Master's Thesis
석사 학위논문

An Algorithm for Local Dynamic Map Generation for Safe UAV Navigation

Jin-Woo Lee(이 진 우 李 鎭 宇)

Department of
Information and Communication Engineering

DGIST

2021

Master's Thesis
석사 학위논문

An Algorithm for Local Dynamic Map Generation for Safe UAV Navigation

Jin-Woo Lee(이 진 우 李 鎭 宇)

Department of
Information and Communication Engineering

DGIST

2021

An Algorithm for Local Dynamic Map Generation for Safe UAV Navigation

Advisor: Professor Kyoung-Dae Kim
Co-advisor: Professor Sunghoon Im

by

Jin-Woo Lee
Department of Information and Communication Engineering
DGIST

A thesis submitted to the faculty of DGIST in partial fulfillment of the requirements for the degree of Master of Science in the Department of Information and Communication Engineering. The study was conducted in accordance with Code of Research Ethics[1]

07. 01. 2021

Approved by

Professor Kyoung-Dae Kim (signature)
(Advisor)
Professor Sunghoon Im (signature)
(Co-Advisor)

[1] Declaration of Ethical Conduct in Research**:** I, as a graduate student of DGIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

An Algorithm for Local Dynamic Map Generation for Safe UAV Navigation

Jin-Woo Lee

Accepted in partial fulfillment of the requirements for the degree of Master of Science.

05. 26. 2021

Head of Committee Prof. Kyoung-Dae Kim (signature)

Committee Member Prof. Sunghoon Im (signature)

Committee Member Prof. Hoon Sung Chwa (signature)

MS/IC
201922033

이 진 우. Jin-Woo Lee. An Algorithm for Local Dynamic Map Generation for Safe UAV Navigation. Department of Information and Communication Engineering. 2021. p23. Advisors Prof. Kyoung-Dae Kim, Co-Advisors Prof. Sunghoon Im

ABSTRACT

In recent years, researches for perception of the flight environment and collision avoidance control have been actively conducted for the safe navigation for unmanned aerial vehicles (UAVs) used in various fields such as surveillance, agriculture, transportation, rescue and military. Accurate and real-time perception of the surrounding environment, such as the free area and the recognition of dynamic and static obstacles, is essential for path planning and control for navigation to avoid collision. The perception system of the UAV needs to recognize information such as the position and velocity of all objects in the surrounding local area regardless of the type of the object. At the same time, a probability based representation taking into account the noise of the sensor is also essential. In addition, a software design with efficient memory usage and operation time is required in consideration of the hardware limitations of the UAVs.

In this paper, we propose a 3D Local Dynamic Map(LDM) with essential elements of the aforementioned UAV perception system. The proposed LDM uses a circular buffer as a data structure to ensure low memory usage and fast operation speed. Probability based occupancy map is created using sensor data, and the position and velocity of each object are calculated through clustering between grid voxels using this occupancy map and velocity estimation based on particle filter. The objects are predicted using the position and velocity of each object, and this is reflected in the occupancy map. This process is continuously repeated, and the flying environment of the UAV can be expressed in a three-dimensional grid map and the state of each object.

For the evaluation of the proposed LDM, we constructed a simulation environment and the UAV for outdoor flying. In the simulation, multiple environments with dynamic and static obstacles are created, and the UAV equipped with a virtual LiDAR sensor is created in those situations. The UAV for outdoor flying is constructed with a LiDAR sensor and computing board for sensor data processing and LDM algorithm running. As an evaluation factor, the occupancy grid accuracy is evaluated, and the ground truth velocity and the estimated velocity are compared.

Keywords: Local Dynamic Map(LDM), UAV, Clustering, Probabilistic Grid

List of Contents

List of Figures

List of Tables

I. Introduction

In recent years, the use of unmanned aerial vehicles (UAVs) has increased significantly in various fields, including surveillance, agriculture, transportation, rescue and military [1–5]. Accordingly, for safe UAV navigation that avoid collisions, researches on perception of flight environments such as object detection, tracking and mapping are actively conducted [6–10]. For planning and control for safe UAV navigation, accurate and real-time perception of the surrounding environment such as occupied and free areas, dynamic and static obstacles are essential. We believe that the following three are essential considerations for the perception system for safe navigation of UAVs.

- It is necessary to know the state such as position and velocity for all objects existing in a local area around the UAV regardless of the type of object. When the UAV is flying, it is necessary to distinguish between places where collisions may occur and not. And further, in the case of dynamic objects, we need to predict the motion of objects and avoid them. Therefore, the current position and velocity of objects in the surrounding 3D local area of UAV are essential factors that must be recognized.
- Probabilistic representation of the surrounding environment which considering the noise of the sensor is needed. We can recognize the surrounding environment through sensors such as LiDAR, camera, and radar. However, since sensor data always has noise, it must be expressed as a probabilistic expression that considers sensor noise like sensor noise models. This can further be useful for integrating different sensors.
- The limit of hardware system of UAV should be considered when configuring the UAV's perception system. Compared to other mobility platforms, UAV has a relatively limited payload weight. Due to this, the usable sensors and computing boards are limited. Therefore, we need to build a software with efficient memory and execution time within the limited hardware situation.

As a perception system, Occupancy Grid Map (OGM) represents an environment by using probabilistic grid cells. OGMs discretize the space into grid cell space to improve memory efficiency and represent occupancy probability of each cell by using sensor noise model. [9] extended this to 3D space and made it possible to apply OGM for UAVs. However, since most OGMs are used as a prior map of wide areas for the purpose of global mapping or SLAM, it couldn't be guaranteed that memory usage and computation time are used in real-time. As

UAVs have limited hardware specifications that can be implemented, we need to reduce memory usage and computational load by limiting the perception area to only adjacent surround area of the UAV. [11] proposed an occupancy grid map using a circular buffer for this purpose. Occupancy map of [11] is limited only for a local area around the current UAV location for efficiency and it is suitable for UAV. However, it is still regrettable that [11] does not have object-level expressions such as the position and velocity of each object which are useful for navigation.

Most of object-level perception algorithms are based on deep learning approach such as [10,12,13]. In recent years, many studies have been continuously conducted, and object recognition and classification are performed using camera and LiDAR. However, deep learning based algorithms can recognize only pre-trained objects. In the case of autonomous vehicles, the objects that can appear on the road are limited, but since UAVs do not have the concept of a driving road, a wide variety of objects can appear, so it is difficult to fully recognize the surrounding environment only with deep learning approach. In addition, it requires vast amounts of training data, but UAVs' flying data is insufficient, making it difficult to apply. Therefore, an additional method is needed, and OGMs are appropriate for perception system for collision avoidance navigation of UAVs.

As mentioned above, the limitation of OGM is no representation of object-level state. So, OGM that contains object-level representation can break the limitation and Occupancy Grid Filter(OGF) algorithms such as [14–16] are most similar to this purpose. OGF expresses the occupancy and velocity of each cell as probability. Furthermore, [17,18] calculated the position and velocity of each object cluster through clustering and tracking between cells. However, these algorithms are studied for autonomous vehicles so they are conducted in 2D space. For UAVs, [19] expresses even the dynamic situation by utilizing the Bin-occupancy filter for the local area around the UAV, but the computational load is heavy.

In this paper, we propose an algorithm for 3D Local Dynamic Map(LDM) generation that includes occupancy grid map and object state that position and velocity with reduced computation load considering the hardware limitations of UAVs. We use circular buffer based mapping algorithm from [11] to reduce used memory and computation time. Occupancy probability of grid map is predicted by using previous state of objects and updated by using sensor measurement of current time. Using this occupancy grid, occupied area and velocity of each object are obtained through particle filter based velocity estimation and clustering. We evaluate proposed LDM algorithm with ground truth of occupancy grid map and state of objects such as position and velocity in simulation. In addition, it is also tested and evaluated using an outdoor UAV composed of LiDAR sensor and onboard PC.

This paper is structured as follows. Section 2 describes related works of LDM. Section 3 explains LDM algorithm that we proposed, and evaluation of algorithm is described in section 4. Finally, section 5 conclude the paper.

II. Related Works

OGM is an algorithm that represents the surrounding environment for safe navigation. Various sensors such as a sonar sensor, LiDAR, RGB-D camera are used and applied to various platforms such as mobile robot and autonomous vehicle, [20–24]. However, these OGMs express the environment in two dimensions, [9] makes 3D grid map by using octree data structure to reduce memory usage and computation time. [25,26] use Octomap as a perception system for 3D Navigation of UAV. However, OGMs have weakness in update speed and prediction for dynamic objects because they accumulate measurements and shows only the occupancy probability of current state.

[14] proposed one of the OGF, Bayesian Occupancy Filter (BOF), which extended OGM to a 4D-grid that consists of velocity probabilities of each occupancy grid cell to compensate for the weaknesses of OGM that can only consider occupancy. In BOF, the state of each cell is predicted using the previous occupancy and velocity probability. Through this, a corresponding grid map is created for dynamic objects, but there is an inefficient aspect of having a velocity grid for all cells include free cells.

[16] and [15] applied particle filter to BOF to solve the above inefficient problem, and expressed the velocity of each cell as particle distribution. In addition, the cell is divided into static, dynamic and free, and the efficiency of calculation is improved by allocating and removing particles.

The aforementioned OGF algorithms express the occupancy probability and velocity of a grid cell, but these are all two-dimensional representations and are not suitable for 3D navigation of UAVs. [19] creates 3D grid map for a local area around the UAV to avoid collision based on bin-occupancy filter which predicts the movement of particle in each cell. However, it is limited in use due to insufficient evaluation of the accuracy of mapping and the heavy computational load. [27,28] use Euclidean Signed Distance Fields (ESDF) grid map as 3D representation for UAV navigation. These obtain the shortest distance to the occupied voxel and obtains the possible collision distance in real-time. However, these algorithms have no representation of velocity field or object state so the trajectory of dynamic objects cannot be predicted.

[11] proposed an algorithm that creates a circular buffer based local grid map for UAV replanning. The data structure of occupancy grid is circular buffer to reduce computation time. We take the circular buffer data structure proposed by [11] for our proposed LDM. However, unlike [11], we perform the occupancy prediction process to respond to dynamic objects.

Clustering of occupancy grid has been proposed for a planning technique that using object recognition information. In [29], the states of grid cells are determined through comparison between cell occupancy probability of the current and previous time, and cell clustering is applied using these states. [18] project the clustering result from superpixel approach, which pixel clustering algorithm, to grid space and assign the clusters to each cell. [17] applied clustering and tracking between cells using BOF grid.

III. Local Dynamic Map(LDM) Generation

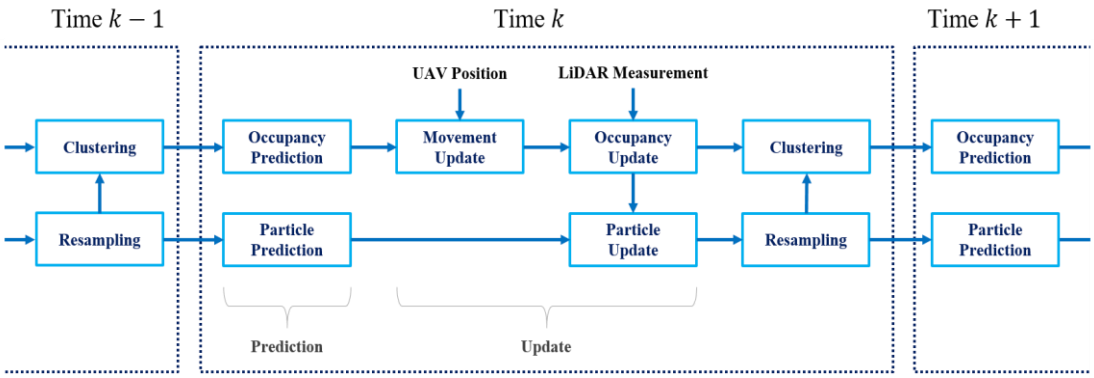

Figure 1. An overview of proposed Local Dynamic Map algorithm.

We propose an algorithm for Local Dynamic Map(LDM) generation that expresses the surrounding environment by occupancy grid and obstacle clusters. Proposed algorithm represents the space that occupied by obstacles in the local area around the UAV as occupancy grid and at the same time recognizes the obstacles' states such as position and velocity. Occupancy grid is predicted by using previous state of objects and updated by using sensor measurement of current time. To extract obstacle state, LDM use grid voxel clustering and particle filter based velocity of cluster estimation. As shown in Figure 1, the overall process proceeds in the order of prediction, update, resampling, and clustering. In this section, the main states of LDM are defined and then each process is described in detail.

3.1 LDM State Representation

In LDM, the environment around the UAV is divided into 3-dimensional grid composed of voxels. Each voxel in the form of a cube expresses occupancy and dynamic state by using the occupancy probability and velocity. Occupancy probability is stored as log-odds notation as shown in equation 1 for computational benefits.

$$l(Occ_k^i) = \ log\left(\frac{P(Occ_k^i)}{1 - P(Occ_k^i)}\right), \tag{1}$$

with $l(Occ_k^i)$ is log-odds notation of voxel i and $P(Occ_k^i)$ is occupancy probability of it.

In order to recognize only the local area around the UAV, the grid map region continuously moves based on the location of the UAV. The easiest way to do this is to match center of grid map region with the location of the UAV. However, it requires occupancy probability update process for whole voxels every time when the UAV moves. And in most cases, the accuracy of occupancy probability may be lost due to overlap between voxels. To prevent this, we update the grid map region so that the UAV is located only inside the center grid voxel of the map. Due to the movement of UAV, if the UAV located voxel moves $\mathbf{\Delta X}$ = (ΔX,ΔY,ΔZ), where X, Y, Z are multiples of voxel resolution from the previous UAV located voxel(same as center grid voxel), the grid map region moves by the same distance so that the UAV located voxel becomes the center grid voxel. Also, the rotation of the grid map due to the rotation of the UAV is not considered, and the orientation of the map is fixed as the orientation of the initial. Therefore, since the grid map region moves only in multiples of the voxel resolution, the overlap between voxels can be blocked. Figure 2 describes it with example situation.

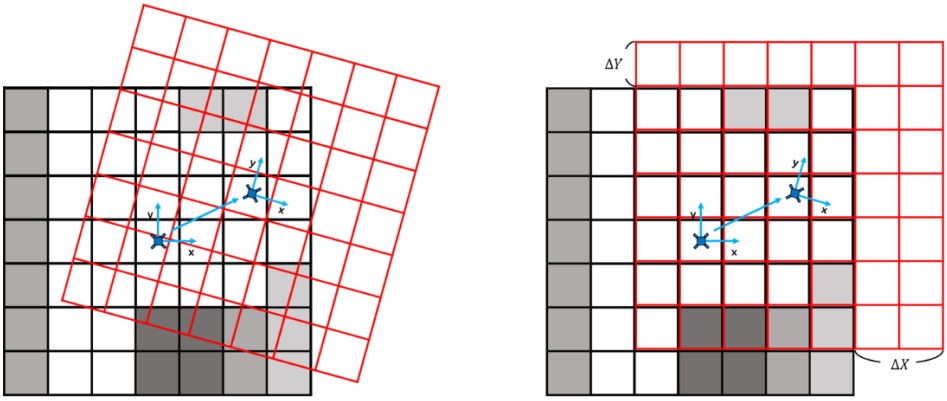

Figure 2. An example of update process of grid map in proposed LDM. For visualization, we reduce dimension to 2D. Left case is the example of overlap situation when update grid map region depends on UAV position and orientation. Right image is that update process that used in our LDM. If UAV located voxel moves (ΔX, ΔY) as shown in right image, grid map region is also moves (ΔX, ΔY). LDM prevent overlap issues by using this update process.

The estimated velocity of each voxel is expressed as a set of particles. The particles have their position and velocity. To represent the reliability of the state of particle that position and velocity, each particle has the weight. The representation of particle state is:

$$\mathbf{x}^i = \left(p_x, p_y, p_z\right) \ , \qquad \mathbf{v}^i = \left(v_x, v_y, v_z\right) \ , \tag{2}$$

where $\mathbf{x}^i$ is position of particle i and $\mathbf{v}^i$ is velocity of particle i. Obstacles can move randomly, so we have to predict the probability of movement in all possible directions. Therefore, we use weighted particles as a probability of obstacle state to recognize the velocity and position of each obstacle.

The particles of LDM are initialized so that it is evenly distributed over the entire grid map region. So, the position of particle i, $\mathbf{x}^i \sim U(-\frac{\mathrm{M}}{2}, \frac{\mathrm{M}}{2})$ is uniform distribution over the grid map region with $\mathrm{M} = (M_x, M_y, M_z)$ is size of grid map region. The velocity of particle i, $\mathbf{v}^i \sim N(0, \Sigma)$ is normal distribution with covariance Σ. The weight of each particle, w^i, is set to the same value.

3.2 Prediction

Because traditional occupancy grid measurement update methods such as [9,11] are based on accumulation of measurements, reaction of dynamic obstacles is slow in a dynamic environment. To prevent this, occupancy prediction is applied using the clustering result. Through the clustering to be described in 3.5, we collected the clusters with their corresponding voxels and the velocity of clusters. By using this, the current moving position of clusters are predicted and the corresponding occupancy probability can be predicted as described in Figure 3.

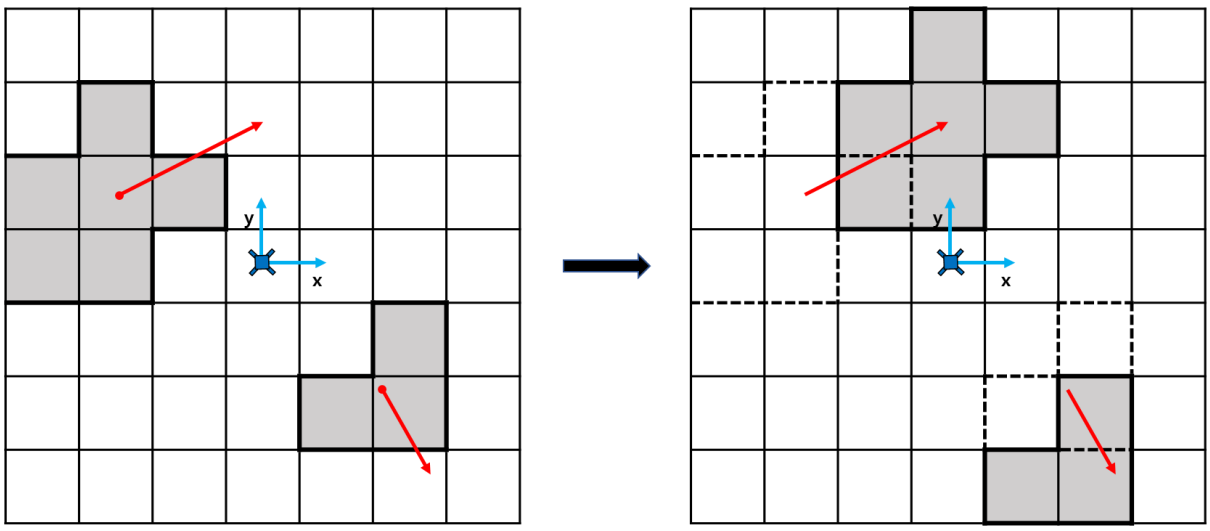

Figure 3. Example of occupancy prediction. The gray color cells are occupied cells and the part where the border is drawn with a thick line is one cluster. In left image, red dots are position of each cluster, red arrows are velocity of each cluster. The end of the arrow is the predicted position of each cluster. In this case left upper cluster is predicted to right upper direction and right lower cluster is predicted to right lower direction. The occupancy probabilities are moved as same direction of clusters.

Particles that distributed over the grid map region are predicted using prediction model from [16].

$$\mathbf{v}_k^i = \mathbf{v}_{k-1}^i + \sigma \ , \qquad \mathbf{x}_k^i = \mathbf{x}_{k-1}^i + dt \cdot \mathbf{v}_k^i \ , \tag{3}$$

where dt is time difference between time k and time $k-1$, $\sigma \sim N(0, \Sigma)$ is zero mean normal distribution noise with covariance Σ. If the predicted location of the particle is outside of grid map, it is removed and is not used in the subsequent processes.

3.3 Update

3.3.1 Movement Update

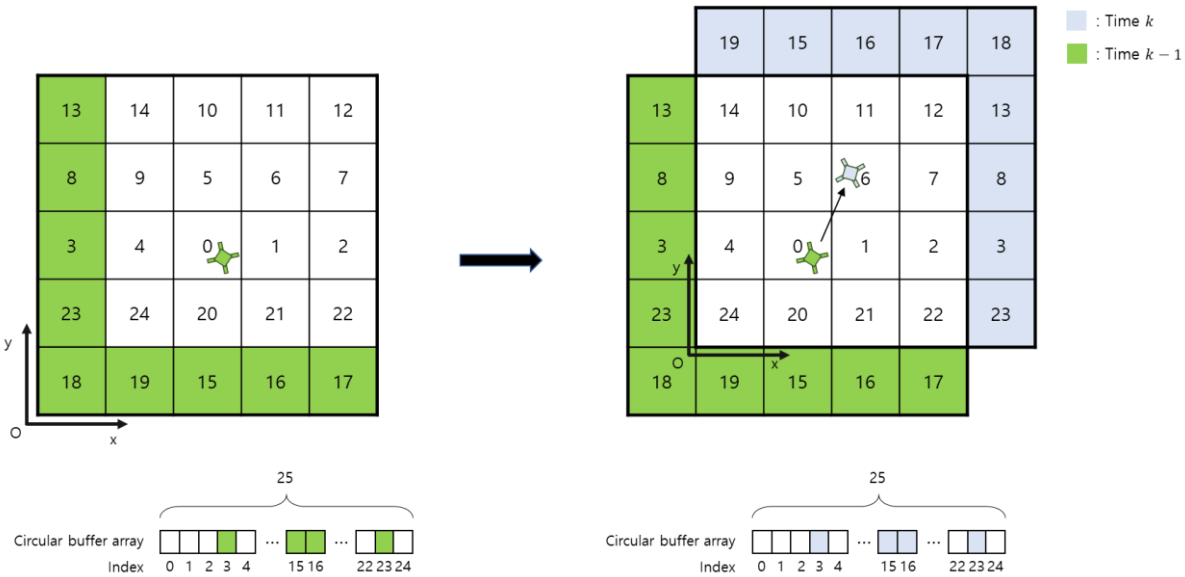

Figure 4. Movement update process. For visualization, we reduce the dimension to 2-dimension. The number of each voxel is index of circular buffer array. At time $k-1$, the voxel that index of the circular buffer equals 18, is the origin voxel of grid voxel coordinate. And the offset voxel $o_{k-1} = (2,2)$, and $X_{k-1}^{k-1} = (2,2)$ where the UAV located. At time k, after the UAV moved, the voxel where the UAV located relative to grid voxel coordinate at time $k-1$, $X_k^{k-1} = (3,3)$. The offset voxel is updated as $o_k = o_{k-1} - (X_k^{k-1} - X_{k-1}^{k-1}) = (1,1)$. Now the voxel that index of the circular buffer equals 24, is the origin voxel of grid voxel coordinate. By update offset and grid voxel coordinate, the voxels in green zone are now out of the grid map region and voxels in blue zones are newly entered the grid map region. To reduce the time that used by data update, the circular buffer array indexes of green zone voxels such as 3, 15, 16, 23 are allocated to blue zone voxels with 0 value.

We use circular buffer data structure of [11] for our grid map data structure. It is composed of a circular buffer array that stores occupancy probability of each voxel in 3D grid space and offset voxel $o = (o_x, o_y, o_z)$ indicating the grid voxel that corresponding to first index of circular buffer array. To update grid map region, the offset voxel is updated using equation 4.

$$o_k = o_{k-1} - \left(X_k^{k-1} - X_{k-1}^{k-1}\right), \tag{4}$$

with $X_k^{k-1} = (ix_k, iy_k, iz_k)$ means grid voxel where UAV located at time k relative to grid voxel coordinate at time $k-1$ and o_k means the offset voxel at time k. In this process, voxels that move out of grid map region are cleaned and voxels that included newly to the grid map region are initialized. To do this simply and fast, circular buffer indexes of moved out voxels are allocated to newly entered voxels and initialized to log-odds

notation of unknown probability, zero as shown in Figure 4. Through this, grid map update according to movement of UAV can be performed with high speed.

3.3.2 Occupancy Update

We use 3D-LiDAR, which provides point measurements with low noise compared to vision and radar sensors, to recognize the surrounding environment of UAV. In order to update measurements to occupancy grid, a measurement flag grid that the size is same as occupancy grid is created to indicate the presence or absence of measurement of each voxel. In the measurement flag grid, the voxel with the measurement is marked as occupied. And we applying the ray-casting algorithm to the occupied voxels and voxels that passing by rays are marked as free.

For each voxel of the occupancy grid, the occupancy probability is updated by using the marking of the measurement flag grid. Since we use log-odds notations, we can simplify update equation 5 to equation 6.

$$\frac{P(Occ_{k|k}^{i})}{1-P(Occ_{k|k}^{i})} = \frac{P(Occ_{k|k-1}^{i})}{1-P(Occ_{k|k-1}^{i})} \cdot \frac{P\left(Occ^{i}\middle|z_k\right)}{1-P\left(Occ^{i}\middle|z_k\right)} \ , \tag{5}$$

$$l\left(Occ_{k|k}^{i}\right) = l\left(Occ_{k|k-1}^{i}\right) + \ l\left(Occ^{i}|z_k\right) \ , \tag{6}$$

where z_k is measurement at time k, $P(Occ_{k|k-1}^{i})$ is predicted occupancy probability of voxel i at time k, $P(Occ_{k|k}^{i})$ is occupancy probability posterior of voxel i at time k and $P(Occ^{i}|z_k)$ means measurement probability. The equation is:

$$P\left(Occ^{i}\middle|z_k\right) = \begin{cases} p_{occ} \ , & \text{if flag of voxel } i \text{ is occupied} \\ p_{free} \ , & \text{if flag of voxel } i \text{ is free} \end{cases} \tag{7}$$

with p_{occ} and p_{free} are constant parameters and it is recommended to set p_{occ} to more than 0.5 and p_{free} to less than 0.5.

Some voxels may not have any flags due to the influence of sensing field of view or interference from other objects. In dynamic environment, voxels without sensor information couldn't be guaranteed that the previous

occupancy probability is reasonable for present occupancy probability. Therefore, LDM that we proposed updates these voxels using the survival probability so that the influence of the previous occupancy probability gradually decreases over time. Now we update voxels that not have any flags by using equation 8.

$$l\left(Occ_{k|k}^{i}\right) = l\left(Occ_{k|k-1}^{i}\right) \cdot P_s^i \tag{8}$$

where, $P_s^i < 1$ is survival probability that the state of voxel i can be remained. If a lot of time passes without sensor measurement, log-odds notation of occupancy probability of voxel is converged to 0, which is the middle of free and occupied state.

Survival probability is set differently for each voxel in consideration of occupancy grid of time $k-1$ and flag grid of current time. Voxels that not have any flags are divided into three cases. First, the voxels that are occupied by a static object at $k-1$ time. In this case, it can be said that the occupancy probability of these voxels is the same as before because they are occupied by the same object even after time passes. Therefore, if the velocity of a voxel at $k-1$ time is less than the threshold velocity, it is determined that the voxel is occupied by a static object and the P_s^i is set to 1.

Except for the above case, voxels can be divided into the case where they are located outside the sensor range and the part of the voxels where they are inside the sensor range but interfered by other objects. In the former case, the current situation is unknown due to the hardware limitation of the sensor, so all voxels in this case have the same $P_s^i < 1$. In the latter case, different P_s^i is determined according to the distance from the object causing the interference. The distance close to the interfering object is more likely to be occupied by the object due to the effect of the object's thickness, motion, etc., but this decreases as the distance increases. Therefore, for voxels passing by extending the ray between the voxel occupied by the interference object(same as occupied flag voxel) and the sensor origin to the end of the map, voxels at a certain distance from the occupied flag voxel have high P_s^i, and subsequent voxels are set so that P_s^i decreases in inverse proportion to the distance.

3.3.3 Particle Update

The reliability of the prediction of particle is high if the occupancy probability of the voxel that the predicted particle is located is high. Therefore, the weight of the particle is updated using the occupancy probability updated to the current measurement. Particle update equation is:

$$w_k^i = w_{k-1}^i + P\left(Occ_{k|k}^j\right) \ , \tag{9}$$

where w_k^i is weight of particle i at time k and j is index of voxel that particle i is located.

We can express state of voxel using particles. Velocity of each voxel is expressed as the weighted average of particles in each voxel:

$$V_k^j = \left(\frac{\sum_{i=1}^n w_k^i \cdot \mathbf{v}_k^i}{\sum_{i=1}^n w_k^i}\right) \ , \tag{10}$$

where, $V_k^j = \left(VX_k^j, VY_k^j, VZ_k^j\right)$ means velocity of voxel j, n is the number of particles in voxel j and $\mathbf{v}_k^i$ is velocity of particle i. In this process, some occupied voxels may not be properly updated due to insufficient number of particles. Therefore, to prevent this, add particles to occupied voxels that not enough particles located. The position of new particles, $\mathbf{x}_{k,new}^i \sim U(-\frac{\mathbf{r}}{2}, \frac{\mathbf{r}}{2})$ ($\mathbf{r} = (r, r, r)$, r is voxel resolution) is uniform distribution within the voxel and velocity is:

$$\mathbf{v}_{k,new}^i = V_k^j + \sigma \tag{11}$$

with $\sigma \sim N(0, \Sigma)$ is zero mean normal distribution noise with covariance Σ. The weight of each particle, $w_{k,new}^i$, is set as $P\left(Occ_{k|k}^j\right)$ where j is voxel where particle i is located.

The last part of update process, weight of particles is normalized and equation is:

$$w_k^i = \frac{1}{\mu_k} \cdot w_k^i \ , \tag{12}$$

where μ_k is normalization factor that:

$$\mu_k = \sum_{i=1}^{N} w_k^i \ , \tag{13}$$

with N is number of total particles.

3.4 Resampling

The total number of particles has changed due to the particles deletion or addition through the prediction and update process. Therefore, to keep the number of particles as same as initial state, resampling process is essential. The resampling sequence is as follows. First, a discrete distribution is created based on the weight of particles, and a particle is randomly selected using this distribution and added to the new particle array. This is done until the size of the new array becomes N_{init}, which is the initial number of particles. Through this, particles can be selected in proportion to the weight, and therefore, more particles can be placed in occupied voxels.

3.5 Voxel Clustering

Objects with different states are clustered using the occupancy probability of the voxels. We consider the connectivity between 26-neighborhood voxels for only voxels with an occupancy probability higher than the threshold. The position of each cluster is expressed as the average value of the voxels included in the cluster. Velocity of each cluster is expressed as weighted average of particles existing in the cluster,

$$V_k^{cluster,j} = \left(\frac{\sum_{i=1}^{n} w_k^i \cdot \mathbf{v}_k^i}{\sum_{i=1}^{n} w_k^i} \right) \tag{14}$$

where $V_k^{cluster,j}$ is velocity of cluster j at time k, n is the number of particles in cluster j. From this process, grid voxels included in each cluster and the velocity of the cluster are obtained.

After clustering, particles located in the clusters additionally adjust the velocity. We can know the velocity of each cluster, grid voxels and particles corresponding to the cluster. Among the resampled particles, they are located in the same cluster, but the velocity can be very different. Therefore, the velocity of these particles is readjusted using the velocity of the cluster. The equation is

$$\mathbf{v}_k^i = V_k^{cluster,j} \tag{15}$$

where $V_k^{cluster,j}$ is velocity of cluster j where particle i is located.

IV. Evaluation

To evaluate LDM algorithm, we build a testbed in both simulation and outdoor UAV with LiDAR sensor. Occupancy grid and estimated velocity accuracy is evaluated in various scenarios.

4.1 Simulation

4.1.1 Simulation Setup

Simulation constructed a virtual environment using V-REP simulator. We created a number of static and dynamic objects in a virtual space and constructed a virtual UAV equipped with 3D LiDAR. LDM algorithm is implemented with C++ based ROS node. UAV position, orientation and LiDAR sensor data of V-REP are communicated to the LDM algorithm node as a ROS topic using V-REP-ROS communication node. The parameters for proposed LDM algorithm are initialized before activate UAV. N_{init}, which is the initial number of particles is set to 200000, resolution of voxel is set to 0.15m and number of grid voxel is 32768(32^3). p_{occ} and p_{free} are set in the same manner as in [9] and [11].

To evaluate the occupancy probability and estimated velocity of LDM, we generated the corresponding ground truth values. For evaluation of the scenarios which dynamic obstacles are exist, a ground truth grid map is created based on the current position of the obstacle for every time. By comparing the occupancy probability with this, we define an evaluation indicator:

$$Occupancy\ \ Grid\ \ Accuracy(\%) = \frac{\sum^t X_{number}}{\sum^t D}\ , \tag{16}$$

with X_{number} is number of voxels matched to the same state({Occupied, Free}) by comparing ground truth and occupancy grid of LDM, D is number of voxels at onetime step and t is number of time step. We also know the ground truth velocity of each object in the simulation, so we compared this value with the estimated velocity of each cluster. We created various scenarios with dynamic and static obstacles through simulation and measured the accuracy of LDM algorithm. The scenarios are:

- Scenario1: Dynamic obstacles and UAV flying
- Scenario2: Static obstacles and UAV flying
- Scenario3: Dynamic and static obstacles and UAV flying

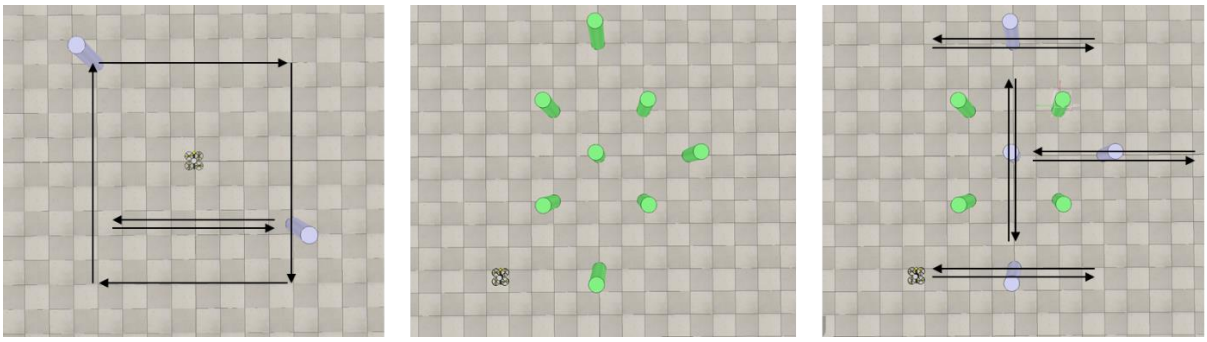

Figure 5. Simulation scenarios. Left image is scenario1, middle image is scenario2 and right image is scenario3. The blue obstacles are static obstacles and green obstacles are static obstacles. The black arrows are trajectory of dynamic obstacles.

4.1.2 Occupancy Grid Accuracy

	Scenario 1		Scenario 2		Scenario 3	
Algorithm	Proposed	[11]	Proposed	[11]	Proposed	[11]
Occupancy Grid Accuracy	99.55%	99.49%	99.22%	99.26%	99.50%	99.48%
Number of case1 false voxels	29355	40215	65034	57853	44520	49506
Number of case2 false voxels	148406	158700	241245	233250	150593	154824

Table 1. Occupancy Grid Accuracy of each scenario. The number of case1 false voxels means the number of false expressed voxels that the ground truth is occupied but expressed as free. The number of case2 false voxels means the number of false expressed voxels that the ground truth is free but expressed as occupied.

We evaluate occupancy grid accuracy of proposed LDM and circular buffer based grid map in [11]. And also we measure the number of false expressed voxels divided into two cases. Case1 means that the ground truth is occupied but expressed as free, and case2 means that the ground truth is free but expressed as occupied.

Compared with [11], proposed LDM predicts occupancy probability by using velocity of clusters and Table 1 shows the effect of these approaches. In scenario 1 where there are dynamic obstacles, the number of case1 and case2 false voxels of proposed algorithm less than that [11]. This is because the accumulation of the occupancy probabilities continued more rapidly by the prediction of the occupancy probability using the predicted velocity of the dynamic obstacle. Therefore, proposed algorithm provides more accurate representation of free space by decrease the number of case1 false voxels and also represents occupied space better.

In scenario 2, which is composed of only static obstacles, the accuracy of proposed algorithm is slightly lower. Proposed algorithm applies the survival probability to voxels without measurement due to object interference or sensor field of view. On the other hand, [11] keep the previous occupancy probability of not observed voxels. Therefore, it can be seen that [11] is more advantageous in a static environment, but it is not

appropriate to say that it is advantageous even in general scenarios involving dynamic obstacles such as scenario 3, because the accuracy of proposed algorithm is slightly higher. The average of total computation time of proposed LDM is 98ms with 200000 number of particles and 32768(32^3) number of voxels.

4.1.3 Velocity Estimation

To evaluate estimated velocity from proposed algorithm, we compared velocity of dynamic and static obstacles with ground truth. There are many algorithms that estimate velocity in 2D grid, but in 3D grid, there is no velocity estimation with voxels for local grid map. So, we used a method that applied the particle filter based velocity estimation part of our proposed algorithm to the occupancy grid map of [11] as a comparison algorithm of proposed algorithm, and this is expressed as '[11] with PF' in this evaluation.

Figure 6 shows the absolute value of the estimated velocity error of a static obstacle. The velocity estimation of proposed algorithm is more accurate for static obstacles in almost all times compared to [11] with PF. The average velocity error of proposed algorithm is 0.009m/s, while [11] with PF showed an average velocity error of 0.019m/s.

Figure 7, 8 and 9 show the result of velocity estimation of dynamic obstacle. The measured obstacle moved only x and y directions. The time for the estimated velocity to reach the ground truth velocity of the dynamic obstacle is similar between proposed LDM and [11] with PF. However, in the case of [11] with PF, the estimated velocity is not constant, whereas in the case of proposed LDM, the estimated velocity is almost similar to the ground truth velocity and is estimated at a constant. The average velocity error of proposed LDM is 0.12m/s, while [11] with PF shows 0.15m/s average velocity error.

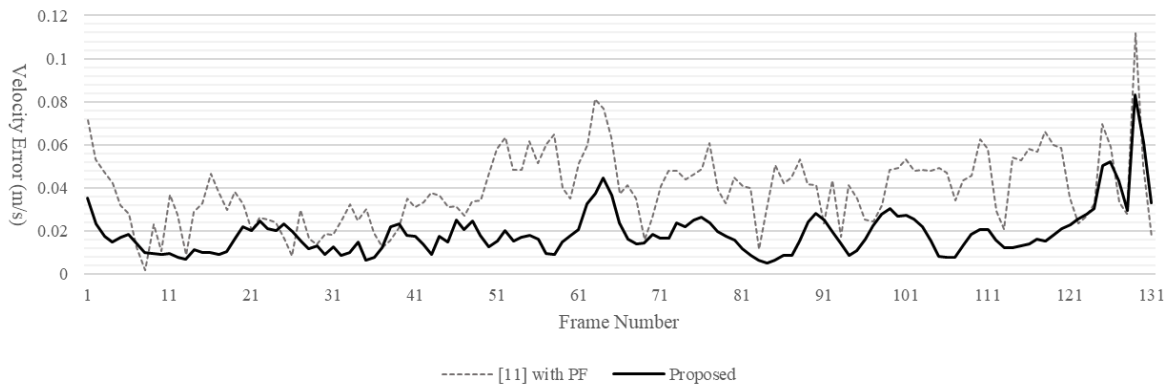

Figure 6. Estimated velocity error of static obstacle. Dotted line is [11] with PF method and solid line is proposed LDM generation algorithm.

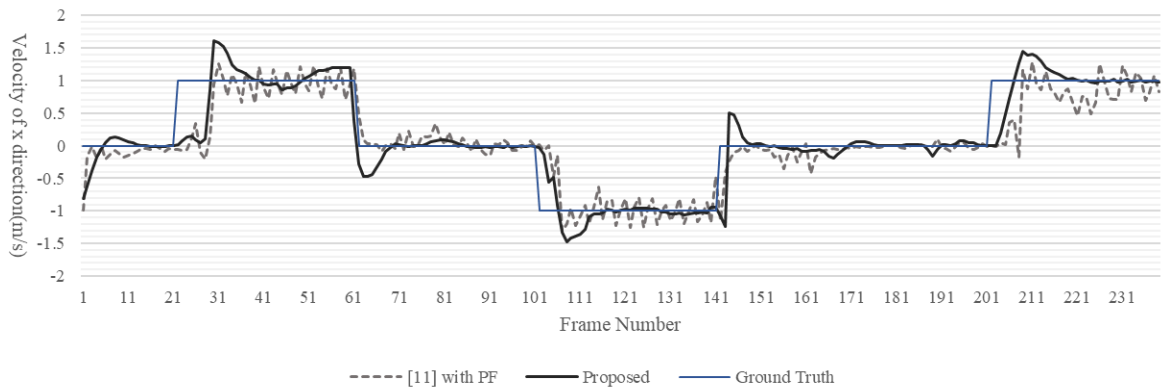

Figure 7. Estimated x-direction velocity and ground truth velocity of dynamic obstacle. Dotted line is [11] with PF method, black solid line is proposed LDM generation algorithm and blue solid line is ground truth.

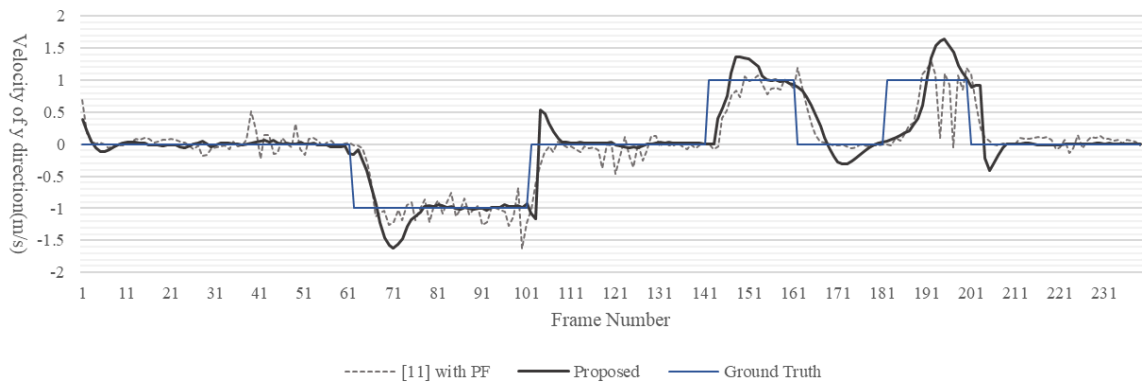

Figure 8. Estimated y-direction velocity and ground truth velocity of dynamic obstacle. Dotted line is [11] with PF method, black solid line is proposed LDM generation algorithm and blue solid line is ground truth.

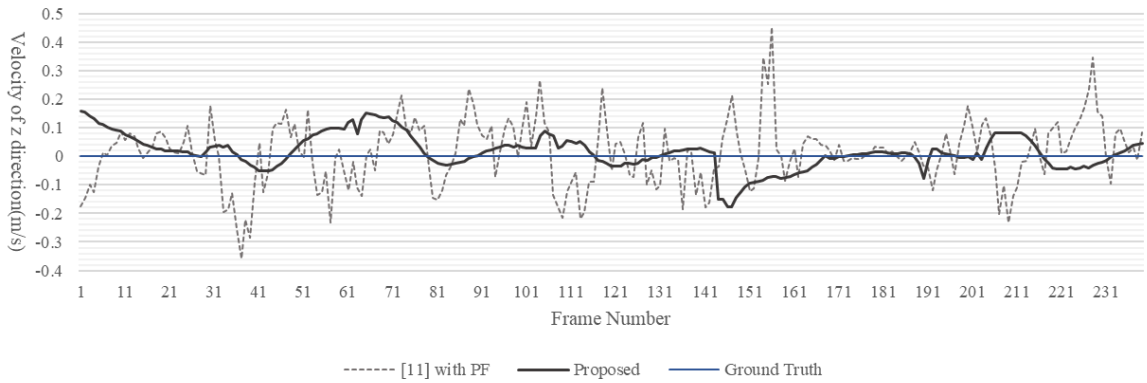

Figure 9. Estimated z-direction velocity and ground truth velocity of dynamic obstacle. Dotted line is [11] with PF method, black solid line is proposed LDM generation algorithm and blue solid line is ground truth.

4.2 Outdoor UAV Experiment

4.2.1 Experimental Setup

Figure 10. The structure of implemented UAV. This consists of LiDAR, onborad PC, GPS, IMU, flight controller and battery.

We implement the hardware and software system for outdoor UAV to evaluate proposed LDM algorithm. as shown in Figure 10. As the body frame, Matrice 100, which includes GPS, IMU and flight controller, is used. Ouster 16-channel 3D LiDAR is used as a sensor to recognize the flying environment. Jetson TX2 board is used to acquire sensor data and run the algorithm, and an extra battery is additionally installed to operate board. Proposed LDM algorithm implementation is same as simulation's one.

The evaluation is conducted in two scenarios. Scenario 1 is a scenario consisting of 2 moving people and 1 stationary person, and Scenario 2 is a situation where the UAV and the person move in the same direction as shown in figure 11.

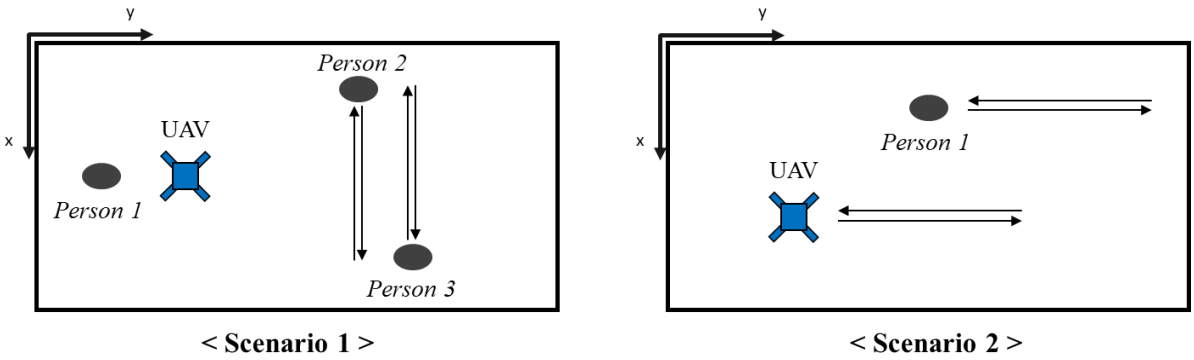

Figure 11. Scenarios of outdoor UAV experiment. In scenario 1, UAV is stopped and there are three people. In scenario 2, UAV and one person move same direction.

4.2.2 Experiment Results

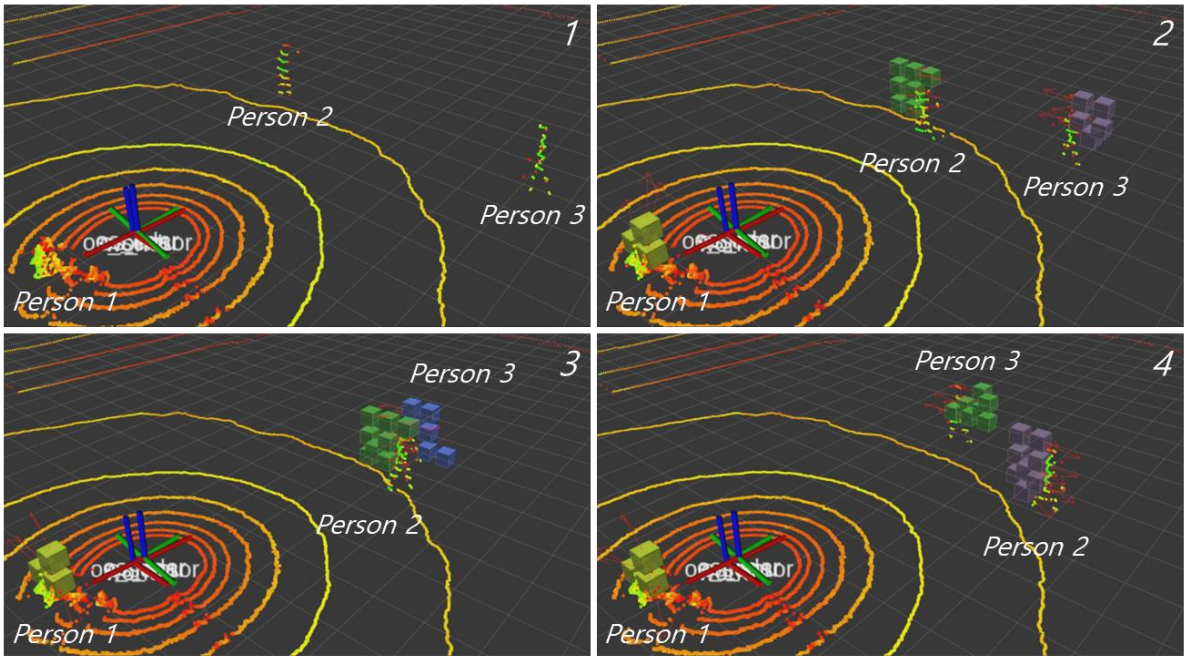

Figure 12. Snapshots of proposed LDM for scenario 1 of outdoor UAV experiment. Person1 is static obstacle that no moved, person2 and 3 are moved to each other. The colored points represent LiDAR measurements, and only occupied voxels among all voxels are visualized. The color of each voxel means the cluster number and the direction of the velocity of each voxel is marked with a red arrow.

Figure 12 shows part of the result of proposed LDM. For the experiment, we created an environment with one static obstacle (*Person1*) and two dynamic obstacles (*Person2* and *Person3*). Among the dynamic obstacles, *Person2* moves to *Person3* (+x direction) and *Person3* moves to *Person2* (-x direction). When each other reaches the other's starting position, they come back to their own starting position. Interference of *Person3* by *Person2* occurs in image 3 in Figure 12, but it is not lost the cluster because of occupancy prediction. It can be seen that the velocity direction is estimated according to the moving direction of the dynamic objects.

Figures 13, 14 and 15 show the velocity, which are estimated by proposed LDM, of obstacles in scenario of Figure 12. Figure 13 is the estimated velocity of static obstacle (*Person1*). The proposed LDM shows an error of up to 0.05m/s in all directions for the estimated velocity of a static obstacle. Figure 14 represent the estimated velocity of *Person2* and Figure 15 represent the estimated velocity of *Person3*. From 91 to 181 frames, when *Person2* moves in the +x direction, we can see that *Person3* moves in the –x direction. And it is also seen that after frame 181, these peoples are return to their starting points.

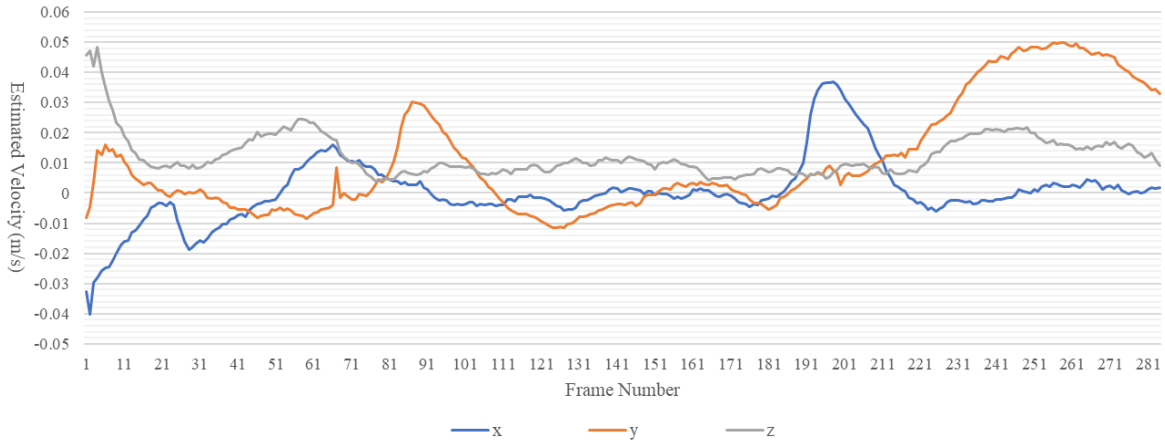

Figure 13. Estimated velocity of static obstacle in Figure 12.

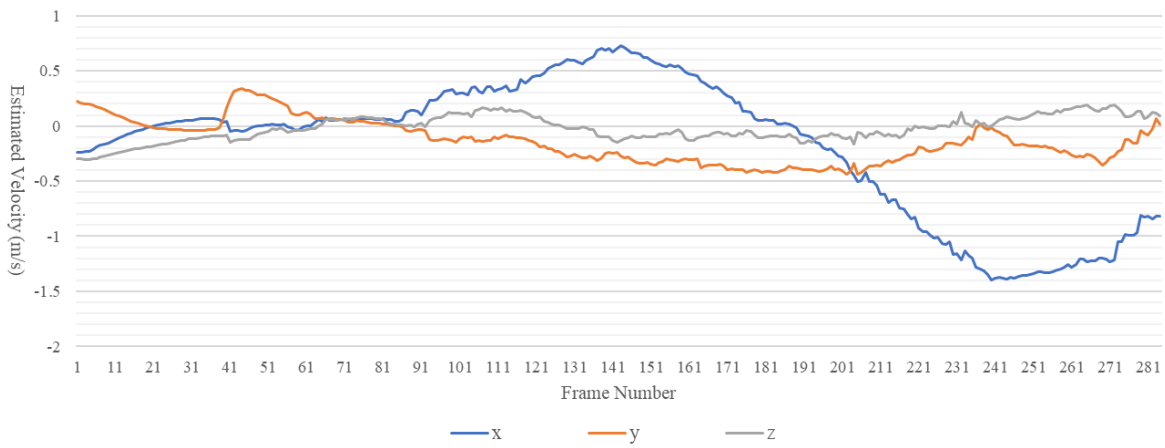

Figure 14. Estimated velocity of dynamic obstacle that started at upper left (*Person2*) in Figure 12.

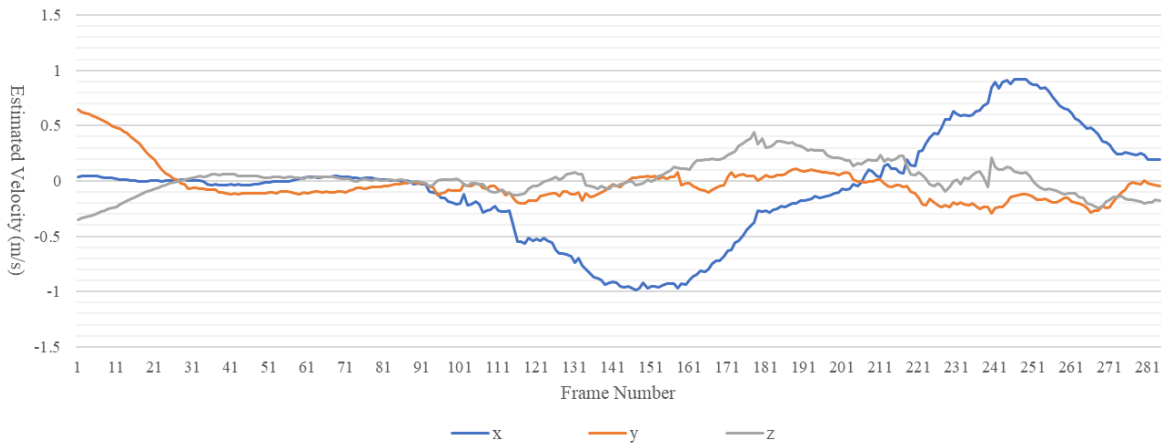

Figure 15. Estimated velocity of dynamic obstacle that started at lower right (*Person3*) in Figure 12.

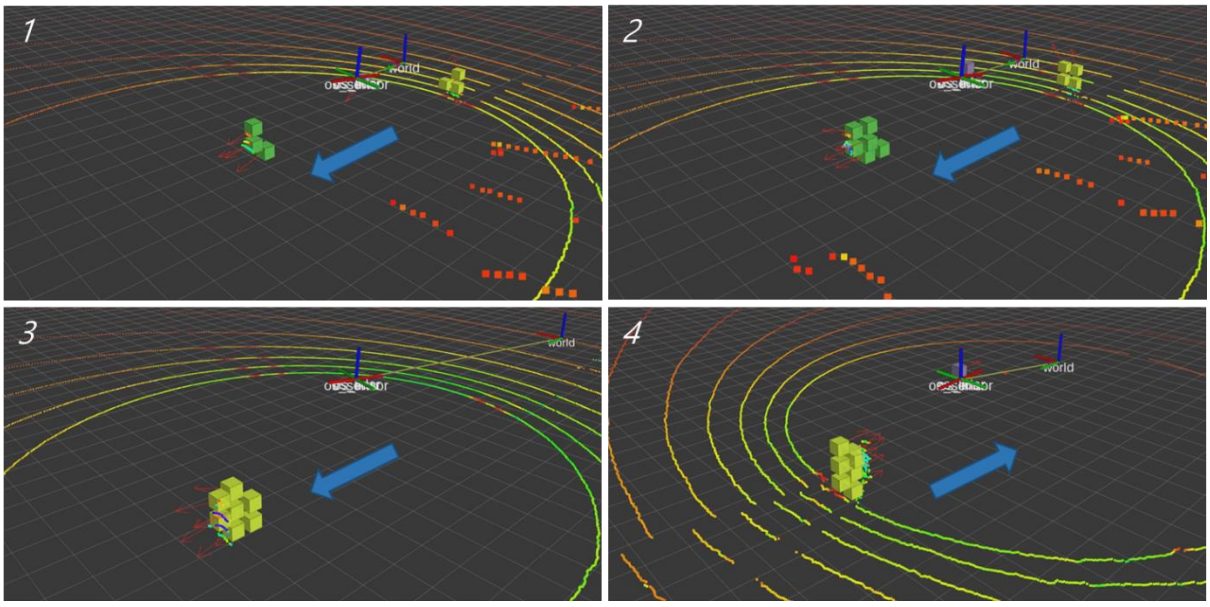

Figure 16. Snapshots of proposed LDM for scenario 2 of outdoor UAV experiment with dynamic obstacle. In this scenario, UAV and obstacle are moving same direction and the big blue arrow is the direction of UAV and obstacle. The colored points represent LiDAR measurements, and only occupied voxels among all voxels are visualized. The color of each voxel means the cluster number and the direction of the velocity of each voxel is marked with a red arrow.

Figure 16 shows another scenario for test the proposed LDM and result of it. In this scenario, UAV and an obstacle are moving same direction always. In image 1, 2, 3 of Figure 16, UAV is moved to the +y direction and obstacle is also moved in the same direction. In image 4 of Figure 16, UAV is moved to the –y direction and obstacle is also moved. We can see that the direction of the obstacle and the direction of estimated velocity of the obstacle are the same.

V. Conclusions

We proposed LDM generation algorithm that represents a local area around the UAV using 3D occupancy grid and object clusters. Proposed LDM provides a 3D occupancy grid map suitable for UAVs that are considering dynamic obstacles. In addition, it provides object-level state information so that the movement of obstacles can be predicted and avoided. By providing grid-level and object-level information at the same time, it can be used in many collision avoidance navigation algorithms.

However, since the calculation time approaches 100ms, it may be difficult to use in real time when the size of the grid map increases. So, it is necessary to improve the calculation time by applying parallel processing. There is also a need to further improve the accuracy of the estimation velocity.

References

[1] Alotaibi, E.T.; Alqefari, S.S.; Koubaa, A. Lsar: Multi-uav collaboration for search and rescue missions. IEEE Access2019,7, 55817–55832.

[2] Tisdale, J.; Kim, Z.; Hedrick, J.K. Autonomous UAV path planning and estimation. IEEE Robotics & Automation Magazine2009,16, 35–42.

[3] Pretto, A.; Aravecchia, S.; Burgard, W.; Chebrolu, N.; Dornhege, C.; Falck, T.; Fleckenstein, F.; Fontenla, A.; Imperoli, M.;Khanna, R.; others. Building an Aerial-Ground Robotics System for Precision Farming: An Adaptable Solution. arXiv preprintarXiv:1911.030982019.

[4] Brunner, G.; Szebedy, B.; Tanner, S.; Wattenhofer, R. The urban last mile problem: Autonomous drone delivery to your balcony. 2019 international conference on unmanned aircraft systems (icuas). IEEE, 2019, pp. 1005–1012.

[5] Condomines, J.P. Nonlinear Kalman Filter for Multi-Sensor Navigation of Unmanned Aerial Vehicles: Application to Guidance and Navigation of Unmanned Aerial Vehicles Flying in a Complex Environment; Elsevier, 2018.

[6] Saha, S.; Natraj, A.; Waharte, S. A real-time monocular vision-based frontal obstacle detection and avoidance for low cost UAVs in GPS denied environment. 2014 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology. IEEE, 2014, pp. 189–195.

[7] Stegagno, P.; Basile, M.; Bülthoff, H.H.; Franchi, A. A semi-autonomous UAV platform for indoor remote operation with visual and haptic feedback. 2014 IEEE international conference on robotics and automation (ICRA). IEEE, 2014, pp. 3862–3869.

[8] Florence, P.R.; Carter, J.; Ware, J.; Tedrake, R. Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data. 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 7631–7638.

[9] Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous robots 2013,34, 189–206.

[10] Ammour, N.; Alhichri, H.; Bazi, Y.; Benjdira, B.; Alajlan, N.; Zuair, M. Deep learning approach for car detection in UAV imagery. Remote Sensing 2017,9, 312.

[11] Usenko, V.; Von Stumberg, L.; Pangercic, A.; Cremers, D. Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 215–222.

[12] Rahman, M.M.; Tan, Y.; Xue, J.; Lu, K. Recent advances in 3D object detection in the era of deep neural networks: A survey. IEEE Transactions on Image Processing 2019,29, 2947–2962.

[13] Fraga-Lamas, P.; Ramos, L.; Mondéjar-Guerra, V.; Fernández-Caramés, T.M. A review on IoT deep learning UAV systems for autonomous obstacle detection and collision avoidance. Remote Sensing 2019,11, 2144.

[14] Coué, C.; Pradalier, C.; Laugier, C.; Fraichard, T.; Bessière, P. Bayesian occupancy filtering for multitarget tracking: an automotive application. The International Journal of Robotics Research 2006,25, 19–30.

[15] Nuss, D.; Yuan, T.; Krehl, G.; Stuebler, M.; Reuter, S.; Dietmayer, K. Fusion of laser and radar sensor data with a sequential Monte Carlo Bayesian occupancy filter. 2015 IEEE intelligent vehicles symposium (IV). IEEE, 2015, pp. 1074–1081.

[16] Nègre, A.; Rummelhard, L.; Laugier, C. Hybrid sampling bayesian occupancy filter. 2014 IEEE Intelligent Vehicles Symposium Proceedings. IEEE, 2014, pp. 1307–1312.

[17] Mekhnacha, K.; Mao, Y.; Raulo, D.; Laugier, C. Bayesian occupancy filter based" fast clustering-tracking" algorithm. IROS 2008,2008.

[18] Oh, S.I.; Kang, H.B. Fast occupancy grid filtering using grid cell clusters from LIDAR and stereo vision sensor data. IEEE Sensors Journal 2016,16, 7258–7266.

[19] Odelga, M.; Stegagno, P.; Bülthoff, H.H. Obstacle detection, tracking and avoidance for a teleoperated UAV. 2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016, pp. 2984–2990.

[20] Lu, D.V.; Hershberger, D.; Smart, W.D. Layered costmaps for context-sensitive navigation. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014, pp. 709–715.

[21] Kato, S.; Tokunaga, S.; Maruyama, Y.; Maeda, S.; Hirabayashi, M.; Kitsukawa, Y.; Monrroy, A.; Ando, T.; Fujii, Y.; Azumi, T. Autoware on board: Enabling autonomous vehicles with embedded systems. 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS). IEEE, 2018, pp. 287–296.

[22] Matthies, L.; Elfes, A. Integration of sonar and stereo range data using a grid-based representation. Proceedings. 1988 IEEE International Conference on Robotics and Automation. IEEE, 1988, pp. 727–733.

[23] Shvets, E.; Shepelev, D.; Nikolaev, D. Occupancy grid mapping with the use of a forward sonar model by gradient descent. Journal of Communications Technology and Electronics 2016,61, 1474–1480.

[24] Homm, F.; Kaempchen, N.; Ota, J.; Burschka, D. Efficient occupancy grid computation on the GPU with lidar and radar for road boundary detection. 2010 IEEE Intelligent Vehicles Symposium. IEEE, 2010, pp. 1006–1013.

[25] Mittal, M.; Mohan, R.; Burgard, W.; Valada, A. Vision-based autonomous UAV navigation and landing for urban search and rescue. arXiv preprint arXiv:1906.013042019.

[26] Vanegas, F.; Gaston, K.J.; Roberts, J.; Gonzalez, F. A framework for UAV navigation and exploration in GPS-denied environments.2019 ieee aerospace conference. IEEE, 2019, pp. 1–6.

[27] Oleynikova, H.; Taylor, Z.; Fehr, M.; Siegwart, R.; Nieto, J. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 1366–1373.

[28] Han, L.; Gao, F.; Zhou, B.; Shen, S. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. arXiv preprint arXiv:1903.021442019.

[29] Bouzouraa, M.E.; Hofmann, U. Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors. 2010 IEEE Intelligent Vehicles Symposium. IEEE, 2010, pp. 294–300.

요 약 문

무인항공기의 안전한 주행을 위한 로컬 동적 맵 작성 알고리즘

최근 몇 년간 무인기는 정찰, 감시, 농업, 운송, 구조, 국방 등의 다양한 분야에서 사용이 크게 증가하고 있다. 이에 따라 충돌을 회피하는 안전한 무인기의 주행을 위해 주행 환경의 인지, 충돌 회피 제어와 관련된 연구도 활발히 이루어지고 있다. 충돌을 회피하는 주행에 대한 경로 계획, 제어를 위해서는 주행 가능 영역, 동적 및 정적 장애물 인지 등 주변 환경에 대한 정확하고 실시간 인지가 필수적이다. 무인기의 인지 시스템은 물체의 종류와 관계없이 주변 로컬 영역에 존재하는 모든 물체의 위치 속도 등의 정보를 인지함과 동시에 센서의 노이즈를 고려한 확률기반의 표현이 필요하다. 또한 무인기의 하드웨어적 한계를 고려하여 메모리 사용 및 연산 시간이 효율적인 소프트웨어 설계가 필요하다.

본 논문에서는 앞서 언급한 무인기 인지 시스템의 필수 요소들을 갖춘 3 차원 로컬 동적 맵을 제안한다. 제안하는 로컬 동적 맵은 데이터 구조로 원형 버퍼를 활용하여 적은 메모리 사용량과 빠른 연산속도를 확보한다. 센서 데이터를 활용하여 확률 기반의 누적 맵을 작성하며 이 누적 맵을 이용한 그리드 셀 간 클러스터링과 파티클 필터 기반의 속도 추정을 통해 각 물체 별 위치 및 속도를 연산한다. 각 물체 별 위치와 속도를 이용하여 이동할 위치를 예측하고 이를 기존 누적 맵에 반영한다. 이 과정이 지속적으로 반복되어 무인기 주행 환경을 3 차원 그리드 맵과 물체 별 상태로 표현할 수 있다.

제안하는 로컬 동적 맵의 평가를 위해 우리는 시뮬레이션 환경과 실외 주행용 무인기를 구성했다. 시뮬레이션에서는 동적 및 정적 장애물이 존재하는 여러 환경을 만들고 해당 상황에서 가상의 라이다 센서가 장착된 무인기를 생성했다. 실외 주행용 무인기는 시뮬레이션과 동일하게 라이다 센서를 장착하여 구성했다. 평가 지표로 실제 장애물 점유 영역에 대한 정확성을 평가했고 실제 속도와 추정된 속도의 비교를 수행했다. 그 결과 속도 추정 결과를 반영한 예측과정으로 인해 기존의 누적 맵 작성 방식에 비해 동적환경에서 더 정확한 그리드 맵을 작성할 수 있었고 추정된 속도 또한 실제 속도와 유사한 것을 확인할 수 있었다.

핵심어: 무인항공기, 로컬 동적 맵, 클러스터링, 확률기반 그리드