Study on Risk Reduction in Localization of Cloud-Supported Autonomous Mobile Robots

Mao Nabeta[1], Kazuteru Tobita[2,*], Seiya Nakamura[1] and Kazuhiro Mima[3]

[1] *Graduate school, Shizuoka Institute of Science and Technology*
[2] *Department of Mechanical Engineering, Shizuoka Institute of Science and Technology*
[3] *Department of Electrical and Electronic Engineering, Shizuoka Institute of Science and Technology*
* Corresponding author E-mail: tobita.kazuteru@sist.ac.jp

October 17, 2024

Abstract

In recent years, outdoor robotics applications have trended towards the "cloud robotics" approach, which involves offloading processing to other computation-rich locations via communication with the cloud, due to the increasing complexity of the tasks the robots are required to perform. However, network conditions in outdoor environments are often volatile, leading to significant performance degradation or unstable behavior in robots due to poor communication with the cloud. To address this issue, we propose a method that combines minimal self-localization capabilities on the robot side with advanced self-localization processing on the cloud side, integrating and interpolating the two. This paper discusses implementing 2D self-localization on the robot and 3D self-localization on the cloud, clarifies the characteristics of each and proposes a fusion method that leverages the features of both self-localization techniques for high accuracy and robustness against communication failures. Adaptive Monte Carlo localization (AMCL), which is a common algorithm for two-dimensional self-localization, was used on the robot side. We implemented two fusion methods: a time-varying weighted average (TVWA) and an unscented Kalman filter (UKF), and showed that these methods can reduce the maximum error compared to using the AMCL alone by approximately 0.11 m with the TVWA and by approximately 0.15 m with the UKF.

Keywords:

Autonomous mobile robots, self-localization, sensor fusion, cloud robotics, communication failure resilience

1 Introduction

In the development of autonomous mobile robots, accurately determining the robot's self-position is especially important [1–3]. Currently, autonomous mobile robots that move on the ground mainly use 2D light detection and ranging (LiDAR) for planar scan matching, as noted in [4–8]. However, detecting objects higher or lower than the plane where the 2D-LiDAR is mounted is difficult, leading to blind spots. This is particularly problematic in outdoor or indoor environments with few distinct features, where self-localization results can significantly diverge from the actual position.

Cost reduction of 3D-LiDAR and acceleration of personal computers (PCs), along with progress in open-source resources related to three-dimensional self-localization, are bringing 3D simultaneous localization and mapping (SLAM) closer to being a practical alternative. However, a challenge remains: the computational cost for three-dimensional self-localization is still significantly higher than that for two-dimensional self-localization.

To address this issue, efforts are being made to enhance processing capabilities by equipping robots with high-performance PCs. However, when a large number of robots is required, such as for agricultural tasks or delivery tasks, this approach can be cost-prohibitive [9]. Consequently, the concept of "cloud robotics" has emerged as an approach to perform three-dimensional self-localization processing while limiting the performance of the PCs mounted on the robots [10]. This approach aims to improve processing capabilities by performing three-dimensional self-localization processing in computation-rich locations and receiving the results via communication networks, offering a promising method for efficiently operating numerous robots.

However, cloud robotics faces several challenges. First, because it relies on data transmission over networks, network latency can become a significant issue for robots [11]. Second, processing in the cloud requires transmitting information externally, increasing the risk of data leakage and security breaches. Third, complete reliance on the cloud can lead to communication breakdowns in poor network conditions, potentially immobilizing the robots.

Our previous research, based on Annex A of ISO 13482, examined 85 risk factors associated with autonomous mobile robots, revealing that several high-risk factors (risk scores above 18) could result from loss of self-localization. We validated the validity of this risk assessment with actual robot tests, confirming that loss of self-localization leads to unstable robot behavior, posing dangers to users and workers.

Related studies have explored self-localization methods that make use of multiple sensors, such as the combination of a real-time kinematic global navigation satellite system (RTK-GNSS) and LiDAR. Zhang et al. combined 3D-SLAM and RTK-GNSS, and Niu et al. combined Visual-SLAM and RTK-GNSS. However, RTK-GNSS faces issues in that it can misinterpret the location due to signal reflection off buildings or perform poorly in areas with bad radio conditions, like forests, especially when the distance to the base station is far [15].

In this study, we propose and evaluate a new approach to enhance the accuracy of self-localization in autonomous mobile robots. This approach involves performing low-cost two-dimensional self-localization using 2D-LiDAR on the robot side, and high-accuracy, high-reliability three-dimensional self-localization using 3D-LiDAR on the cloud side. By combining the results of these two self-localization methods, with the two-dimensional results on the robot side as the primary data and integrating the cloud side results through network communication, we aim to create a system that maintains self-localization accuracy while allowing the robot to continue self-localization even if communication is interrupted. This approach mitigates the impact of communication delays and disconnections due to poor network environments.

The implementation of autonomous control is beyond the scope of this study. Instead, we focus on evaluating the combined self-localization methods using manual control. This enables a detailed evaluation of self-localization accuracy and discussions on the integration of different self-localization methods. In future research, we plan to implement autonomous control using the self-localization results obtained from the proposed method and to evaluate control algorithms that utilize self-localization signals.

This paper first describes the overall system configuration used in this study. Then, it evaluates the accuracy of two-dimensional and three-dimensional self-localization using optical motion capture technology. The accuracy is discussed, and a fusion algorithm for combining the two-dimensional and delayed three-dimensional self-localization results is proposed. Finally, the evaluation results are discussed.

2 System Configuration

2.1 System Overview

This section describes the overall system architecture used in the present study. A diagram of the system's configuration is shown in Fig. 1. The system is built around the middleware known as the Robot Operating System (ROS), with the ROS distribution being noetic and the corresponding Ubuntu OS version being 20.04. Using the ROS, the robot is equipped with 2D-LiDAR, 3D-LiDAR, and an inertial measurement unit (IMU), enabling autonomous navigation. Additionally, the computationally intensive task of three-dimensional self-localization is offloaded to another computer via wired communication. Although ”cloud” typically refers to a service that provides computing resources such as servers, storage, databases, networking, and software over the internet, in this context, this other computer is referred to as a virtual cloud.

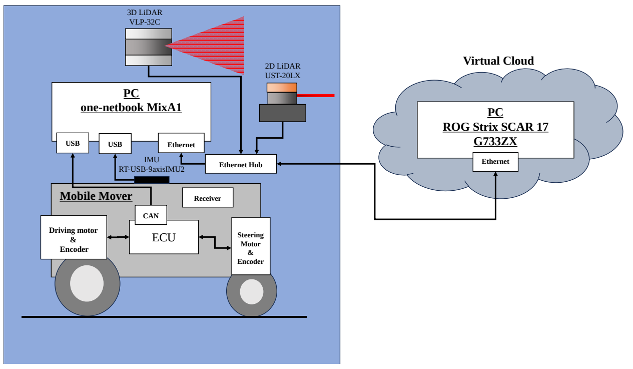


Figure 1: System configuration of Agrino.

2.2 Hardware

In this study, various sensors are mounted on a radio-controlled agricultural work cart, Mobile Mover, to robotize it and enable autonomous mobility. The robot is named Agrino. Fig. 2 shows Agrino with the sensors attached. The specifications of Agrino are presented in Table 1. Agrino is equipped with 2D-LiDAR, 3D-LiDAR, an IMU, and odometry sensors, from which information is gathered.

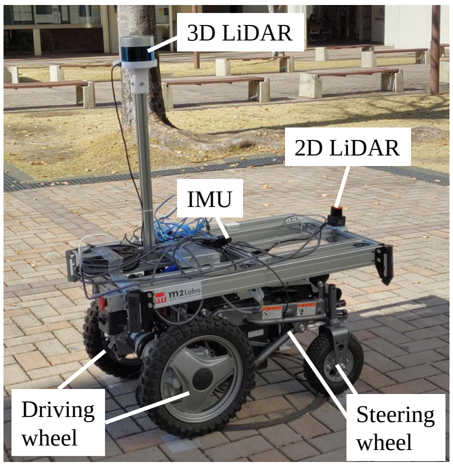


Figure 2: Schematic of Agrino.

2.3 Details of the Sensors

The main specifications of the 2D-LiDAR, 3D-LiDAR, and IMU used in this study are summarized in Table 2.

2.3.1 2D-LiDAR: Hokuyo UST-10LX

In this study, a Hokuyo UST-10LX is utilized as the 2D-LiDAR sensor. This sensor scans the environment

Table 1: Agrino specifications and dimensions

Item	Specification
Size (mm)	905(L) × 600(W) × 500(H)
Vehicle Weight (including battery) (kg)	92
Maximum Load Capacity (kg)	100
Top Speed (km/h)	6
Minimum Turning Radius (m)	6
Maximum Operating Angle (deg)	8
Continuous Running Distance (km)	39
Electric Motor	DC24V 210W×2 (30% rated output)
Battery (5-hour rate)	SC38-12 (12V35Ah)×2
Tire Size	2.50-4 (Front), 2.50-10 (Rear)
Drive System	Rear Wheel Two-Wheel Drive
Braking System	Electromagnetic Reel Type, Mechanical Ground Braking

within a 270-degree field of view and measures the distance to objects. It operates by measuring the time it takes for a laser beam emitted from the sensor to reflect off an object, and calculates the distance based on this time. This sensor is used for two-dimensional self-localization and obstacle detection of the robot, with a maximum measurement distance of 10 m.

2.3.2 3D-LiDAR: Velodyne Ultra Puck

For 3D-LiDAR, a Velodyne Ultra Puck is employed. This sensor uses 32 laser beams to scan a wide area in three dimensions. It operates on a similar principle to the 2D-LiDAR, measuring the time it takes for each laser beam to hit an object and reflect back to calculate the distance. In this study, this sensor is utilized for 3D SLAM, enabling advanced self-localization and environmental recognition.

2.3.3 IMU:RT-USB-9axisIMU2

An RT-USB-9axisIMU2 is used as the IMU. This incorporates a 9-axis sensor (3-axis accelerometer, 3-axis gyroscope, 3-axis compass) and measures the robot's movement (acceleration, rotation, direction) with high precision. The data from this sensor are combined with the 3D-LiDAR data for use in 3D SLAM.

2.3.4 Sensor coordination

This study leverages the synergy of these sensors by utilizing them together. Specifically, the 2D-LiDAR is used for basic self-localization and obstacle detection, the 3D-LiDAR for advanced environmental recognition

Table 2: Specifications of sensors

Sensor	Type	Specifications
2D-LiDAR	Hokuyo UST-10LX	Field of View: 270 deg, Max Distance: 10 m Accuracy: ±40 mm
3D-LiDAR	Velodyne Ultra Puck	Laser Beams: 32 Max Distance: 200 m Accuracy: ±30 mm
IMU	RT-USB-9axisIMU2	Axes: 9 (3 Acceleration, 3 Gyro, 3 Compass),

and self-localization, and the IMU to complement the movement data of the robot. This approach is expected to be able to take advantage of the strengths of each sensor while compensating for their weaknesses, resulting in a synergistic effect.

2.4 Software and Self-Localization Algorithms

In this study, two separate algorithms are utilized for the robot's self-localization. The software configuration is illustrated in Fig. 3.

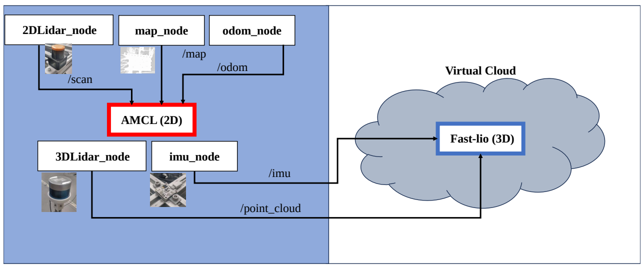


Figure 3: Software configuration.

2.4.1 AMCL (Adaptive Monte Carlo Localization)

AMCL (Adaptive Monte Carlo Localization) [16] is a probabilistic self-localization algorithm widely used in 2D environments. It performs self-localization based on information from 2D-LiDAR and odometry, as well as a pre-prepared environmental map. AMCL is capable of real-time position estimation and operates at a low computational cost, making it suitable for basic self-localization in 2D environments. Henceforth, AMCL will be referred to as the two-dimensional self-localization algorithm. The principles of AMCL are outlined below.

- **Initialization:**

$$X_0 = \{x_0^m | m = 1, ..., M\}$$

Here, X_0 is the initial particle set, x_0^m is the initial state of the m-th particle, and M is the total number of particles.

- **Motion Model:**

  $$x_t^m = \text{motion\_model}(x_{t-1}^m, u_t)$$

  At each time t, the state of each particle x_t^m is updated based on the previous state x_{t-1}^m and the control input u_t.

- **Measurement Model:**

  $$w_t^m = \text{measurement\_model}(x_t^m, z_t)$$

  The weight w_t^m of each particle is calculated based on the state of the particle x_t^m and the observation data z_t.

- **Resampling:**

  $$X_t = \text{resample}(X_{t-1}, W_t)$$

  The new particle set X_t is resampled based on the previous set X_{t-1} and the set of weights of all particles W_t.

2.4.2 Fast-lio (Fast LiDAR-inertial Odometry)

Fast-lio [17] is an algorithm that integrates data from 3D-LiDAR and an IMU (Inertial Measurement Unit) to perform high-accuracy self-localization and 3D mapping. It is particularly suited for fast-paced and dynamic environments, capable of real-time processing. In this study, Fast-lio is employed for advanced self-localization and environmental mapping in 3D environments. Henceforth, Fast-lio will be referred to as the three-dimensional self-localization algorithm. The principles of Fast-lio are outlined below.

- **IMU Pre-integration:** IMU pre-integration is formulated as follows:

  $$\Delta \boldsymbol{p}_{ij} = \int_{t_i}^{t_j} \int_{t_i}^{t} R_{t_i} (\boldsymbol{\omega}_t - \boldsymbol{b}_\omega)\, dt'\, dt$$

  Here, $\Delta \boldsymbol{p}_{ij}$ represents the change in position, R_{t_i} is the rotation matrix at time t_i, $\boldsymbol{\omega}_t$ is the angular velocity, and $\boldsymbol{b}_\omega$ is the bias.

- **Alignment of LiDAR Point Clouds and Maps:** The formula for aligning LiDAR point clouds to a map is as follows:

  $$\text{minimize} \sum_k \|\boldsymbol{p}_k - \boldsymbol{T}_{\text{LiDAR}} \boldsymbol{p}_k^{\text{map}}\|^2$$

  Where $\boldsymbol{p}_k$ are points from a new scan, $\boldsymbol{T}_{\text{LiDAR}}$ is the transformation matrix of the LiDAR sensor, and $\boldsymbol{p}_k^{\text{map}}$ are the corresponding points on the map.

- **Optimization of State Estimation:** Optimization of state estimation is expressed as:

  $$\text{minimize} \sum_{\text{all measurements}} \|\boldsymbol{r}(\boldsymbol{x})\|^2$$

  Here, $\boldsymbol{r}(\boldsymbol{x})$ represents the residuals between the measurements and the predictions, and $\boldsymbol{x}$ is the state vector.

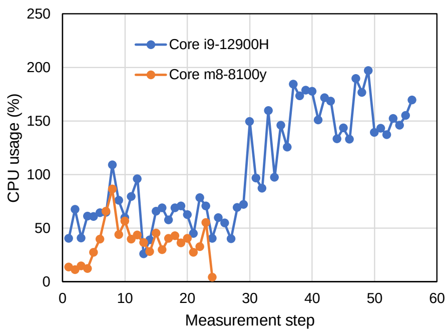
2.5 Programs and Computing Resources

The program executed on the computer mounted on Agrino includes AMCL for two-dimensional self-localization, a move base for route planning, various sensor programs, and programs for processing the estimated robot posture. In the virtual cloud, computations for three-dimensional self-localization and the creation of 3D maps are performed using 3D SLAM and Fast-lio. The computer used as the virtual cloud is an ASUS ROG Strix SCAR 17 G733ZX (G733ZX-I9R3080T), and the computer mounted on Agrino is a One-netbook One MixA1. The specifications of these two computers are summarized in Table 3. It is evident that the ROG Strix SCAR 17 G733ZX offers higher performance than the One-netbook, but when comparing costs, the ROG Strix SCAR 17 G733ZX is significantly more expensive than the One-netbook's MixA1. When attempting to minimize the cost per robot, the One-netbook's MixA1 proves to be more cost-effective when mounted on a robot.

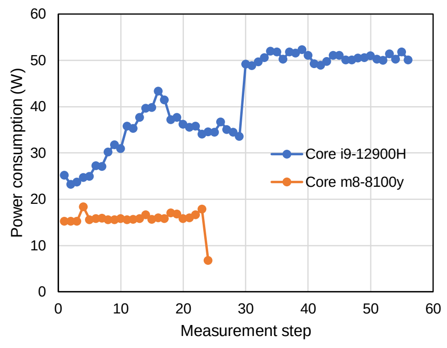
As a preliminary experiment, we measured the CPU utilization and power consumption of each PC running Fast-lio using the Linux top command, and the results are shown in the Fig. 4. The One-netbook A1 has the lowest CPU utilization and power consumption. The On One-netbook A1, the Fast-lio program failed at the 24th step after about 10 minutes had elapsed. The graph shows only one example of the results, but it was confirmed that the program terminated after about 10 minutes in each of the five trials. From these results, we can conclude that the One-netbook A1 with low power consumption is suitable for minimal self-location estimation on the edge side, and the ROG with high CPU resources is suitable for 3D self-location estimation on the cloud side in the experiments described in this paper.

Table 3: Comparison of computer specifications

Item	Details
PC Name	ROG Strix SCAR 17 G733ZX
Processor (CPU)	Intel Core i9-12900H Processor
Memory (RAM)	32GB DDR5 4800MHz
Graphics Card (GPU)	NVIDIA GeForce RTX3080 Ti Laptop GPU
Price	$2867
PC Name	One Netbook A1
Processor (CPU)	Intel Core m3-8100y
Memory (RAM)	8GB
Graphics Card (GPU)	None
Price	$589



(a) CPU usage.



(b) Power consumption.

Figure 4: CPU usage and power consumption for both PCs when Fast-lio was processed.

2.6 Communication Methods and Data Exchange

Ideally, high-speed and high-capacity communication methods like 5G would be preferable for interactions between the robot and the cloud. However, achieving the nominal value of 1 ms latency with 5G communication is challenging. In addition, we do not want to be affected by unintentional wireless communication failures because we want to evaluate the system in the absence of uncontrolled delays in order to examine the effect of intentionally created delays or data loss. Therefore, this paper adopts wired Ethernet for data exchange.

3 Outdoor Self-Localization Evaluation Experiment

3.1 Objective of the experiment

In this experiment, the position of the robot was measured externally using motion capture, and this measurement was taken as the true value. By comparing these measurements with the self-localization results obtained from the sensors, we verified the self-localization accuracy of both Fast-lio and AMCL. This verification method is based on the study by Z. Niu et al. [14], where the accuracy of self-localization is determined by the error between the externally measured true value and the estimated position of the robot. Additionally, we investigated the conditions under which self-localization is likely to become inaccurate.

3.2 Experimental method

Using Agrino, self-localization is performed with 2D AMCL on a mini-computer, while 3D self-localization with Fast-lio is conducted outdoors using a virtual cloud. Markers are installed on Agrino as shown in Fig. 5. These markers are to allow the motion capture system, OptiTrack, to record Agrino's movements. The experimental setup is shown in Fig. 6.

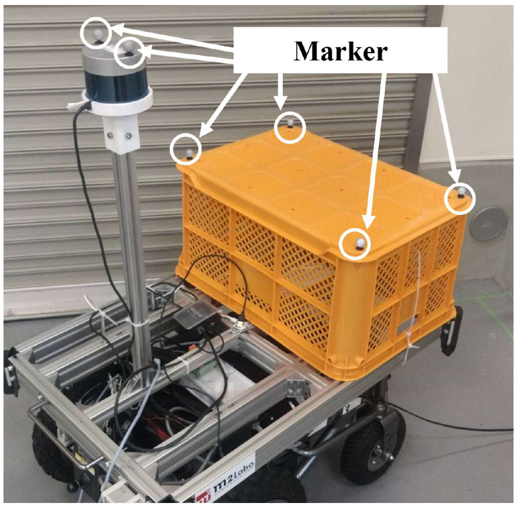


Figure 5: Agrino with attached markers.

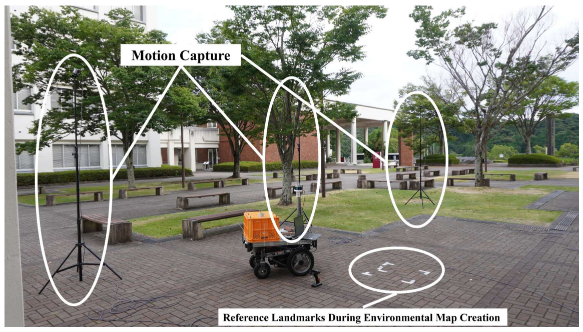


Figure 6: Scene during outdoor experiments.

Table 4 presents the specifications of the optical motion capture system, OptiTrack. It is capable of accurately measuring the movement of objects with 6 degrees of freedom, with tracking possible at a maximum rate of 1,000Hz. In this experiment, we utilized this motion capture system to measure the movements of Agrino.

The experimental procedure is as follows:

- Set the initial position of Agrino and operate it manually to create a 2D environment map using SLAM gmapping.
- Start AMCL on the robot-side computer and Fast-lio on the virtual cloud-side computer.
- Place the robot at the initial position. Also, start the program to record the bag file and the posture of the robot at this time.
- Operate Agrino with the controller and drive it.

Table 4: Motion capture system specifications

Software	
Degrees of Freedom	6
Tracking Frequency	20 Hz to 1,000 Hz
Camera Control	Bulk control of connected cameras
Accuracy	Error of 0.1mm or less
Rigid Body	Creation of rigid bodies possible
Camera	
Lens	Horizontal Field of View: 56° Vertical Field of View: 46°
Standard Filter	850nm IR Bandpass 700nm Visible Light
LED	10 pieces, 850nm IR adjustable brightness Max. range with reflective markers: 16 m Max. range with active markers: 25 m
Image Sensor	Resolution: 1,280 × 1,024 pixels Frame Rate: 30-240 fps (2.X), 20-1,000 fps (3.X) Shutter Type: Global Shutter Speed: 0.01 ms 3.9 ms
Data I/O	Data: GigE / PoE (1000BASE-T) Camera Synchronization: Ethernet

- In this experiment, drive the robot to trace a square approximately 3 m wide and 4 m tall, as shown in Fig. 7.

3.3 Experimental results and discussion

In Fig. 8, AMCL is represented by red dots, Fast-lio by blue dots, and motion capture by green dots, indicating their trajectories. The x-axis and y-axis of the graph represent the displacement in the direction of the robot's movement.

From Fig. 8, it can be observed that while Fast-lio exhibits some deviation from the trajectory of the motion capture around (-1,4) during turns, the trajectories are very similar, indicating high accuracy in self-localization. On the other hand, in the case of AMCL, significant discrepancies, particularly on the left side of Fig. 8, where the robot tends to overshoot after turning inward, are evident. Moreover, the deviation on the outer side in the upper right corner of Fig. 8 suggests errors between the estimated position and the actual position of Agrino. Furthermore, focusing on the goal points, it can be inferred that AMCL exhibits errors compared to Fast-lio.

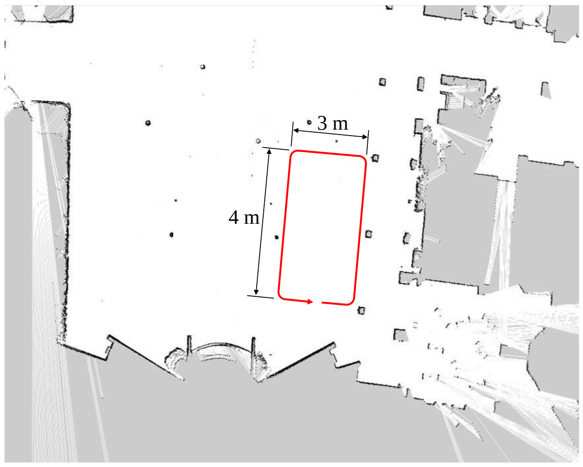


Figure 7: Outdoor running route.

To provide a clearer understanding of the detailed movements, the time variation of x and y are illustrated in Figs. 9 and 10. From around 40 seconds to 50 seconds in Fig. 9, significant deviations between AMCL and motion capture can be observed, which are not present in Fast-lio. Fig. 11 compares the errors for AMCL and Fast-lio with motion capture values as the reference. It should be noted that motion capture, AMCL, and Fast-lio have different timestamps due to the frequency of data updates and experimental constraints. Therefore, data resampling and correction were performed to align the start and end times of measurements. The errors of AMCL and Fast-lio during this time were approximately -0.18 m to 0.23 m and -0.13 m to 0.1 m, respectively.

Next, to quantitatively analyze the errors, the root mean square errors (RMSEs) of Fast-lio and AMCL were measured from Fig. 11. These were 0.1239 m for AMCL and 0.0693 m for Fast-lio.

AMCL relies on odometry topics and is susceptible to odometry errors influenced by environmental changes such as road conditions and steps. Therefore, in environments with long-distance self-localization measurements or terrain with slippery surfaces and steps, such as fields, the errors are expected to be even larger. Particularly for outdoor robotic applications, these errors can become significant, and even with particle filter correction by AMCL, accurately estimating the robot's self-position based on the particle distribution over time becomes challenging.

3.4 Summary of the experiment

Self-localization using Fast-lio demonstrated superior accuracy compared to AMCL. Ideally, it would be desirable to equip the robot with a computer rich in computational resources to perform advanced self-localization using Fast-lio. However, considering the realistic scenario of limited computational resources, solely deploying Fast-lio on the robot platform for control is not optimal. Therefore, contemplating autonomous navigation using Fast-lio in a computational resource-rich cloud seems plausible. Nonetheless, relying entirely on

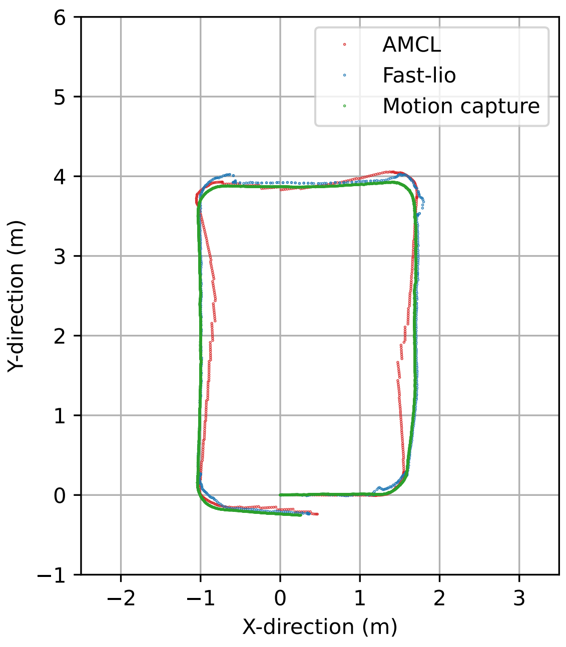


Figure 8: Motive: Tracker's measured trajectory for Agrino.

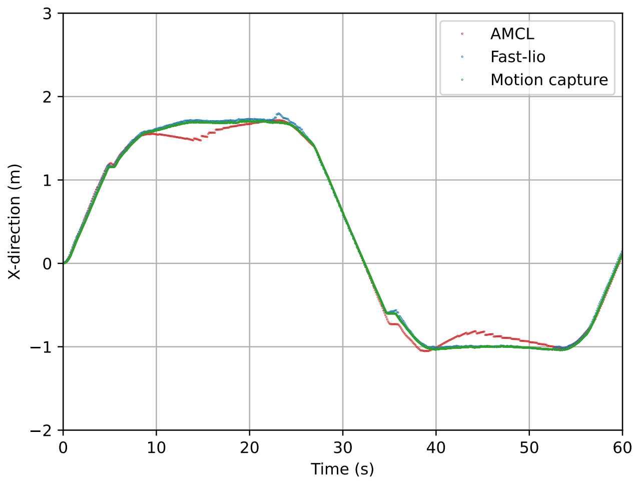


Figure 9: Comparison of estimated X-axis position over time.

cloud-based control for the robot poses risks of instability due to communication loss, jeopardizing the safety of both the surroundings and the robot itself. Hence, it is imperative to integrate both algorithms, equipping the robot platform and the cloud with suitable algorithms respectively, to create a system that leverages the strengths of both, ensuring high accuracy and safety.

4 Fusion of Heterogeneous Self-Localization Algorithms

4.1 Necessity and Approach for Fusion

AMCL, which performs two-dimensional self-localization, and Fast-lio, which performs three-dimensional self-localization, are systems based on different algorithms, and each possesses distinct characteristics. Both systems use an environmental map for localization, but AMCL is lighter in processing compared to Fast-lio. However, since AMCL relies on scan matching with the environmental map, it is susceptible to mislocalizations caused by obstacles not reflected in the map. Additionally, as AMCL is a two-dimensional localization system, it cannot detect obstacles that are not at the height of the 2D-LiDAR, which can be problematic when carrying loads; for example, a robot transporting a load might choose a path under a table where the robot itself can pass, but the load may get caught.

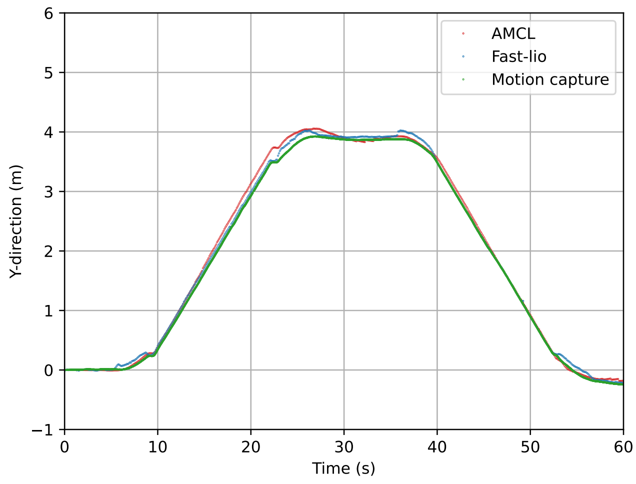


Figure 10: Comparison of estimated Y-axis position over time.

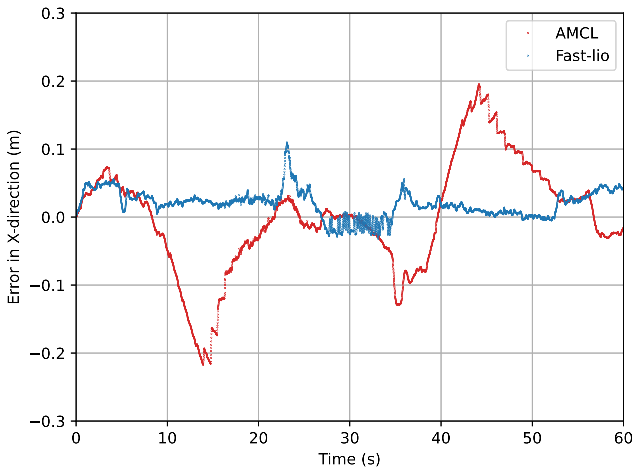


Figure 11: AMCL and Fast-lio X-axis position error compared to ground truth (Optitrack).

Fast-lio, being a 3D SLAM system, performs self-localization while simultaneously creating environmental maps, offering robustness against dynamic environmental changes. It uses 3D-LiDAR, which enables the detection of objects in three dimensions, a significant advantage. However, compared to AMCL, Fast-lio requires more processing power, which can be challenging on low-cost, low-performance computers.

In this study, two-dimensional self-localization with AMCL is performed on the onboard computer of Agrino, while three-dimensional localization with Fast-lio is handled on a virtual cloud. By fusing these two systems, improvements in the accuracy and robustness of self-localization are anticipated. Such an approach is particularly important in outdoor areas where communication networks are prone to degradation. Specifically, in cases of deteriorating communication, it is possible to compensate for continuous self-localization by degrading the reliability of Fast-lio results or by switching to wired AMCL results.

First, we tried to detect a quantity corresponding to the deviation from the ground truth from the components of the AMCL covariance matrix, but preliminary experiments showed that they were not correlated, so we excluded this method.

Here, two potential approaches are explored: time-varying weighted average (TVWA) and the unscented Kalman filter (UKF). The data used for validating the fusion particularly involve outdoor experimental data, where significant discrepancies between AMCL, Fast-lio, and motion capture were observed. Additionally, the fusion of self-localization using covariance matrices was also considered, but it did not show significant results.

4.2 Examination of Self-Localization Fusion Method Using TVWA

4.2.1 Reason for Choosing TVWA

Even when running AMCL and Fast-lio simultaneously locally, the update rate of AMCL's self-localization results is approximately 800Hz, while that of Fast-lio's self-localization is about 10Hz, which is significantly slower. Network latency can exacerbate this delay further. One possible method to handle this issue is to apply a weighted average for the state transition of self-localization, where Fast-lio has higher accuracy but slower updates, and AMCL has lower accuracy but faster updates. However, instead of simply averaging the results of AMCL and Fast-lio, the proposed method proposes introducing a metric that varies over time called "reliability," and computing the weighted average according to this reliability. We call this fusion method TVWA. The algorithm formula is presented below. In this study, the reliability value for AMCL is fixed at 50%, and the reliability (weight) for Fast-lio is set at 100%. With each loop of the program, the reliability value for Fast-lio is reduced by 10%, and this reliability is directly used as the weight in the weighted average. When Fast-lio's reliability drops below 60%, it is reset to 100% to reflect the acquisition of new data.

4.2.2 Self-localization Fusion Results

The results of the self-localization fusion using TVWA are shown in Fig. 12. Orange points show results of TVWM. The self-localization results obtained through TVWA appear to be closer to those of the motion capture than the results from AMCL alone. To examine the results more closely, Figs. 13 and 14 show the temporal changes along the X and Y axes, respectively. The RMSE of TVWA was 0.0958 m. The self-localization results using the TVWA indeed provide better outcomes than the standard AMCL results (0.1239 m).

Algorithm 1 TVWA transformation

```
Require: ROS environment, tf and tf2 libraries
Ensure: Weighted average of transformations
 1: $W_{2D} = 0.5$
 2: $W_{3D} = 1.0$
 3: while ROS is running do
 4:     try
 5:     $X_{wa} = \frac{X_{2D} \times W_{2D} + X_{3D} \times W_{3D}}{W_{2D} + W_{3D}}$
 6:     $W_{3D} - 0.1$
 7:     if $W_{3D} < 0.6$ then
 8:         $W_{3D} = 1.0$
 9:     end if
10: end while
```

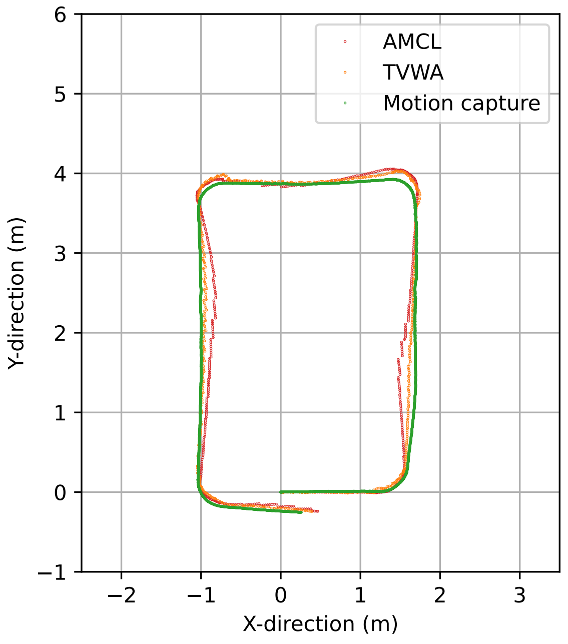


Figure 12: Self-localization results obtained through TVWA.

4.3 Exploration of Self-localization Fusion Method Using UKF

4.3.1 Reason for Choosing the UKF

As another method for fusing two self-localization results, we propose the fusion of self-localization results using the UKF [18]. The UKF is a method for estimating the state of nonlinear systems. It is frequently used in robotics and aerospace engineering. A distinctive feature of the UKF is its use of sigma points, which allows for a higher accuracy compared to the extended Kalman filter, another method for estimating states in nonlinear systems. This is because the UKF can approximate

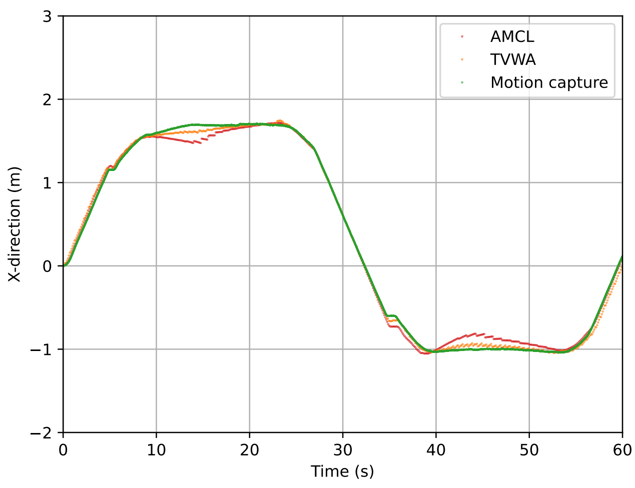


Figure 13: Self-localization results along X axis obtained through TVWA.

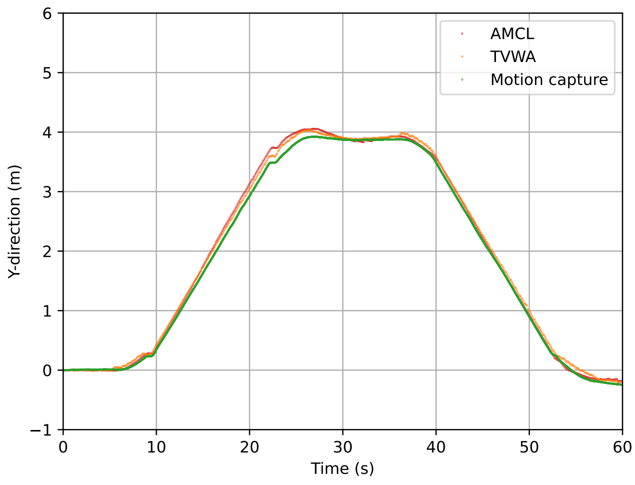


Figure 14: Self-localization results along Y axis obtained through TVWA.

up to the second-order terms using Taylor expansion, unlike the extended Kalman filter. Below, we present the essential components needed for the UKF.

i. **State Vector**: A collection of variables representing the state of the system. For a robot, this includes position (x, y, z) and attitude (roll, pitch, yaw).

ii. **Process Model**: A model that represents the time evolution of the system. It defines the transition from the previous state to the next state.

iii. **Observation Model**: A model that represents the relationship between observations from sensors and the state vector. It can accommodate nonlinear models.

iv. **Noise Covariance**: Represents the uncertainty in the system. The process noise covariance accounts for the uncertainty in the process model, while the observation noise covariance represents the uncertainty in sensor measurements.

v. **Sigma Points**: A unique feature of the UKF that enables predicting the next state directly from the nonlinear system with very high accuracy. However, calculating sigma points requires generating $(2n+1)$ points for a system of dimension n, which incurs computational costs.

The basic steps of the UKF are as follows:
Basic Steps:

i. **Prediction Step**:
Using the previous state and the process model, compute the estimated value of the next state and its uncertainty (covariance). At this point, sigma points are used to predict the next state from the nonlinear system.

ii. **Update Step**:
Use new observation data to update the predicted state and reduce uncertainty.

By repeating these steps, the states of various systems are estimated.

4.3.2 Role of the UKF

In this section, the components of the UKF are as follows:

i. **State Vector**:
A collection of variables representing the state of the system. In the case of a robot which moves in plane field, this includes position (x, y) and the robot's velocity ($\dot{x}$, $\dot{y}$). Therefore the state variable is as below.

$$\boldsymbol{X} = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} \tag{1}$$

Initial State

$$\boldsymbol{X_0} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2}$$

The position is directly used as the state vector from the self-localization results output by AMCL and Fast-lio. The angular velocity is derived by dividing the displacement of coordinates from the most recent data of AMCL and Fast-lio by the time interval Δt obtained up to that point.

ii. **Process Model**:
A model that represents the time evolution of the system. It defines the transition from the previous state to the next state. This includes using the velocity at each moment, calculated by dividing the displacement in coordinates (x, y) obtained from AMCL and Fast-lio by the difference in time between readings. The process model for Agrino is presented below as an equation.

$$x_{t+1} = x_t + \dot{x}_t \Delta t \tag{3}$$
$$y_{t+1} = y_t + \dot{y}_t \Delta t \tag{4}$$

At each time t, the state vector $\mathbf{x}_t$ is updated based on the current state, velocity, and time interval Δt. The state transition matrix F is given below.

$$F = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

iii. **Observation Model**:
A model that represents the relationship between observations from sensors and the state vector. It can accommodate nonlinear models. Since there are no sensor inputs available in this instance, the self-localization results from AMCL and Fast-lio are used as a virtual observation model. The observation noise matrix R is given below.

$$R = \begin{pmatrix} 1.0 & 0 \\ 0 & 0.5 \end{pmatrix} \tag{6}$$

iv. **Noise Covariance**:
Represents the system's uncertainty. Process noise covariance accounts for the uncertainty in the process model, while observation noise covariance represents the uncertainty in sensor measurements. In this study, since Fast-lio provides more accurate self-localization results than AMCL, a fixed value is used for the noise covariance.

The initial error covariance matrix P is given below.

$$P = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{pmatrix} \tag{7}$$

The process noise matrix Q is given below.

$$Q = \begin{pmatrix} 0.05 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{pmatrix} \tag{8}$$

4.3.3 Fusion Results with the UKF

The fusion results obtained using the UKF are shown in Fig. 15. Orange points show the results with UKF. The graph shows that the results fused with the UKF closely resemble those from the motion capture, more so than those output by AMCL.

For a more detailed review of the results, the temporal changes along the X and Y axes are shown in Figs. 16 and 17, respectively. The RMSE of UKF was 0.0944 m. The self-localization results using the TVWA indeed provide better outcomes than the standard AMCL results (0.1239 m) and TVWM result (0.0958 m). Table 5 summarizes the RMS Error of each method. This improvement in accuracy is likely because the errors from AMCL were smaller where Fast-lio's errors were largest, thus enhancing precision through fusion. Fig. 18 shows this, with the largest errors for Fast-lio occurring at around 57 to 60 seconds.

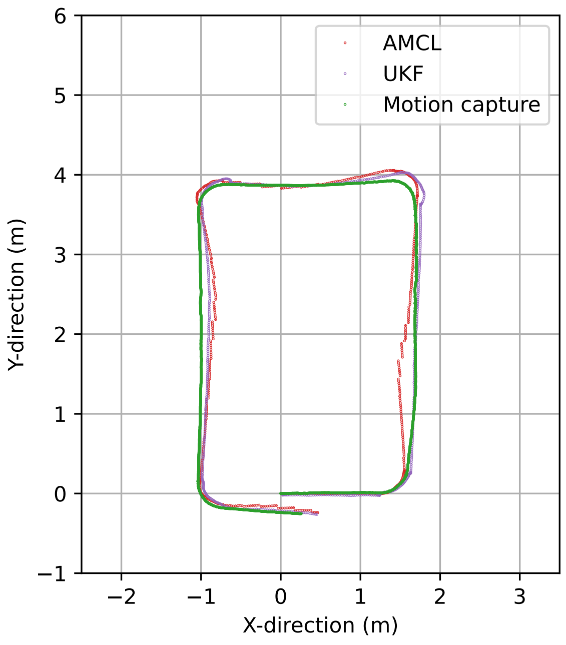


Figure 15: Data fusion results with UKF.

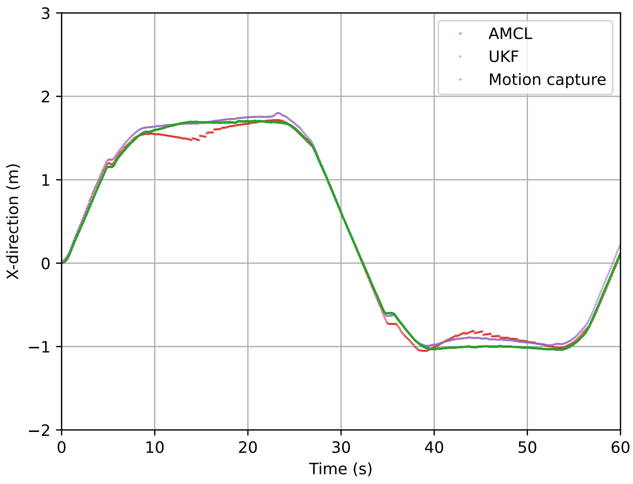


Figure 16: Fusion results of UKF along X axis.

Additionally, in this instance, the robot was treated as a single mass point for the application of the UKF, meaning that only linear system information was input into a filter designed for estimating nonlinear systems. Hence, the full potential of the filter has not been realized. In future work, by treating the robot not just as a point but as a nonlinear system with a yaw angle inclination, the UKF can be better utilized and is expected to produce more accurate results. The system model equation expected at that time would look something like the equation below, where $\theta(t)$ represents the robot's yaw angle component inclination, which can be

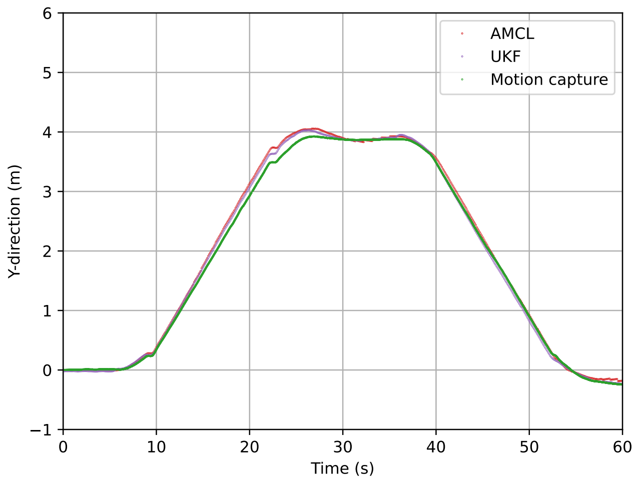


Figure 17: Fusion results of UKF along Y axis.

Table 5: Comparison of RMS error for each method

Method	RMSE (m)
AMCL	0.1239
Fast-lio	0.0693
TVWA	0.0958
UKF	0.0944

utilized in estimating nonlinear systems with the UKF.

- **Process Model Equation Considering Yaw Angle:**

$$x_{t+1} = x_t + v_t \cos\theta_t \Delta t \tag{9}$$
$$y_{t+1} = y_t + v_t \sin\theta_t \Delta t \tag{10}$$
$$\theta_{t+1} = \theta_t + \omega_t \Delta t \tag{11}$$

  At each time t, the state vector $\mathbf{x}_t$ is updated based on the current state x, y, θ and velocities v_t, ω_t, and the time interval Δt. The yaw angle inclination of the robot at this time can be represented as θ_t.

4.4 Summary

In this section, we examined methods for fusing self-localization results estimated by different algorithms using several techniques. It was determined that the method of switching data based on the covariance matrix, which was used to judge the uncertainty of AMCL, is difficult to implement because there was little correlation found between the covariance matrix and the errors. Furthermore, the fusion using time-varying weighted averages, which adjusts the reliability over time to blend the two datasets, was found to produce results superior to those of the original AMCL. This fusion method can be implemented more simply and with less information than the UKF, and by adjusting the reliability parameters or the timing parameters for reliability updates, it is conceivable that even more accurate fusions of self-localization could be achieved in the future. Fusion using the UKF can achieve very high precision, but compared to fusion using time-varying weighted averages, it requires separate calculations for velocity and time intervals, and also the computational cost of generating sigma points must be considered, as it is higher than for other Kalman Filters. Therefore, the choice of algorithm can vary depending on the performance of the computer onboard the robot. For very low-performance computers or when high real-time performance is required, fusion using time-varying weighted averages should be employed, while for computers with slightly higher performance, fusion using the UKF is more appropriate.

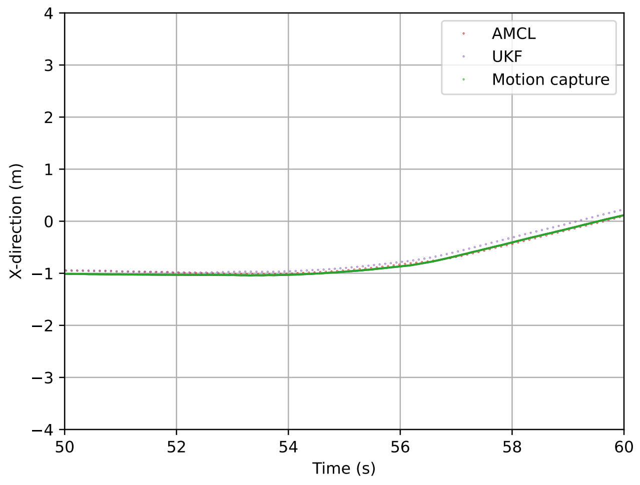


Figure 18: Detailed results for each self-localization algorithm along X axis.

5 Fusion of Self-localization Considering Communication Interruptions

5.1 Deterioration of Network Conditions

Up to this point, we have proposed fusion methods for cases without communication delays. However, network conditions can vary, with both good and bad situations due to changes in the network environment. Particularly, poor network conditions can negatively impact autonomous mobile robots. In this experiment, Agrino and the virtual cloud are connected via a wired connection, so significant data loss is unlikely. To evaluate the impact of poor network conditions, we intentionally caused data loss to simulate conditions similar to deteriorating wireless communication.

5.1.1 Communication Delays

The term "communication delay" refers to the time lag that occurs when data travels from the sender to the receiver. In internet and network environments, it can take time for data packets to reach their destination, and these communication delays can be caused by various factors. For example, data congestion, network congestion, processing times at routers or switches, and even physical media or weather conditions can impact this.

5.1.2 Communication Interruptions

Unlike communication delays, communication interruptions refer to a complete loss of network connectivity. This means that the data does not reach the intended receiver at all, causing a disruption in communication. Interruptions can be caused by hardware failures, software issues, natural disasters, or radio signal conditions. These interruptions can lead to serious accidents in autonomous mobile robots.

5.2 Fusion of Self-Localization Considering Temporary Communication Interruptions

The fusion of self-localization conducted outdoors was based on the assumption that communication between Agrino and the virtual cloud was optimal. However, in actual usage environments, it is necessary to consider risks such as communication delays and interruptions. For this scenario of communication interruptions, we created data with simulated communication delays and evaluated the results using two implemented fusion methods for self-localization.

5.3 Creation of Simulated Data During Communication Interruptions

It is challenging to intentionally obtain data with communication delays from experiments. Therefore, for this study, we modified the data obtained from experiments using a Python program to intentionally create simulated data with communication delays and interruptions. The data being processed are the self-localization results output by Fast-lio from an outdoor experiment. Figure 19 shows the processed data. Figure 19 depicts the data randomly removed by the Python program at intervals from 0.1 seconds to 2.0 seconds, at 0.1-second intervals. The data with these missing points are then used in the fusion methods. In this Figure 19, gaps in the robot's position are shown. The larger these gaps are, the longer the robot loses track of its position, which can negatively impact the robot's operation. Therefore, we aim to verify that the fusion of self-localization reduces these gaps.

Figures 20 and 21 show the fusion results using TVWA and the UKF, respectively, along with the actual positions of the robot measured by motion capture. In the fusion using TVWA, the trust value changes sharply from 60% to 100%, resulting in more instances where the data is not continuous compared to the fusion using the UKF.

Additionally, Figures 22 and 23 show the coordinate differences before and after time steps with missing data in the x and y directions, respectively, with both TVWA and UKF. The orange line shows the results with TVWA, and the purple line shows those with UKF. Both these figures show that the difference of our fusion methods are smaller than the missing data. Additionally, since the fusion using the UKF combines two observation datasets, the graph is smoother compared to the TVWA.

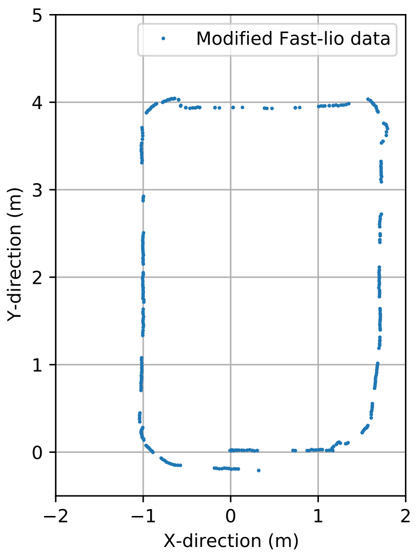


Figure 19: Processed results for 3D self-localization.

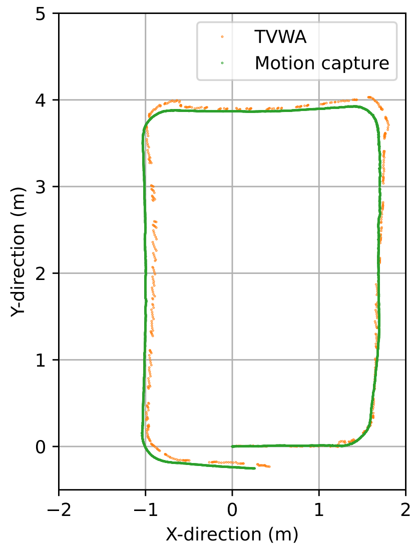


Figure 20: Fusion results for data using TVWA and actual position of robot observed by motion capture.

5.4 Summary

It was observed that in both methods, there are places where the self-localization changes abruptly. Particularly with the TVWA, these abrupt changes were more frequent compared to the UKF. It is necessary to further investigate how these sudden changes in self-localization affect programs related to autonomous navigation.

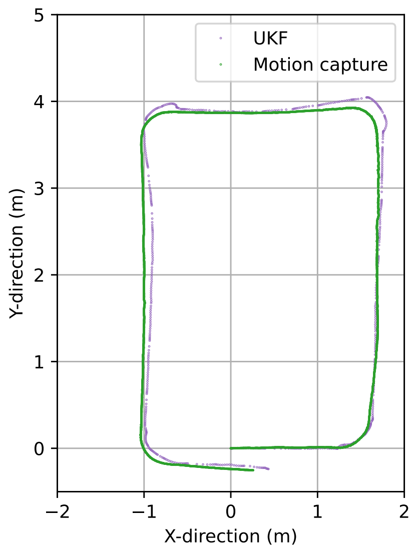


Figure 21: Fusion results for data using UKF and actual position of robot observed by motion capture.

6 Conclusion

In cloud robotics, aiming to achieve high-precision and safe self-localization methods, we designed a system where computationally intensive 3D self-localization is processed by a remote computer (virtual cloud), while more cost-effective 2D self-localization is performed by the local computer. We evaluated the results of each localization method and examined their fusion techniques, leading to the following conclusions.

- When evaluating the difference between the self-localization results and the reference data obtained through motion capture, the accuracy of the 3D self-localization by Fast-lio was found to be about 0.1 m closer to the reference data compared to the 2D self-localization accuracy by AMCL.
- As methods for fusing 3D and 2D self-localization results, two techniques, the TVWA and the UKF, were proposed and compared. It was demonstrated that fusion using UKF offers higher accuracy. However, UKF requires preprocessing, so depending on the performance of the computer onboard the robot, the TVWA could also be a viable option.
- In scenarios with intermittent communication failures or complete disconnections, using partially missing self-localization data showed that UKF resulted in less abrupt changes in self-localization compared to the TVWA, demonstrating higher robustness against communication interruptions.

Future efforts will include the following

- In the fusion method using UKF, only the coordinates of the robot were considered as a linear system. Moving forward, the fusion will be performed using a nonlinear system model that also includes the robot's orientation angles.

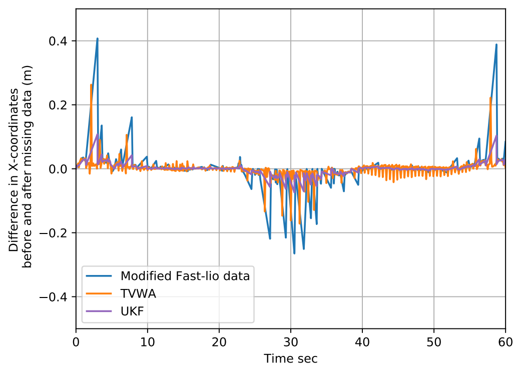


Figure 22: X-direction coordinate differences before and after time steps with missing data.

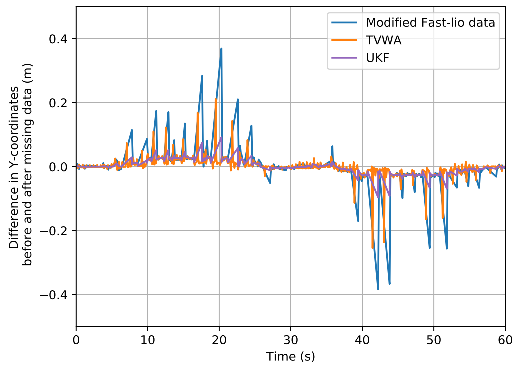


Figure 23: Y-direction coordinate differences before and after time steps with missing data.

- Instead of relying solely on Fast-lio's SLAM, by implementing self-localization according to a pre-created 3D map, actual operation in real environments will be carried out.
- Since actual operations are expected to involve multiple robots, the system will be updated and ported from ROS 1 to ROS 2.

Acknowledgment

This paper is based on results obtained from a project, JPNP20017, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] X. Zhang, J. Lai, D. Xu, H. Li, and M. Fu, "2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots," *J. Adv. Transp.*, vol. 2020, pp. 1–14, 2020. DOI: 10.1155/2020/8867937

[2] T. Ruiz, S. De Raucourt, Y. Petillot, and D. M. Lane, "Concurrent mapping and localization using sidescan sonar," *IEEE J. Oceanic Eng.*, vol. 29, no. 2, pp. 442–456, 2004. DOI: 10.1109/JOE.2004.829790

[3] L. von Stumberg, V. Usenko, J. Engel, J. Stückler, and D. Cremers, ”From monocular SLAM to autonomous drone exploration,” in *Proc. Euro. Conf. Mobile Robot., 2017*. DOI: 10.1109/ECMR.2017.8098709

[4] A. Ohya and N. Shimaji, “Commercial Production of Optical Range-sensor for Intelligent Robot,” *J. Robot. Soc. Jpn.*, vol. 23, no. 2, pp. 181–184, 2005. DOI: 10.7210/jrsj.23.181

[5] S. Raikwar, H. Yu, and T. Herlitzius, “2D LIDAR SLAM Localization System for a Mobile Robotic Platform in GPS Denied Environment,” *J. Biosyst. Eng.*, vol. 48, no. 2, pp. 123–135, 2023., doi:DOI: 10.1007/s42853-023-00176-y

[6] J. Xiong, Y. Liu, X. Ye, L. Han, H. Qian, and Y. Xu, ”A hybrid lidar-based indoor navigation system enhanced by ceiling visual codes for mobile robots,” in *2016 IEEE Int. Conf. Robot. and Biomimetics (ROBIO)*, Qingdao, China, 2016, pp. 1715-1720, doi: DOI: 10.1109/ROBIO.2016.7866575.

[7] W. L. Liu, X. P. Zhao, and B. G. Xu, “Application of Constructing Three-Dimensional Model Using Laser Scanning Technology,” *Appl. Mech. Mater.*, vol. 94–96, pp. 86–89, 2011., doi:DOI: 10.4028/www.scientific.net/amm.94-96.86 DOI: 10.4028/www.scientific.net/AMM.94-96.86

[8] J. Y. Ren et al. ”LIDAR EKF-SLAM Technology and Its Application in Automated Guided Vehicle Navigation.” *DEStech Trans. Comput. Sci. Eng.*, 2018,DOI: 10.12783/dtcse/iciti2018.29102

[9] Y. Kanuki, N. Ohta, N. Nakazawa, ”Development of Autonomous Moving Robot Using Appropriate Technology for Tsukuba Challenge,” *J. Robot. Mechatron.*, vol. 35, no. 2, pp. 279-287, 2023. DOI: 10.20965/jrm.2023.p0279

[10] G. Hu, W. P. Tay, and Y. Wen, “Cloud robotics: Architecture, challenges and applications,” *IEEE Netw.*, vol. 26, no. 3, pp. 21–28, 2012. DOI: 10.1109/MNET.2012.6201212

[11] K. Konno, “Advantages and Disadvantages of Using Cloud Computing: A Comparison of Cloud Computing and On-Premise System,” *Ger. Policy Stud.*, vol. 22, pp. 17–33, 2012.

[12] M. Nabeta, K. Mima, and K. Tobita, “Research on Risk Assessment in Cloud Control of Autonomous Mobile Robots,” in *JSME Conf. Robot. Mechatron.*, June 28 – July 1, 2023. DOI: 10.1299/jsmermd.2023.1A2-I06

[13] J. Zhang, W. Wen, F. Huang, Y. Wang, X. Chen, and L.-T. Hsu, “GNSS-RTK Adaptively Integrated with LiDAR/IMU Odometry for Continuously Global Positioning in Urban Canyons,” *Appl. Sci*, vol. 12, no. 10, p. 5193, 2022. DOI: 10.3390/app12105193

[14] Z. Niu, X. Zhao, J. Sun, L. Tao, and B. Zhu, “A Continuous Positioning Algorithm Based on RTK and VI-SLAM with Smartphones,” *IEEE Access*, vol. 8,185638–185650, 2020. DOI: 10.1109/ACCESS.2020.3028119

[15] D. Lu and E. Schnieder, “Performance evaluation of GNSS for train localization,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 1054–1059, 2014. DOI: 10.1109/TITS.2014.2349353

[16] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, “Robust Monte Carlo localization for mobile robots,” *Artif. Intell.*, vol. 128, no. 1-2, pp. 99–141, 2001. (Elsevier.) DOI: 10.1016/S0004-3702(01)00069-8

[17] W. Xu and F. Zhang, “Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3317–3324, 2021. DOI: 10.1109/LRA.2021.3064227

[18] S. J. Julier and J. K. Uhlmann, “Unscented filtering and nonlinear estimation,” in *Proc. IEEE*, vol. 92, no. 3, pp. 401–422, Mar. 2004., doi:DOI: 10.1109/JPROC.2003.823141