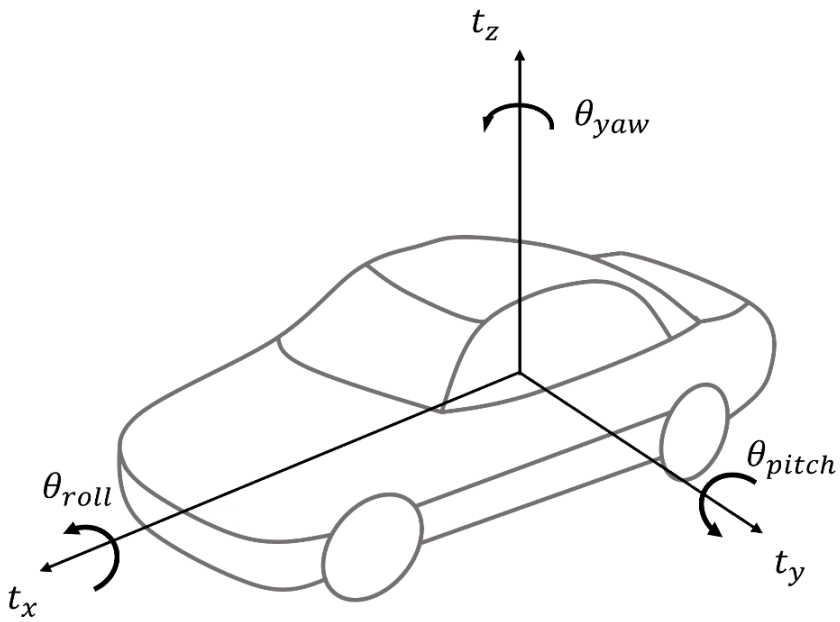


Figure 7. Local coordinates of LeGO-LOAM.

2.2.4. LiDAR SLAM with Static Point Cloud

In the proposed algorithm, GMFA and LeGO-LOAM work in conjunction to estimate LiDAR odometry from a LiDAR point cloud of a static environment. As shown by the overview of the entire algorithm in Figure 1, with the interaction between STOM and GMFA, cluster points that correspond to moving objects and the state values, inclusive of the average position, can be obtained from the moving object detection module. Then, the bounding box that envelops the moving object in the ego-vehicle's local coordinates can be obtained. Following this, points in the raw LiDAR point cloud that correspond to the bounding box can be removed, resulting in the formation of a static point cloud. However, the comparatively slow cycle rate of GMFA may be a bottleneck slowing down the overall system. GMFA is capable of providing information regarding a moving object up to a rate of 15Hz. On the other hand, the cycle rate for the input LiDAR point cloud into the system was set to 20Hz. The frequency for pose estimation was set to 20Hz as well to guarantee safe maneuvers from the motion planning module of an autonomous vehicle. I.e. a delay exists for the state information of moving objects detected by GMFA. To overcome this issue, LeGO-LOAM and the moving object detection module were designed to run in parallel. Furthermore, compensation for the difference in time between the moving objects and the raw LiDAR point cloud was added to ensure that detected targets were accurately removed. The time compensation was implemented through a point mass model utilizing the position, velocity, and angular velocity, all included in the state of a moving object with the assumption of constant velocity. Points included in the compensated bounding box were removed to obtain the static point

cloud, which was given as input to LeGO-LOAM, finally achieving the estimated 6-DOF pose of an autonomous vehicle.

Chapter 3. Experiments

3.1. Experimental Setup

The test vehicle, equipped sensors, and specifications of the sensors used for validation are depicted and summarized in Figure 8 and Table 1. The KIA Motors Carnival was selected as the test vehicle, equipped with a 32 Channel Velodyne Ultra Puck LiDAR on the top of the vehicle. The aforementioned LiDAR is of a rotary type which operates with 360 degrees of range with a horizontal resolution of 0.1 to 0.4 degrees. The sampling time for the LiDAR can be set to any value between 5 to 20Hz, where 20Hz was chosen for the experiment. Additionally, a Septentrio AsteRx SBi was used as an RTK GPS to obtain the ground truth of the ego-vehicle as a reference. To ensure accurate comparisons, a 6-DOF transformation was performed to standardize measurements for the LiDAR and RTK GPS measurements to the center of the rear axle. The sensors and algorithms shown in Figure 1 were all implanted in ROS and C++ under Ubuntu Linux. Experiments for this study were conducted in urban environments where multiple moving vehicles were present to check if moving objects were effectively removed on LiDAR SLAM. Three scenarios were planned. Scenario 1 describes a situation where the ego-vehicle drives in a straight line for 250m at 30km/h with multiple vehicles traveling in the same direction. Scenario 2 describes a situation where the ego-vehicle remains stationary for 50 seconds while multiple vehicles are traveling in both directions of a lane. Finally, scenario 3 describes a situation in which the ego-vehicle drives in a straight line for 300m at 50kph in a static environment without moving objects.

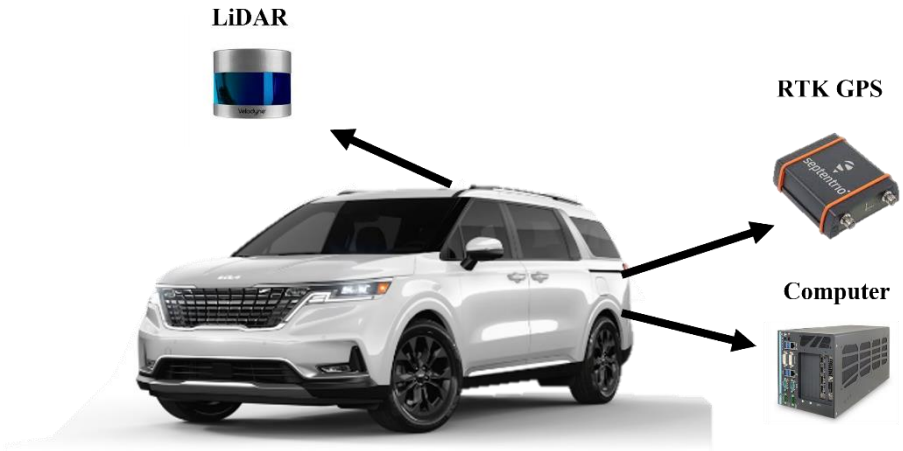


Figure 8. Sensor placement on experimental vehicle.

Table 1. Specifications of experimental components.

Component/Model	Specification
Test Car/ KIA Carnival	- Type: Recreational Vehicle (RV) - Length: 5,155mm - Width: 1,995mm - Height: 1,775mm
LiDAR/ Velodyne Ultra-Puck 32Ch	- Channels: 32 - Update rate: 5-20Hz - Horizontal resolution: 0.1-0.4deg - Vertical resolution: 0.33deg
Computer/ Neousys Technology Nuvo-8108GC	- CPU: Intel Xeon E-2278GE(8C, 16T, 16M, 3.3GHz) - VGA: EVGA NVIDIA RTK 3070 8GB
RTK GPS/ Septentrio AsteRx SBi	- Tracking signals: GPS (L1, L2), GLONASS (L1, L2), Galileo (E1, E5b), BeiDou (B1, B2), QZSS (L1, L2) - Horizontal position accuracy: 0.6cm + 0.5ppm (RTK-INS) - Vertical position accuracy: 1cm + 1ppm (RTK-INS) - Heading accuracy: 0.15deg (RTK-Dual antenna)

3.2. Error Metrics

Once the pose is acquired from the LiDAR odometry estimation process, it is aligned to the coordinates of the first synchronized GPS. The first GPS pose obtained is set as the origin of the global coordinates. The 6-DOF targets to be compared are shown in equation (7), where R represents the 3-axis rotation matrix and $\angle(R)$ represents a vector obtained through the conversion of the rotation matrix to the corresponding axis values. Then, error metrics are calculated for each component of the vehicle state through comparisons made with the RTK GPS measurements.

$$\begin{aligned} \mathbb{x} &= \left[x\ y\ z\ \theta_x\ \theta_y\ \theta_z\right] \\ &= \left[p^T\ \angle(R)^T\right] \\ p = [x\ y\ z]^T \ \text{and}\ \angle(R) &= \left[\theta_x\ \theta_y\ \theta_z\right]^T \end{aligned} \tag{7}$$

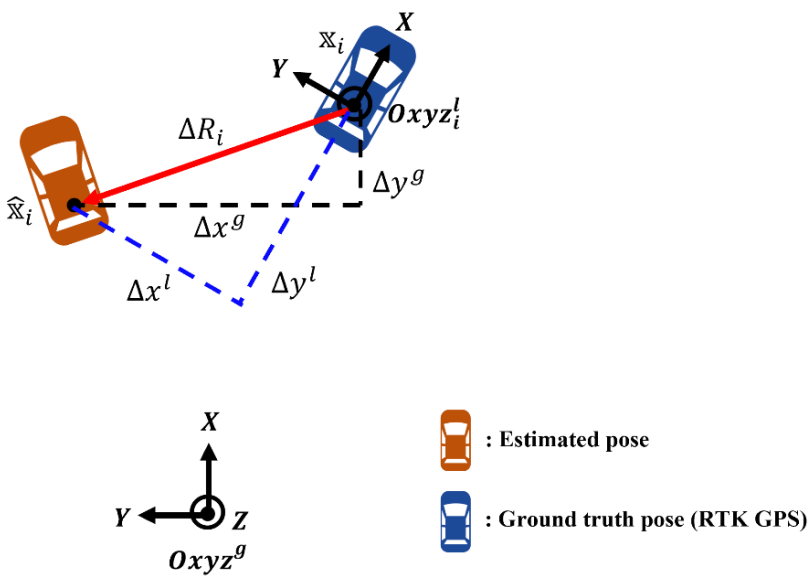


Figure 9. Pose error in local coordinates and global coordinates.

The estimation error can be represented as the difference in pose to the synchronized pose of the RTK GPS. This difference, between estimated rotation ($\hat{R}_i$) and ground truth (R_i), can be represented as follows:

$$\Delta R_i = \hat{R}_i (R_i)^T \tag{8}$$

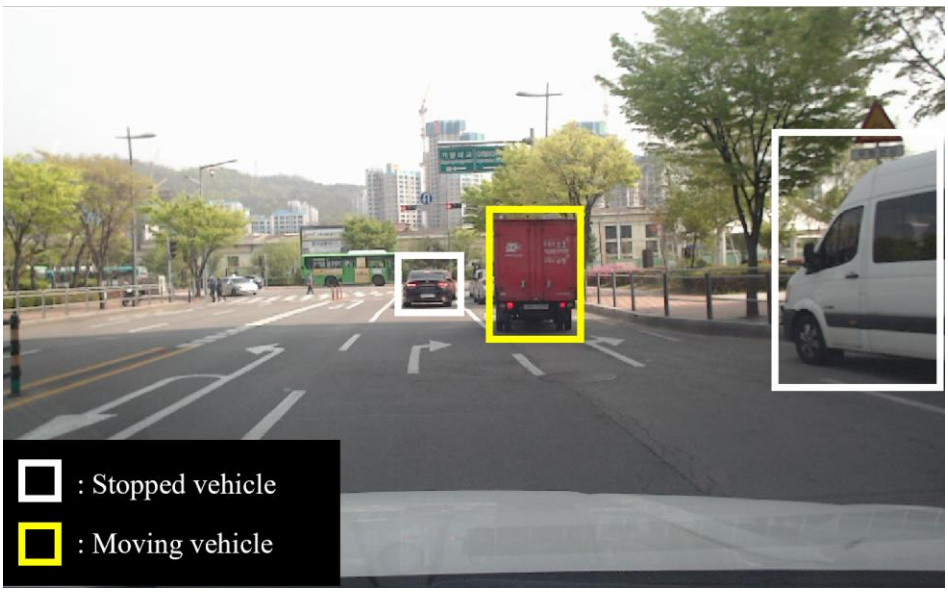
On the other hand, translation error is computed for 2 coordinates, one in the global coordinates ($Oxyz^g$) and another in the local coordinates of i-th ground truth pose ($Oxyz_i^l$), visualized in Figure 9. The corresponding global (Δp_i^g) and local translation error (Δp_i^l) are defined in equation (9) and (10) below.

$$\begin{aligned} \Delta p_i^g &= \left[\Delta x_i^g \ \Delta y_i^g \ \Delta z_i^g\right]^T \\ \Delta x_i^g &= \hat{x}_i^g - x_i^g \\ \Delta y_i^g &= \hat{y}_i^g - y_i^g \\ \Delta z_i^g &= \hat{z}_i^g - z_i^g \end{aligned} \tag{9}$$

$$\begin{aligned} \Delta p_i^l &= \left[\Delta x_i^l \ \Delta y_i^l \ \Delta z_i^l\right]^T \\ &= (R_i)^T \Delta p_i^g \end{aligned} \tag{10}$$

Furthermore, there exists another metric for quantifying a trajectory, most commonly referred to as the absolute trajectory error (ATE), widely used for its calculation simplicity [53]. ATE utilizes root mean square error values in its calculations, shown in equation (11).

$$\begin{aligned} ATE_{pos} &= \left(\frac{1}{N}\sum_{i=0}^{N-1} \| \Delta p_i \|^2\right)^{1/2} \\ ATE_{rot} &= \left(\frac{1}{N}\sum_{i=0}^{N-1} \| \angle(\Delta R_i) \|^2\right)^{1/2} \end{aligned} \tag{11}$$



(a) Front view image

(b) Raw point cloud

(c) After moving object rejection

Figure 10. Result of moving object detection and rejection.

Figure 11. Front view of ego-vehicle in scenario 1.

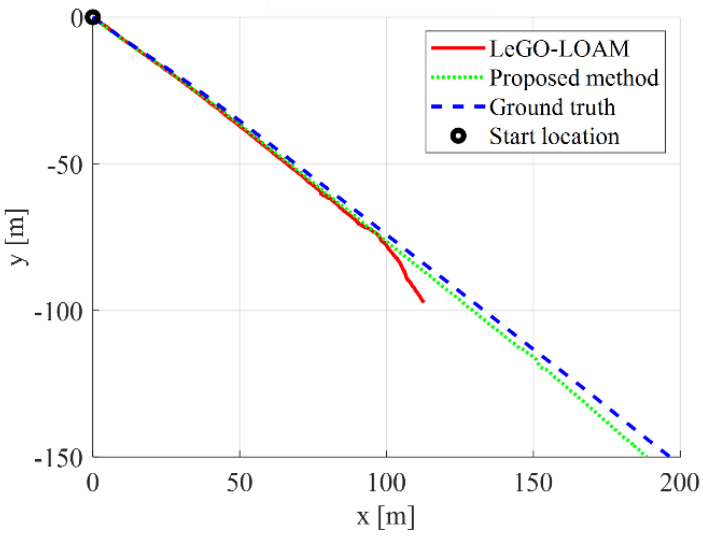


Figure 12. Global trajectory of scenario 1.

3.3. LiDAR SLAM using Static Point Cloud

Figure 10 depicts the results for the detection and rejection of moving objects. The object detection carried out by the simultaneous implementation of GMFA and STOM not only provides information on whether an obstacle is present or not, but also estimations on the speed of these objects allowing the algorithm to differentiate between moving and stationary vehicles. Hence, valid features can be extracted from static objects while rejecting outlier features from moving objects. Figure 10 shows a moving vehicle, a red truck, being differentiated and removed through the GMFA method.

As shown by Figure 11, the only static objects where valid features can be extracted are generally trees planted on the left side. Hence, in a single LiDAR scan, feature points extracted from moving vehicles are usually dominant over that of stationary objects, which has been known to cause performance degradation in LiDAR odometry estimations A comparison was made between the proposed method and LeGO-LOAM, the results of which have been quantitatively analyzed and summarized in Figure 12, Table 2, and Table 3. Results for a raw LiDAR point cloud input into LeGO-LOAM showed that the estimated odometry in the driving direction of the vehicle was less than the ground truth values. The reduction in odometry can be explained by feature points moving at a similar speed to the ego vehicle, resulting in the false estimation of sensor motion, where the sensor is perceived to be stationary. Consequently, the final translation errors of the x and y-axis in local coordinates are -111.23m and -9.96m, respectively. In addition, the final yaw (z-axis) error, one of the most influential factors for autonomous driving, is -17.62deg. On

the other hand, the proposed algorithm, which uses a static point cloud, was able to reduce estimation error in the longitudinal direction by up to 5m.

Table 2. Final and mean translation errors in scenario 1.

Metric	Method	Δx^g	Δy^g	Δz^g	Δx^l	Δy^l	Δz^l
Final [m]	LeGO-LOAM	-94.02	60.25	1.52	-111.23	-9.96	1.21
	Proposed method	-8.00	-1.79	-0.73	-5.22	-6.33	-0.66
Mean [m]	LeGO-LOAM	-27.17	17.62	0.11	-32.23	-3.19	0.16
	Proposed method	-2.84	-0.84	-0.19	-1.70	-2.43	-0.16

Table 3. Final and mean rotation errors in scenario 1.

Metric	Method	$\Delta\theta_x$	$\Delta\theta_y$	$\Delta\theta_z$
Final [°]	LeGO-LOAM	-1.99	-2.58	-17.62
	Proposed method	-0.46	0.32	-0.92
Mean [°]	LeGO-LOAM	-0.79	-1.15	-5.88
	Proposed method	-0.69	0.32	-2.75

Figure 13. Front view of ego-vehicle in scenario 2.

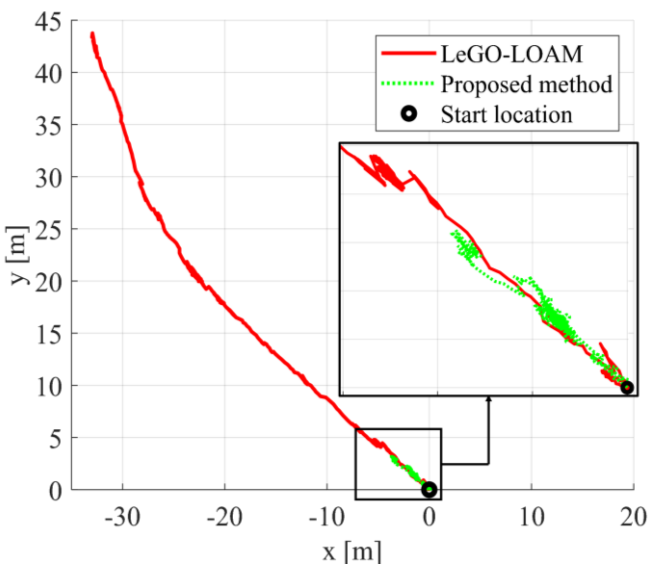


Figure 14. Global trajectory in scenario 2.

Validation results for scenario 2 provided evidence that feature points extracted from moving vehicles result in a degradation of odometry estimation performance for LeGO-LOAM, leading to an erroneous final trajectory. In this scenario, dominance was observed for feature points extracted from vehicles moving in the direction of the ego-vehicle's heading. Thus, LeGO-LOAM estimated the pose of the ego-vehicle as though the vehicle was reversing. On the contrary, the proposed method was able to reject most feature points from moving vehicles. Through this exclusion, estimation faults arising from moving feature points could be prevented in a situation where the ego-vehicle is stationary. Quantitative comparisons for scenario 2 are shown in Figure 14, Table 4, and Table 5.

Table 4. Final and mean translation errors in scenario 2.

Metric	Method	Δx^g	Δy^g	Δz^g	Δx^l	Δy^l	Δz^l
Final [m]	LeGO-LOAM	-34.23	44.46	1.34	-54.38	13.78	1.67
	Proposed method	-0.32	0.24	-0.04	-0.45	0.06	-0.02
Mean [m]	LeGO-LOAM	-13.03	12.69	0.25	-18.08	1.95	0.31
	Proposed method	-0.10	0.07	-0.09	-0.27	-0.13	0.08

Table 5. Final and mean rotation errors in scenario 2.

Metric	Method	$\Delta\theta_x$	$\Delta\theta_y$	$\Delta\theta_z$
Final [°]	LeGO-LOAM	0.40	1.94	-33.95
	Proposed method	-1.42	0.10	-0.49
Mean [°]	LeGO-LOAM	0.12	0.39	-6.26
	Proposed method	1.14	-0.36	0.16

Figure 15. Front view of ego-vehicle in scenario 3.

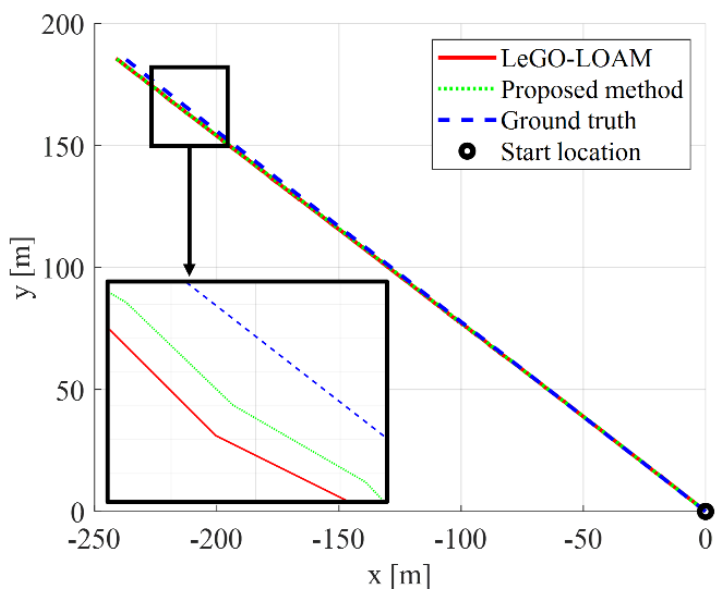


Figure 16. Global trajectory in scenario 3.

Table 6. Final and mean translation errors in scenario 3.

Metric	Method	Δx^g	Δy^g	Δz^g	Δx^l	Δy^l	Δz^l
Final [m]	LeGO-LOAM	-1.14	-2.15	-2.84	-0.35	2.33	-2.91
	Proposed method	-1.24	-1.80	-1.92	-0.07	2.13	-1.98
Mean [m]	LeGO-LOAM	-0.29	-0.88	-0.91	-0.32	0.85	-0.93
	Proposed method	-0.38	-0.71	-0.83	-0.14	0.77	-0.85

Table 7. Final and mean rotation errors in scenario 3.

Metric	Method	$\Delta\theta_x$	$\Delta\theta_y$	$\Delta\theta_z$
Final [°]	LeGO-LOAM	-1.21	0.54	1.20
	Proposed method	0.23	-0.39	1.37
Mean [°]	LeGO-LOAM	-0.93	1.18	0.77
	Proposed method	0.21	-0.20	0.89

Finally, for scenario 3, the ego-vehicle was operated in a completely static environment. Accordingly, faults in pose estimations were not observed for this scenario, as shown in Figure 16, Table 6, and Table 7. Computation results for the error metrics have shown no noticeable difference between the two methods, demonstrating that a parallel structure with GMFA does not have a negative effect on the LiDAR odometry estimation performance. Furthermore, the final Δx^l of the proposed method in scenario 3 was computed to be -0.07m, much smaller than that of scenarios 1 and 2 (-5.22m and -5.63m respectively). This implies that the features of the dynamic objects within the environment were not completely rejected, and similar results can be observed for absolute trajectory errors (Table 7) as well. Despite the fact that values for ATE_{pos} and ATE_{rot} of the proposed method were considerably reduced in scenarios 1 and 2, these values were still larger than those

of LeGO-LOAM in scenario 3, a completely static environment. In other words, a residual degradation can be observed. This can be attributed to a limitation in system structure, where GMFA has been simply implemented as an additional module, and insufficient feature points arising from the occupancy of the moving objects.

Table 8. Absolute trajectory errors in total scenarios.

Metric	Method	Scenario 1	Scenario 2	Scenario 3
ATE_{pos} [m]	LeGO-LOAM	5.71	4.27	1.13
	Proposed method	1.59 (-72.1%)	1.55 (-63.7%)	1.10 (-2.6%)
ATE_{rot} [°]	LeGO-LOAM	20.22	19.61	8.13
	Proposed method	9.34 (-53.8%)	8.95 (-54.3%)	7.96 (-2.09%)

In addition, a serial processing system, in which LiDAR SLAM waits for the output of GMFA to receive a static LiDAR point cloud, has a limitation in the operating frequency of the entire system at the frequency of GMFA. In this paper, the output frequency of the LiDAR was set to 20Hz, whereas the operating frequency of GMFA was limited to 15Hz. This issue was handled by running LeGO-LOAM and GMFA in parallel threads. This solution does have its downsides with regard to computation speed, where sharing computational resources may potentially slow down the processing speed for pose estimation. Hence, to observe whether the real-time operation was possible, the computation time taken for a single LiDAR SCAN in each module was measured and summarized in Table 9. Here, no significant difference in computation time can be observed between LeGO-LOAM and the proposed method. Therefore, real-time operation for pose estimation can be guaranteed, despite the implementation of an additional perception module.

Table 9. Mean of computation time in each module for one LiDAR scan.

Scenario	Method	Segmentation	Extraction	Odometry	Mapping
1 [ms]	LeGO-LOAM	3.45	0.26	0.26	53.38
	Proposed method	3.77	0.23	0.23	50.12
2 [ms]	LeGO-LOAM	3.27	0.24	0.24	46.86
	Proposed method	3.47	0.22	0.21	36.66
3 [ms]	LeGO-LOAM	3.61	0.26	0.26	55.58
	Proposed method	3.67	0.24	0.24	56.88

Chapter 4. Conclusion

In this paper, a parallel architecture consisting of moving object detection and LiDAR SLAM techniques was proposed with the objective of improving estimation performance against moving objects within a LiDAR point cloud. Vehicle experiments carried out in congested urban scenarios have proven that dynamic objects in the vicinity of the ego-vehicle may result in erroneous estimation results from feature-based LiDAR SLAM techniques which generally require a static environment. Through the proposed algorithm, estimation errors in environments with multiple dynamic objects were significantly reduced. Furthermore, process delay issues for LiDAR SLAM arising from the additional perception module could be avoided by operating the modules in parallel threads.

However, the proposed method still shows room for improvement. Firstly, GMFA utilized for detecting moving objects is not complementary to LiDAR SLAM. GMFA requires the longitudinal speed and yaw rate of the ego-vehicle, which are information obtained from vehicle chassis sensors, not LiDAR SLAM. Consequently, the operation of GMFA as an independent module does not fully utilize all available outputs. Secondly, GMFA does not fully utilize all data available from LiDAR points, but rather only the two-dimensional positions by projecting the 3D points to the ground. Thus, static objects located on top of detected objects are likely to be rejected. To overcome these limitations, an integrated LiDAR SLAM system is required, which will be a closed-loop system that utilizes 3D point cloud information along with the estimated odometry to form a static environment.

Bibliography

[1] Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, *2*(4), 31-43.

[2] Li, S., & Lee, D. (2017). RGB-D SLAM in dynamic environments using static point weighting. *IEEE Robotics and Automation Letters*, 2(4), 2263-2270.

[3] Chatterjee, J. (2020). *Moving object removal for robust visual SLAM in dynamic environment* (Doctoral dissertation, GIPSA Lab, 11 Rue des Mathématiques, 38400 Saint-Martin-d'Hères; Grenoble INP Ensimag; University of Grenoble Alpes (UGA)).

[4] Kaestner, R., Maye, J., Pilat, Y., & Siegwart, R. (2012, May). Generative object detection and tracking in 3d range data. In *2012 IEEE International Conference on Robotics and Automation* (pp. 3075-3081). IEEE.

[5] Pomerleau, F., Krüsi, P., Colas, F., Furgale, P., & Siegwart, R. (2014, May). Long-term 3D map maintenance in dynamic environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3712-3719). IEEE.

[6] Petrovskaya, A., & Thrun, S. (2009). Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, *26*(2), 123-139.

[7] He, M., Takeuchi, E., Ninomiya, Y., & Kato, S. (2016, October). Precise and efficient model-based vehicle tracking method using Rao-Blackwellized and scaling series particle filters. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 117-124). IEEE.

[8] Schauer, J., & Nüchter, A. (2018). The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid. *IEEE robotics and automation letters*, *3*(3), 1679-1686.

[9] Pagad, S., Agarwal, D., Narayanan, S., Rangan, K., Kim, H., & Yalla, G. (2020, May). Robust method for removing dynamic objects from point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 10765-10771). IEEE.

[10] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous robots*, *34*(3), 189-206.

[11] Meagher, D. (1982). Geometric modeling using octree encoding. *Computer graphics and image processing*, *19*(2), 129-147.

[12] Lee, H., Yoon, J., Jeong, Y., & Yi, K. (2020). Moving object detection and tracking based on interaction of static obstacle map and geometric model-free approach for urban autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, *22*(6), 3275-3284.

[13] Granstrom, K., Baum, M., & Reuter, S. (2016). Extended object tracking: Introduction, overview and applications. arXiv preprint arXiv:1604.00970.

[14] Derpanis, K. G. (2004). The harris corner detector. *York University*, *2*, 1-2.

[15] Ng, P. C., & Henikoff, S. (2003). SIFT: Predicting amino acid changes that affect protein function. *Nucleic acids research*, *31*(13), 3812-3814.

[16] Bay, H., Tuytelaars, T., & Gool, L. V. (2006, May). Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404-417). Springer, Berlin, Heidelberg.

[17] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision* (pp. 2564-2571). IEEE.

[18] Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE transactions on pattern analysis and machine intelligence*, *29*(6), 1052-1067.

[19] Chiuso, A., & Soatto, S. (2000). MFm: 3-d motion from 2-d motion causally integrated over time-part i: Theory. In *In European Conference on Computer Vision*.

[20] Davison, A. J. (2005, October). Active search for real-time vision. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (Vol. 1, pp. 66-73). IEEE.

[21] Klein, G., & Murray, D. (2007, November). Parallel tracking and mapping for small AR workspaces. In 2007 6th IEEE and ACM international symposium on mixed and augmented reality (pp. 225-234). IEEE.

[22] Strasdat, H., Montiel, J. M., & Davison, A. J. (2012). Visual SLAM: why filter?. Image and Vision Computing, 30(2), 65-77.

[23] Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics, 31(5), 1147-1163.

[24] Mur-Artal, R., & Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE transactions on robotics, 33(5), 1255-1262.

[25] Engel, J., Schöps, T., & Cremers, D. (2014, September). LSD-SLAM: Large-scale direct monocular SLAM. In *European conference on computer vision* (pp. 834-849). Springer, Cham.

[26] Engel, J., Sturm, J., & Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on*

computer vision (pp. 1449-1456).

[27] Eade, E. (2014). Lie groups for computer vision. *Cambridge Univ., Cam-bridge, UK, Tech. Rep*, *2*.

[28] Jing, C., Potgieter, J., Noble, F., & Wang, R. (2017, November). A comparison and analysis of RGB-D cameras' depth performance for robotics application. In *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)* (pp. 1-6). IEEE.

[29] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., ... & Fitzgibbon, A. (2011, October). KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (pp. 559-568).

[30] Zhang, Z. (2012). Microsoft kinect sensor and its effect. IEEE multimedia, 19(2), 4-10.

[31] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., & Davison, A. J. (2013). Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1352-1359).

[32] Tateno, K., Tombari, F., & Navab, N. (2016, May). When 2.5 D is not enough: Simultaneous reconstruction, segmentation, and recognition on dense SLAM. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 2295-2302). IEEE.

[33] Concha, A., & Civera, J. (2017, September). RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 6756-6763). IEEE.

[34] Sun, Q., Yuan, J., Zhang, X., & Duan, F. (2020). Plane-Edge-SLAM: Seamless fusion of planes and edges for SLAM in indoor environments. *IEEE Transactions on Automation Science and Engineering*, *18*(4), 2061-2075.

[35] Li, B., Zou, D., Sartori, D., Pei, L., & Yu, W. (2020, May). Textslam: Visual slam with planar text features. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2102-2108). IEEE.

[36] Scherer, S., Rehder, J., Achar, S., Cover, H., Chambers, A., Nuske, S., & Singh, S. (2012). River mapping from a flying robot: state estimation, river detection, and obstacle mapping. *Autonomous Robots*, 33(1), 189-214.

[37] Anderson, S., & Barfoot, T. D. (2013, May). Towards relative continuous-time SLAM. In *2013 IEEE International Conference on Robotics and Automation* (pp. 1033-1040). IEEE.

[38] Furgale, P., Barfoot, T. D., & Sibley, G. (2012, May). Continuous-time batch estimation using temporal basis functions. In *2012 IEEE International Conference on Robotics and Automation* (pp. 2088-2095). IEEE.

[39] Bosse, M., Zlot, R., & Flick, P. (2012). Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, *28*(5), 1104-1119.

[40] Geer, S. A., van de Geer, S., & Williams, D. (2000). *Empirical Processes in M-estimation* (Vol. 6). Cambridge university press.

[41] Zhang, J., & Singh, S. (2014, July). LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems* (Vol. 2, No. 9, pp. 1-9).

[42] Biasutti, P., Aujol, J. F., Brédif, M., & Bugeau, A. (2018). Range-Image: Incorporating sensor topology for LiDAR point cloud processing. *Photogrammetric*

Engineering & Remote Sensing, *84*(6), 367-375.

[43] Moré, J. J. (1978). The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis* (pp. 105-116). Springer, Berlin, Heidelberg.

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

[45] Himmelsbach, M., Hundelshausen, F. V., & Wuensche, H. J. (2010, June). Fast segmentation of 3D point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium* (pp. 560-565). IEEE.

[46] Bogoslavskyi, I., & Stachniss, C. (2016, October). Fast range image-based segmentation of sparse 3D laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 163-169). IEEE.

[47] Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., & Rus, D. (2020, October). Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 5135-5142). IEEE.

[48] Dellaert, F., & Kaess, M. (2017). Factor graphs for robot perception. *Foundations and Trends® in Robotics*, *6*(1-2), 1-139.

[49] Jiao, J., Ye, H., Zhu, Y., & Liu, M. (2021). Robust odometry and mapping for multi-lidar systems with online extrinsic calibration. *IEEE Transactions on Robotics*, *38*(1), 351-371.

[50] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based

algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, No. 34, pp. 226-231).

[51] Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons.

[52] Velodyne Lidar. (2019). *63-9278-Rev-F Ultra Puck Datasheet*. https://velodynelidar.com/wp-content/uploads/2019/12/63-9378_Rev-F_Ultra-Puck_Datasheet_Web.pdf

[53] Zhang, Z., & Scaramuzza, D. (2018, October). A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 7244-7251). IEEE.

초 록

도심도로에서 자율주행차량의 라이다 기반 강건한 위치 및 자세 추정

본 연구는 복잡한 도심 환경에서 라이다 기반 동시적 위치 추정 및 맵핑(Simultaneous localization and mapping, SLAM)의 이동량 추정 오류를 방지하는 방법론을 제안한다. 대부분의 SLAM은 이전 스텝과 현재 스텝의 센서 측정치를 비교하여 자차량의 이동량을 추정한다. 따라서 SLAM에는 정적인 환경이 필수적이다. 그러나 센서는 도심환경에서 동적인 물체에 쉽게 노출되고 동적 물체로부터 출력되는 라이다 점군들은 이동량 추정 성능을 저하시킬 수 있다. 이에, 본 연구는 동적인 도심환경에서 정적인 점군을 기반한 3차원 라이다 SLAM 시스템을 제안하였다. 제안된 방법론은 이동 물체 인지와 3차원 라이다 SLAM을 통한 위치 및 자세 추정으로 구성된다. 우선, 기하학적 모델 프리 접근법과 정지 장애물 맵의 상호 보완적인 관계에 기반한 참고된 알고리즘을 이용해 자차량 주변의 이동 물체의 동적 상태를 실시간으로 추정한다. 그 후, 추정된 이동 물체가 차지하는 경계선을 이용하여 동적 물체에 해당하는 점들을 기존 라이다 점군에서 제거하고, 결과로 얻은 정적인 라이다 점군은 라이다 SLAM에 입력된다. 다음으로, 제안된 방법론은 라이다 SLAM을 통해 자차량의 위치 및 자세를 추정한다.

이를 위해 본 연구는 라이다 SLAM의 프레임워크인 LeGO-LOAM을 채택하였다. 특징점 기반 SLAM인 LeGO-LOAM은 라이다 점군을 거리 기반 이미지로 변환시켜 특징점인 모서리 점과 평면 점을 추출한다. 또한 거리 기반 이미지를 사용한 전처리 과정을 통해 계산 효율을 높인다. 추출된 현재 스텝의 특징점과 이에 대응되는 이전 스텝의 특징점으로 이루어진 기하학적 구조와의 거리를 잔차로 설정하여 6 자유도 변환식에 대한 모델 방정식을 얻을 수 있다. 참고한 LeGO-LOAM은 해당 방정식을 Levenberg-Marquardt 방법을 통해 최적화를 수행한다. 또한, 본 연구는 참고된 인지 모듈의 처리 지연 문제를 보완하기 위해 이동 물체 인지 모듈과 LeGO-LOAM의 병렬 처리 구조를 고안하였다. 실험은 도심환경에서 32채널 3차원 라이다와 고정밀 GPS를 장착한 실험차량으로 진행되었다. 성능 검증 결과, 제안된 방법은 목표 출력 속도를 보장하면서 움직이는 특징점으로 인한 추정 오차를 유의미하게 줄일 수 있었다.

주요어: 측위, 자율주행차량, 이동 물체, 이동량, 정적 환경, 병렬 처리, 라이다

학 번: 2021-28230