IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 23, 2020, accepted September 2, 2020, date of publication September 21, 2020, date of current version October 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3025537

Robust 3-Dimension Point Cloud Mapping in Dynamic Environment Using Point-Wise Static Probability-Based NDT Scan-Matching

SUMYEONG LEE[1], (Student Member, IEEE), CHANSOO KIM[2], (Member, IEEE), SUNGJIN CHO[2], (Student Member, IEEE), SUNWOO MYOUNGHO[2], (Life Member, IEEE), AND KICHUN JO[3], (Member, IEEE)

[1]Robot Recognition Development Team, LG Electronics, Seoul 07795, South Korea
[2]Department of Automotive Engineering, Hanyang University, Seoul 04763, South Korea
[3]Department of Smart Vehicle Engineering, Konkuk University, Seoul 05029, South Korea

Corresponding author: Kichun Jo (kichun.jo@gmail.com)

This work was supported in part by the BK21 Plus Program through the Ministry of Education, South Korea, under Grant 22A20130000045, in part by the Industrial Strategy Technology Development Program under Grant 10039673, Grant 10060068, Grant 10042633, Grant 10079961, and Grant 10080284; in part by the International Collaborative Research and Development Program through the Ministry of Trade, Industry and Energy (MOTIE, South Korea) under Grant N0001992, in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MEST) under Grant 2011-0017495, Grant 2019R1G1A1099806, and Grant 2020R1C1C1007739, and in part by the Energy Resource Research and Development Program through the Ministry of Knowledge Economy, South Korea, under Grant 2006ETR11P091C.

ABSTRACT Graph-based simultaneous localization and mapping (SLAM) is one of the methods to generate point cloud maps which are used for various applications in autonomous vehicles. Graph-based SLAM represents the pose of the vehicle as a node and the odometry between two different nodes as an edge. Among the edge generating methods, scan matching, light detection and ranging (LiDAR) based method, can provide an accurate pose between two nodes based on the high distance accuracy of the LiDAR. However, the point cloud in real driving situations contains numerous moving objects, which degrade the scan-matching performance. Therefore, this article defines the static probability which means the likelihood that an acquired point is from a static object, and proposes the weighted normal distribution transformation (NDT), which is achieved by modifying NDT. Weighted NDT is a scan-matching algorithm which can reflect the static probability of each point as a weight. The odometry from the weighted NDT is utilized for graph construction to generate a robust point cloud map even in a dynamic environment. Finally, the proposed algorithm was compared with the existing object removal algorithms in two areas: dynamic object classification and scan-matching performance. Based on the scan-matching results, the accuracy of the point cloud map generated by the proposed algorithm was evaluated with a reference map using high-performance global navigation satellite system (GNSS). It was confirmed that the proposed algorithm has higher classification accuracy and lower scan-matching error compared with other dynamic object removal methods. The proposed algorithm was able to generate a point cloud map, despite the presence of many dynamic objects, that was similar to a map generated in the absence of dynamic objects in the same environment.

INDEX TERMS Static probability, scan-matching, weighted NDT, LiDAR characteristic, graph-based SLAM.

I. INTRODUCTION

Light detection and ranging (LiDAR), which is an essential sensor in autonomous vehicles, has the following two characteristics. First, it can provide precise distance information on vehicles, buildings, and people in the form of points. Second, a LiDAR sensor can represent the surroundings of the vehicle with a high resolution. Such a reconstructed three-dimensional (3D) environment is called a point cloud map, which can be utilized in various

The associate editor coordinating the review of this manuscript and approving it for publication was Razi Iqbal.

applications (e.g., localization and path planning) in autonomous vehicles [1]–[3].

To generate a point cloud map, the vehicle with LiDAR acquires point cloud data while driving along the road. The center of the reference coordinates is set as the starting point of data acquisition. The sensor coordinates of each point cloud change in real-time during driving relative to the reference coordinates. Therefore, to generate a point cloud map, it is necessary to estimate the sensor coordinates of the LiDAR at each time step.

To estimate the sensor coordinates of the LiDAR while simultaneously generating the point cloud map, an optimization technique called graph-based simultaneous localization and mapping (SLAM) is generally used [4]–[6]. In graph-based SLAM, all the sensor poses are represented as nodes. The nodes are optimized based on edge constraints, which are generated by the relationship between two nodes. In general, an inertial navigation system (INS) or environment-aware sensor (e.g., camera, LiDAR) is used to generate edge constraints [7], [8]. Among the various methods for constructing graph edge constraints, scan-matching can be used in graph-based SLAM for LiDAR-based generation [9]–[11]. LiDAR provides accurate distance information on objects and detailed depictions of the surrounding environment at a high resolution. These characteristics of LiDAR can be used in scan-matching to match the same static object in two different point clouds. Thus, we can estimate the relative pose between two acquired poses in the point clouds. Then, the constraint between nodes is created with the estimated relative pose.

If only static objects exist in the environment where point cloud maps are created, then scan-matching can accurately estimate the relative pose. However, in a real driving environment, there are numerous dynamic objects such as cars and pedestrians, which change their pose during the data acquisition for mapping. This can result in incorrect estimates of the relative pose while the point clouds are aligned owing to the effect of the dynamic objects [12]. If the wrong match result is used as a constraint of the graph, then the graph optimization is carried out incorrectly, which in turn leads to inaccurate map generation. Thus, the effect of dynamic objects must be eliminated during scan-matching to create an accurate point cloud map.

This article proposes a novel point cloud mapping method to generate an accurate map in an environment with numerous dynamic objects. The proposed algorithm consists of three steps. The first is to estimate the static probability of each point. In this study, we define static probability as the possibility of a point being acquired from a motionless object. The probability is calculated using point cloud information obtained from the LiDAR sensor and the two-dimensional (2D) pose of the sensor. The static probability of the point cloud is calculated by comparing the previous point cloud information according to probability theory and LiDAR characteristics. In the second step, scan-matching is carried out, reflecting the static probability of each point. Among the many scan-matching methods, this study utilizes the normal distribution transform (NDT) algorithm, which is proven to have excellent matching performance [13], [14]. The basic NDT algorithm to reflect the static probabilities, called weighted NDT, is also proposed. The weighted NDT algorithm conducts point cloud matching in a dynamic environment by assigning higher and lower weights to points with higher and lower probabilities, respectively. Finally, the result is used as a constraint in graph-based SLAM. Thus, we could generate an accurate point cloud map with the point cloud acquisition spots from an optimized graph.

The proposed algorithm was verified according to three indicators. First, the accuracy of the estimated static probability was evaluated according to a confusion matrix. Second, the scan-matching error was evaluated scene-by-scene and illustrated in an error histogram. Third, the accuracy of the generated point cloud map was evaluated quantitatively by calculating the closest distance from the reference map, which was generated from the high-precision global navigation satellite system (GNSS). All experiments were conducted in a dynamic environment and compared with other algorithms; these are covered in Section 2.

This article makes two main contributions. First, it calculates the static probability of an individual point based on the probability theory and LiDAR's characteristics. By considering the latter, more accurate static probabilities can be calculated compared with other dynamic object classification methods. Second, the paper proposes a modified NDT algorithm that reflects the static probability of an individual point as a weight. Basic NDT algorithms assign the same weight to all points. In contrast, the modified NDT algorithm reflects the calculated static probability to exclude the influence of dynamic points. This method can estimate the related poses of two-point clouds accurately in a dynamic environment. Finally, an accurate point cloud map in a dynamic environment is generated based on the accurate relative pose.

II. PREVIOUS STUDIES

To carry out accurate point cloud mapping in dynamic environments, the relative pose of point clouds must by estimated by reducing the effects of dynamic objects in scan matching. Previous research on point cloud mapping that considered the effects of dynamic objects can be divided into two major categories: classifying dynamic objects and estimating the relative pose of point clouds using scan matching for mapping.

A. CLASSIFICATION OF DYNAMIC OBJECTS

To eliminate the influence of dynamic objects, it is necessary to classify dynamic objects and remove the classified points. If the dynamic objects are effectively removed, then scan matching can be performed as in a static environment. There are two representative dynamic point classification algorithms: occupancy grid maps and detection and tracking moving objects (DATMO).

An occupancy grid map is a static map of the surrounding environment created by accumulating measured values [15].

The surrounding environment must be divided into a 2D or 3D space to make the map. Each grid cell has a uniform size and fixed structure. Each also has an occupancy level, which is updated by the laser's ray tracing. The occupancy level increases when the rays are reflected back after encountering a structure and decreases when no rays are reflected back when passing through the area. The occupancy level of a grid increases based on probability theory [16] and evidence theory [17]. Consequently, a static object increases the occupancy level as its location does not change. In contrast, dynamic objects do not increase the occupancy level because their locations continuously change. In this process, the occupancy grid map can remove dynamic objects by considering a grid with an occupancy level below the threshold as free. Occupancy grid map-based classification has been studied for a long time in this field. However, the method has limited real-time applications in autonomous vehicles. When the occupied grid map has a small cell size, a considerable amount of memory is required for storage because all the space around the vehicle is divided into grids or voxels to represent occupancy. It also requires a long time to trace each LiDAR beam and update the occupancy level of each cell. Moreover, when the grid map has few cells, discretization errors can occur because the space is divided discretely.

DATMO is another static point classification algorithm [18]–[20]. This field studies how to detect and track moving objects around the ego vehicle. At the object detection level, the nearby points are clustered around a point cloud. Among the clustered point clouds, the algorithm creates a bounding box to the point cloud, which is considered to represent dynamic objects. In the tracking step, this algorithm can track the bounding box to obtain position, heading, velocity, and acceleration information through various filtering techniques. Despite considerable research on such point-clustering methods, it remains difficult to accurately distinguish between static and dynamic objects with only point cloud information. In particular, in an environment with many moving objects, it is difficult to create the vehicle bounding box as the points of each vehicle are hidden behind the surrounding vehicles. Furthermore, because the tracking method requires the initial time to create a new track, it is impossible to remove the dynamic objects before the track is created.

To enhance the classification performance, a method that can estimate the static probability of each point is proposed. The static probability of a point is calculated using the laser beam model of LiDAR. Because this method does not separate the space discretely, there is no discretization error or erroneous clustering, which occurs in occupancy grid maps and DATMO, respectively.

B. ESTIMATION OF THE RELATIVE POSE SCAN-MATCHING

The scan-matching method used as the constraint for graph-based SLAM has been actively studied since the 2000s [21]. This method aligns the same object contained in two different point clouds. Through this, it is possible to calculate the relative pose between the two poses where the point clouds are acquired. There are two representative scan-matching techniques: iterative closest point (ICP) [22] and NDT [23]. The ICP method generates a correspondence between points and finds the rotation and translation that minimizes the distance between all correspondences. This research is extended to generate a correspondence between lines and faces [24], [25]; improved studies with many different versions exist [26]. NDT divides the standard point cloud into equal cell sizes and calculates the normal distribution. Then, it scores how well it compares to other normal distributions that are derived from other point clouds. The algorithm also finds the rotation and translation that increases the scores given. NDT has been applied to a 2D grid cell and 3D voxel cell by Magnusson [23]. Magnusson also proved that NDT performs better than ICP when there is little overlap between two point clouds and no distinct geometric structure through experiments and under various conditions [27].

Although many matching methods exist and their performances have been compared, it is difficult to apply them in a dynamic environment. Because these methods are often evaluated in environments with static objects, their performance cannot be guaranteed in dynamic environments. Therefore, this article proposes a weighted NDT, which is modified to reflect the static probability of each point based on the existing NDT algorithm. By using the weighted NDT, scan-matching results for graph-based SLAM constraints are generated more accurately.

III. SYSTEM ARCHITECTURE

The objective of this algorithm is to generate an accurate 3D point cloud map in a dynamic environment. To do so, the LiDAR coordinates $(x_1, x_2, \cdots, x_m)$ and point cloud information $(Z_1, Z_2, \cdots, Z_m)$ at each time step were used as inputs of the algorithm. The entire algorithm of this system is illustrated in Fig.1.

First, the proposed algorithm received the 3D point cloud and its acquisition pose as input. The point cloud at time $= k$ consists of laser beams and its multi-echoes. The point cloud element can be represented as $z_t^{n,m}$, where n denotes the index of the laser beam from 1 to N, and m represents the multi-echo index for each laser beam, which often has a value less than 2. The LiDAR pose at time $= k$ is denoted as x_t, where x_t is calculated by integrating the motion information from an initial pose x_1. x_t consists of three states, i.e., position x, position y, and angle yaw. We can utilize various types of motion information, such as inertial measurement unit(IMU) or on-board motion sensors based on a motion model(constant velocity or constant acceleration). In this study, the motion sensor was not determined for scalability reasons. When all the point clouds $Z_1, Z_2, \cdots, Z_m$, and all the LiDAR poses $x_1, x_2, \cdots, x_m$ are given, the proposed algorithm is performed in the following three steps.

In the first step, the algorithm calculates the static probability of the individual points. Static probability refers to the probability that an individual point measured from LiDAR is

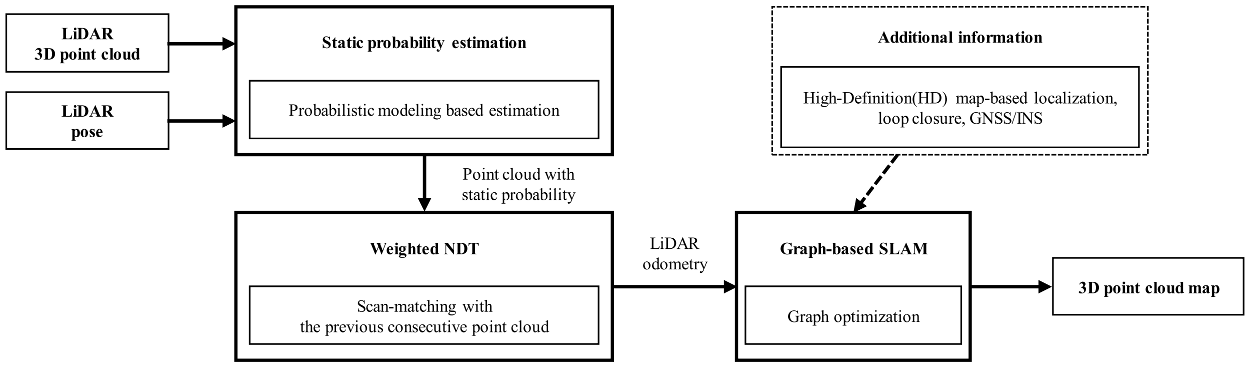

LiDAR 3D point cloud
LiDAR pose
Static probability estimation
Probabilistic modeling based estimation
Point cloud with static probability
Weighted NDT
Scan-matching with the previous consecutive point cloud
LiDAR odometry
Additional information
High-Definition(HD) map-based localization, loop closure, GNSS/INS
Graph-based SLAM
Graph optimization
3D point cloud map

FIGURE 1. **System architecture of point cloud mapping using static probability-based NDT.**

obtained from a static object. If the point acquired at the current time was measured at the same position as in the previous time, then it can be assumed that the point was acquired from a fixed static object. On the other hand, if the point acquired at the current time is located in the ray area of the previous point, then the current point is expected to have been obtained from a dynamic object. To estimate the static probability of Z_t, the total W of point clouds $Z_{t-1}, \cdots, Z_{t-W}$ and sensor poses $x_{t-1}, \cdots, x_{t-W}$ of the previous time are required. The W is the window size to estimate the static probability of Z_t, and it can be optimized according to estimation performance. Using the immediately preceding point cloud Z_{t-1} and its pose x_{t-1}, the areas in which the object does and does not exist can be represented in the 3D map. In this situation, if the point in Z_t is located near the area with the object, then the point will have a high static probability. However, if the point in Z_t is located in the sweep area by the ray of the previous point, then it will have a low static probability. Similarly, the static probability of Z_t was calculated using not only Z_{t-1} but also $Z_{t-2}, \cdots, Z_{t-W}$. Multiple calculated static probabilities of Z_t can be integrated into one through Bayes' rule.

Although we already know the LiDAR poses from motion information, the IMU and on-board sensor have a bias accumulation problem. Thus, it is impossible to generate an accurate point cloud map using only motion information. The next step is to conduct scan matching between two consecutive point clouds. Scan matching is a method that aligns different point clouds using their static objects to estimate the relative pose. This relative pose can be used as a LiDAR odometry. However, in a dynamic environment, it is difficult to perform scan matching because of the dynamic objects. Therefore, we must utilize the static probability that was previously calculated. The basic NDT algorithm matches all points with the same weight. However, the proposed algorithm aligns the weight of each point according to their static probability to reduce the effect of dynamic objects in LiDAR odometry estimation. If a point has a high static probability, then it is assigned a large weight to significantly influence the scan matching. Likewise, a point with low static probability is assigned a low weight.

All the LiDAR odometry was estimated between consecutive point clouds. These values were applied as constraints in the graph-based SLAM. According to the generated constraints, the graph-based SLAM optimizes all the graphs to minimize the constraint error. As a result, the LiDAR poses at each time are obtained as an output of the algorithm. Furthermore, accumulating point clouds based on the calculated LiDAR poses can generate the point cloud map. In this study, scan matching-based LiDAR odometry is explained as a constraint in the graph. However, to improve the mapping performance, additional information, such as high-definition (HD) map-based localization, loop closure, and GNSS/INS can be added as a constraint on the graph.

IV. POINT-WISE STATIC PROBABILITY ESTIMATION

A. CHARACTERISTICS OF LiDAR POINT CLOUD

LiDAR rotates and fires a laser beam to the environment at an angle. When the laser beam hits an object, it is reflected back. At this time, the time-of-flight (ToF) is measured by calculating the difference between the time when the pulse is emitted from the diode and when it returns. The distance to the reflected object can be estimated by multiplying the measured ToF by the laser's velocity.

The laser beam has several characteristics, which were applied in this study. First, it does not travel in a straight line; the cross-section of the beam increases as it moves forward. The rate of increase in cross-sectional area in the horizontal and vertical directions is determined by the individual LiDAR specifications. Second, the laser beam can measure multiple echoes; hence, it can produce several points within a beam. Third, its distance information has an uncertainty. As the distance to the measured object is calculated through ToF, the accuracy of the time measurement is directly related to that of the measured distance. This section describes how to calculate the static probability of individual points using these LiDAR characteristics.

The basic concept of scan matching is aligning the pose of static objects in two point-clouds. Therefore, when we generate accurate scan-matching constraints in a dynamic environment, points from dynamic objects reduce the matching accuracy. To reduce the effect of dynamic points, we estimate the static probability of individual points and assign different weights based on this probability in the scan-matching process. To estimate the static probability, the point cloud is represented by a spherical coordinate and Cartesian coordinate. In the spherical coordinate, the point cloud at time $= t$ can be represented by the horizontal and vertical beam angles. It can be expressed as $Z_{r\theta\phi,t} = z^{1,m}_{r\theta\phi,t}, \cdots, z^{i,m}_{r\theta\phi,t}, \cdots, z^{N,m}_{r\theta\phi,t}$. Because of the multi-echo capability of LiDAR, more than one measurement can be generated in a single beam. Therefore, the individual beam $z^{i,m}_{r\theta\phi,t}$ has several measurements with different distances $r^{i,1}_t, r^{i,2}_t, \cdots, r^{i,m}_t$ in certain vertical and horizontal angles θ^i_t, ϕ^i_t. The Cartesian coordinate system has 3D information, which consists of x, y, z values.

B. PROBABILISTIC MODELING FOR LiDAR POINT STATIC PROBABILITY

The goal of this step is to obtain the static probability of the most recent point cloud through stochastic modeling. The static probability of point cloud Z_t is represented as $p(S_t)$. $p(S_t)$ includes the static probability of individual points $p(s^{1,m}_t), p(s^{2,m}_t), \cdots, p(s^{N,m}_t)$. To calculate the static probability of individual points of the current point cloud Z_t, the previously buffered point clouds $Z_{t-1}, \cdots, Z_{t-W+1}, Z_{t-W}$ and sensor coordinates $x_{t-1}, \cdots, x_{t-W+1}, x_{t-W}$ are utilized. The sensor pose can be obtained from the node of graph-based SLAM before graph optimization, which is generated from vehicle motion. The static probability of the current point cloud Z_t calculated with Z_t, Z_{t-k}, x_t, x_{t-k} can be expressed by the conditional probability shown in Equation (1).

$$p(S_{(t-k)\to t}) = p(S_t|Z_t, Z_{t-k}, x_t, x_{t-k}) \tag{1}$$

where $p(S_t)$ is the sum of the static probability of individual point $\{p\left(s^1_t\right), p\left(s^2_t\right), \cdots, p\left(s^N_t\right)\}$. Equation (1) can be transformed into Equation (2).

$$p\left(S_{(t-k)\to t}\right) = \{p\left(s^{1,m}_t \middle| Z_t, Z_{t-k}, x_t, x_{t-k}\right), \cdots, p\left(s^{N,m}_t \middle| Z_t, Z_{t-k}, x_t, x_{t-k}\right)\} \tag{2}$$

Each point consists of two probabilities, the static probability $p(s)$ and nonstatic probability $p(!s)$, the sum of which is always 1. The static probability of a point is expressed as Equation (3), which is rearranged according to Bayes' rule.

$$p(s^{i,m}_t|z^{i,m}_t, Z_{t-k}, x_t, x_{t-k}) = \frac{p(z^{i,m}_t|s^{i,m}_t, Z_{t-k}, x_t, x_{t-k})p(s^{i,m}_t|Z_{t-k}, x_t, x_{t-k})}{p(z^{i,m}_t|Z_{t-k}, x_t, x_{t-k})} \tag{3}$$

In Equation (3), $p\left(z^{i,m}_t \middle| s^{i,m}_t, Z_{t-k}, x_t, x_{t-k}\right)$ is the likelihood function when the point is in a static state. $p\left(s^{i,m}_t \middle| Z_{t-k}, x_t, x_{t-k}\right)$ is a predicted probability density function since $s^{i,m}_t$ is determined by the given Z_{t-k}, x_t, x_{t-k}, and state consists of only static or nonstatic states. Therefore, we can assign the $p\left(s^{i,m}_t \middle| Z_{t-k}, x_t, x_{t-k}\right)$ as 0.5. Finally,

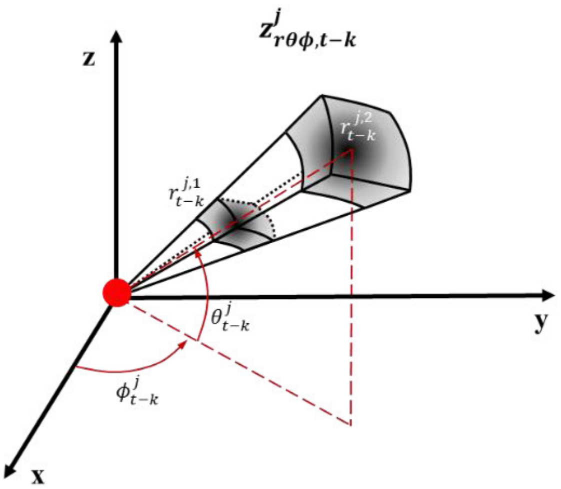


FIGURE 2. Likelihood field for the point $z^{i,m}_t$ is constructed by previous point $z^{j,l}_{r\theta\phi,t-k}$.

$p\left(z^{i,m}_t \middle| Z_{t-k}, x_t, x_{t-k}\right)$ is the normalization constant. In summary, the static probability of a point can be calculated if there is a likelihood function for a static state.

$$p\left(s^{i,m}_t \middle| z^{i,m}_t, Z_{t-k}, x_t, x_{t-k}\right) = \eta\, p(z^{i,m}_t|s^{i,m}_t, Z_{t-k}, x_t, x_{t-k}) \tag{4}$$

where η is the normalization constant of the likelihood function to satisfy the condition that the sum of the static and nonstatic probabilities is 1. The normalization is multiplied with the likelihood function $p\left(!s^{i,m}_t \middle| z^{i,m}_t, Z_{t-k}, x_t, x_{t-k}\right)$ to obtain the final static probability.

C. LIKELIHOOD FUNCTION FOR THE STATIC STATE OF POINT CLOUD

To calculate the static probability $p\left(S_{(t-k)\to t}\right)$ of the current point cloud, the likelihood function $p(z^{i,m}_t|s^{i,m}_t, Z_{t-k}, x_t, x_{t-k})$ is required. The likelihood function $p(z^{i,m}_t|s^{i,m}_t, Z_{t-k}, x_t, x_{t-k})$ indicates how static points are statistically distributed when Z_{t-k}, x_t, x_{t-k} are given. The likelihood function of $z^{i,m}_t$ for a static state is expressed as a spherical coordinate system based on the sensor pose x_{t-k} at the previous time $= t-k$. The region with the black in Figure 3 is the region where the object is detected at time $= t-k$. Therefore, the current point $z^{i,m}_t$ in the likelihood region located near the black region is likely to be a static point. On the other hand, the static probability of the point located farther away from that region decreases. The point cloud from LiDAR is detected with horizontal and vertical angles. Therefore, it would be easier to understand if point cloud Z_{t-k} is converted into a spherical coordinate.

$$Z_{t-k} = Z_{r\theta\phi,t-k} = z^{1,l}_{r\theta\phi,t-k}, \cdots, z^{j,l}_{r\theta\phi,t-k}, \cdots, z^{J,l}_{r\theta\phi,t-k} \tag{5}$$

One of the elements of Z_{t-k}, $z^j_{r\theta\phi,t-k}$ can have more than one return value in one laser beam because

]

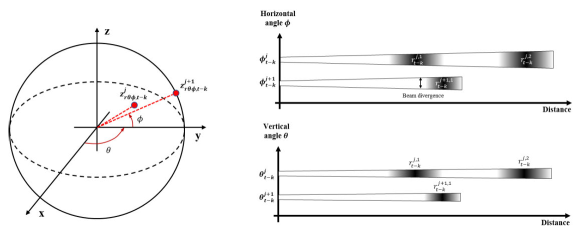


FIGURE 3. Two-dimensional likelihood field of the previous measurements $z_{t-k}^{j,1}$, $z_{t-k}^{j,2}$, and $z_{t-k}^{j+1,1}$ in the (r-θ) plane and (r-ϕ) plane.

of the multi-echo described in the LiDAR characteristic. Therefore, it can be represented as $z_{r\theta\phi,t-k}^{j} = \left\{r_{t-k}^{j,1}, r_{t-k}^{j,2}, \cdots, r_{t-k}^{j,m}, \theta_{t-k}^{j}, \phi_{t-k}^{j}\right\}$. The state likelihood function represented in three dimensions can be represented by 2D planes: (r-θ) plane and (r-ϕ) plane. In the $(r-\theta)$ plane, $p\left(z_t^{i,m}\middle|s_t^{i,m}, Z_{t-k}, x_t, x_{t-k}\right)$ is converted into $p\left(r_t^{i,m}, \phi_t^{i}\middle|s_t^{i,m}, Z_{t-k}, x_t, x_{t-k}\right)$, and converted into $p\left(r_t^{i,m}, \theta_t^{i}\middle|s_t^{i,m}, Z_{t-k}, x_t, x_{t-k}\right)$ in the $(r-\phi)$ plane. Figure 3 shows the likelihood function of the elements z_{t-k}^{j} and z_{t-k}^{j+1} of the point cloud at time $= t-k$ in the $(r-\theta)$ and $(r-\phi)$ planes.

Each likelihood function reflects beam divergence according to the actual LiDAR specifications and multi-echo characteristics, in which multiple measurements are returned in one laser beam. Figure (4) shows the static likelihood function when Z_{t-k}, x_t, x_{t-k} are given. When the previous measurement z_{t-k}^{j} has two multi-echoes, it can be expressed as $\left\{r_{t-k}^{j,1}, r_{t-k}^{j,2}, \theta_{t-k}^{j}, \phi_{t-k}^{j}\right\}$. As mentioned before, if the point at the current time is located in the same area where the point was measured at a previous time, then it is likely to be static. The uncertainty of the distance $r_t^{i,m}$, which is measured according to the LiDAR's ToF principle, is determined by LiDAR's ToF measurement accuracy. To account for the uncertainty, the likelihood of a static state is expressed as a Gaussian distribution, as in Equation (6).

$$p\left(r_t^{i,m}\middle|s_t^{i,m}, z_{t-k}^{j}, x_t, x_{t-k}\right) = \max\left(\frac{1}{\sigma\sqrt{2\pi}}e^{-(r_{t-k}^{j,1}, -r_t^{i,m})}, \frac{1}{\sigma\sqrt{2\pi}}e^{-(r_{t-k}^{j,2}, -r_t^{i,m})}\right) \quad (6)$$

The standard deviation of the measured distance σ is determined by the LiDAR used. However, the likelihood of a nonstatic state can be determined by subtracting the static likelihood value at that position from the maximum likelihood of the static state, as in Equation (7).

$$p\left(r_t^{i,m}\middle|!s_t^{i,m}, z_{t-k}^{j}, x_t, x_{t-k}\right) = MaxLikelihood_{t-k}^{j} - p\left(r_t^{i,m}\middle|s_t^{i,m}, z_{t-k}^{j}, x_t, x_{t-k}\right) \quad (7)$$

where $MaxLikelihood_{t-k}^{j}$ denotes the maximum likelihood value of the previous measurement z_{t-k}^{j}. It indicates when the distance between the current and previous measurements

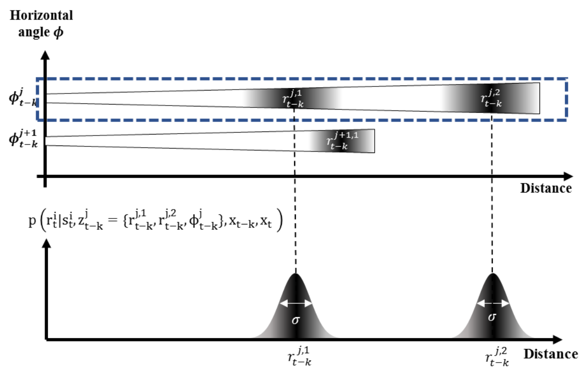


FIGURE 4. Likelihood of static state for one laser beam.

is the same, as described in Equation (8).

$$MaxLikelihood_{t-k}^{j} = p(r_{t-k}^{j,m}|s_t^{i,m}, z_{t-k}^{j}, x_t, x_{t-k}) \quad (8)$$

Thus, the likelihood of both the static and nonstatic states in one measurement at the current time can be defined. After calculating both likelihoods, we calculated the normalization constant b, which makes the sum of the two probabilities equal to 1, and is reflected in the calculation of the static probability of the measured value. Given the point cloud and sensor pose Z_t, x_t of the current point cloud, and the sensor pose Z_{t-k}, x_{t-k} at the previous time $= t-k$ through the above process, the static probability $p(S_{(t-k)\rightarrow t})$ is calculated.

D. FINAL STATIC PROBABILITY ESTIMATION BY INTEGRATING THE STATIC PROBABILITIES

The static probability of the current point cloud can be calculated using point clouds from time $= t-W$ to time $= t-1$ just as $p\left(S_{(t-k)\rightarrow t}\right) = p(S_t|Z_t, Z_{t-k}, x_t, x_{t-k})$ was calculated in Section 4.3. Therefore, it is necessary to integrate the static probabilities of the current point clouds classified through different previous point clouds $p\left(S_{(t-1)\rightarrow t}\right), \cdots, p\left(S_{(t-k)\rightarrow t}\right), \cdots, p\left(S_{(t-W)\rightarrow t}\right)$ into one integrated static probability. The integrated static probability of the current point cloud is expressed as $p(S_t|Z_t, Z_{t-W:t-1}, x_t, x_{t-W:t-1})$. To calculate this value, the definition of the log odds ratio is used. The log odds ratio is the ratio of the probability that an event will occur and the probability that an event will not occur. The static probability of the current point cloud classified through the information at time $= t-k$ is $p(S_t|Z_t, Z_{t-k}, x_t, x_{t-k})$ and the nonstationary probability is $p(!S_t|Z_t, Z_{t-k}, x_t, x_{t-k})$. Since the sum of the two is always 1, the log odds ratio of the static probability $l(S_t|Z_t, Z_{t-k}, x_t, x_{t-k})$ can be defined by Equation (9).

$$\begin{aligned} l(S_t|Z_t, Z_{t-k}, x_t, x_{t-k}) &= \log\frac{p(S_t|Z_t, Z_{t-k}, x_t, x_{t-k})}{p(!S_t|Z_t, Z_{t-k}, x_t, x_{t-k})} \\ &= \log\frac{p(S_t|Z_t, Z_{t-k}, x_t, x_{t-k})}{1-p(S_t|Z_t, Z_{t-k}, x_t, x_{t-k})} \end{aligned} \quad (9)$$

This log odds ratio can be accumulated as the sum of individual log odds ratios by the binary Bayes' filter in

Equation (10).

$$l(S_t|Z_t, Z_{t-W:t-1}, x_t, x_{t-W:t-1}) = l(S_t|Z_t, Z_{t-W}, x_t, x_{t-W}) + \cdots + l(S_t|Z_t, Z_{t-1}, x_t, x_{t-1}) \quad (10)$$

When modifying Equation (10), the final static probability can be obtained as in Equation (11).

$$p(S_t|Z_t, Z_{t-W:t-1}, x_t, x_{t-W:t-1}) = 1 - \frac{1}{1+\exp(l(S_t|Z_t, Z_{t-W:t-1}, x_t, x_{t-W:t-1}))} \quad (11)$$

V. NDT SCAN-MATCHING-BASED EDGE CONSTRAINT GENERATION

A. BASIC NDT SCAN-MATCHING

The basic principle of scan matching is to align points from the same object contained in two point-clouds. Among the various scan-matching algorithms, we used the NDT algorithm, which has robust matching performance under various conditions. In this study, a point cloud used as a scan-matching reference is called a target point cloud, while a point cloud that is matched to a target point cloud is defined as a source point cloud. A characteristic of the NDT algorithm is that the target point cloud is transformed into a combination of normal distributions to express the surface of the surrounding environment. NDT is performed by determining the degree of matching with the normal distribution, and not directly with the target points. If the number of points contained in the divided voxel is greater than five, then the normal distribution is obtained by calculating the mean (μ) and covariance matrix (Σ). Assuming that a set of point clouds contained in a voxel is $Y = y_1, \cdots, y_m$, the mean and covariance matrix calculations are as follows.

$$\mu = \frac{1}{m}\sum_{k=1}^{m} y_k, \quad \Sigma = \frac{1}{m}\sum_{k=1}^{m}(y_k-\mu)(y_k-\mu)^T \quad (12)$$

When the mean and covariance matrices are calculated, the normal distribution of the D-dimension is expressed by Equation (13).

$$p(x) = \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma|}} exp\left(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right) \quad (13)$$

The score function of the individual source point is calculated from the above equation. The goal is to maximize the product of the scores of all points. However, this normal distribution is vulnerable to the effects of outliers. Previous studies have solved this problem by mixing the normal distribution and uniform distribution. The problem has also been modified to calculate the sum of all scores by simply taking the log-likelihood of the whole expression rather than calculating the product of scores.

$$score(\vec{p}) = -\sum_{k=1}^{n}\tilde{p}(T(\vec{p}, x_k)) \quad (14)$$

where $T(\vec{p}, x_k)$ are the means to transform x_k by vector $\vec{p}$. The scoring sum of all source points converted by $\vec{p}$ is the final score of $\vec{p}$. An optimization technique based on Newton's law finds a vector $\vec{p}$ that minimizes the score.

$$H\Delta\vec{p} = -\vec{g} \quad (15)$$

Newton's law is expressed as Equation (15), and H and $\vec{g}$ represent the Hessian matrix and gradient vector, respectively. The Hessian matrix and gradient vector can be obtained by Equation (16) and Equation (17), respectively, when x'_k defined as $T(\vec{p}, x_k) - \mu$.

$$H_{ij} = \sum_{k=1}^{n} d_1 d_2 exp(-\frac{d_2}{2}{x'_k}^T\Sigma_k^{-1}x'_k) (-d_2({x'_k}^T\Sigma_k^{-1}\frac{\delta x'_k}{\delta\vec{p}_i})({x'_k}^T\Sigma_k^{-1}\frac{\delta x'_k}{\delta\vec{p}_j}) + {x'_k}^T\Sigma_k^{-1}\frac{\delta^2 x'_k}{\delta\vec{p}_i\delta\vec{p}_j} + \frac{\delta x'_k}{\delta\vec{p}_j}\Sigma_k^{-1}\frac{\delta x'_k}{\delta\vec{p}_i}) \quad (16)$$

$$g_i = \sum_{k=1}^{n} d_1 d_2 {x'_k}^T\Sigma_k^{-1}x'_k\frac{\delta x'_k}{\delta\vec{p}_i} exp\left(-\frac{d_2}{2}{x'_k}^T\Sigma_k^{-1}x'_k\right) \quad (17)$$

NDT calculates the Hessian matrix and gradient vector at each point and finds Δp, which minimizes the score by adding all the values. The calculated Δp is added to $\vec{p}$, which was calculated in the previous step to obtain a new $\vec{p}$. Finally, the optimized transformation vector $\vec{p}$ between two point-clouds can be obtained by repeating the same process. Algorithm 1 presents the pseudo code of all the processes.

B. WEIGHTED NDT SCAN-MATCHING BASED ON THE STATIC PROBABILITY

Basic NDT scan-matching gives equal weight to all points in the scan-matching process. In a dynamic environment, if all points have equal weight in matching, points from dynamic objects can degrade the matching performance. Therefore, to reduce the influence of dynamic points, an NDT scan-matching algorithm that can reflect the static probability of each point is proposed. The static probability of each point is utilized as the weight of each point. The weight of each point is reflected in the matching process in two ways. First, the weight of the target point cloud can be applied in the calculation of the normal distribution. The target point cloud is represented as a combination of normal distributions. The normal distribution is obtained by calculating the mean and covariance matrix of points, which exist inside the voxel. The calculation of the normal distribution with the static probability is shown in Equation (18) and Equation (19).

$$q^* = \frac{\sum_{k=1}^{m}\omega_k y_k}{\sum_{k=1}^{m}\omega_k} \quad (18)$$

where ω_k represents the static probability of the kth point. The weighted covariance matrix can be calculated using the weighted mean.

$$\Sigma^* = \frac{\sum_{k=1}^{m}\omega_k}{\left(\sum_{k=1}^{m}\omega_k\right)^2 - \sum_{k=1}^{m}\omega_k^2}\sum_{k=1}^{m}\omega_k\left(y_k-q^*\right)\left(y_k-q^*\right)^T$$

Algorithm 1 Basic NDT Algorithm

```
Input:
    The source point cloud $X$
    The target point cloud $Z$
    Initial guess of transformation $\vec{p}_{ini}$
Output:
    Final transformation $\vec{p}$ between $X$ and $Z$
 1: $\vec{p} \leftarrow \vec{p}_{ini}$
 2: for all points $z_i \in Z$ do
 3:     find the cell $Y$ that contains $z_i$
 4:     classify $z_i$ to entire cells $Y$
 5: end for
 6: for all cells $Y$ do
 7:     $Y = \{y_1, \cdots, y_m\}$
 8:     $q = \frac{1}{m}\sum_{k=1}^{m} y_k,\ \Sigma = \frac{1}{m}\sum_{k=1}^{m}(y_k - q)(y_k - q)^T$
 9: end for
10: while not converged do
11:     score $\leftarrow 0$, $\vec{g} \leftarrow 0$, $H \leftarrow 0$
12:     for all points $x_i \in X$ do
13:         find the cell $Y$ that contains $T(\vec{p}, x_k)$
14:         update $\vec{g}$, $H$
15:     end for
16:     solve $H\Delta\vec{p} = -\vec{g}$
17:     $\vec{p} \leftarrow \vec{p} + \Delta\vec{p}$
18: end while
```

$$= \frac{\sum_{k=1}^{m} \omega_k (y_k - q^*)(y_k - q^*)^T}{V_1 - (V_2/V_1)} \tag{19}$$

If we define $V_1 = \left(\sum_{k=1}^{m} \omega_k\right)^2$ and $V_2 = \sum_{k=1}^{m} \omega_k^2$, then the whole equation can be summarized as Equation 5–9. Next, Newton's law of each point reflects the static probability of the point in the Hessian matrix and gradient vector computation. The pseudo code of the static probability-based NDT scan-matching is as follows.

Algorithm 2 Weighted NDT Algorithm

```
Input:
    The source point cloud $X$
    The target point cloud $Z$
    The static probability of point clouds $p(S)$
    Initial guess of transformation $\vec{p}_{ini}$
Output:
    Final transformation $\vec{p}$ between $X$ and $Z$
 1: $\vec{p} \leftarrow \vec{p}_{ini}$
 2: for all points $z_i \in Z$ do
 3:     find the cell $Y$ that contains $z_i$
 4:     classify $z_i$ to entire cells $Y$
 5: end for
 6: for all cells $Y$ do
 7:     $Y = \{y_1, \cdots, y_m\}$
 8:     $w_k \leftarrow p(s_k)$
 9:     $q^* = \frac{\sum_{k=1}^{m} w_k y_k}{\sum_{k=1}^{m} w_k},\ \Sigma = \frac{\sum_{k=1}^{m} \omega_k (y_k - q^*)(y_k - q^*)^T}{V_1 - (V_2/V_1)}$
10: end for
11: while not converged do
12:     score $\leftarrow 0$, $\vec{g} \leftarrow 0$, $H \leftarrow 0$
13:     for all points $x_i \in X$ do
14:         find the cell $Y$ that contains $T(\vec{p}, x_k)$
15:         $w_k \leftarrow p(s_k)$
16:         update $w_k * \vec{g}$, $w_k * H$
17:     end for
18:     solve $H\Delta\vec{p} = -\vec{g}$
19:     $\vec{p} \leftarrow \vec{p} + \Delta\vec{p}$
20: end while
```

VI. POINT CLOUD MAPPING BASED ON GRAPH-BASED SLAM

To construct the point cloud map, graph-based SLAM estimates the acquired pose of the point cloud at each time step. Graph-based SLAM can be divided into the front-end and back-end. The front-end generates nodes of the graph each time a point cloud comes in from LiDAR. Graph-based SLAM generates edges through constraints from information, such as GNSS, motion, and scan matching. Based on the generated nodes and edges, the back-end optimizes the all of the graphs to obtain the accurate pose at each time step. Recently, iSAM [28] and g2o [29] have been widely used as the back-end of graph-based SLAM.

A. GRAPH CONSTRUCTION (FRONT-END)

Assuming that the environment in which the vehicle moves is 2D, the sensor pose is represented as $x_t = \{x_t, y_t, \theta_t\}$. The graph is a vector $x = x_{1:n}$ consisting of x_t representing the pose of the vehicle. The relative pose between two nodes x_i and x_j can be represented with the edge constraint. The information of the edge constraint can be obtained from the in-vehicle motion sensor, INS sensor, and scan matching, and even a static probability-based algorithm, which was described in the previous section.

B. GRAPH OPTIMIZATION (BACK-END)

In the constraint, x_i and x_j are the location of the two nodes before optimization, and $< z_{ij}, \Omega_{ij} >$ are the mean and information matrix of a virtual measurement of x_j obtained using GNSS, motion, or scan matching at pose x_i. The error between the pose of the virtual measurement and that of the node before optimization is denoted by $e_{ij}(x_i, x_j)$. Using these constraint information, graph optimization iteratively computes the final graph that minimizes the sum of these errors based on least squares.

$$x^* = argmin_x \sum_i e_{i,j}^T(x_i, x_j)\Omega_{i,j} e_{i,j}(x_i, x_j) \tag{20}$$

VII. EXPERIMENTS

A. EXPERIMENTAL ENVIRONMENT

To evaluate the proposed algorithm, the autonomous vehicle A1 of Hanyang University ACE Lab was used in the experiment. The A1 was equipped with two LiDARs and a GNSS/INS sensor. The two LiDARs in the vehicle enable

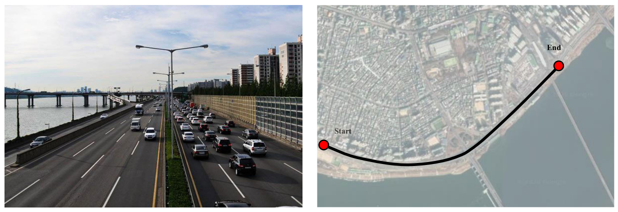


FIGURE 5. Test site with numerous dynamic objects.

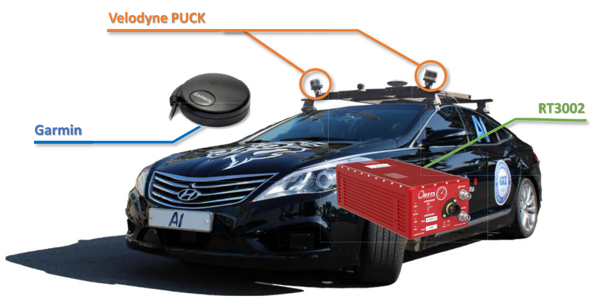


FIGURE 6. Sensor configuration of test vehicle A1.

point cloud acquisition in real-time during driving. The mounted LiDAR is Velodyne's Puck (VLP-16) sensor with horizontal and vertical angles of 360° and 30° degrees, respectively, in the region of interest (ROI). Through the Ethernet interface, the vehicle receives point clouds with an accuracy less than 3 cm and frequency of 10 Hz. Both LiDARs were calibrated and acquired a point cloud in the same coordinate system. A1 also has the RT3002 vehicle-mounted GNSS/INS sensor to evaluate the performance of the proposed algorithm. It can provide the absolute position and velocity with errors of less than 2 cm and 0.03 m/s, respectively, using real-time kinematic (RTK) GNSS correction. In addition, A1 has another low-cost GNSS sensor from Garmin, which was used as a constraint of graph-based SLAM to reduce the cumulative scan-matching error. The performance of the proposed algorithm was evaluated with following three experiments. As shown in Figure 5, all the experiments were conducted in Gangbyun Expressway, Seoul, Korea, which has numerous dynamic objects.

B. HOW EXACTLY THE STATIC PROBABILITIES OF INDIVIDUAL POINTS ARE ESTIMATED

To evaluate the performance of static probability estimation, it is necessary to determine the property of each point, i.e., whether it is dynamic or static. To tag the property of each point, we can manually classify the points and assign the property. However, the manual process also can cause errors and requires a large amount of computational power. Thus, we conducted the evaluation in the place where all points can be considered as static or dynamic. The test in the static environment was conducted at dawn when there were no vehicles on the road. On the other hand, to acquire the dataset of the dynamic environment, we drove to the center of highway which had numerous vehicles, and we only utilized the points from other vehicles within 8 m of the ego vehicle.

Based on the explained actual class of each point, we evaluated the static probability estimation performance of the tracking and proposed algorithm. The tracking deterministically derived the predicted class of each point as dynamic or static according to bounding boxes. On the other hand, the output of the proposed algorithm is the static probability of each point. Therefore, we made classifications based on whether the static probability of the point exceeds a certain value. The result is shown in Figure 7. The accuracy and recall rate of the two algorithms were calculated based on confusion matrices. It was confirmed that both the accuracy and recall of the proposed algorithm were higher than those of the tracking algorithm by at least 10%.

C. HOW EXACTLY DOES THE STATIC PROBABILITY BASED SCAN-MATCHING CALCULATE THE SENSOR ODOMETRY

The scan-matching performance of the proposed algorithm was compared with the basic NDT algorithm using a dynamic object removal algorithm. All adjacent point clouds were matched to estimate the odometry between the two point-clouds, and the results were compared with RTK-GPS-based ground truth. Before evaluating the proposed algorithm in the dynamic environment, an independent performance evaluation of the proposed algorithm was first performed in an environment without dynamic objects. Specifically, the test was conducted at dawn without moving vehicles. The scan-matching performance of the proposed algorithm is shown in Figure 8. Figure 8-(a) shows the distance and heading errors of the predicted odometry over the entire area of the test site. The distance error of the predicted odometry at the entire test site has a maximum value of 0.0590 m and an root-mean-square (RMS) value of 0.0190 m. The heading error has a maximum value of 0.2196° and an RMS value of 0.2196°. Figure 8-(b) shows the predicted absolute position of the vehicle by accumulating scan-matching results obtained with the proposed algorithm. Although matching errors were accumulated for all test sites, the predicted position of the proposed algorithm followed the reference position well. It can be confirmed that the proposed algorithm has good matching performance in an environment without dynamic objects.

Next, the performance of the proposed algorithm in an environment with many dynamic objects was compared with matching based on other dynamic object removal algorithms. As in the previous experiment, scan matching was performed for each algorithm in each scenario. Four algorithms were used in the experiment: the proposed algorithm, NDT without dynamic object removal, NDT with DATMO, and NDT with an occupancy grid map. To evaluate the performance of individual algorithms, error histograms of scan-matching results were compared. In the histograms, the more samples with errors close to 0, the better the scan-matching performance. The proposed algorithm had the largest number of samples with heading and distance errors close to 0 compared with other algorithms. Figure 10-(a) shows the distance and

Confusion matrix			Actual class		Performance	
			Static	Non-static		
Predicted class	Tracking	Static	670066	146591	Accuracy	78.2%
		Non-static	266600	816157	Static Recall	71.5%
					Non-static Recall	84.8%
	Proposed algorithm	Static	474597	35569	Accuracy	90.0%
		Non-static	103324	648292	Static Recall	82.1%
					Non-static Recall	94.8%

FIGURE 7. Confusion matrices of the tracking-based algorithm and proposed algorithm.

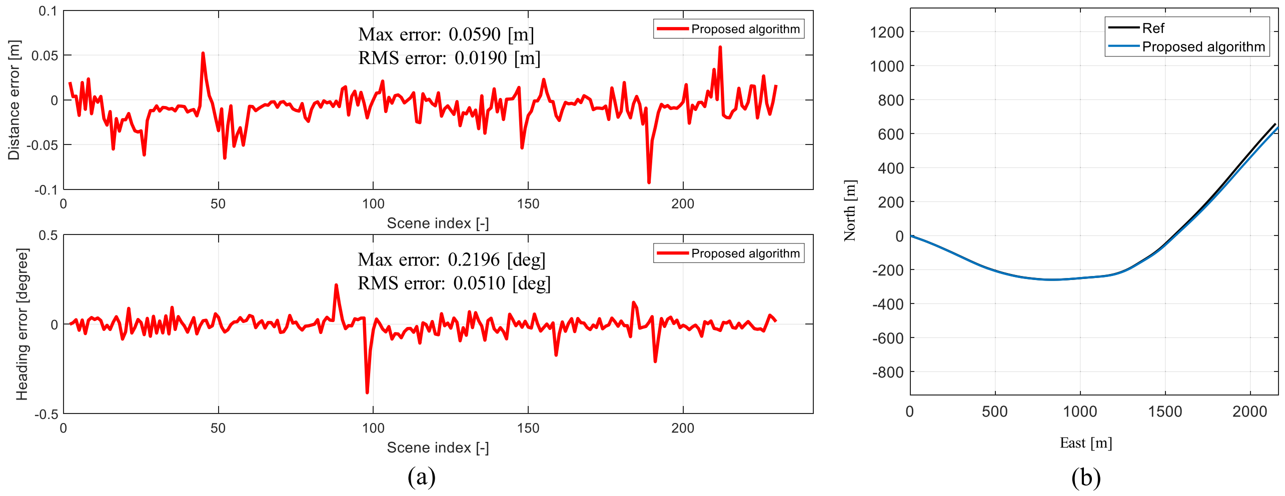


FIGURE 8. (a) Distance and heading error of the estimated odometry based on the proposed algorithm in a static environment (b) Comparison between the reference pose from RTK-GPS and the estimated pose from the proposed algorithm.

TABLE 1. RMS and maximum error of the proposed algorithm and the other dynamic object removal algorithms.

Error type	Distance error [m]		Heading error [deg]	
	RMS	Max	RMS	Max
Proposed algorithm	**0.0135**	**0.0365**	**0.0304**	**0.0973**
Non-filtering	0.3143	0.9221	1.1100	3.5873
Tracking	0.1244	0.2165	0.3597	0.1805
Occupancy grid map	0.0543	1.4866	0.0453	1.0320

heading graph for the entire area. The proposed algorithm, represented by the solid red line, has a lower error level than the other algorithms. The results of vehicle position estimation by accumulating scan-matching results are shown in Figure 10-(b). Because the proposed algorithm has a lower level of peak error, the estimated position of the proposed algorithm is closer to the reference position than those of the other three algorithms. RMS and maximum values of matching errors are summarized in Table 1. Numerically, it can be confirmed that the proposed algorithm has a lower matching error compared with the other algorithms.

D. HOW ACCURATELY THE POINT CLOUD MAP BASED ON THE SENSOR ODOMETRY ARE GENERATED

To evaluate the performance of the point cloud map generated through the proposed algorithm, the closest distance from the reference map was used as an evaluation index. The test site was used twice, i.e., when there were no dynamic objects, and when there were many dynamic objects. The two sets of driving data were used to produce a point cloud map using only high-precision GNSS. To produce the reference map, the dynamic objects were manually removed from the point cloud maps. The point cloud maps for performance evaluation were produced under three conditions. The first is the map in the environment with no dynamic objects. The second is the map in the environment with many dynamic objects, which were not filtered. The third is the map in the environment with many dynamic objects, which were considered based on static probability. In the three cases, a point cloud map was generated based on scan matching and low-cost GNSS. In the proposed static probability-based algorithm, only points with a static probability of 0.5 or higher in the point cloud map were considered as static points. To calculate the

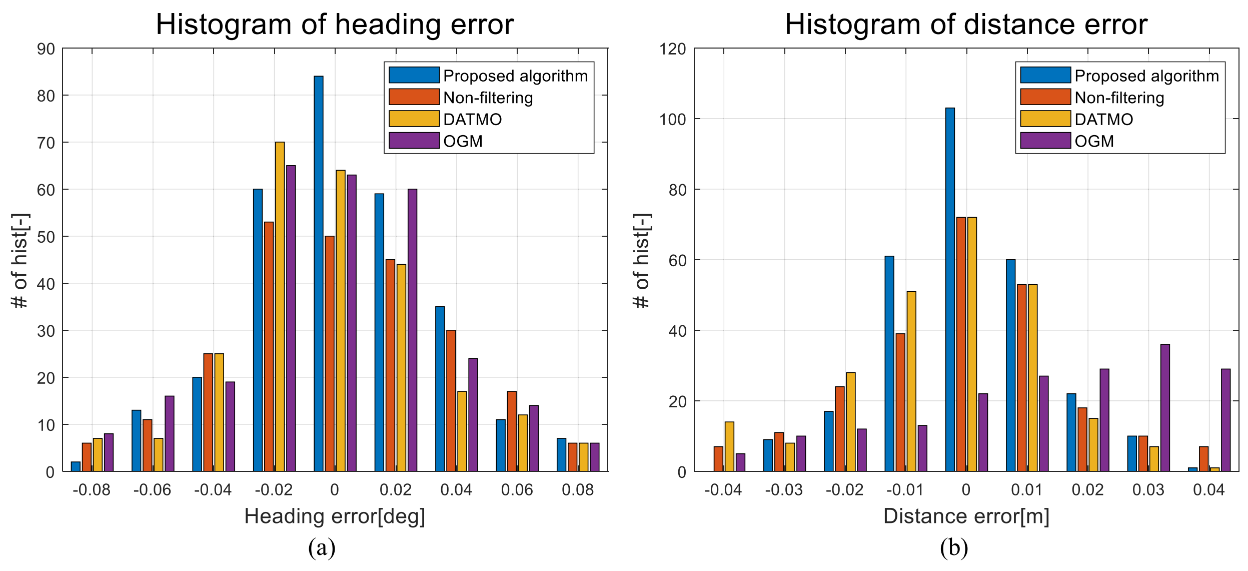


FIGURE 9. Error histograms of heading and distance of various algorithms.

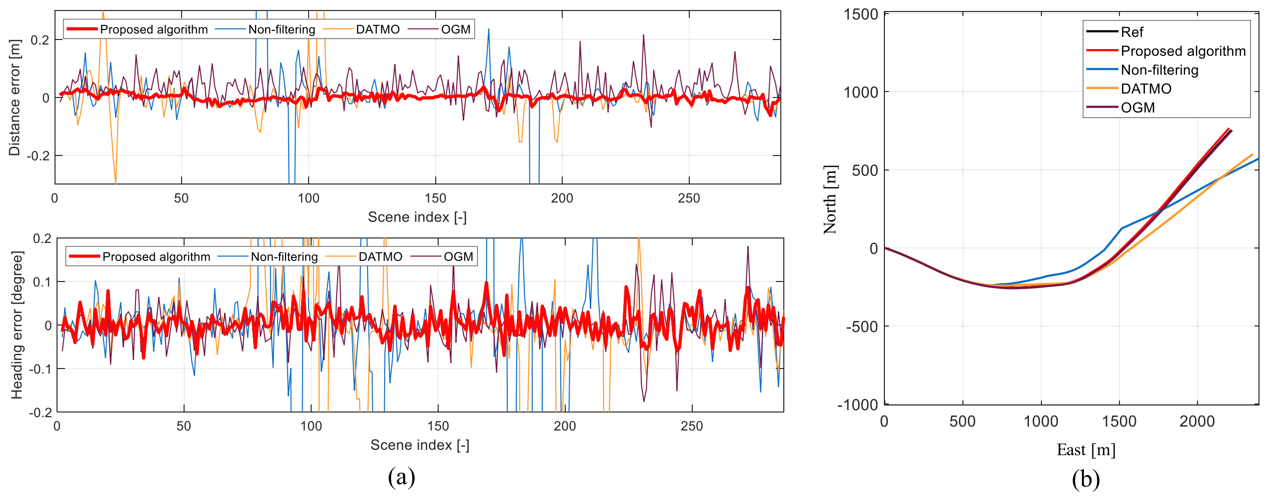


FIGURE 10. (a) Distance and heading error of the estimated odometry of various algorithms in a dynamic environment (b) Comparison between the reference pose from RTK-GPS and the estimated pose from various algorithms.

correspondence between the reference map and the map to be evaluated, $1/1000{*}N$ among the N points included in the map to be evaluated was randomly extracted. For each extracted point, the closest corresponding point in the reference map was found. The distance between all corresponding points was calculated, and their RMS errors were compared.

The results confirm that the map generated by the proposed algorithm has RMS nearest distance of 0.5777 m. The map is twice as accurate as the map without dynamic object filtering, which has RMS nearest distance of 1.1643 m. In addition, the proposed algorithm provides more accurate results than the RMS nearest distance of 0.7444 m, which is the result of an environment without a dynamic object. There are two reasons for this result. First, the effect of the greenbelt created in the road environment can degrade the scan-matching performance. The location of the greenbelt can be changed by the wind, and it also causes scattering of LiDAR points. However, in the proposed algorithm, these green spaces have a weak effect on scan-matching performance by giving a low static probability compared with previously acquired point clouds. Second, it was confirmed that the error of the low-cost

TABLE 2. RMS distance between reference map and the generated map in various conditions.

Error type		Nearest point distance [m]
		RMS
Static environment	Non-filtering	0.7444m
Dynamic environment	Proposed algorithm	**0.5777m**
	Non-filtering	1.1643m

GPS data acquired under two different driving conditions, i.e., with and without a dynamic object, influenced the result to a certain extent.

VIII. CONCLUSION

This article proposed a 3D point cloud mapping algorithm using a static probability-based NDT algorithm in a dynamic environment. To generate an accurate point cloud map in a dynamic environment, the effect of dynamic objects on scan matching must be eliminated. For this reason, we estimated the static probability of each point and performed scan matching. The calculated odometry from scan matching was used as a constraint of point cloud mapping. These three steps were evaluated by comparison with other dynamic object removal algorithms.

First, the static probability estimation algorithm was compared with a tracking algorithm. Based on the predicted and actual class of each point, we generated the confusion matrix of each algorithm. The static probability estimation algorithm has an accuracy of 90.0%, 82.1%, and 94.8%, static recall, and nonstatic recall. On the other hand, the tracking algorithm had an accuracy of 78.2%, 71.5%, and 84.8%, static recall, and nonstatic recall. The proposed estimation algorithm had higher accuracy and recall than the tracking algorithm. Second, the scan-matching performance of the proposed algorithm (Weighted NDT) was evaluated by comparison with other dynamic object removal algorithms (tracking, occupancy grid map). The distance and heading errors were represented in histograms, and there were more elements with near-zero errors in the proposed algorithm compared with the other algorithms. Furthermore, the proposed algorithm has 0.0135 m and 0.0304° of RMS distance and heading errors, respectively, making it much more accurate than the other algorithms. Third, we evaluated the accuracy of the point cloud map generated using the proposed algorithm in a dynamic environment. The RMS of the closest distance from reference map was calculated by both the proposed algorithm and nonfiltering algorithm. The RMS nearest distance error of the nonfiltering and proposed algorithm was 1.1643 m and 0.5777, respectively.

REFERENCES

[1] E. Javanmardi, M. Javanmardi, Y. Gu, and S. Kamijo, "Autonomous vehicle self-localization based on multilayer 2D vector map and multi-channel LiDAR," in *Proc. IEEE Intell. Vehicles Symp.*, no. 4, Jun. 2017, pp. 437–442.

[2] J. Li, H. Bao, X. Han, F. Pan, W. Pan, F. Zhang, and D. Wang, "Real-time self-driving car navigation and obstacle avoidance using mobile 3D laser scanner and GNSS," *Multimedia Tools Appl.*, vol. 76, no. 21, pp. 23017–23039, Nov. 2017.

[3] R. W. Wolcott and R. M. Eustice, "Visual localization within LIDAR maps for automated urban driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 176–183.

[4] P. de la Puente and D. Rodriguez-Losada, "Feature based graph-SLAM in structured environments," *Auto. Robots*, vol. 37, no. 3, pp. 243–260, Oct. 2014.

[5] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, 2010.

[6] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus, "Mapping with synthetic 2D LIDAR in 3D urban environment," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, vol. 3, no. 2, 2013, pp. 4715–4720.

[7] R. Mascaro, L. Teixeira, T. Hinzmann, R. Siegwart, and M. Chli, "GOMSF: Graph-optimization based multi-sensor fusion for robust UAV pose estimation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 644128, May 2018, pp. 1421–1428.

[8] X. I. Wong and M. Majji, "Extended Kalman filter for stereo vision-based localization and mapping applications," *J. Dyn. Syst., Meas., Control*, vol. 140, no. 3, Mar. 2018, Art. no. 030908.

[9] A. Diosi and L. Kleeman, "Laser scan matching in polar coordinates with application to SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 1439–1444.

[10] T. Kaminade, T. Takubo, Y. Mae, and T. Arai, "2P2-C14 the generation of environmental map based on a high resolutional NDT grid mapping," in *Proc. JSME Annu. Conf. Robot. Mechatronics (Robomec)*, vol. 2008, Jun. 2008, p. 2P2-C14_1, doi: 10.1299/jsmermd.2008._2P2-C14_1.

[11] R. Iser and F. M. Wahl, "Building local metrical and global topological maps using efficient scan matching approaches," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 1023–1030.

[12] W. Wen, L.-T. Hsu, and G. Zhang, "Performance analysis of NDT-based graph SLAM for autonomous vehicle in diverse typical driving scenarios of Hong Kong," *Sensors*, vol. 18, no. 11, p. 3928, Nov. 2018.

[13] M. Magnusson, "The three-dimensional normal-distributions transform: An efficient representation for registration, surface analysis, and loop detection," M.S. thesis, School Sci. Technol., Örebro Univ., Örebro, Sweden, 2009, no. 36.

[14] M. Magnusson, N. Vaskevicius, T. Stoyanov, K. Pathak, and A. Birk, "Beyond points: Evaluating recent 3D scan-matching algorithms," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2015, pp. 3631–3637.

[15] K. Jo, S. Cho, C. Kim, P. Resende, B. Bradai, F. Nashashibi, and M. Sunwoo, "Cloud update of tiled evidential occupancy grid maps for the multi-vehicle mapping," *Sensors*, vol. 18, no. 12, p. 4119, Nov. 2018.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics* (Intelligent Robotics and Autonomous Agents), vol. 45, no. 3. Cambridge, MA, USA: MIT Press, 2002.

[17] H. Pagac, D. Nebot, and E. M. Durrant-Whyte, "An evidential approach to probabilistic map-building," in *Proc. Int. Workshop Reasoning Uncertainty Robot.*, in Lecture Notes in Computer Science, 1996, pp. 164–170.

[18] K. Na, J. Byun, M. Roh, and B. Seo, "RoadPlot-DATMO: Moving object tracking and track fusion system using multiple sensors," in *Proc. Int. Conf. Connected Vehicles Expo (ICCVE)*, Oct. 2015, pp. 142–143.

[19] A. H. Abd Rahman, H. Zamzuri, S. A. Mazlan, and M. A. A. Rahman, "Model-based detection and tracking of single moving object using laser range finder," in *Proc. 5th Int. Conf. Intell. Syst., Model. Simul.*, Jan. 2014, pp. 556–561.

[20] D. Kim, K. Jo, M. Lee, and M. Sunwoo, "L-shape model switching-based precise motion tracking of moving vehicles using laser scanners," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 598–612, Feb. 2018.

[21] O. Bengtsson and A.-J. Baerveldt, "Robot localization based on scan-matching—Estimating the covariance matrix for the IDC algorithm," *Robot. Auto. Syst.*, vol. 44, no. 1, pp. 29–40, Jul. 2003.

[22] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Proc. Sensor Fusion 4th Control Paradigms Data Struct.*, Apr. 1992, pp. 239–256.

[23] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Oct. 2003, pp. 2743–2748.

[24] A. Censi, "An ICP variant using a point-to-line metric," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 19–25.

[25] S. Segal, A. Haehnel, and D. Thrun, "Generalized-ICP," *Robot., Sci. Syst.*, vol. 2, p. 4, p. 435, 2009.

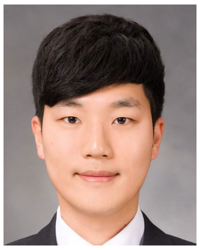
[26] F. A. Donoso, K. J. Austin, and P. R. McAree, "How do ICP variants perform when used for scan matching terrain point clouds?" *Robot. Auto. Syst.*, vol. 87, pp. 147–161, Jan. 2017, doi: 10.1016/j.robot.2016.10.011.

[27] M. Magnusson, N. Vaskevicius, T. Stoyanov, K. Pathak, and A. Birk, "Beyond points: Evaluating recent 3D scan-matching algorithms," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 3631–3637.

[28] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[29] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2O: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3607–3613.

SUMYEONG LEE (Student Member, IEEE) received the M.S. degree in automotive engineering from the Automotive Control and Electronics Laboratory (ACE Laboratory), Hanyang University, Seoul, South Korea, in 2019. His research interests include real-time embedded systems for autonomous localization and mapping and simultaneous localization and mapping. His current research interest includes precise map generation for an autonomous car.



CHANSOO KIM (Member, IEEE) received the B.S. degree in mechanical engineering from Hanyang University, Seoul, South Korea, where he is currently pursuing the Ph.D. degree with the Automotive Control and Electronics Laboratory (ACE Laboratory). His research interests include object detection and recognition, deep learning, localization, mapping, simultaneous localization and mapping, map updating, real-time systems for autonomous cars, and systems of autonomous vehicle. His current research interest includes precise map generation for an autonomous car.

SUNGJIN CHO (Student Member, IEEE) received the B.S. degree in electronic engineering from Hanyang University, Seoul, South Korea, in 2014, where he is currently pursuing the Ph.D. degree with the Automotive Control and Electronics Laboratory. His research interests include precise positioning and localization, information fusion theories, and real-time systems for an autonomous car. His current research interest includes the design of parallel computing architec tures for vehicle localization.

SUNWOO MYOUNGHO (Life Member, IEEE) received the B.S. degree in electrical engineering from Hanyang University, in 1979, the M.S. degree in electrical engineering from The University of Texas at Austin, in 1983, and the Ph.D. degree in system engineering from Oakland University, in 1990. He joined the General Motors Research (GMR) Laboratories, Warren, MI, in 1985, where he has worked in the areas of automotive electronics and control for 30 years. During his nine-year tenure at GMR, he worked on the design and development of various electronic control systems for powertrains and chassis. Since 1993, he has been leading research activities as a Professor with the Department of Automotive Engineering, Hanyang University. His research interest includes automotive electronics and controls, such as modeling and control of internal combustion engines, design of automotive distributed real-time control systems, intelligent autonomous vehicles, and automotive education programs.

KICHUN JO (Member, IEEE) received the B.S. degree in mechanical engineering and the Ph.D. degree in automotive engineering from Hanyang University, Seoul, South Korea, in 2008 and 2014, respectively. From 2014 to 2015, he was with the ACE Laboratory, Department of Automotive Engineering, Hanyang University, where he is doing research on system design and implementation of autonomous cars. From 2015 to 2018, he was with the Driving Assistance Research, Valeo, Bobigny, France, where he is working on the highly automated driving. He is currently an Assistant Professor with the Department of Smart Vehicle Engineering, Konkuk University, Seoul. His current research interests include localization and mapping, objects tracking, information fusion, vehicle state estimation, behavior planning, and vehicle motion control for highly automated vehicles.