Engineering Science and Technology, an International Journal 64 (2025) 101998

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Full Length Article

Robust robot localization with visually adaptive consensus filters in dynamic corridor environments

Suhyeon Kang, Heoncheol Lee *

Department of IT Convergence Engineering, School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Gyeongbuk 39177, Republic of Korea

ARTICLE INFO

Keywords:
Robot localization
Adaptive consensus filter
Dynamic corridor environments
Object detection

ABSTRACT

This paper deals with the problem of robot localization in dynamic corridor environments. If a robot uses only a LiDAR (light detection and ranging) for its localization, the accuracy of robot localization degenerates as time goes due to the occlusions by moving people around a robot and the lack of scan features in corridors. This paper proposes a robust robot localization method with visually adaptive consensus filters (VACF) to solve the problem. The VACF consists of LiDAR odometry estimation, probabilistic localization, visual odometry estimation, optical flow recognition, object detection and adaptive consensus filters. To deal with long corridor environments, optical flow methods are used to correct the robot's position. For robust localization in dynamic environments, object detection algorithm is used to detect dynamic objects, and localization algorithms are adaptively used as input to a consensus filter based on the number of dynamic objects detected. The VACF was tested in real-world experiments in dynamic corridor environments and showed better accuracy than other existing methods when compared to pre-determined ground truth points.

1. Introduction

Indoor autonomous driving is a key technology for developing autonomous robots. In order to perform indoor autonomous driving, robot localization is the basis for the robot to know where it is. Since autonomous driving is performed based on the results of localization, localization is a highly important technology for robot autonomy. Robot localization is performed using various sensors such as LiDAR, camera, IMU (inertia measurement unit), and GPS (global positioning system), each of which has its own advantages and disadvantages. LiDAR-based localization algorithms are relatively accurate compared to camera-based localization algorithms, but they can produce inaccurate localization results in corridor environments and when there is little change in features [1,2]. On the other hand, camera-based localization algorithm is less accurate than LiDAR-based localization algorithm, but it shows relatively better results in dynamic and corridor environments [3,4]. Therefore, it is necessary to use various sensors to improve the accuracy of the localization algorithm and to prevent false localization in dynamic and corridor environments, and various research is being conducted to address dynamic and corridor environments [5–7].

There are many SLAM and robot localization algorithms that are the basis for robotic autonomy. GMapping [8] is a representative localization algorithm using 2D LiDAR. GMapping is a grid-based SLAM algorithm that uses a probabilistic grid map to simultaneously estimate the robot's position and the map. Particle filter is used to perform the initialization and resampling process. It initially assumes a number of probabilistic positions, keeps the best particles, removes the rest, and repeats the process. With each iteration, the robot's position and map information are estimated more accurately. Other 2D LiDAR-based SLAM methods that use particle filters and grid maps include Hector SLAM [9] and AMCL [10] (adaptive monte carlo localization). GMapping is generally suitable for environmental mapping, Hector SLAM has the advantage of fast execution time, and AMCL is useful for accurate real-time robot localization. Other LiDAR-based localization algorithms include cartographer [11], EKF-SLAM [12], and FastSLAM [13]. Cartographer algorithm can be performed with 2D LiDAR and 3D LiDAR, and consists of two main subsystems: local SLAM [14], global SLAM [15]. First, local SLAM is primarily used to estimate the current position of the robot and a map of the environment. It utilizes information related to the current submap to estimate the robot's path of travel and local features of the environment to determine the robot's local position. Second is global SLAM. Global SLAM is mainly used to estimate the overall path and map of the environment. It integrates multiple submaps and estimates the global position of the robot and a

* Corresponding author.
E-mail address: hclee@kumoh.ac.kr (H. Lee).

https://doi.org/10.1016/j.jestch.2025.101998
Received 22 October 2024; Received in revised form 31 January 2025; Accepted 3 February 2025
Available online 20 February 2025

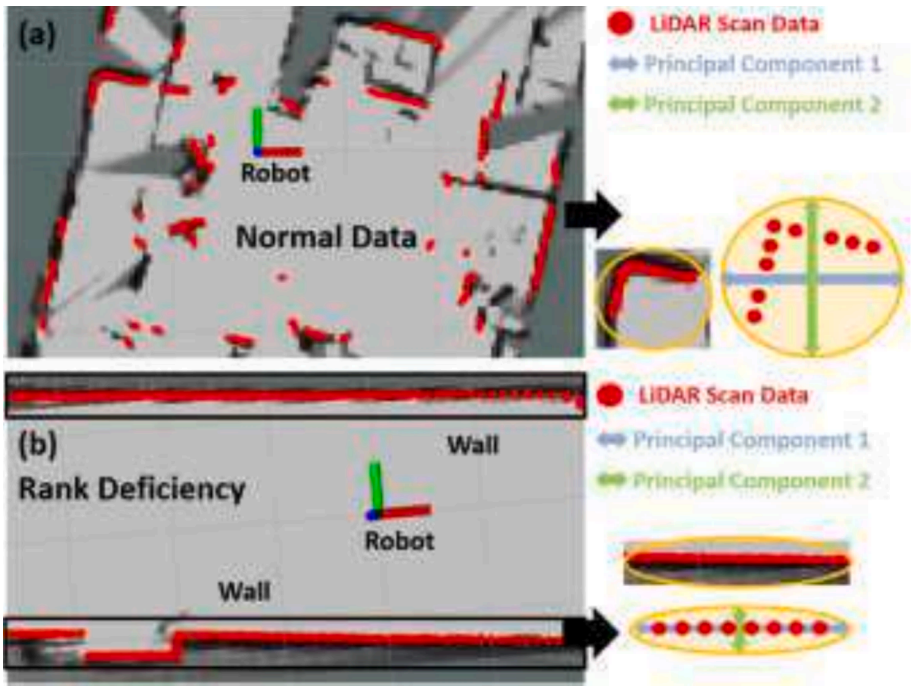

Fig. 1. An example of the problem of rank deficiency in corridor environments. (a) A general environment and general LiDAR scan data from it without rank deficiency. (b) A corridor environment and problematic LiDAR scan data from it with rank deficiency, which may decrease the accuracy of LiDAR scan matching for robot localization.

map of the environment. Next is EKF-SLAM, which uses the extended Kalman filter method. It uses the continuously updated Kalman filter state to estimate the robot's position and environment map. Finally, FastSLAM uses particle filters to model the environment and estimate the robot's position in a probabilistic manner. Localization algorithms are performed by employing various sensors, not limited to algorithms based on LiDAR, for position estimation. In addition to LiDAR, the most commonly used sensor is camera. Representative localization algorithms using camera include ORB SLAM [16], LSD SLAM [17], and MonoSLAM [18]. ORB SLAM has good performance for real-time 3D environment mapping and robot localization, and LSD SLAM is a large-scale direct SLAM that is suitable for large-scale environments. MonoSLAM is SLAM using a monocular camera. RGB-D SLAM uses both RGB image information and depth data from the camera and performs better than visual SLAM that lacks depth information.

Each sensor-based algorithm has strengths and weaknesses. LiDAR-based SLAM or localization algorithms are relatively more accurate than camera-based algorithms. But, LiDAR-based algorithms are vulnerable to dynamic environments or low variation in features such as corridors. Camera-based SLAM or localization algorithms are more robust to dynamic environments and corridors than LiDAR-based algorithms, but they tend to be less accurate overall. Therefore, other methods such as sensor fusion [19], optical flow [20], and consensus filter [21] are needed for robust localization algorithms in dynamic corridor environments. There is a lot of research in this area, and the most popular approach is to recognize dynamic objects and then remove them from the mapping process without using them [22,23]. However, these methods are computationally expensive and are likely to lead to incorrect mapping if dynamic objects are half-covered or poorly recognized. Also, these algorithms don't consider the corridor environment with the problem of rank deficiency. In addition, the accuracy of localization algorithms is vulnerable to decrease in dynamic corridor environments with data occlusion and rank deficiency problems. For accurate localization in dynamic corridor environments, it is necessary to use more than one localization algorithm and to take advantage of the strengths of each sensor-based localization algorithm.

Therefore, this paper mainly improves the localization algorithm accuracy for two environments: dynamic environment and corridor environment. To address the decrease in localization accuracy in corridor environments, an optical flow method is applied to perform position correction when position is being incorrectly estimated. To address the decrease in localization accuracy in the dynamic environment, adaptive consensus filter is applied to be used adaptively according to the number of recognized people. camera and LiDAR-based localization results are adaptively used to maximize the strengths of each sensor without removing dynamic objects. A rule-based optical flow method and a consensus filter method with rapid execution time are used to efficiently use the strengths of each sensor.

The contributions of this paper are as follows:

- The VACF which integrates object detection, optical flow, and an adaptive consensus filter has been proposed to achieve robust localization in dynamic corridor environments.
- The problem of the accuracy of the localization algorithm decreasing with the Rank Deficiency in corridor environments has been solved by optical flow-based approach.
- The problem of the accuracy of the localization algorithm decreasing with scan data occlusion due to dynamic objects which becomes worse in corridor environments has been solved by an object detection-based adaptive consensus filter.
- The proposed method was quantitatively compared with several other methods in real dynamic corridor environments and showed the better results than the others.

The remainder of this paper is organized as follows. Section 2 describe the problems of LiDAR-based localization in dynamic corridor environments. Section 3 describes the overall structure of the method proposed in this paper, the localization algorithms used, and defines the VACF proposed in this paper. Finally, Section 4 describe the robot hardware configuration and sensors used, compare the proposed algorithm with other algorithms in a static and dynamic corridor environment, and analyze the results. The environment is validated by applying

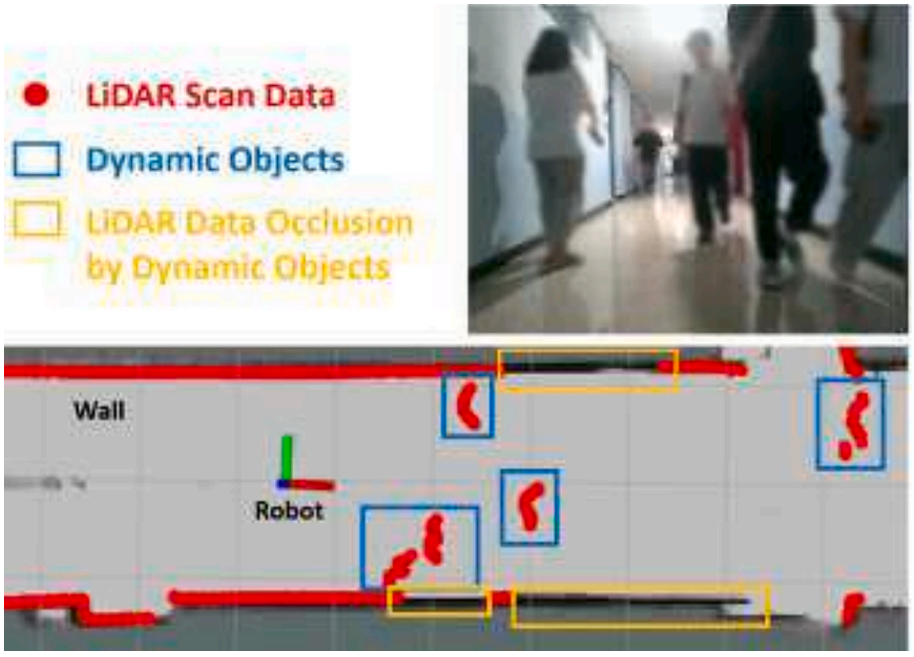

Fig. 2. An example of the problem of dynamic objects or human in corridor environments. The LiDAR scan data is significantly occluded by dynamic human, which may decrease the accuracy of LiDAR scan matching for robot localization.

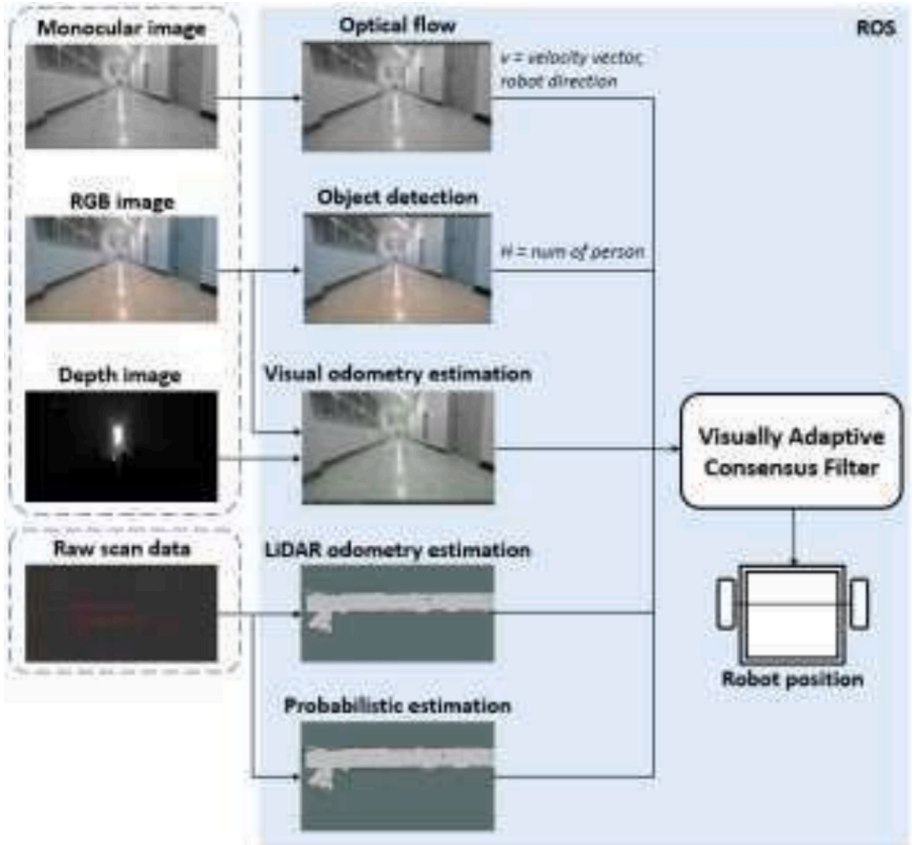

Fig. 3. Overall structure of the proposed method.

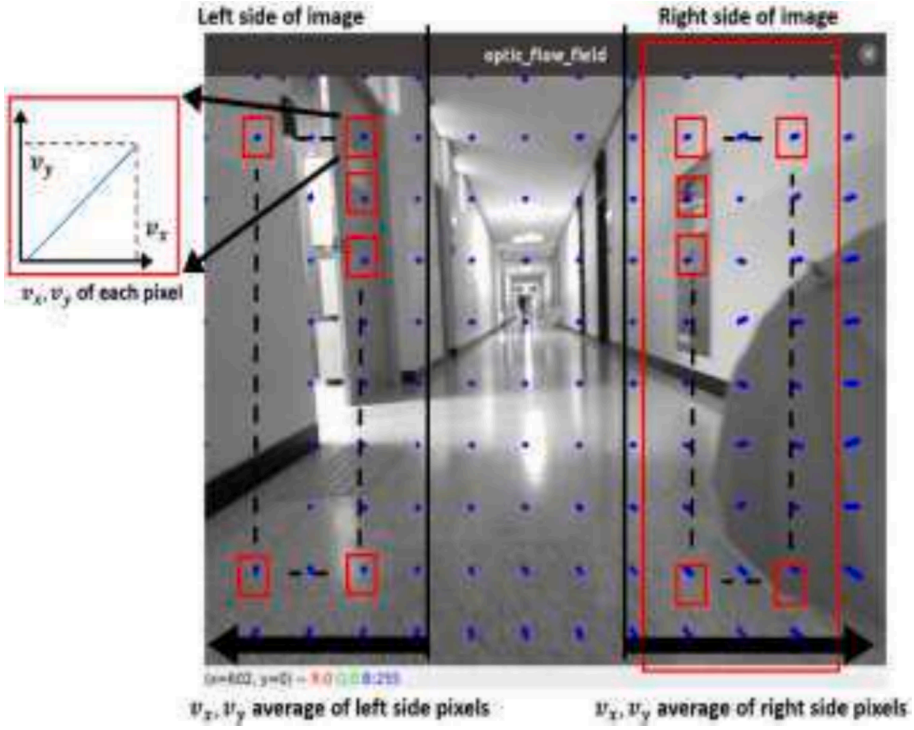

Fig. 4. An example of conducting the optical flow method and calculating velocity vectors. The blue points represent each feature, and each feature has an optical flow value in each x and y direction. The features on the left and right sides of the image are used to determine the robot's direction of movement. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Pseudo code of optical flow.

Algorithm 1 Optical flow node

Input: Image, Float threshold T
Output: *OpticalFlowValue*
1: Create Publisher for '*OpticalFlowValue*' ROS topic
2: $Vx_{left} = LeftSide(Image.resolution_{width})$
3: $Vx_{right} = RightSide(Image.resolution_{width})$
4: $OpticalFlowValue.Avg_{Vx_{left}} = \text{Mean}(Vx_{left})$
5: $OpticalFlowValue.Avg_{Vx_{right}} = \text{Mean}(Vx_{right})$
6: **if** $Avg_{Vx_{left}} > T$ and $Avg_{Vx_{right}} > T$ **then** // Calculate the current direction of the robot using optical flow values
7: *OpticalFlowValue*.Robot_direction = 'left'
8: **else if** $Avg_{Vx_{left}} < -T$ and $Avg_{Vx_{right}} < -T$ **then**
9: *OpticalFlowValue*.Robot_direction = 'right'
10: **else if** $Avg_{Vx_{left}} < -T$ and $Avg_{Vx_{right}} > T$ **then**
11: *OpticalFlowValue*.Robot_direction = 'forward'
12: **else if** $Avg_{Vx_{left}} > T$ and $Avg_{Vx_{right}} < -T$ **then**
13: *OpticalFlowValue*.Robot_direction = 'back'
14: **else**
15: *OpticalFlowValue*.Robot_direction = 'stop'
16: **end if**
17: Publish *OpticalFlowValue*

Table 2
Pseudo code of object detection.

Algorithm 2 Object detection node

Input: Image
Output: *NumofPerson*
1: Create Publisher for '*NumofPerson*' ROS topic
2: *NumofPerson* = ObjectDetection(Image)
3. Publish *NumofPerson*

the approach proposed in this paper in both static and dynamic environments for the same corridor environment.

2. Problem description

2.1. The problem of rank deficiency in corridor environments

As shown in Fig. 1(a), 2D LiDAR-based localization algorithms perform strongly in feature-rich environments. However, the performance of scan matching can be decreased in situations with low variation in the scan data, such as corridor environments.

As seen in Fig. 1(b), LiDAR scan data within a corridor environment is predominantly aligned along a single directional axis, leading to decreased localization accuracy. In the data presented in Fig. 1(a), the eigenvalues of each principal component 1 and principal component 2 are comparable. Conversely, in the data shown in Fig. 1(b), the data is concentrated along principal component 1, resulting in a significantly larger eigenvalue for principal component 1, while the eigenvalue for principal component 2 approaches zero. This issue, where scan data is insufficient in either the x or y direction, is identified as a rank deficiency problem. As demonstrated in Fig. 1(b), the matrix rank is reduced, leading to a rank deficiency problem. Therefore, for accurate LiDAR-based localization in feature-poor, long, narrow environments such as corridors, the integration of camera sensors or other supplementary methods is necessary.

2.2. The problem of dynamic objects in corridor environments

In addition to environments with sparse features, such as corridors, significant discrepancies between pre-constructed 2D maps and real-time LiDAR data can adversely impact localization accuracy. As shown in Fig. 1. (a), even in feature-rich environments, the presence of numerous dynamic objects poses a challenge for precise robot localization.

Furthermore, as shown in Fig. 2, in environment with fewer features,

Table 3
Pseudo code of visually adaptive consensus filter.

Algorithm 3 Visually adaptive consensus filter node

Input: *OpticalFlowValue, NumofPerson,* Float Max_Linear_Velocity$L$, Current_Robot_Direction C, Int threshold T^{min}_{human}, T^{max}_{human}, LiDAR_Odometry_Estimation *LOE,* Probabilistic_Localization *PL,* Visual_Odometry_Estimation *VOE*
Output:Avg_{final}
1: **if** $NumofPerson \leq T^{min}_{human}$ **then**
2: $a_i = Compute\_a_i(LOE, PL)$ // Compute each a_i using (5), Using the LiDAR-based localization results
3: **else if** $NumofPerson \geq T^{max}_{human}$ **then** // Using the visual odometry estimation
4:$a_i = Compute\_a_i(VOE)$
5: **else**
6: $a_i = Compute\_a_i(LOE, PL, VOE)$ // Using all the algorithms
7: **end if**
8: **if** *OpticalFlowValue.*Robot_direction $\neq C$ **then**
9: $Avg_{final} = \frac{1}{N}\sum(a_i + \frac{v}{L})$ // Equation (6), using optical flow values when the robot moves in wrong direction
10: **else**
11: $Avg_{final} = \frac{1}{N}\sum(a_i)$ // Using a normal consensus filter without optical flow values when the robot moves in right direction
12: **end if**

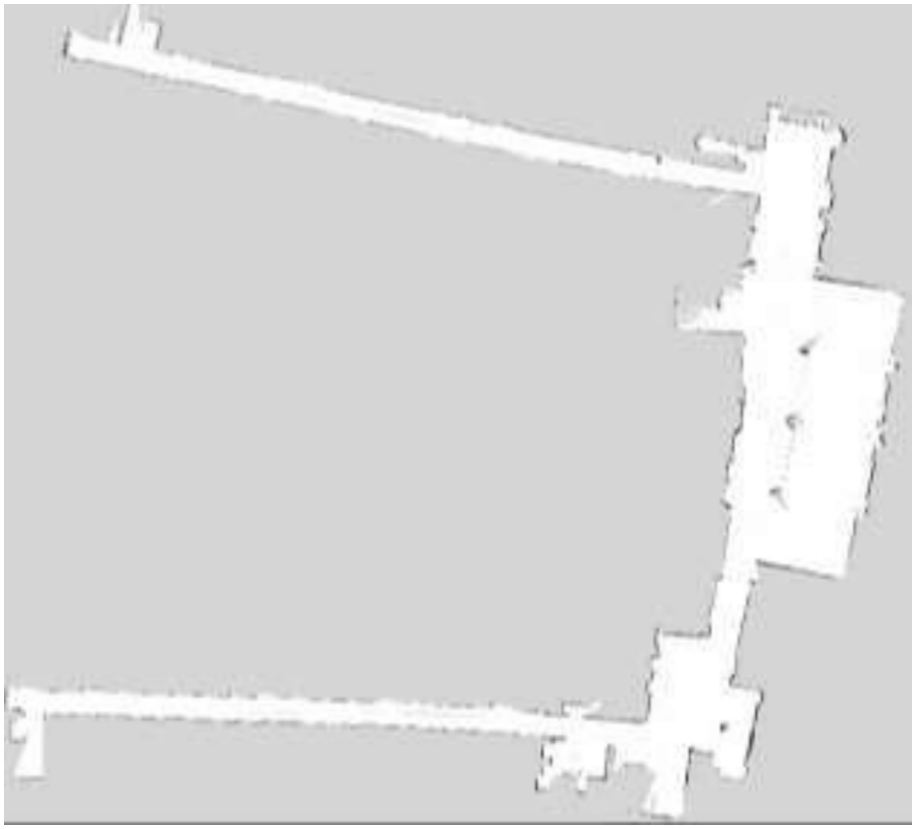

Fig. 5. The grid map obtained from the first environment.

the occlusion of scan data caused by dynamic objects exacerbates the difficulty of accurate localization. Therefore, to achieve robust localization in environments characterized by sparse features, such as corridors, and in dynamic environments with moving objects, an adaptive consensus filter that leverages the strengths of both camera-based and LiDAR-based localization algorithms is essential.

3. Proposed method

3.1. Overview of the proposed method

The overall structure of the proposed method is shown in Fig. 3. Three localization algorithms are used in this paper: LiDAR odometry, monte carlo localization (MCL), and visual odometry. Both LiDAR and camera-based localization algorithms are used to improve the accuracy of the localization. In addition, a computer vision-based algorithm called optical flow is utilized to solve the problem in a corridor environment. Optical flow will be described in more detail in Section 3.2. Localization based on LiDAR odometry, MCL, and visual odometry algorithms are performed independently, and optical flow is performed by the camera. To further improve performance in dynamic environments, dynamic objects are recognized through object detection algorithm, and a consensus filter is adaptively applied to the localization results of each algorithm with optical flow according to the number of dynamic objects. This will be discussed in detail in Section 3.3. The robot operating system [24] (ROS) is applied to all the algorithms performed.

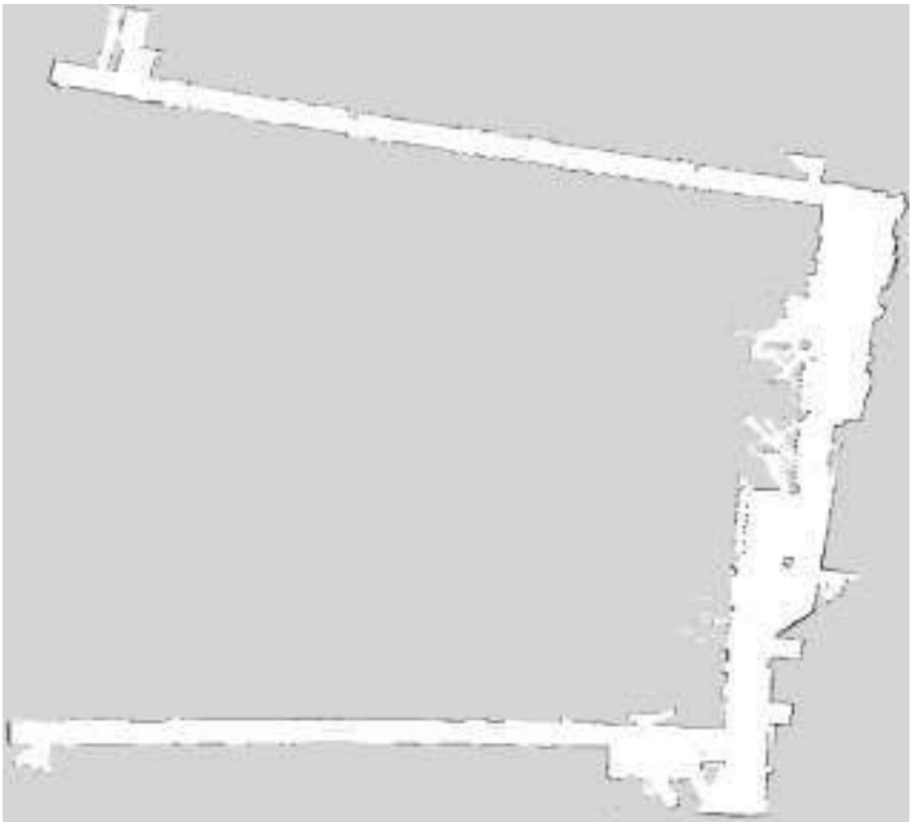

Fig. 6. The grid map obtained from the second environment.

Fig. 7. Robot system configuration.

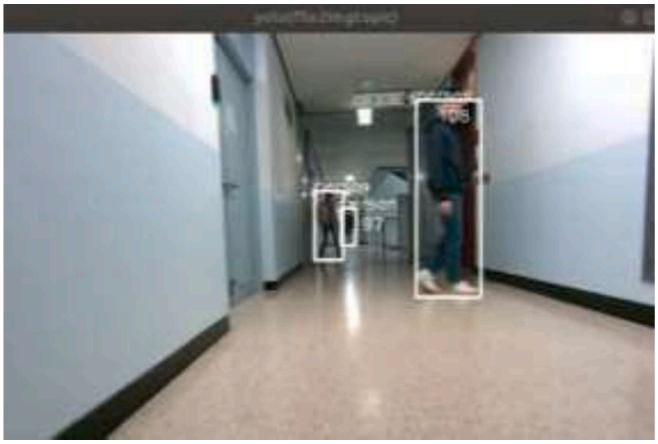

Fig. 8. The result of performing object detection. The dynamic object, i.e. the number of people, obtained through object detection is used as input to the VACF.

1) LiDAR Odometry Estimation (LOE).

The cartographer algorithm is used for LiDAR odometry. Basically, cartographer produces high-performance mapping and localization results compared to other algorithms. It can be applied to various sensors such as 2D LiDAR and 3D LiDAR, and has the advantage of being able to adjust the appropriate parameters according to the surrounding environment, so it can improve performance according to the situation. In addition, the loop closing function, which detects the difference between the current area and the previously visited area, enables position correction when a cumulative error in localization occurs. In this paper, cartographer is performed using 2D LiDAR and use cartographer for both mapping and localization. Maps based on localization are created with cartographer, and maps created with both cartographer and MCL are used. The maps used will be described in Section 4. In the proposed method, LiDAR odometry is used in the case where dynamic objects are not abundant because LiDAR-based robot localization algorithms are more robust to static environments than dynamic environments.

2) Probabilistic Localization (PL).

MCL algorithm is one of the most popular algorithms for robot localization. MCL uses a particle filter to determine the position of the robot, and particles with probabilities are used to represent the potential positions of the robot. It starts with the assumption that the robot has no

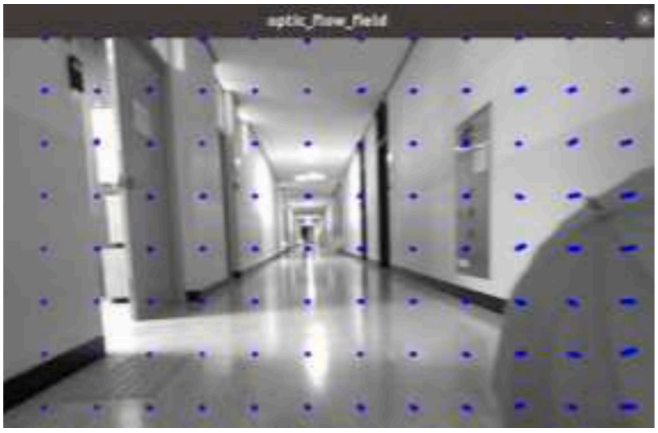

Fig. 9. The result of performing optical flow. The robot's direction and robot velocity vector obtained through optical flow are used as input to the VACF.

information about its current position and could be at any point. The robot resamples the particles using recursive Bayesian estimation whenever it detects an object in its environment. By repeating this process, it gradually gets closer to its true position, and the larger the number of particles, the more accurate the localization. Among the various MCL algorithms, the EMCL2 (MCL with expansion resetting) [25] algorithm is used, which has an expansion resetting feature. EMCL2 performs localization based on the map created by the cartographer above. As with LiDAR odometry, MCL is used in cases where dynamic objects are few and far between, as it performs localization based on LiDAR.

3) Visual Odometry Estimation (VOE).

Finally, ORB SLAM2 [26] algorithm is used for visual odometry. ORB SLAM2 is a representative visual SLAM algorithm using a camera. It uses the ORB (oriented FAST and rotated BRIEF) extraction method to extract features of the environment. It is mainly divided into three threads: tracking, local mapping, and loop closing. ORB SLAM2 has several build types. In this paper, the RGB-D build version is utilized, which uses depth information to show more accurate results than the monocular and stereo versions. Unlike LiDAR-based localization algorithms, visual odometry is used in cases with a lot of dynamic objects because it is less affected by the dynamic environment.

3.2. Optical flow method

The optical flow method is one of the most important concepts used in image processing and computer vision. Optical flow uses the difference between the previous and current frames obtained from the input image to calculate the movement of each pixel through the relationship between each pixel to determine the motion of objects. The optical flow method is used in a variety of ways. The Lucas-Kanade [27] and Horn-Schunck [28] methods are representative. The Lucas-Kanade method has the advantage of being computationally lightweight and can be processed in real time. It can accurately trace local behavior, and provides stable performance with relatively simple implementation. On the other hand, the Horn-Schunck method can smoothly track the motion in the whole image and maintain the continuity of the motion. However, it is computationally intensive, which can be burdensome for real-time processing. Therefore, due to the processing time, the Lucas-Kanade method is chosen in this paper. The Lucas-Kanade algorithm has the following three conditions as prerequisites. 1) Constant brightness. 2) The flow should be locally smooth. 3) The optical flow in the window region N(x, y), centered at pixel (y, x), is the same, i.e., it is assumed that neighboring pixels in the frame move together in the same direction and that the difference in brightness between the previous frame and the current frame is zero or very small. The expression for the first of these three prerequisites can be expressed as follows:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{1}$$

In the equation (1), $I(x,y,t)$ represents the brightness at time t. Prerequisite 2 asserts that dt assumes a considerably small value to ensure the smoothness of time t with minimal motion. Consequently, by employing Taylor's formula on (1) and disregarding terms of higher

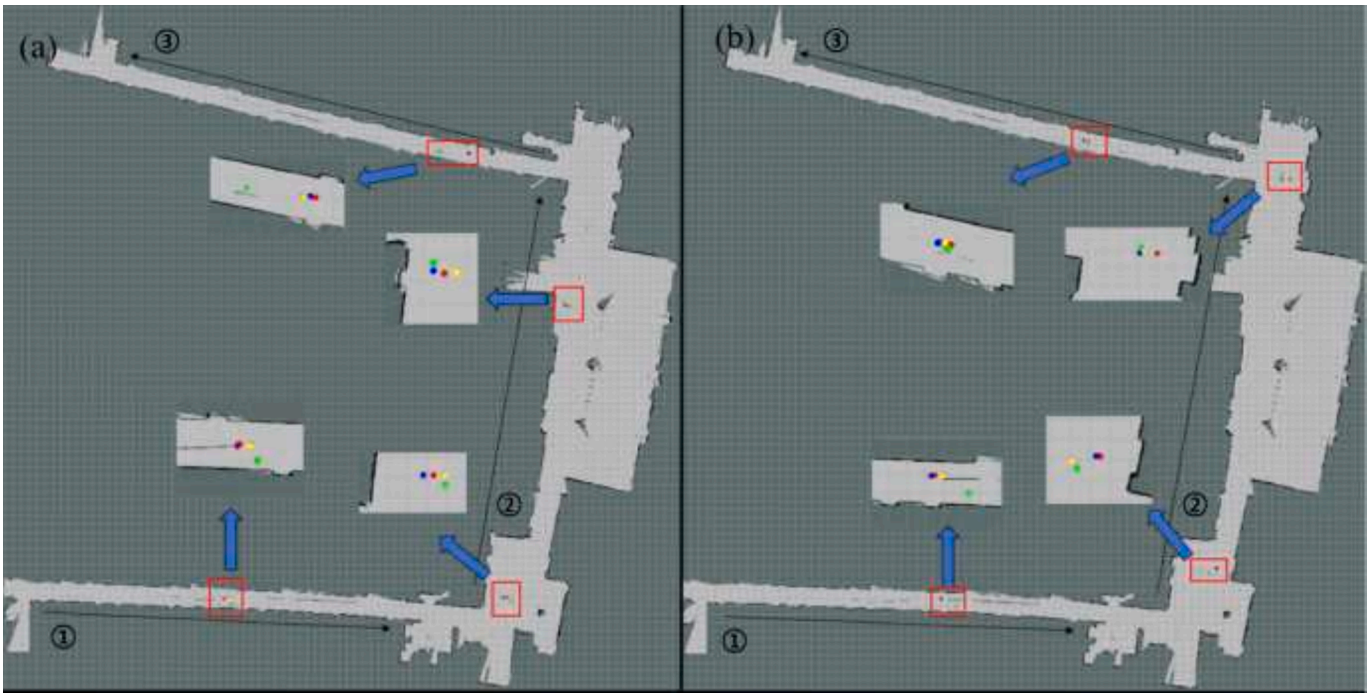

Fig. 10. Localization results with the map of the first environment. Red corresponds to MCL, blue to LiDAR odometry, green to visual odometry, and yellow to VACF proposed in this paper. (a) Results of each localization in a static corridor environment. (b) Results of each localization in a dynamic corridor environment. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

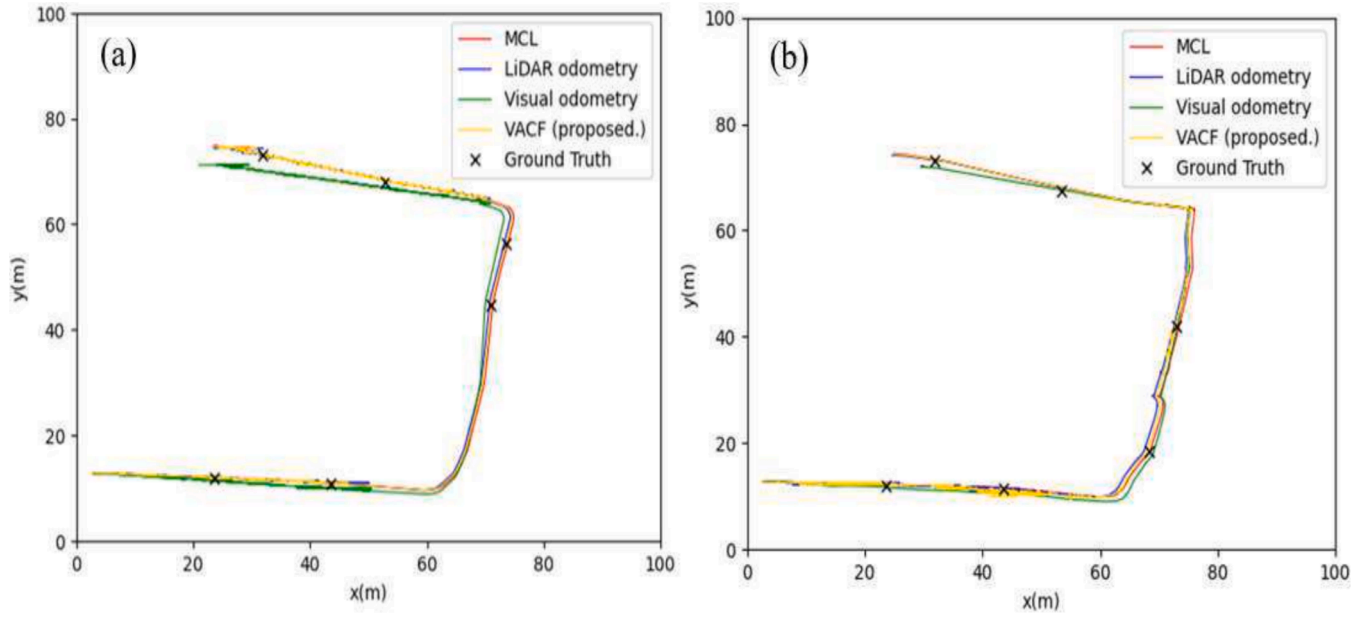

Fig. 11. Results of each localization algorithm trajectory in a static and dynamic corridor environment including proposed method used map of the first environment. Red corresponds to MCL, blue to LiDAR odometry, green to visual odometry, yellow to VACF proposed in this paper, and X is ground truth point. (a) Results of each localization trajectory in a static corridor environment. (b) Results of each localization trajectory in a dynamic corridor environment. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 4
Quantitative average comparison results for static corridor environments used map of the first environment.

–	X(m)	Y(m)	θ(angle)
MCL[25]	0.0792	0.0801	0.825
LiDAR Odometry[11]	0.108	0.0985	0.604
Visual Odometry[26]	1.032	3.137	1.2
VACF(Proposed.)	**0.063**	**0.079**	**0.563**

Table 5
Quantitative average comparison results for dynamic corridor environments used map of the first environment.

–	X(m)	Y(m)	θ(angle)
MCL[25]	1.3904	0. 3878	1.298
LiDAR Odometry[11]	1.5445	0.394	1.139
Visual Odometry[26]	1.1045	1.4301	1.607
VACF (Proposed.)	**0.081**	**0.0527**	**1.0304**

Table 6
Quantitative average comparison results for static corridor environments used map of the second environment.

–	X(m)	Y(m)	θ(angle)
MCL[25]	0.0908	0.0736	0.564
LiDAR Odometry[11]	**0.0301**	0.0605	0.509
Visual Odometry[26]	0.089	0.0825	1.032
VACF (Proposed.)	0.0303	**0.0412**	**0.387**

order, equation (2) is derived in the subsequent manner:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0 \tag{2}$$

Define $\partial I/\partial x$ as I_x, $\partial I/\partial y$ as I_y, $\partial I/\partial t$ as I_t, each is represented by the brightness gradient of one pixel. v_x, v_y are the optical flow solutions in each x, y direction. Expressing (2) as a matrix is equivalent to (3), and expressing (3) as $A\boldsymbol{V} = c$ finally yields $\boldsymbol{V}(v_x, v_y)$, the optical flow solution.

Table 7
Quantitative average comparison results for dynamic corridor environments used map of the second environment.

–	X(m)	Y(m)	θ(angle)
MCL[25]	0.0467	0.0609	0.49
LiDAR Odometry[11]	0.039	0.0473	0.348
Visual Odometry[26]	0.034	0.56	1.01
VACF (Proposed.)	**0.0297**	**0.03409**	**0.288**

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ I_{t2} \\ \vdots \\ I_{tn} \end{bmatrix} \tag{3}$$

$$A^T A\mathbf{V} = A^T c \rightarrow \mathbf{V} = (A^T A)^{-1} A^T c \tag{4}$$

An example and the pseudo code of conducting the optical flow method and calculating velocity vectors are shown in Fig. 4 and Table 1, respectively. The final velocity vectors v_x and v_y are used for robot velocity input commands in each of the x and y directions. Within a single image frame, the values of the pixels on the left and right sides of the frame are calculated in the form of a matrix, which is represented by the value of the velocity vector. These two velocity values are used to obtain four states about the robot's direction: "left", "right", "forward", and "back".

In summary, the velocity vector value $\mathbf{V}(v_x, v_y)$ and the robot's direction information is obtained by performing optical flow. The robot's direction information is used to determine whether the current localization is incorrect, and the velocity vector value is used to correct the robot's position when it is determined that the localization is incorrect. For example, in a long corridor section, if the robot is currently moving forward and the localization result is increasing backward or stuck, it is determined that the localization is incorrect and the velocity vector value is applied appropriately to correct the incorrect position. This will be discussed in more detail in Section 3.3.

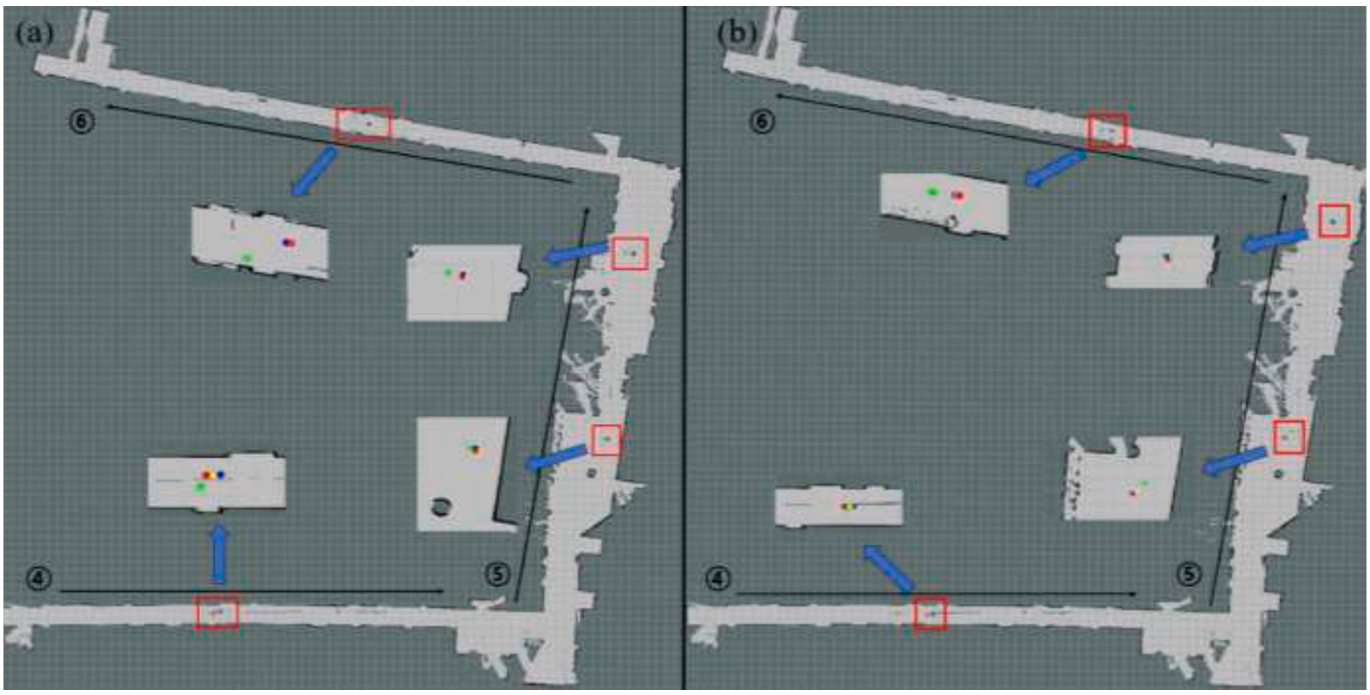

Fig. 12. Localization results with the map of the second environment. Red corresponds to MCL, blue to LiDAR odometry, green to visual odometry, and yellow to VACF proposed in this paper. (a) Results of each localization in a static corridor environment. (b) Results of each localization in a dynamic corridor environment. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

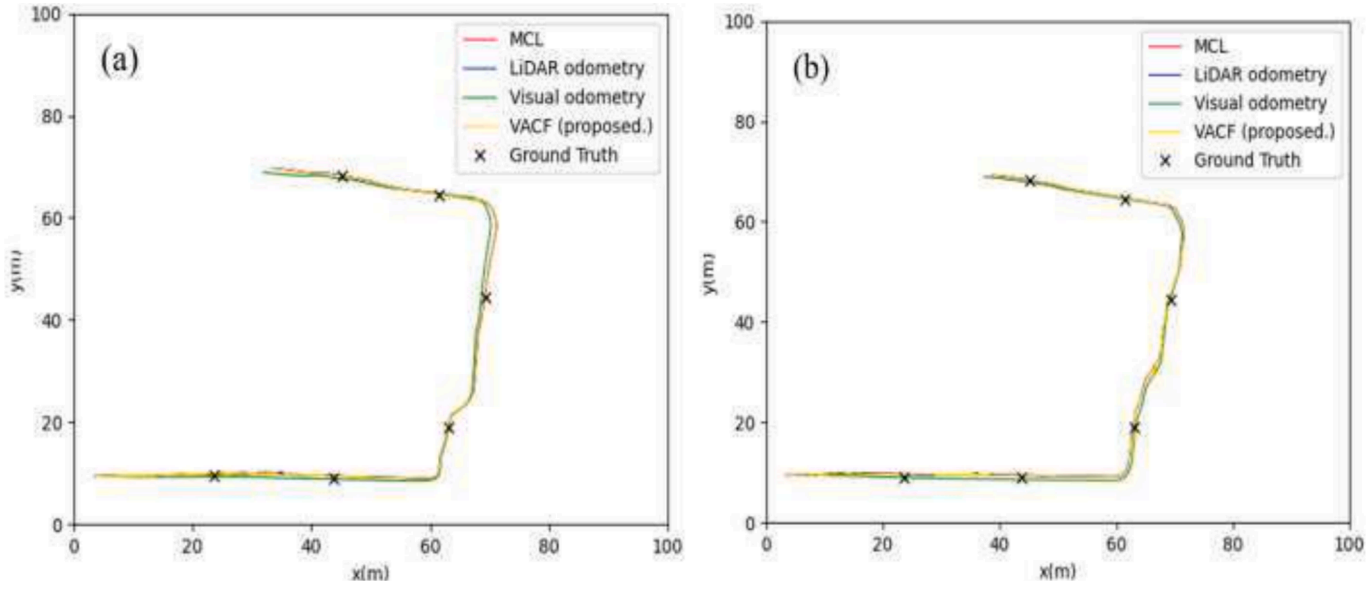

Fig. 13. Results of each localization algorithm trajectory in a static and dynamic corridor environment including proposed method used map of the second environment. Red corresponds to MCL, blue to LiDAR odometry, green to visual odometry, yellow to VACF proposed in this paper, and X is ground truth point. (a) Results of each localization trajectory in a static corridor environment. (b) Results of each localization trajectory in a dynamic corridor environment. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.3. Visually adaptive consensus filter

Consensus filter algorithms have gathered a lot of research and attention in areas such as multi-sensor fusion and sensor networks. Consensus filters can be applied in several ways, among which the average consensus algorithm [29] is used. The average consensus filter is a commonly used method for solving for an average solution. In a network with N nodes, assume that each node, denoted by i, has a state a_i. The goal is to find the average solution of these node states. Each node in the Average consensus filter algorithm starts by initializing its consensus state to $a_i = a_i(0)$, and then iterates to compute the average value. The single-step consensus formula is given by (5).

$$a_i(K) = a_i(K-1) + \epsilon \sum^{j \in N_i} (a_j(K-1) - a_i(K-1)) \tag{5}$$

$$Avg_{final} = \frac{1}{N} \sum_{i=1}^{N} (a_i + (\frac{v}{L})) \tag{6}$$

where $a_j(K-1)$ is the previous value of the neighboring node, $a_i(K-1)$ is the previous value of its own node, the number of nodes N, the current state K, and the ratio parameter ∈. Repeat for the number of nodes N to find the current value a_i for that node, and use equation (6) to derive the final consensus mean value Avg_{final}. We reformulate the consensus average formula by adding the v/L. where v is the optical flow solution obtained in Section 3.2, and L is the maximum linear velocity parameter of the robot. This can be adjusted according to the maximum linear velocity of the used robot.

In addition, to deal with dynamic environments, object detection algorithm is used to recognize dynamic objects such as people, and a consensus filter is adaptively applied according to the number of people. If there is no or single person in the environment, the LiDAR-based localization algorithm is mainly performed because it can overcome the LiDAR data distortion and occlusion problems when their portion is not large. However, if there are many people in the environment, the VIO estimation algorithm is conducted because the VIO estimation algorithm is more robust than the LiDAR-based localization algorithm when the LiDAR data distortion and occlusion portion is large. Except for these two cases, all the algorithms are conducted to find the consensus on robot localization. The details of this process are included in Table 3. Table 3 shows the pseudo code for the VACF. The parameters T_{human}^{min}, T_{human}^{max} can be appropriately adjusted by experiments.

In Table 3, *NumofPerson* is the number of dynamic objects, i.e. people, currently recognized by object detection, N is the number of localization algorithms used in the final result, and the inputs of a_i to be computed are represented by PL, LOE, and VOE, respectively.

The value obtained by performing the optical flow is used as one of the inputs. In addition, the number of currently recognized dynamic objects (people) obtained through object detection in Table 2 and the localization results of PL, LOE, and VOE are used as inputs, respectively. Depending on the value of the parameter *NumofPerson*, the inputs to the Final Position are changed. Equations (5) and (6) correspond to the Compute the Final Position in the pseudo code in Table 3. The VACF function is the function that performs all the calculations of the proposed method, including calculating the final position of the robot. The final output is the Avg_{final} variable, which is the position of the robot applied with the VACF.

4. Experimental results

4.1. Environmental setup

The experimental environment is compared for static and dynamic environments for each map of the environment. Each map of the environment contains two long corridor segments with few features. The maps used in the experiments are shown in Fig. 5 and Fig. 6. Both environments are large-scale, covering an area of about 3000 square meters, and include long corridor sections of about 60 m. To validate the proposed method for dynamic environments, it is compared for both static and dynamic environments for each map. The dynamic corridor environment is defined as an environment with the following conditions: 1) one or more long corridor segments. 2) one or more dynamic objects such as a person or a group with unpredictable trajectories.

4.2. Robot and sensor system HW/SW configuration

The mobile robot and sensor system configuration for real experiments is shown as Fig. 7. The sensor system consisted of a LiDAR sensor and a camera sensor on the top front. The camera was used to perform dynamic object detection, optical flow, and visual odometry, and the LiDAR was used to perform MCL and LiDAR odometry localization algorithms. The entire algorithms were implemented based on ROS and conducted on a laptop.

4.3. Experimental results and analysis

The results of the object detection algorithm and optical flow performed using the camera are shown in Fig. 8 and Fig. 9, respectively. The number of dynamic objects currently being recognized is obtained through object detection, and the velocity vector value and robot direction are obtained through optical flow. The number of dynamic objects, velocity vector values, and robot direction are all used as inputs to the VACF.

Applying the map of the first environment shown in Fig. 5, the localization results in a static environment are shown in Fig. 10(a). Fig. 10(a) shows the localization results of each algorithm using ROS visualization tool in a static environment, and Fig. 11(a) shows the corresponding trajectory results in a static environment used map of first environment. For comparison of results, MCL, LiDAR Odometry, and Visual Odometry are used. First, MCL is a Rao-Blackwellized particle filter (RBPF)-based localization algorithm and popular when a LiDAR sensor is used. Second, the LiDAR odometry, Cartographer developed by Google, is also a widely-used algorithm when a LiDAR sensor is used. Finally, the visual odometry algorithm, ORB SLAM2, is also a widely-used algorithm when a visual sensor is used. The most used and powerful algorithms are used for the comparison. In Fig. 10 and Fig. 11, red corresponds to MCL, blue to LiDAR odometry, green to visual odometry, and yellow to proposed VACF in this paper. Except for the proposed method in this paper, all of localization algorithm results show pure localization results without applying the VACF. Except for the visual odometry algorithm, which is not affected by the corridor environment, the results of MCL and LiDAR odometry algorithms are affected, although the difference is small.

The start point is at the far end of corridor 1, and the destination is at the end of corridor 3. The bottom left corner in Fig. 10 is the same start point as the bottom left corner in Fig. 11, and the bottom left corner in Fig. 11 corresponds to the origin. The ground truth values were obtained by specifying a few points in advance. The quantitative comparison results for localization in a static environment used map of the first environment are shown in Table 4. The result of comparing the predicted localization result (x, y, θ) with the pre-specified GT using the RMSE (Root Mean Square Error), the proposed method in this paper shows the best results, and secondly, the MCL algorithm shows good results overall. In Tables 4–7, bold indicates the best result and underline indicates the second-best result. The RMSE equation used for validation is shown in (7). For x, y, θ (pred), calculate the RMSE result for each environment by comparing the error to the predetermined ground truth (gt) at the same timestamp.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i}^{N}(pred_i - gt_i)^2} \tag{7}$$

The results of localization in a dynamic corridor environment used map of the first environment are shown in Fig. 10(b) and Fig. 11(b). Unlike a static environment with no dynamic objects or changes in the surrounding environment, differences exist in whether the door is open or closed, changes in static obstacles, and dynamic objects. A dynamic environment includes one or more dynamic objects, such as people or groups, with unpredictable trajectories. The colors corresponding to each algorithm are the same as in the static environment, and it can be seen that the method proposed in this paper shows the best results. The quantitative comparison results for localization in dynamic environments used map of the first environment are shown in Table 5.

Compared to the quantitative comparison results in a static environment, the presence of dynamic objects and changes in static obstacles definitely affect the robot localization. In summary, the overall results show that the method proposed in this paper is effective for robust localization in dynamic corridor environments.

The localization results of each algorithm with map of the second environment shown in Fig. 6 are shown in Fig. 12 and Fig. 13. Second environment is a slightly larger environment than first environment and similarly has two corridor segments. Fig. 12(a) shows the localization results of each algorithm in a static environment, and Fig. 12(b) shows the localization results of each algorithm in a dynamic environment using ROS visualization tool. The trajectory results of each algorithm's localization in the static and dynamic environments used map of the second environment are shown in Fig. 13. The quantitative average comparison results of applying map of the second environment in a static environment are shown in Table 6. And the quantitative average comparison results of used map of the second environment in a dynamic environment are shown in Table 7. The results in Table 6 and Table 7 show that the proposed method in this paper is more effective than other methods in both static and dynamic corridor environments.

5. Conclusion

In this paper, a robust robot localization method with the VACF was proposed to solve the problem of robot localization in dynamic corridor environments. The VACF consisted of LiDAR odometry estimation, probabilistic localization, visual odometry estimation, optical flow recognition, object detection and adaptive consensus filters. The proposed method was tested by real-world experiments in dynamic corridor environments and showed better accuracy than other existing methods when compared to pre-determined ground truth points. In future work, the proposed method will be extended to deal with other complex environments such as glassy environments or illumination changing.

CRediT authorship contribution statement

Suhyeon Kang: Writing – original draft, Visualization, Software, Methodology, Data curation. **Heoncheol Lee:** Writing – review & editing, Validation, Resources, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Korea Institute of Marine Science & Technology Promotion (KIMST) funded by the Ministry of Trade, Industry and Energy in 2023 (Project Number 20210630), and in part by the MSIT(Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program(IITP-2024-RS-2024-00437190, 50 %) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jestch.2025.101998.

References

[1] L. Zhang, et al., Indoor mobile robot localization applying IMU/stereo camera/LiDAR and graph-based optimization, IEEE Sens. J. 24 (13) (2024) 21466–21478.
[2] Z. Wan, et al., MS-Mapper: A LiDAR mapping system by switching between mobile and stationary configurations, IEEE Sens. J. 24 (5) (2024) 6655–6665.
[3] Cai, et al., Improving SLAM techniques with integrated multi-sensor fusion for 3D reconstruction, Sens. 24 (7) (2024) 2033.
[4] P. Liu, Y. Bi, J. Shi, T. Zhang, C. Wang, Semantic-assisted LIDAR tightly coupled SLAM for dynamic environments, IEEE Access 12 (2024) 34042–34053.
[5] Z. Zou, et al., DY-LIO: Tightly-coupled LiDAR-inertial odometry for dynamic environments, IEEE Sens. J. 24 (21) (2024) 34756–34765.
[6] Z. Shen, J. Wang, C. Pang, Z. Lan, Z. Fang, A LiDAR-IMU-GNSS fused mapping method for large-scale and high-speed scenarios, Meas. 225 (2024) 113961.
[7] Q. Ji, Z. Zhang, Y. Chen, E. Zheng, DRV-SLAM: an adaptive real-time semantic visual SLAM based on instance segmentation toward dynamic environments, IEEE Access 12 (2024) 43827–43837.
[8] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with rao-blackwellized particle filters, IEEE Trans. Robot 23 (1) (2007) 34–46.
[9] S. Kohlbrecher, O. von Stryk, J. Meyer and U. Klingauf, A flexible and scalable SLAM system with full 3D motion estimation, in: IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, Japan, 2011, pp. 155–160.
[10] F. Dellaert, D. Fox, W. Burgard and S. Thrun, Monte Carlo localization for mobile robots, in: Proc. IEEE International Conference on Robotics and Automation, Detroit, MI, USA, 1999, pp. 1322–1328.
[11] W. Hess, D. Kohler, H. Rapp and D. Andor, Real-time loop closure in 2D LIDAR SLAM, in: IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 1271-1278.
[12] S. Yavuz, Z. Kurt and M. S. Bicer, Simultaneous localization and mapping using Extended Kalman Filter, in: IEEE 17th Signal Processing and Communications Applications Conference, Antalya, Turkey, 2009, pp. 700-703.
[13] M. Montemerlo, S. Thrun, D. Koller, W.B. FastSLAM, A factored solution to the simultaneous localization and mapping problem, in: Proc. of AAAI02, 2002.
[14] C. Chen, et al., Trajectory optimization of LiDAR SLAM based on local pose graph, in: In: China Satellite Navigation Conference (CSNC), 2019, pp. 360–370.
[15] D. Droeschel, S. Behnke, Efficient continuous-time SLAM for 3D lidar-based online mapping, in: IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 5000–5007.
[16] R. Mur-Artal, J.M.M. Montiel, J.D. Tardós, ORB-SLAM: a versatile and accurate monocular SLAM system, IEEE Trans. Robot 31 (5) (2015) 1147–1163.
[17] Engel, Jakob, Thomas Schöps, Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. in: European conference on computer vision, 2014, pp. 834–849.
[18] A.J. Davison, I.D. Reid, N.D. Molton, O. Stasse, MonoSLAM: real-time single camera SLAM, IEEE Trans. Pattern Anal. Mach. Intell 29 (6) (2007) 1052–1067.
[19] H. Chang, J. Choi and M. Kim, Probabilistic Localization of Service Robot by Sensor Fusion, in: SICE-ICASE International Joint Conference, Busan, Korea (South), 2006, pp. 3626–3631.
[20] H. Chao, Y. Gu, M. Napolitano, A survey of optical flow techniques for UAV navigation applications, in: International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 2013, pp. 710-716.
[21] R. Olfati-Saber, J. S. Shamma, Consensus Filters for Sensor Networks and Distributed Sensor Fusion, in: Proc of the 44th IEEE Conference on Decision and Control, Seville, Spain, 2005, pp. 6698-6703.
[22] J. Konno, Y. Ando, Improvement of 3D-SLAM Accuracy by Removing Moving Objects on 3D-LiDAR Point Cloud Using Image Recognition in Web Camera, in: 22nd International Conference on Control, Automation and Systems (ICCAS), Jeju, Korea, 2022, pp. 527-531.
[23] Y. Wang, Z. Wang and X. Wu, An Efficient and Accurate 3D SLAM Method for Dynamic Environment, in: 4th International Conference on Robotics and Computer Vision (ICRCV), Wuhan, China, 2022, pp. 148–153.
[24] Q. Morgan, et al. ROS: an open-source Robot Operating System. ICRA workshop on open source software. 3 (3.2) (2009).
[25] R. Ueda, T. Arai, K. Sakamoto, T. Kikuchi and S. Kamiya, Expansion resetting for recovery from fatal error in Monte Carlo localization - comparison with sensor resetting methods, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, 3, 2004, pp. 2481-2486.
[26] R. Mur-Artal, J.D. Tardós, ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras, IEEE Trans. Robot 33 (5) (2017) 1255–1262.
[27] Z. Wang, X. Yang, Moving Target Detection and Tracking Based on Pyramid horn-Kanade Optical Flow, in: IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, China, 2018, pp. 66–69.
[28] O. A. Omer, Region-based Horn-Schunck optical flow estimation, in: Japan-Egypt Conference on Electronics, Communications and Computers, Alexandria, Egypt, 2012, pp. 73–78.
[29] R. Olfati-Saber, J.A. Fax, R.M. Murray, Consensus and cooperation in networked multi-agent systems, Proc. IEEE 95 (1) (2007) 215–233.