2020 IEEE Intelligent Vehicles Symposium (IV)
October 20-23, 2020. Las Vegas, USA

3D Monte Carlo Localization with Efficient Distance Field Representation for Automated Driving in Dynamic Environments

Naoki Akai[1], Takatsugu Hirayama[2], and Hiroshi Murase[1]

Abstract— **This paper presents a LiDAR-based 3D Monte Carlo localization (MCL) with an efficient distance field (DF) representation method. To implement 3D MCL, high computing capacity is required because the likelihood of many pose candidates, i.e., particles, must be calculated in real time by comparing sensor measurements and a map. Additionally, a large-scale map is needed for allocation to embedded computers since autonomous vehicles are required to navigate wide areas. These make it difficult for 3D MCL implementation. This paper first presents an efficient DF representation method while considering the 3D LiDAR-based localization characteristics. Because each DF voxel has the closest distance from occupied voxels, swift comparison of the sensor measurements and map can be achieved. Consequently, 3D MCL using the likelihood field model (LFM) can be executed in real time. Furthermore, this paper presents a method for improving the localization robustness to environmental changes without increasing memory and computational cost from that of the LFM-based MCL. Through experiments using the SemanticKITTI dataset, we show that the presented method can efficiently and robustly work in dynamic environments.**

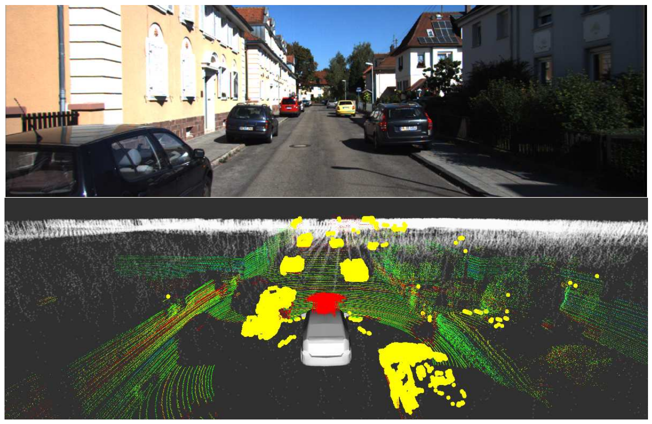
Fig. 1. An example of an estimation by the presented method. The red arrows and yellow points depict the particles and unmapped obstacles; exactly categorizing the cars as unmapped obstacles.

I. Introduction

An autonomous vehicle needs to localize its own 6DoF pose exactly for achievement of precise automated driving. The Bayes filter-based localization methods, in particular Monte Carlo localization (MCL) [1], [2], are popular since these methods are robust and can be easily implemented. To implement 3D LiDAR-based MCL, a large computing capacity is required because the likelihood of many pose candidates, i.e., particles, must be calculated in real time by comparing sensor measurements to a map. Additionally, a large-scale map must be allocated to embedded computers since autonomous vehicles must navigate wide areas. These requirements make 3D MCL implementation difficult.

Almost all recent 3D localization methods used for automated driving are based on the optimization approaches [3], [4] like the iterative closest point (ICP) [5] or normal distributions transform (NDT) [6]. These optimization approaches, particularly the NDT-based methods, enable registration of 3D LiDAR measurements to a map in real time. However, they are not suitable for automated driving in terms of safety because prior distribution estimated using motion sensors, e.g., odometry, cannot be considered. In other words, the optimization approaches are weak to noise and sometimes drastically fail its estimate.

[1]Naoki Akai and Hiroshi Murase are with the Graduate School of Informatics, Nagoya University, Nagoya 464-8603, Japan

[2]Takatsugu Hirayama is with the Institute of Innovation for Future Society (MIRAI), Nagoya University, Nagoya 464-8601, Japan

E-mail of the corresponding author `akai@nagoya-u.jp`

This paper first presents an efficient distance field (DF) representation method for real-time 3D MCL implementation. The DF is a voxel map and each voxel contains the closest distance from occupied voxels [7]. The likelihood field model (LFM) [2] can be efficiently calculated using the DF.

For a straightforward DF implementation, a voxel map memory that completely covers a target environment is required. However, it is almost impossible to implement a simple 3D DF since it requires an extremely large memory cost. For example, to model $2000\ \mathrm{m} \times 2000\ \mathrm{m} \times 40\ \mathrm{m}$ size environment with $0.1\ \mathrm{m}$ resolution, 64 GB memory usage is required if each voxel has a 4 byte representation range.

In this work, we introduce two assumptions by considering the characteristics of the 3D LiDAR-based localization to efficiently represent the 3D DF. The assumptions are (1) the voxels do not need to contain a large distance value, and (2) most areas do not require modeling since there are many areas that do not contain obstacles. Based on these assumptions, we achieve the efficient 3D DF representation. Through experiments, we show that these assumptions do not cause fatal problems.

This paper also presents a method for improving localization robustness. The LFM is efficient, however it does not consider any environmental changes. As a result, the LFM-based MCL is weak to environmental variations. The presented method increases localization robustness by simultaneously estimating measurement classes while localizing a vehicle pose. Where "measurement classes" categorize sensor measurements as those obtained from mapped or unmapped obstacles as shown in Fig. 1. Additionally, the simultaneous estimation can be achieved without increasing memory and computational costs from that of the LFM-based MCL.

The presented method was originally proposed in [8], [9]. However, the method was only applied to the 2D LiDAR-based localization problem. This work extends the previous work and applies the method to the 3D LiDAR localization problem. Through experiments using the SemanticKITTI dataset [10], we show that the presented method efficiently and robustly works in dynamic environments.

The contribution of this work is two-fold.

- Presenting the efficient 3D DF representation method
- Presenting the simultaneous localization and measurement class estimation method in the 3D LiDAR-based localization problem

The rest of this paper is organized as follows. Section II summarizes related work. Section III describes the efficient 3D DF representation method. Section IV describes the simultaneous localization and measurement class estimation method. Section V details the implementation methodology using the SemanticKITTI dataset. Section VI shows the presented method performance. Section VII concludes this work.

II. Related work

Occupancy grid (or voxel) mapping is widely used for modeling an environment [2]. However, straightforward environment modeling using a 3D voxel map quickly outgrows the memory limitations of modern computers. 3D map representation should be innovated for modeling a large-scale environment.

An elevation map models an environment using a 2D grid map with height values with the assumption that a ground surface can be represented using a single surface [11]. Multi-level surface maps relax the assumption [12]. Multi-volume occupancy grid maps model an environment using a list of occupied height areas and one of free height areas [13]. Recently in the robotics field, OctoMap, based on Octrees [14], is widely used since it allows efficient modeling of an environment by hierarchically partitioning a space [15]. In [16], particle filter-based SLAM with octree data structure is presented. However, these environment modeling methods do not contribute to efficient DF representation.

The ICP can be applied if an environment is modeled by point cloud. However, the process to find corresponding points between map and measurement points, i.e., nearest neighbor search (NNS), is time consuming. Although accelerated NNS methods, e.g., approximate NNS [17], [18], were proposed, using the NNS for solving the on-line localization problem is not practical since much of the NNS process has to be executed in real time.

The NDT scan matching was proposed in [6], [19]. Although the basic NDT algorithm is similar to the ICP, it enables efficient search of the corresponding pairs between source and target point clouds by approximately representing point clouds with normal distribution. Additionally, it was presented that the NDT is fast and has a wider valley of convergence, more so than the ICP in [20]. Automated driving demonstrations have been conducted using the NDT-based localization [3].

Because the NDT is based on the optimization approach, it cannot approximate its estimation uncertainty. In [21], [22], the NDT is extended to the particle filter-based localization. We also presented probabilistic localization methods using the NDT [4], [23]. Although such probabilistic approaches improve the localization robustness since these methods can handle unconfident localization results, the NDT-based methods still suffer from the non-smooth likelihood distribution due to discrete representation of NDT maps [9]. In [24], an NDT-based occupancy mapping method, called NDT occupancy mapping, is presented. The NDT maps also achieve higher accuracy at the cost of increased memory utilization.

In [25], a map representation based on the decay rate model, called DCT maps [26], is presented. The DCT maps store the map parameters in the discrete frequency domain and the continuous extension of the inverse discrete cosine transform is used to convert the map parameters in the spacial domain. In [25], the authors argued that the DCT maps accurately model the environment more than state-of-the-art mapping methods that do not require specific map resolution, called Hilbert maps [27] and Gaussian process occupancy maps [28]. Moreover, the DCT maps allow inferring the measurement probabilities of the forward sensor model in closed form. However, these map representation methods have not been applied to represent DFs.

In [29], efficient incremental map updating methods for maps suited for robotics applications including DFs are presented. [30] presented a combined map representation method using truncated and Euclidean signed DFs and the authors argued that the map representation is better-suited for robotic applications. A localization method using vector DFs that contain a direction toward the closest occupied cell is presented in [31]. Adaptively sampled DFs that adaptively divide a space to efficiently represent the DFs are presented in [32]. The idea used in this work is similar to the method presented in [32]. However, we utilize this representation to solve the 3D localization problem in large-scale environments.

III. Efficient 3D distance field representation

This section describes an efficient DF representation method. To achieve efficient representation, we use two assumptions that are (1) the voxels do not need to contain a large distance value, and (2) most areas do not need modeling since there are many areas with no obstacles.

Figure 2 illustrates an overview of the 3D DF representation. An environment is first modeled using a voxel map with resolution r_M. Width, depth, and height sizes of the voxel map are denoted as W, D, and H, respectively. Each voxel contains a sub voxel map that represents an actual DF.

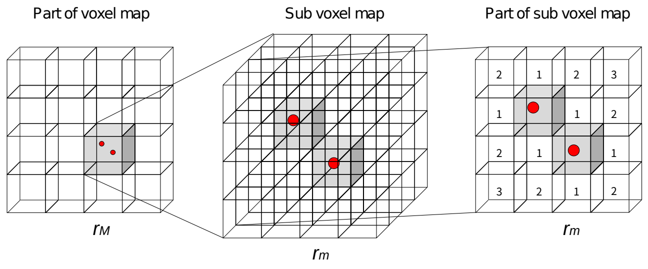

Fig. 2. The 3D DF representation used. The red points and gray voxels are map points and occupied voxels. r_M and r_m ($r_M > r_m$) are resolutions of the voxel and sub voxel maps.

Algorithm 1 Get the distance from the 3D DF

```
Input: p: 3D point on the map coordinates
Output: d: Distance from the point to the nearest occupied
    voxels containing the map point
    $i_{\mathrm{VM}} = \mathrm{getVoxelMapIndex}(\mathbf{p})$
    $i_{\mathrm{SVM}} = \mathrm{getSubVoxelMapIndex}(\mathbf{p})$
    $\mathrm{distanceList} = \mathrm{getDistanceList}(i_{\mathrm{VM}})$
    $d = \mathrm{distanceList}(i_{\mathrm{SVM}})$
    return $d$
```

We assume that the environment is modeled by a point cloud and refer it to *map points*; plotted to the corresponding sub voxel maps. If a point is included in a sub voxel map, memory for the sub voxel map is allocated. Because voxels containing no points are unallocated, an environment can be efficiently modeled.

An example of the actual DF is shown to the right in Fig. 2. The voxels of the sub voxel map each contain an unsigned single byte of data. A list variable containing corresponding distances to the unsigned data is also implemented for each sub voxel map. As a result, the closest distance can be efficiently obtained just by finding a corresponding voxel in the sub voxel map.

Algorithm 1 shows the pseudo code to get the distance from the 3D DF. A 3D point on the map coordinates, $\mathbf{p}$, is input and corresponding indices of the voxel and sub voxel maps, i_{VM} and i_{SVM}, are first computed. Then, the list variable corresponding to the i_{VM}-th sub voxel map is obtained. Finally, the closest distance, d, can be obtained from the corresponding list.

Because the sub voxel maps contain the unsigned single byte of data, the DF represents 256 value types. One value must correspond to an invalid value to handle invalid localization data. Hence, the DF is presented by 255 tones. If the resolution r_m is set to 0.1 m, the DF can represent up to about 2.8 m. This value is adequate since a measurement valiance of LiDAR is sufficiently smaller than the maximum distance. This is detailed in Section VI-B.1.

IV. Improving localization robustness

A. General localization problem

The general probabilistic localization problem described in [2] is formulated as the posterior estimation problem:

$$p(\mathbf{x}_t|\mathbf{u}_{1:t},\mathbf{z}_{1:t},\mathbf{m}) = \eta p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}) \int p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t-1},\mathbf{z}_{1:t-1},\mathbf{m})d\mathbf{x}_{t-1}, \tag{1}$$

where $\mathbf{x}$ is a pose, $\mathbf{u}$ is a control input, $\mathbf{z}$ are sensor measurements, $\mathbf{m}$ is a map, η is a normalization constant, and t and $1:t$ represent current and time sequence data. $p(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m})$ and $p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)$ are referenced as measurement and motion models, respectively [2]. When the LFM is used, the measurement model is denoted as:

$$p_{\mathrm{LFM}}(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}) = \begin{pmatrix} z_{\mathrm{hit}} \\ z_{\mathrm{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\mathrm{hit}}(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}) \\ p_{\mathrm{rand}}(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}) \end{pmatrix}, \tag{2}$$

where z_{hit} and z_{rand} are arbitrary constants satisfying $z_{\mathrm{hit}} + z_{\mathrm{rand}} = 1$, and $p_{\mathrm{hit}}(\cdot)$ and $p_{\mathrm{rand}}(\cdot)$ model things related to the measurement of the mapped obstacles and random noise, respectively[1].

$p_{\mathrm{hit}}(\cdot)$ and $p_{\mathrm{rand}}(\cdot)$ are respectively denoted as:

$$p_{\mathrm{hit}}(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{d(\mathbf{z}_t,\mathbf{x}_t,\mathbf{m})^2}{2\sigma^2}\right), \tag{3}$$

$$p_{\mathrm{rand}}(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}) = \mathrm{unif}(0,R), \tag{4}$$

where σ^2 is a measurement variance, R is the maximum measurement range, $d(\cdot)$ returns the closest distance from a scan point which is transformed based on the given pose, $\mathbf{x}_t$, to obstacles existing on the map, $\mathbf{m}$, and $\mathrm{unif}(\cdot)$ is a uniform distribution within a given range.

B. Simultaneous localization and measurement class estimation

As shown in equation (2), the LFM does not consider any environmental changes. The simultaneous localization and class estimation method presented in [8], [9] introduces class of sensor measurements, $\mathbf{c}$, as a hidden variable. Where "class" categorizes the sensor measurements as those obtained from mapped or unmapped obstacles. In the model, we assume that the sensor measurements depend on the measurement classes.

The method estimates the following joint posterior distribution:

$$p(\mathbf{x}_t,\mathbf{c}_t|\mathbf{u}_{1:t},\mathbf{z}_{1:t},\mathbf{m}). \tag{5}$$

This joint distribution is factorized using the multiplication theorem as:

$$p(\mathbf{x}_t|\mathbf{u}_{1:t},\mathbf{z}_{1:t},\mathbf{m})p(\mathbf{c}_t|\mathbf{x}_t,\mathbf{u}_{1:t},\mathbf{z}_{1:t},\mathbf{m}). \tag{6}$$

[1]In [2], a thing that the measurement is to be the maximum value is considered, however we do not consider the thing since a 3D point cloud does not usually have such measurements.

The first term of equation (6) can be deformed to the same equation shown in equation (1). However, the measurement model can be re-written using the law of total probability as:

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}) = \sum_{\mathbf{c}_t \in C} p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{c}_t, \mathbf{m}) p(\mathbf{c}_t), \quad (7)$$

where C is a list of the measurement classes and is denoted as $C = \{\text{mapped}, \text{unmapped}\}$. We refer $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{c}_t, \mathbf{m})$ to *class conditional measurement model (CCMM)* since the measurement classes are given as a condition.

The measurement model with the mapped condition is modeled using the LFM:

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{c}_t = \text{mapped}, \mathbf{m}) = p_{\text{LFM}}(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}). \quad (8)$$

The measurement model with the unmapped condition is modeled using the exponential distribution as:

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{c}_t = \text{unmapped}, \mathbf{m}) = \frac{\lambda \exp(-\lambda r_t)}{1 - \exp(-\lambda R)}, \quad (9)$$

where λ is the hyperparameter and r_t is the current measurement range. Both models can be calculated without complex processes. Therefore, the computational and memory costs are equivalent to those of the LFM.

Because the CCMM allows us to explicitly consider environmental changes via the unmapped class measurement model, localization robustness can be improved even though the LFM is used to model the mapped class measurement. However, this yields a trade-off problem between accuracy and robustness. We discuss this problem in Section VI-C.

V. Implementation

In this work, we used the SemanticKITTI dataset [10] that provides semantic labels of the 3D LiDAR measurements included in the KITTI odometry dataset [33] to evaluate the presented method. This section describes the methodology of implementation and evaluation using this dataset.

A. Building 3D distance field

The SemanticKITTI dataset contains annotated LiDAR measurements and the ground truth vehicle trajectories. The object classes include static and dynamic classes, e.g., person, bicycle, road, and building. We plot only measurements obtained from the static objects according to the ground truth trajectory and build the map points. Note that we assumed that the ground surface is flat. The voxel grid filter with a 0.1 m resolution is applied to the map points to remove unnecessary redundant points.

The 3D DF is built based on the map points. The resolutions of voxel and sub voxel maps were set as: $r_M = 5$ m and $r_m = 0.1$ m. To build the DFs, we used the algorithm described in [34].

B. Control input

We simulate a noisy control input, denoted as $\mathbf{u}_t = (\Delta d_t, \Delta\theta_t)$, using the ground truth trajectory. Movement amounts according to xy plane, Δd_t^{GT}, and yaw axis, $\Delta\theta_t^{\text{GT}}$, are first calculated using the ground truth. Then, the noisy control input is simulated as:

$$\Delta d_t \sim \mathcal{N}(\gamma_{\Delta d} \Delta d_t^{\text{GT}}, \sigma_{\Delta d}^2), \quad (10)$$

$$\Delta\theta_t \sim \mathcal{N}(\gamma_{\Delta\theta} \Delta\theta_t^{\text{GT}}, \sigma_{\Delta\theta}^2), \quad (11)$$

where $\mathcal{N}(a, b^2)$ is Gaussian with mean a and variance b^2, and γ and σ^2 are arbitrary constants to add the noise. These parameters were set as: $\gamma_{\Delta d} = 0.97$, $\gamma_{\Delta\theta} = 1.03$, $\sigma_{\Delta d}^2 = 0.1\ \text{m}^2$, and $\sigma_{\Delta\theta}^2 = 1.72\ \text{degree}^2$.

C. Motion model

The vehicle pose, $\mathbf{x}$, is composed of a 3D point, x, y, and z, and roll, pitch, and yaw angles, ϕ, ψ, and θ. The state of the particles is also composed of these variables. To update the pose of the particles, the following motion model is used:

$$\begin{pmatrix} x_t \\ y_t \\ z_t \\ \phi_t \\ \psi_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ z_{t-1} \\ \phi_{t-1} \\ \psi_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} \Delta\hat{d}_t \cos\theta_{t-1} \\ \Delta\hat{d}_t \sin\theta_{t-1} \\ \Delta\hat{z}_t \\ \Delta\hat{\phi}_t \\ \Delta\hat{\psi}_t \\ \Delta\hat{\theta}_t \end{pmatrix}, \quad (12)$$

$$\begin{aligned} \Delta\hat{d}_t &\sim \mathcal{N}(\Delta d_t, \alpha_1 \Delta d_t^2 + \alpha_2 \Delta\theta_t^2), \\ \Delta\hat{z}_t &\sim \mathcal{N}(0, \alpha_3 \Delta d_t^2 + \alpha_4 \Delta\theta_t^2), \\ \Delta\hat{\phi}_t &\sim \mathcal{N}(0, \alpha_5 \Delta d_t^2 + \alpha_6 \Delta\theta_t^2), \\ \Delta\hat{\psi}_t &\sim \mathcal{N}(0, \alpha_7 \Delta d_t^2 + \alpha_8 \Delta\theta_t^2), \\ \Delta\hat{\theta}_t &\sim \mathcal{N}(\Delta\theta_t, \alpha_9 \Delta d_t^2 + \alpha_{10} \Delta\theta_t^2), \end{aligned} \quad (13)$$

where $\alpha_1 \sim \alpha_{10}$ are arbitrary constants. These constant values were experimentally determined.

D. Likelihood calculation

We first assume that each element of the sensor measurements, $\mathbf{z} = (\mathbf{z}^1, \mathbf{z}^2, ..., \mathbf{z}^K)$, are independent, where K is the number of the sensor measurements. The sensor measurement classes are denoted as $\mathbf{c} = (c^1, c^2, ..., c^K)$, where c^k is corresponding class to $\mathbf{z}^k$. By assuming the independence, the CCMM is re-written as:

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{c}_t, \mathbf{m}) = \prod_{k=1}^{K} p(\mathbf{z}_t^k|\mathbf{x}_t, c_t^k, \mathbf{m}). \quad (14)$$

From equation (7), the likelihood of the i-th particle is calculated as:

$$\omega_t^i = \prod_{k=1}^{K} \sum_{c_t \in C} p(\mathbf{z}_t^k|\mathbf{x}_t, c_t^k, \mathbf{m}) p(c_t^k). \quad (15)$$

In equation (15), we need to define prior probability over c_t^k, however no conditions are given to estimate the prior. Thus, we assume that the prior probability is uniform, namely $p(c_t^k = \text{mapped}) = p(c_t^k = \text{unmapped}) = 0.5$. The parameters shown in equations (2), (3) and (9) were set as: $z_{\text{hit}} = 0.95$, $z_{\text{rand}} = 0.05$, $\sigma^2 = 0.01\ \text{m}^2$, $\lambda = 0.03$, and $R = 120$ m.

The likelihood calculation process can easily be parallelized. In implementation, we parallelize this process to four threads using OpenMP [35].

E. Pose estimation and re-sampling

After calculating the particles' likelihood, they are first normalized. Then, the current vehicle pose is estimated as the weighted average of the particles' state. The estimated errors are computed by comparing the estimated pose and the ground truth.

Before re-sampling the particles, the effective sample size (ESS) [36] is calculated as:

$$\mathrm{ESS} = \frac{1}{\sum_{i=1}^{M} \left(\omega_t^i\right)^2}, \tag{16}$$

where M is the number of the particles and was set to $M = 1000$. If ESS $< M/2$, then re-sampling is performed.

When the re-sampling is performed, 10 $\%$ random particles are injected and denoted as:

$$\mathbf{x}_t^i \sim \mathcal{N}(\mathbf{x}_t, \Sigma_{\mathrm{rand}}), \tag{17}$$

where $\mathbf{x}_t$ is the current estimated pose and $\Sigma_{\mathrm{rand}} = \mathrm{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\phi^2, \sigma_\psi^2, \sigma_\theta^2)$. These variables were also experimentally determined. This random sampling injection increases localization robustness to failures due to inaccurate motion modeling.

F. Measurement class estimation and evaluation

The simultaneous estimation method presented in this paper estimates not only the vehicle pose but also the sensor measurement classes. The posterior distribution over the measurement classes is denoted as:

$$p(\mathbf{c}_t | \mathbf{x}_t, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{m}) = \eta p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{c}_t, \mathbf{m}) p(\mathbf{c}_t) \tag{18}$$

With implementation, the measurement class is classified to unmapped if $p(c_t = \mathrm{unmapped}) > 0.9$. To evaluate the class estimation accuracy (CEA) measurement, the semantic labels included in the SemanticKITTI dataset are used. The CEA is denoted as:

$$\mathrm{CEA} = \frac{\sum_{k=1}^{K} \mathbb{1}(c^k = \hat{c}^k)}{K} \tag{19}$$

where $\mathbb{1}(\cdot)$ is an indicator function equal to one when the condition within the bracket is true, and zero otherwise, and c^k and $\hat{c}^k$ are the estimated and ground truth classes.

VI. Experiments

A. Comparison methods

We use two comparison methods; the LFM-based MCL and NDT.

1) LFM-based MCL: The LFM-based MCL estimates the posterior shown in equation (1). The LFM shown in equation (2) is used for calculating the particles' likelihood. The same parameters described in Section V are used.

2) NDT: The point cloud library [37] that implements the NDT presented in [19] is used. Resolution of the NDT map was set to 1 m. The voxel grid filter with 2 m resolution is applied to the sensor measurements before the registration. The initial pose for the registration is estimated using the motion model. If the NDT has not converged, the initial pose is used as the estimated pose.

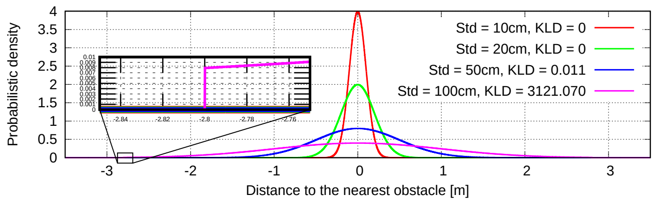

Fig. 3. Likelihood calculation examples of $p_{\mathrm{hit}}(\cdot)$ shown in equation (3) using the 3D DF.

B. Results

1) Likelihood calculation using 3D DF: Figure 3 shows likelihood calculation examples of $p_{\mathrm{hit}}(\cdot)$ shown in equation (3) using the 3D DF. Kullback-Leibler divergence (KLD) denoted as:

$$\mathrm{KLD} = \int p(d) \ln \frac{p(d)}{q(d)} \mathrm{d}d, \tag{20}$$

is calculated to evaluate the similarity of two distributions:

$$p(d) = \begin{cases} p_{\mathrm{hit}}(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}) & (d \le 2.8) \\ 10^{-30} & (\text{otherwise}), \end{cases} \tag{21}$$

$$q(d) = \max(p_{\mathrm{hit}}(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}), 10^{-30}), \tag{22}$$

where $p(\cdot)$ is the likelihood distribution calculated using the 3D DF. If the KLD value is quite small, the two distributions are regarded as almost the same.

KLD was zero when the measurement standard deviation, σ, was set to 0.1 m and 0.2 m. Standard deviation of typical LiDAR measurements is less than 0.2 m. Therefore, we conclude that the 3D DF has enough specification to accurately calculate the LFM.

2) Map memory usage: Table I shows memory usage in each map representation method and environment size. We chose seven sequences from the KITTI odometry dataset for the evaluation. It should be noted that the SemanticKITTI dataset only opens the semantic labels from the sequences 00 to 10, other labels are only used for public evaluation.

The map points correspond to a 3D point cloud map described in Section V-A. The presented distance field (PDF) described in Section III, NDT, and straightforward DF (SDF) map representations are compared. The SDF is a simple voxel map with a resolution of 0.1 m and floating point resolution.

A voxel of the NDT map contains mean, $\boldsymbol{\mu}_i \in \mathcal{R}^3$, and covariance matrix, $\Sigma_i \in \mathcal{R}^{3\times3}$. Because the covariance matrix is symmetric, it can be represented using six elements. Hence, one NDT voxel contains nine elements. To perform smooth registration, at least two NDT maps are needed and must be overlapped. Thus, memory usage for the two NDT maps is computed for the comparison.

The memory usage of the PDF and NDT is significantly smaller than that of the map points and SDF. In particular, with sequence 01 being the largest environment, the PDF could model it with very little memory usage. Figure 4 is the map points of sequence 01. To model such narrow

TABLE I

MAP MEMORY USAGES AND ENVIRONMENT SIZES IN GB AND METERS. THE PERCENTAGE SHOWN IN THE PDF ROW IS THE ACTIVE VOXEL RATE.

Sequence	01	03	04	06	07	09	10
Map points	1.34	1.14	0.09	0.22	0.13	2.23	1.74
PDF	0.15 (1.7 %)	0.04 (12.6 %)	0.02 (37.4 %)	0.02 (17.5 %)	0.04 (21.2 %)	0.09 (9.5 %)	0.04 (11.7 %)
NDT	0.70	0.03	0.01	0.01	0.02	0.07	0.03
SDF	39.53	1.49	0.31	0.69	0.92	4.09	1.60
Size	1908 x 1312 x 40	573 x 302 x 22	106 x 498 x 14	122 x 563 x 25	295 x 294 x 27	564 x 672 x 27	776 x 274 x 19

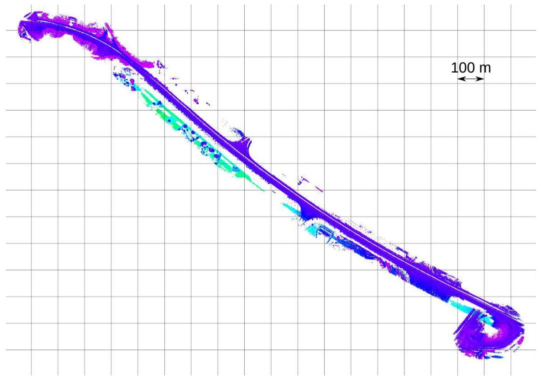

Fig. 4. The map points of the sequence 01.

wide environments, the presented 3D DF representation is effective.

Table I also shows the active sub voxel rate in the PDF row. The rate is denoted as N_{active}/WDH, where N_{active} is the number of active sub voxels that contains the map points. As can be seen from the Table I, the active sub voxel rate is quite small particularly in sequence 01. Based on the results shown in Section VI-B.1 and this section, we conclude that the assumptions used to efficiently represent the 3D DF do not cause fatal localization problems.

3) Position and angle estimation errors: Table II shows position and angle estimation errors by the presented method (PM) described in Section IV-B, LFM-based MCL (MCL), and NDT. The position and angle errors are denoted as: $\sqrt{\Delta x^2 + \Delta y^2}$ and $|\Delta\theta|$. This table also shows the static points rate (SPR) that is the rate of mapped points included in the 3D LiDAR measurements and the CEA by the PM.

The NDT was unusually accurate; however it was not robust to an inaccurate initial guess. Because the NDT is based on the optimization approach, it cannot consider the prior distribution over the initial pose estimated using the motion model like the Bayes filter. Consequently, the NDT sometimes yielded drastic estimation errors.

Overall, estimation accuracy by the MCL is the most accurate. However, the PM more accurately estimated the vehicle pose in sequences 06 and 07.

As can be seen from the average and minimum SPRs, sequences 06 and 07 contain more dynamic obstacles than other sequences. Figure 5 shows the position and angle estimation errors with the SPR and CEA in sequences 06 and 07. The bottom plots in Fig. 5 show errors in the control input, namely the moving amount differences computed between the ground truth and simulated noisy odometry.

Because the LFM does not consider any environmental changes, it could not cope with highly dynamic changes in the environment. Because the CCMM used in PM explicitly considers environmental changes, it coped well with the dynamics. However, the LFM resulted in more accurate estimation than PM in low dynamic environments. We discuss this reason in VI-C.

4) Computation time: Table III shows computation times for PM, MCL, and NDT. For the PM and MCL, time for the likelihood calculation and pose estimation was measured. For the NDT, time for the point cloud registration and pose estimation was measured. All processes were performed using a desktop computer (Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50 GHz).

The NDT computation time was usually enough for real-time localization, i.e., less than 100 msec. When the initial guess is accurate, the NDT achieved fast registration. However, the worst computation time was quite slow.

The MCL was a bit faster than the PM, however these computation times were almost the same. The PM is robust to environmental changes more than the MCL. Additionally, the PM and MCL map usage is identical since they use the same map. Therefore, we conclude that the PM is suitable for automated driving in dynamic environments.

C. Discussion

As we mentioned in V-D, the CCMM explicitly considers environmental changes. Because the measurement model with the unmapped class is modeled using exponential distribution up to the maximum measurement range, it considers the probability where obstacles are measured within the entire measurement range. Consequently, the CCMM has similar performance to the M-estimator utilized in the optimization approach to decrease outliers [38].

In the M-estimator, corresponding pairs between source and target point clouds with a large residual error are ignored in the cost function calculation to increase robustness to outliers. However, large estimation errors cannot be compensated since all correct pairs have a large residual error. Consequently, the trade-off problem between accuracy and robustness exists.

The CCMM also has this trade-off relationship. In the experiments, because we simulated noisy odometry as shown in the bottom of Fig. 5, the initial guess was sometimes inaccurate. As a result, the estimation accuracy using the CCMM was also inaccurate even though environmental changes are quite small.

TABLE II

POSITION / ANGLE ESTIMATION ERRORS BY PM, MCL, AND NDT, STATIC POINTS RATE (SPR), AND CLASS ESTIMATION ACCURACY (CEA). UNITS OF THE POSITION AND ANGLE ERRORS ARE METERS AND DEGREES.

	Sequence	01	03	04	06	07	09	10
PM	Ave	0.20 / 0.20	0.17 / 0.46	0.10 / 0.12	0.12 / 0.21	0.13 / 0.36	0.14 / 0.28	0.15 / 0.62
	Std	0.35 / 0.21	0.10 / 0.77	0.07 / 0.16	0.06 / 0.31	0.09 / 0.60	0.09 / 0.39	0.13 / 0.92
	Min	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.01 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
	Max	5.25 / 3.23	0.66 / 8.16	0.58 / 1.84	0.60 / 4.71	0.87 / 8.62	0.91 / 4.32	0.98 / 7.67
MCL	Ave	0.15 / 0.17	0.12 / 0.54	0.04 / 0.19	0.11 / 0.29	0.15 / 0.55	0.07 / 0.05	0.06 / 0.33
	Std	0.24 / 0.15	0.09 / 0.98	0.04 / 0.19	0.24 / 0.63	0.11 / 0.80	0.05 / 0.48	0.06 / 0.73
	Min	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
	Max	3.45 / 2.19	0.58 / 9.34	0.26 / 1.44	2.39 / 6.72	0.91 / 7.00	0.42 / 8.30	0.58 / 6.87
NDT	Ave	0.36 / 0.76	0.30 / 1.39	0.17 / 0.52	1.38 / 2.51	0.10 / 0.47	0.12 / 0.59	0.07 / 0.34
	Std	0.63 / 0.04	0.51 / 0.14	0.10 / 0.02	7.49 / 0.26	0.08 / 0.03	0.07 / 0.04	0.04 / 0.02
	Min	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
	Max	5.66 / 9.04	4.53 / 14.76	1.73 / 8.29	13.40 / 18.55	2.44 / 10.34	2.03 / 12.41	1.88 / 10.12
SPR	Ave	95.92 %	95.74 %	96.64 %	89.26 %	87.08 %	93.79 %	95.92 %
	Std	2.46 %	2.96 %	2.30 %	6.85 %	7.81 %	4.74 %	3.88 %
	Min	84.75 %	80.62 %	89.79 %	59.37 %	57.97 %	65.66 %	70.22 %
	Max	99.66 %	99.11 %	98.81 %	97.48 %	98.33 %	99.29 %	99.67 %
CEA	Ave	97.19 %	97.00 %	97.10 %	90.47 %	93.61 %	96.28 %	96.29 %
	Std	1.32 %	1.75 %	1.75 %	6.16 %	4.59 %	2.59 %	3.93 %
	Min	91.26 %	86.09 %	90.68 %	62.13 %	62.23 %	81.83 %	70.34 %
	Max	99.57 %	99.30 %	98.82 %	97.61 %	99.24 %	99.37 %	99.66 %

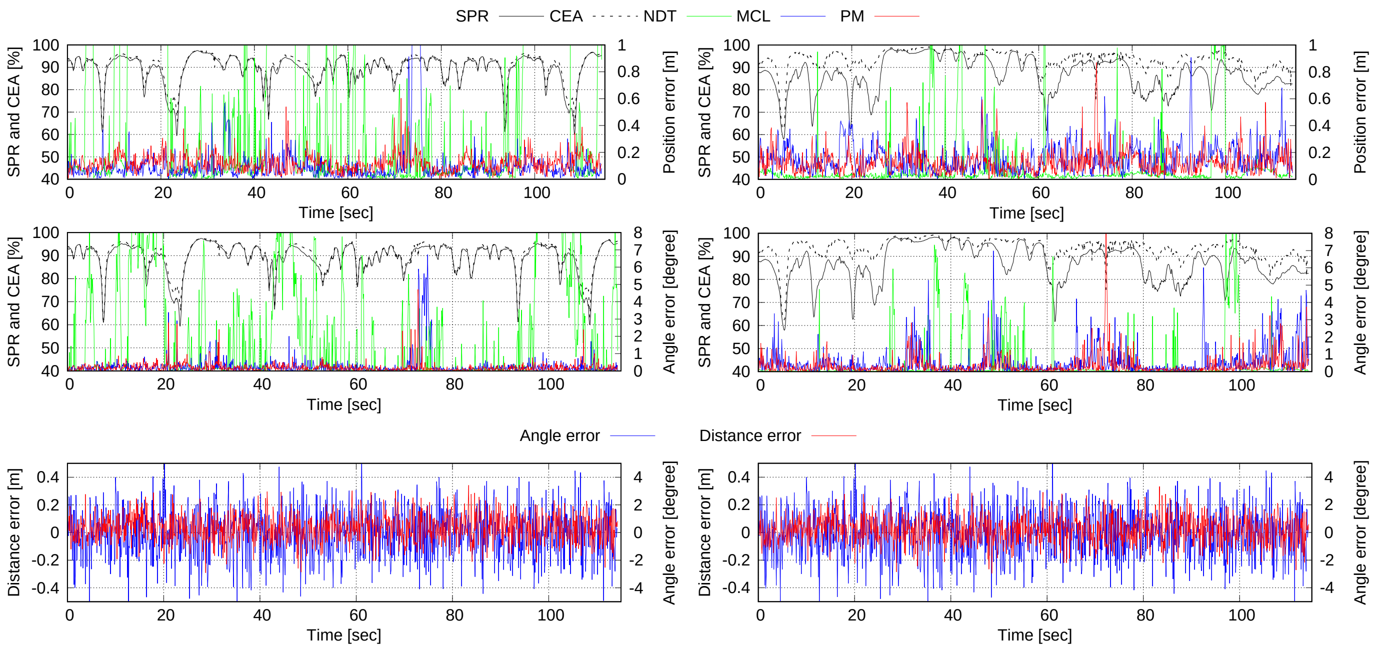

Fig. 5. Position (top) and angle (middle) estimation errors with SPR and CEA and control input errors (bottom) in sequences 06 (left) and 07 (right).

TABLE III

COMPUTATION TIMES IN MILLISECONDS.

	Ave	Std	Min	Max
PM	88.1	4.8	77	122
MCL	86.7	3.9	73	113
NDT	94.0	61.1	37	432

However, the robustness against environmental changes is significantly important for automated driving, in particular LiDAR field of view is limited as described in [9]. Thus, we consider that the CCMM is more effective for automated driving in dynamic environments.

VII. CONCLUSION

This paper has presented an efficient 3D DF representation method. Because the DF enables efficient LFM calculation, real-time 3D MCL could be implemented. Additionally, this paper has presented the robust likelihood calculation method that simultaneously estimates sensor measurement classes while localizing the vehicle pose. We evaluated the presented methods using the SemanticKITTI dataset and showed that the presented method efficiently and robustly works in dynamic environments.

Our future work is implementation of localization failure detection and reliability estimation methods presented in [39] and [40] using the efficient DF representation method.

ACKNOWLEDGMENT

This work was supported by the Center of Innovation Program (Nagoya-COI) funded by the JST Agency, JST-Mirai Program (Grant Number JPMJMI17C6), and KAKENHI under Grant 40786092.

REFERENCES

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. volume 2, pages 1322–1328, 1999.

[2] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[3] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[4] N. Akai, L. Y. Morales, T. Yamaguchi, E. Takeuchi, Y. Yoshihara, H. Okuda, T. Suzuki, and Y. Ninomiya. Autonomous driving based on accurate localization using multilayer LiDAR and dead reckoning. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 1147–1152, 2017.

[5] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[6] P. Biber and W. Strasser. The normal distributions transform: A new approach to laser scan matching. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2743–2748, 2003.

[7] P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248, 1980.

[8] N. Akai, L. Y. Morales, and H. Murase. Mobile robot localization considering class of sensor observations. In *Proceedings of the IEEE/RSJ Intelligent Robots and Systems*, pages 3159–3166, 2018.

[9] N. Akai, L. Y. Morales, T. Hirayama, and H. Murase. Toward localization-based automated driving in highly dynamic environments: Comparison and discussion of observation models. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 2215–2222, 2018.

[10] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[11] M. Herbert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade. Terrain mapping for a roving planetary explorer. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 997–1002, 1989.

[12] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2276–2282, 2006.

[13] I. Dryanovski, W. Morris, and J. Xiao. Multi-volume occupancy grids: An efficient probabilistic 3D mapping model for micro aerial vehicles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1553–1559, 20010.

[14] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

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

[16] N. Fairfield, G. Kantor, and D. Wettergreen. Real-time SLAM with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, 24(1–2):3–21, 2007.

[17] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications*, 2009.

[18] W. Li, Y. Zhang, Y. Sun, W. Wang, W. Zhang, and X. Lin. Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement (v1.0). *CoRR*, abs/1610.02455, 2016.

[19] M. Magnusson, A. Lilienthal, and T. Duckett. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827, 2007.

[20] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal, and J. Hertzberg. Evaluation of 3D registration reliability and speed - A comparison of ICP and NDT. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3907–3912, 2009.

[21] J. Saarinen, H. Andreasson, T. Stoyanov, and A. Lilienthal. Normal distributions transform Monte-Carlo localization (NDT-MCL). In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 382–389, 11 2013.

[22] R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto, and A. J. Lilienthal. Localization in highly dynamic environments using dual-timescale NDT-MCL. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3956–3962, 2014.

[23] N. Akai, L. Y. Morales, E. Takeuchi, Y. Yoshihara, and Y. Ninomiya. Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 1357–1364, 2017.

[24] J. Saarinen, H. Andreasson, T. Stoyanov, J. Ala-Luhtala, and A. J. Lilienthal. Normal distributions transform occupancy maps: Application to large-scale online 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2233–2238, 2013.

[25] A. Schaefer, L. Luft, and W. Burgard. DCT maps: Compact differentiable lidar maps based on the cosine transform. *IEEE Robotics and Automation Letters*, 3(2):1002–1009, 2018.

[26] A. Schaefer, L. Luft, and W. Burgard. An analytical lidar sensor model based on ray path information. *IEEE Robotics and Automation Letters*, 2(3):1405–1412, 2017.

[27] F. Ramos and L. Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35(14):1717–1730, 2016.

[28] M. G. Jadidi, J. V. Miró, and G. Dissanayake. Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Autonomous Robots*, 2017.

[29] B. Lau, C. Sprunk, and W. Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61(10):1116–1130, 2013.

[30] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart. Signed distance fields: A natural representation for both mapping and planning. In *Robotics: Science and Systems 2016 Workshop: Geometry and Beyond - Representations, Physics, and Scene Understanding for Robotics*, 2016.

[31] J. Arukgoda, R. Ranasinghe, L. Dantanarayana, G. Dissanayake, and T. Furukawa. Vector distance function based map representation for robot localization. In *Australasian Conference on Robotics and Automation*, 2018.

[32] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 249–254, 2000.

[33] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[34] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8(19), 2012.

[35] L. Dagum and R. Menon. OpenMP: An industry-standard API for shared-memory programming. *IEEE Computer Society Press*, 5(1):46–55, 1998.

[36] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transaction on Robotics*, 23(1):34–46, Feb. 2007.

[37] R.B. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[38] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2003.

[39] N. Akai, L. Y. Morales, T. Hirayama, and H. Murase. Misalignment recognition using Markov random fields with fully connected latent variables for detecting localization failures. *IEEE Robotics and Automation Letters*, 4(4):3955–3962, 2019.

[40] N. Akai, L. Y. Morales, and H. Murase. Simultaneous pose and reliability estimation using convolutional neural network and Rao-Blackwellized particle filter. *Advanced Robotics*, 32(17):930–944, 2018.