2024

MASTER'S THESIS

Implementation of Object Grasping and Autonomous Inter-Floor Transport for a Tracked Delivery Robot

ACADEMIC SUPERVISOR: Joo-Ho Lee

Graduate School of Information Science and Engineering
Ritsumeikan University

MASTER'S PROGRAM
MAJOR in Advanced Information Science and Engineering

STUDENT ID: 6612220086-9

NAME: Yihe Kang

Abstract

This paper details the development and implementation of an autonomous system for the UDOn (Ubiquitous Delivery On-demand) robot, a tracked mobile platform from previous research. This system enables the robot to navigate between floors using stairs and to perform object grasping and transport tasks. It employs a 2D LiDAR-based SLAM algorithm to ensure robust indoor navigation and generate 2D maps for multi-floor environments. The robot is also fitted with a low-degree-of-freedom robotic arm designed for reliable object handling. Our approach integrates the 2D LiDAR SLAM and navigation modules with the robot and arm control system, facilitating smooth transitions between floors and efficient object transportation. Extensive experiments conducted in the hallways of a multi-floor building validate the system's performance. The results indicate the robot's capability to autonomously navigate complex indoor spaces, ascend and descend stairs, and transport objects between floors. This research advances the development of service robots capable of operating in dynamic, multi-level indoor environments, thus enhancing their practical applications in logistics, healthcare, and facility management.

Contents

Chapter 1

Introduction

1.1 Background

Rapid advances in robotics technology in recent years have led to the development of autonomous robots capable of performing a wide variety of tasks across diverse industries, including logistics, healthcare, and facilities management. These robots are increasingly deployed in dynamic indoor environments, where they must navigate complex spaces and interact with objects to accomplish various tasks. One of the critical challenges in such environments is enabling robots to autonomously navigate between multiple floors and handle objects accurately.

Simultaneous Localization and Mapping (SLAM) technology plays a crucial role in enabling robots to navigate autonomously in various environments by constructing maps while simultaneously determining their own position within these maps. Over the years, SLAM has evolved significantly from early methods based on laser and visual sensors to today's sophisticated approaches integrating multiple sensor modalities. Technologically, SLAM has transitioned from traditional methods that rely on geometry and handcrafted features to modern approaches that integrate deep learning and neural network technologies. Traditional SLAM methods depend on feature extraction, motion estimation, and map construction. In contrast, modern SLAM techniques incorporate neural networks to more accurately handle perception and localization tasks in complex environments, thereby enhancing system robustness and adaptability.

Visual SLAM algorithms, such as ORB-SLAM and LSD-SLAM, leverage camera data to build maps and localize the robot. These methods can provide detailed maps with high accuracy but are sensitive to changes in lighting conditions and may struggle in feature-poor environments. Furthermore, real-time performance can be limited by the high computational demands of image processing. In contrast, 2D LiDAR-based SLAM algorithms, such as Hector SLAM and Cartographer, offer a good balance between accuracy and real-time performance. These algorithms rely on LiDAR data to create accurate 2D maps and are less affected by lighting conditions compared to visual SLAM. However, their limitation lies in their capability to generate only 2D maps, unable to capture height information. This restriction significantly hinders their application in complex 3D environments.

Furthermore, this research explores the use of image-based objects detection algorithms for recognizing and grasping target objects. These algorithms are crucial for enabling robots to visually perceive and interact with their environment. By leveraging computer vision techniques, robots can identify specific objects, determine their spatial location, and plan precise grasping actions. This capability is vital for enhancing the versatility and autonomy of robotic systems in dynamic and unstructured environments, such as those encountered in logistics and household assistance.

1.2 Prior Research

Previous research introduced a simple and cost-effective structure for tracked mobile robots as shown in Fig.1.1 Fig.1.2, along with a method for climbing stairs. This approach demonstrated the feasibility of using tracked mechanisms for multi-layer navigation, addressing key challenges in multi-floor indoor environments. While this tracked robot has basic mobility, it still requires continuous human operation during its tasks. Moreover, it lacks mechanisms for interacting with dynamic external environments, such as the ability to autonomously acquire objects for transportation. Consequently, human intervention is still necessary throughout its operation.

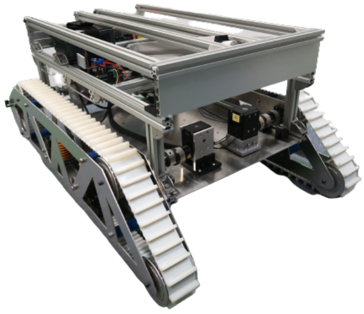

Figure 1.1: UDOn image

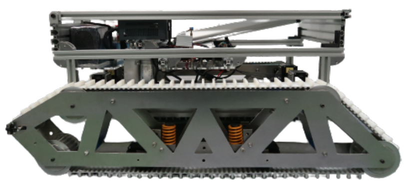

Figure 1.2: UDOn side view

1.3 Objective

This study focuses on deploying the previously developed tracked robot into real-world scenarios, such as transporting items within buildings. Our primary objective is to enhance the UDOn robot with full autonomy, enabling it to independently deliver packages to specified destinations within a building according to user requirements. To achieve this, we concentrate on two key aspects: enabling the UDOn robot to autonomously grasp target objects and navigate between different floors within a building. The subsequent sections will provide detailed explanations of our specific implementation methods.

1.4 Composition

The remainder of this paper is organized as follows. In the next chapter, we will explore related work pertinent to this paper, including algorithms, frameworks, and solutions implemented by other teams facing similar scenarios. In the methodology chapter, we will provide detailed explanations of the specific methods used in both the target grasping and navigation tasks modules. In the experiments chapter, we will present the performance of deploying the methods on the UDOn robot in real-world scenarios, and evaluate the experimental results accordingly. In the conclusion chapter, we will review and summarize the research conducted in this paper and also explore future research directions.

Chapter 2

Related Works

2.1 The Tracked robot UDOn

Previous research[37] has highlighted the significant increase in demand for efficient logistics systems due to the widespread adoption of the internet and the expansion of electronic commerce (EC) . However, last-mile delivery faces significant challenges, including labor shortages, which lead to higher costs and decreased efficiency. To address these issues, various styles of delivery robots have been developed by many teams and companies. These robots are mainly categorized into "outdoor" and "indoor" types. Indoor delivery robots often require integration with elevators, which can be challenging in older buildings without elevators, thereby limiting their ubiquity. As a solution, robots capable of moving between floors using stairs have been proposed. Our previous research developed a crawler-type delivery robot called UDOn, which can navigate between floors using stairs, and we examined the stability of this stair-climbing method. Building on this foundation, the current study aims to further enhance the autonomy and real-world performance of the UDOn robot.

2.2 Target Detection

In the field of object detection, numerous models and algorithms have been developed to enhance accuracy and efficiency. One of the pioneering approaches is the Region-based Convolutional Neural Network(R-CNN)[3] and its variants (Fast R-CNN, Faster R-CNN)[2] [21]. These models use a region proposal network to identify potential object locations and then apply convolutional networks to classify and refine these regions. Despite their high accuracy, R-CNN models tend to be computationally intensive and slower, which limits their effectiveness in real-time applications.

Single shot multiBox detector(Ssd)[10] introduced a more efficient framework by eliminating the region proposal step and directly predicting object classes and bounding boxes in a single pass through the network. This approach significantly speeds up detection while maintaining a good level of accuracy, making it suitable for real-time applications.

Another notable advancement is the You Only Look Once(YOLO)[20] family of models. YOLO models reformulate object detection as a single regression problem, directly predicting bounding boxes and class probabilities from full images in one evaluation. This leads to high-speed detection, which is advantageous for real-time applications, and its overall performance is better than the Ssd. In our research, we utilize the YOLOv5[8] model for object detection due to its balance of speed and accuracy, which is crucial for the efficient operation of our delivery robot. By integrating YOLOv5, we aim to achieve robust object recognition and grasping capabilities, essential for navigating and performing tasks in complex indoor environments.

2.3 SLAM

Simultaneous Localization and Mapping(SLAM) has been a pivotal research area in robotics, enabling robots to autonomously navigate and understand unknown environments. The choice of sensor configurations significantly influences the selection and implementation of SLAM algorithms. Based on the sensor type, SLAM algorithms can be broadly categorized into LiDAR SLAM, visual SLAM, and multi-sensor fusion SLAM.

LiDAR SLAM primarily utilizes LiDAR sensors to capture detailed depth information of the environment, offering key advantages such as high accuracy and robustness against varying lighting conditions. Classical 2D LiDAR SLAM algorithms like Gmapping[4], Hector SLAM[9], and Cartographer[5] are well-regarded, with Cartographer being particularly notable for its widespread adoption and reliability in both research and industrial applications. Cartographer is extensively used for autonomous navigation and environmental mapping across unmanned vehicles, UAVs, and warehouse robots. In the realm of 3D Lidar SLAM, algorithms like LOAM[32], LIO-SAM[24], and FAST-LIO[30], have achieved notable advancements and are increasingly becoming mainstream in various industries. However, due to the high cost of 3D Lidar, the 3D LIDAR solution is not used in our current study.

Visual SLAM utilizes camera sensors to capture visual information of the surrounding environment, offering advantages such as low cost and detailed environmental representation. However, due to intensive image processing requirements, it demands significant computational resources. Prominent works in pure visual SLAM include feature-based methods like ORB-SLAM[17], which was later integrated with IMU in ORB-SLAM3[1] Other notable approaches include the semantic-aware Cube-SLAM[31], the IMU-enhanced VINS-MONO[19], and more recent advancements like Nerf-SLAM[22], which integrates neural radiance fields[15].

2.4 Navigation

While SLAM technology enables robots to perceive and map their surroundings, navigation technology facilitates autonomous movement and decision-making within that environment. Recent advancements in robotics navigation frameworks have significantly enhanced the capabilities of autonomous systems. One prominent example is the Nav2 stack[13], a versatile open-source navigation framework built on ROS (Robot Operating System). Nav2 is designed to provide robust path planning and navigation for various robotic platforms. This high-quality navigation framework offers a wide range of functionalities and algorithms, including perception, planning, control, and localization, making it a valuable tool for implementing autonomous navigation capabilities in our research.

2.5 Similar Studies

Y. K. Tee and Y. C. Han et al.(2021)[26] conducted a comprehensive study reviewing and comparing three common 2D SLAM algorithms (GMapping, Hector-SLAM, and Google Cartographer) using ROS-based SLAM libraries on a mobile robot equipped with a 2D LiDAR, IMU, and wheel encoders. Their work described the strengths and weaknesses of these algorithms and visualized the differences in the constructed maps.

Q. Zou et al.(2021)[35] analyzed and compared various 3D LiDAR SLAM algorithms, evaluating their performance in real indoor environments with a primary focus on industrial application scenarios.

Maxim Sokolov et al.(2017)[25] analyzed the performance of LiDAR SLAM and Visual SLAM algorithms on tracked robots. Their comparative analysis showed that the LiDAR odometer was close to the ground truth, while the visual odometer exhibited significant trajectory deviation. This finding was instrumental in informing our choice of methods for this study.

Xuexi Zhang et al.(2020)[34] evaluated the application of 2D LiDAR SLAM combined with path planning algorithms in indoor rescue environments. Although they achieved good results, the maximum slope in their simulated environment was only about 15 degrees, and no research was conducted on climbing stairs. Many studies deploy 2D LiDAR SLAM algorithms for indoor localization and mapping, primarily focusing on planar scenes, with navigation algorithms based on planar maps.

Indoor service robots are widely used in fields such as cleaning and delivery. However, their mobility is significantly constrained by the indoor environment. For instance, stairs are a major obstacle limiting the accessible areas for these robots. T. Seo and S. Ryu et al.(2023)[23] reviewed the development of stair-climbing robots, covering mechanical structures, sensors, and specific performances. While legged robots have largely addressed the challenge of stairs, their high cost and lower efficiency on flat surfaces compared to wheeled robots are significant drawbacks. Therefore, considering overall mobility, the ability to overcome stairs, and cost-effectiveness, tracked mobile robots might be the better choice.

At present, there are also many studies focusing on the structure, mobility performance and control of tracked mobile robots[18] [6] [7], including the research of Junyan Yang et al.[37] [38]

Chapter 3

Methodology

3.1 Target Detection and grasping

In this section we will introduce the main methods of UDOn to achieve object detection and target grasping.

3.1.1 Arm mechanism

In a previous study[36] on robotic arm mechanism development, we designed and fabricated a cost-effective 3-DOF robotic arm for the UDOn robot. This arm successfully accomplished the task of acquiring an object in front of the robot and placing it on the rear UDOn carrier. Despite its limited degrees of freedom, the arm's unconventional structure enabled effective gripping and placing maneuvers. The structural appearance and kinematic model of the robotic arm are shown in Fig.3.1. The D-H parameters used for the forward kinematics calculations of the robotic arm are presented in Table 3.1. For the inverse kinematics calculations, we simplified the process to a two-link inverse kinematics model, as only two joints are involved in grasping the target object, as shown in Fig.3.2.

Table 3.1: Standard DH table

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	$\pi/6$	0	L_0	θ_1
2	$\pi/2$	L_2	L_1	θ_2
3	0	L_3	0	θ_3

We integrated the hardware and software systems of the robotic arm with UDOn, as shown in Fig.3.3. The entire system is powered by a lithium polymer battery. An Intel NUC Kit serves as the host computer, managing sensor data processing, motion control, and object detection computations based on image data.

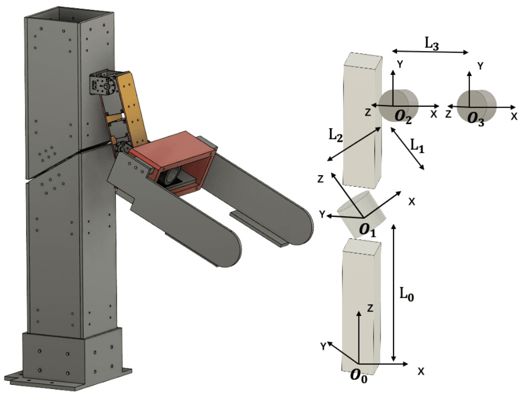

Figure 3.1: Structural appearance and kinematic model of UDOn arm

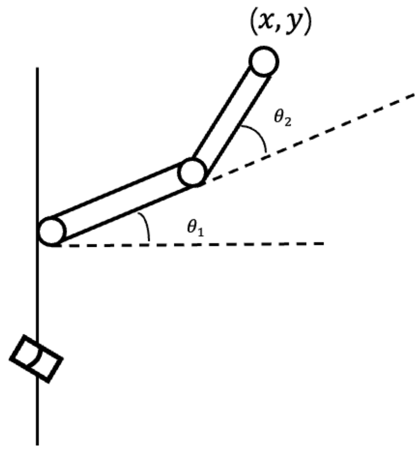

Figure 3.2: Simplified inverse kinematic model

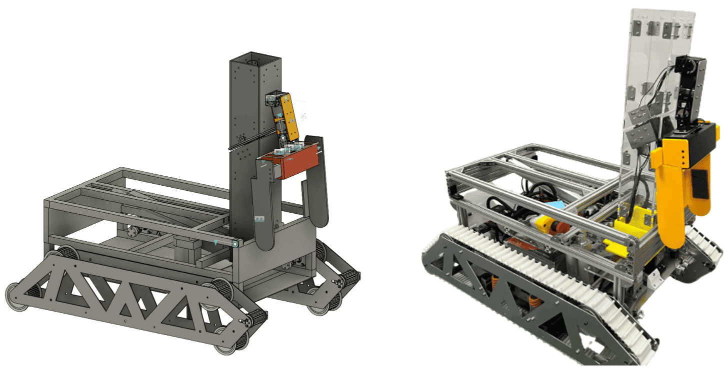

Figure 3.3: Combination of UDOn and arm

3.1.2 Target detection and depth calculation

Object detection is a crucial task in the field of computer vision, aiming to identify and locate specific categories of objects from images or videos. With the development of deep learning techniques, particularly Convolutional Neural Networks (CNN), many deep learning-based target detection methods have been proposed such as YOLO[20], Faster R-CNN[21], Ssd[10], etc. Different algorithms are applied in various fields according to their unique characteristics. Considering the working scenario of the UDOn, target object detection must be carried out in real-time, necessitating specific requirements for detection speed and performance. The YOLO algorithm is advantageous for its real-time performance and computational speed. First proposed in 2015, YOLO has continuously evolved, with the latest iteration being YOLOv8. Juan R. Terven et al.[27] reviewed the development of YOLO, highlighting its consistent advancements in accuracy while retaining high-speed capabilities. Among the various iterations, YOLOv5 has gained widespread popularity due to its end-to-end architecture, lightweight nature, and straightforward deployment process. Consequently, YOLOv5 was selected as the preferred algorithm for the target detection training aspect of this research.

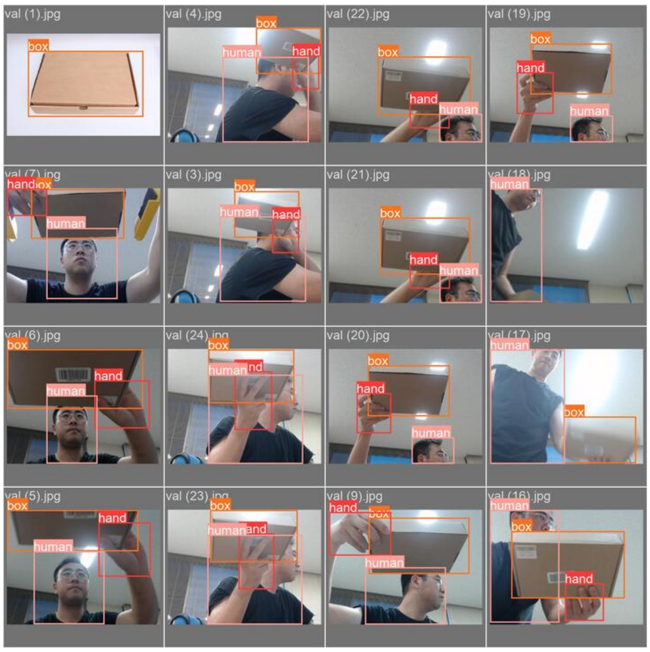

Figure 3.4: Validation set labels

Considering the specific working scenario, the UDOn is currently mandated to identify packages situated in its frontal area, subsequently activating the arm to grip and deposit the package onto its carrier platform. The primary recognition target is rectangular packages. To facilitate this, we constructed a dataset of approximately 300 images reflecting UDOn's operational scenario. Objects within the images were labeled using a dedicated labeling tool, and the dataset includes three categories: box, human, and human hand.

The dataset was trained using the YOLOv5 open-source framework released by Ultralytics. The training results on this dataset are presented in Fig.3.5 and Fig.3.6. The performance of the model on the validation set is shown in Fig.3.4.

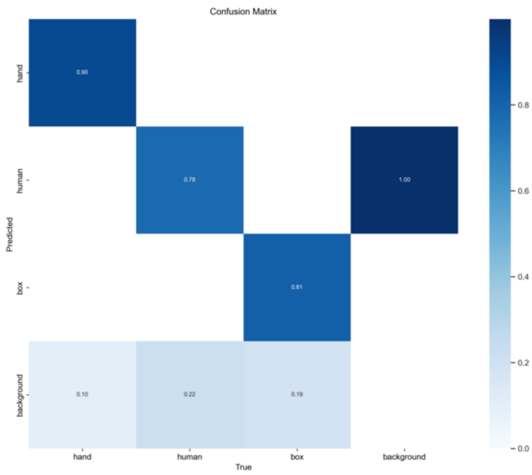

Figure 3.5: Confusion matrix of trained model

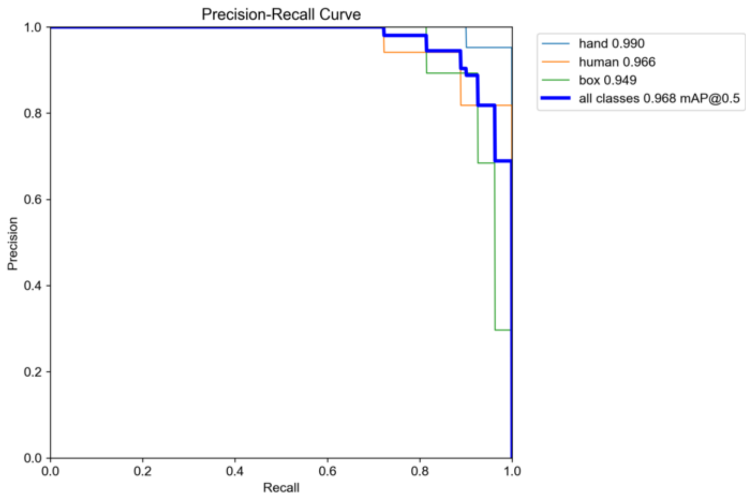

Figure 3.6: PR curve of trained model

We employed an RGB-D camera to capture both color (RGB) and depth (D) information of the environment. To ensure accurate and robust distance measurements, we applied a median filtering algorithm to the depth data. This filtering process reduces noise and outliers, providing more reliable distance estimates.

The median filter algorithm involves three main steps:

1. **Sorting**: Given a set of depth values $D = \{d_1, d_2, \ldots, d_n\}$ within a neighborhood window, sort these values in ascending order.

Figure 3.7: Realsense D435

2. **Selecting the Median Range**: Select the values within a range as you wish, for example from $\frac{3}{n}$ to $\frac{2n}{3}$.

3. **Averaging**: Calculate the filtered depth value d_{filtered} as the average of the depth values within the selected range:

$$d_{\text{filtered}} = \frac{1}{m} \sum_{i=\frac{n}{3}}^{\frac{2n}{3}} d_i$$

   where m is the number of values in the selected range.

The combination of RGB-D data and median filtering allows for precise distance calculations, which are crucial for the accurate operation of the robotic arm during object detection and grasping tasks.

3.1.3 Implementation

Utilizing the PyTorch framework, we deployed a trained model to capture real-time images through the implementation of an RGB-D camera affixed to the UDOn, and after the real-time processing of each frame, the actual detection results was presented in Fig.3.8.

From the derived test results, it is evident that while the dataset employed in this study remains limited in scale, it effectively aligns with the experimental requirements for evaluating the robot arm's grasping functionality. The model demonstrated a remarkable ability to swiftly identify target objects within each frame and exhibited commendable accuracy levels.

Since the entire control system operates in the ROS2 environment, we encapsulated different modules as nodes and used a topic-based communication mechanism for inter-node communication. When the camera node detects a target object within its field of view, it calculates the

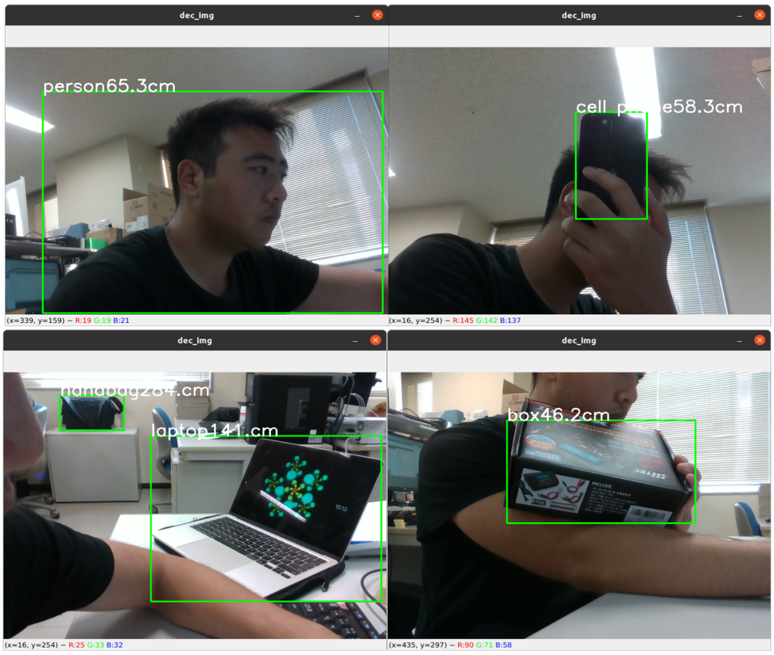

Figure 3.8: Median filter results

distance from the camera to the object and sends the object's position in the image along with the distance data to the data processing node. Upon receiving this data, the data processing node calculates the relative position coordinates of the target object, converts these coordinates to the robotic arm's coordinate system, and sends the coordinates to the robotic arm control node. The robotic arm control node then performs inverse kinematics calculations based on the coordinate data to determine the angles for each joint, controlling each joint to move the robotic arm's end effector to the target object's position for grasping. The communication relationships among the various components of the system are illustrated in Fig.3.9, while the control logic for whether the robotic arm should grasp is shown in Fig.3.10.

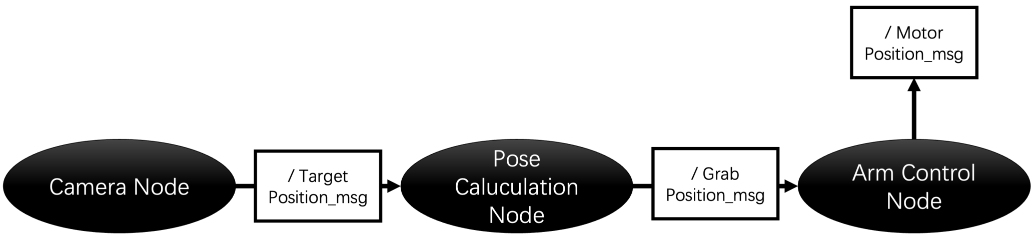

Figure 3.9: Communication relationships among nodes

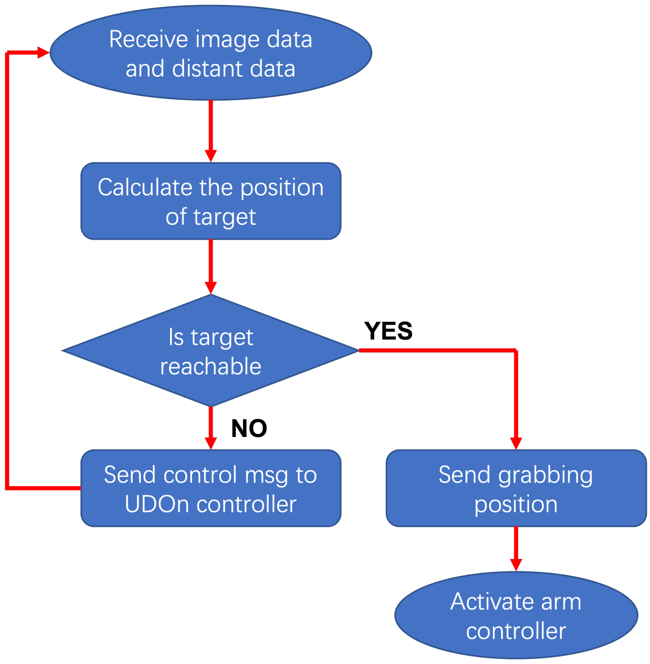

Figure 3.10: Arm control logic

3.2 SLAM and Navigation

In this section we will introduce how UDOn implements SLAM and navigation functions.

3.2.1 2D Lidar SLAM

Simultaneous Localization and Mapping(SLAM) is a technique used to create a map of an environment while simultaneously determining the location of the robot within that map. In this study, we utilized a 2D LiDAR SLAM approach to achieve robust indoor navigation and mapping.

The hardware setup of our 2D LiDAR SLAM system includes a 2D LiDAR (Fig.3.11), an IMU module (Fig.3.12), and UDOn that carries all the hardware devices. The 2D LiDAR sensor that we used was manufactured by SLAMTEC, specifically the RPLIDAR A1 model. The RPLIDAR A1 utilizes triangulation and rotates clockwise to perform distance measurements, enabling 360° scanning of the surrounding environment and generating a comprehensive map of the area. The IMU is a 10-axis inertial navigation ARHS sensor module, equipped with an accelerometer, gyroscope, magnetometer, and barometer.

Figure 3.11: 2D LiDAR

For the SLAM implementation, we used the Cartographer algorithm provided by Google, which is integrated within the ROS2 framework. Cartographer is a robust SLAM algorithm that provides real-time 2D and 3D mapping. It uses a combination of scan matching, loop closure, and global optimization to produce accurate maps. High level system overview of Cartographer is shown in Fig.3.14.

While 2D LiDAR SLAM is effective in many scenarios, it faces significant challenges in environments with repetitive structures, such as hallways. These environments often lack distinctive features, leading to several issues:

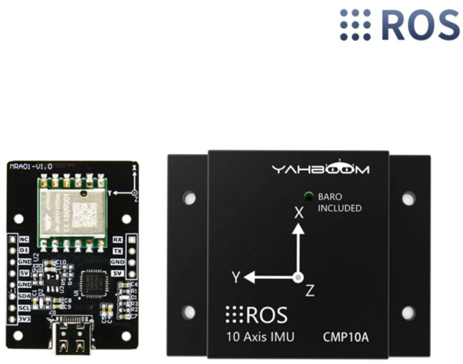

Figure 3.12: IMU

Figure 3.13: Dynamixel servo motor

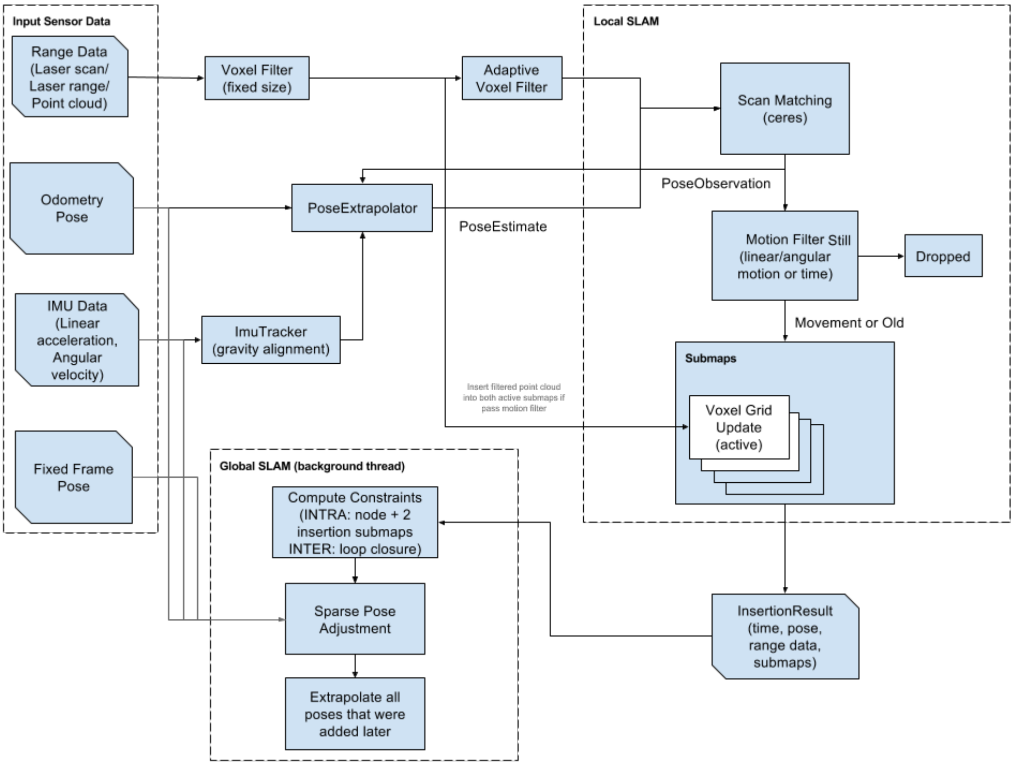

Figure 3.14: system overview of Cartographer. Source: `https://google-cartographer.readthedocs.io/en/latest/`

1. **Lack of Distinctive Features**: Hallways tend to have smooth, featureless walls, which can make it difficult for the SLAM algorithm to identify unique landmarks. This can result in ambiguous data, making it challenging for the algorithm to determine the robot's precise location.

2. **False Loop Closures**: In repetitive environments, the SLAM algorithm might incorrectly identify similar-looking locations as the same place, leading to false loop closures. This can introduce errors into the map and cause the robot to misjudge its position.

3. **Pose Estimation Errors**: Due to the lack of distinctive features, the algorithm may struggle with accurate pose estimation, leading to drift over time. This drift accumulates and can distort the generated map, reducing its overall accuracy.

To mitigate these challenges, we employ several strategies within the Cartographer algorithm framework, which processes LiDAR data in three main steps:

1. **Data preprocessing**: First, the LiDAR data is preprocessed to filter out noise and unreliable measurements, enhancing the accuracy of subsequent processing.

2. **Scan matching**: Cartographer employs scan matching to align the current frame of LiDAR data with the previously constructed submap. The objective is to find a pose (position and orientation) that optimizes the match between the current LiDAR frame and the submap features, typically using optimization algorithms like Ceres Solver. During scan matching, the primary computation involves assessing the distances or similarities between LiDAR scan points and the submap.

3. **Loop Closure**: Cartographer includes a loop closure mechanism to recognize and correct the robot's path when it revisits previously traversed areas. Loop closure involves searching the global map for regions that match the current LiDAR data frame, further refining the map and the robot's global pose.

In summary, the Cartographer algorithm utilizes each LiDAR data frame for efficient mapping and localization. Although explicit feature points are not computed for each frame in 2D LiDAR data, implicit feature matching is still used for data alignment and optimization. In environments with repetitive structures, such as long corridors, the high similarity between LiDAR frames can lead to incorrect position estimates.

To address the challenges of precise robot positioning, we integrated IMU sensors with the encoders of UDOn's wheel control servos as shown in Fig.3.13. Initially, we considered using the servo encoder data from UDOn's four wheels combined with its kinematic model to enhance the localization system. The details of UDOn's kinematic model will be discussed in the next subsection. For tracked robot motion control, the presence of a dimensionless parameter complicates precise control due to significant slippage. Therefore, additional sensors are necessary to monitor the robot's movement accurately.

The IMU sensor, featuring an integrated gyroscope, accelerometer, and sometimes a magnetometer, provides high-precision angle change information in the short term, aiding in tracking the device's orientation and posture. The accelerometer measures linear acceleration, offering information on spatial movement, while the magnetometer measures geomagnetic field strength and direction, providing an absolute directional reference. However, in practical environments, the accelerometer and magnetometer are highly susceptible to interference, leading to drift in displacement data over time, even with filters. Nonetheless, the angular velocity data from the gyroscope remains stable.

Ultimately, we integrated these three data sources to achieve robust localization for UDOn. The kinematic model accurately calculates displacement, the IMU sensor effectively tracks orientation and rotation, and the LiDAR sensor compensates for cumulative errors from the other two sources. Within the Cartographer algorithm framework, this approach results in accurate and highly robust odometry data. Using the Cartographer algorithm, we constructed 2D plane maps of multiple floors in the UDOn robot's working environment. These maps were then used to conduct subsequent navigation function tests.

3.2.2 UDOn kinematic model

Tracked mobile robots are highly effective for applications requiring off-road mobility, such as agriculture, military operations, forestry, mining, and planetary exploration. Their track-based locomotion provides a larger contact area with the ground, resulting in better traction compared to wheeled vehicles on natural terrains. This enhanced grip allows them to navigate challenging environments more effectively.

Many studies have conducted in-depth research on the motion models of tracked mobile robots[14][16][28]. However, the motion control of these robots remains a complex issue, and there is currently no universal solution. Most research is based on simulation results and often incorporates complex vehicle dynamics. Therefore, this paper will primarily discuss the motion model of the most basic configuration of tracked unmanned vehicles. While the robot's configuration may vary across different application scenarios, the fundamental principles of the basic kinematic model remain consistent.

To simplify the model, we make two assumptions:

1. The UDOn robot does not experience slippage during tracked movement.
2. The mass of the UDOn robot is uniformly distributed, with the center of mass and geometric center located along the robot's longitudinal symmetry axis, although they do not necessarily coincide.

The velocity decomposition diagram of the UDOn kinematic model is shown in Fig.3.15. COM represents the center of mass, CENTER represents the geometric symmetry center of the robot, and ICR represents the motion rotation center of the robot.

Given the motion characteristics of the tracked robot, we simplify the model by treating it as a two-wheel differential drive robot. This simplification is based on the requirement that the rotation speeds of the servos controlling each track must be equal. As illustrated in Fig, we define the Instantaneous Center of Rotation (ICR) along the horizontal axis and the Center of

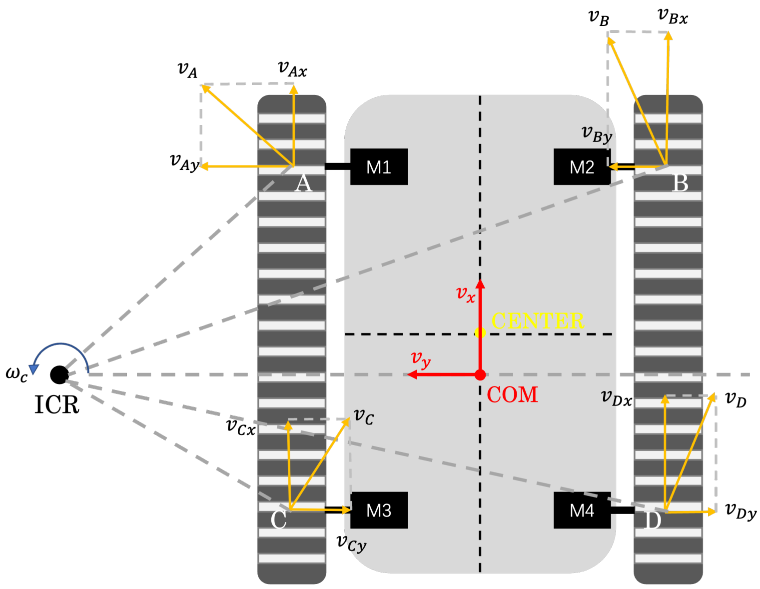

Figure 3.15: Velocity decomposition diagram of the UDOn

Mass (COM) along the vertical axis. The virtual left and right wheels are positioned at points L and R, respectively.

It is important to note that the distance between the virtual wheels (LR) does not necessarily match the actual spacing between the tracks; rather, it varies dynamically and is influenced by sliding friction. Consequently, the interaction between the ground and the tires, which may have different friction coefficients, can significantly impact the robot's actual rotational motion. The kinematic model of the tracked robot follows the approach outlined in reference[29], with the simplified model depicted in Fig3.16.

By applying the motion model of the two-wheel differential drive robot, we derive a simplified forward kinematic model(1) for the tracked robot. This model calculates the speed of the geometric center of mass (COM) based on the speeds of the virtual left and right drive wheels. The following equations detail this kinematic relationship:

$$\begin{bmatrix} v_c \\ w_c \end{bmatrix} = \begin{bmatrix} \frac{v_L+v_R}{2} \\ \frac{v_L-v_R}{d_{LR}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{d_{LR}} & -\frac{1}{d_{LR}} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \tag{3.1}$$

The inverse kinematic model for the left and right drive wheels, based on the speed of the geometric center of mass (COM), is decomposed as follows:

$$\begin{bmatrix} v_L \\ w_R \end{bmatrix} = \begin{bmatrix} v_c + \frac{d_{LR}}{2} w_c \\ v_c - \frac{d_{LR}}{2} w_c \end{bmatrix} = \begin{bmatrix} 1 & \frac{d_{LR}}{2} \\ 1 & -\frac{d_{LR}}{2} \end{bmatrix} \begin{bmatrix} v_c \\ w_c \end{bmatrix} \tag{3.2}$$

In the above equations, d_{LR} represents the distance between the virtual wheels. During calculations, a dimensionless parameter r is introduced, resulting in the expression for d_{LR} as

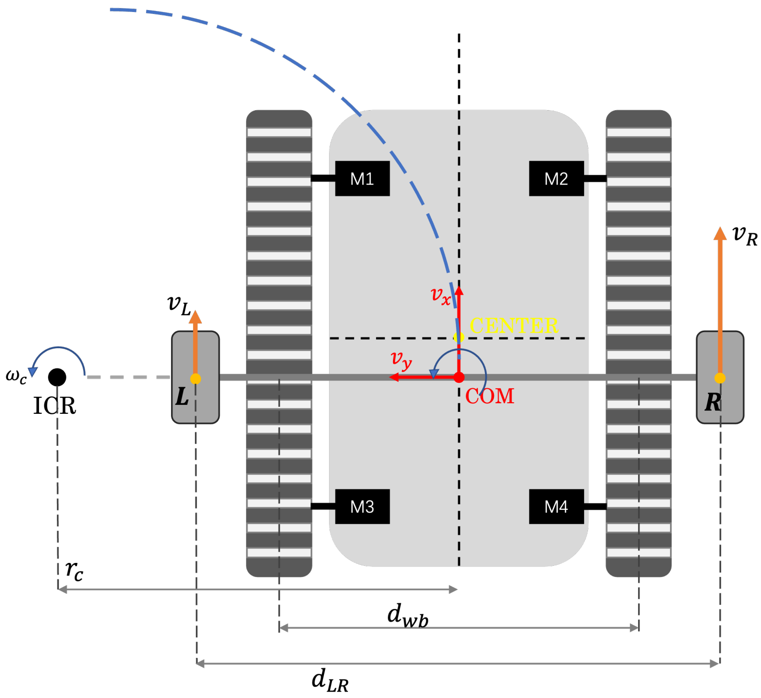

Figure 3.16: Simplified UDOn kinematic model

shown in (6). Here, d_{wb} denotes the actual distance between the two tracks.

$$d_{LR} = \gamma d_{wb} \tag{3.3}$$

The introduction of the dimensionless parameter r results in an unknown variable in the calculation process. Our current challenge is to solve for this unknown parameter γ. This parameter is influenced by the total load of the robot, the relative friction coefficient between the tracks and the ground, the turning radius, and the position of the center of mass. It is a complex parameter, often determined through specific experiments. We conducted practical experiments using the UDOn robot. Due to varying friction between different surfaces and the tracks, we could only obtain an average value. This leads to cumulative errors in odometry calculations when the UDOn robot performs turning operations.

Fortunately, the kinematic errors during straight-line movement of the UDOn robot are minimal, making the odometry data calculated from kinematics relatively accurate for straight-line predictions. This accuracy compensates for the low reliability of displacement data from the IMU sensor's accelerometer, which often suffers from zero drift. Thus, the displacement data during straight-line movement in long corridors is the most reliable for the UDOn robot, significantly improving the accuracy of the fused odometry data.

3.2.3 Nav2 Stack

The navigation functionality for the tracked delivery robot in this research was implemented using the Nav2 stack[13] , a well-known open-source project designed for robust and flexible robotic navigation. The Nav2 stack, part of the ROS2 ecosystem, provides a modular framework that integrates various essential components for autonomous navigation, including localization, path planning, and control(Fig.3.17). This stack was chosen for its versatility, active community support, and extensive documentation, which facilitated the integration process with the robot's hardware and software systems.

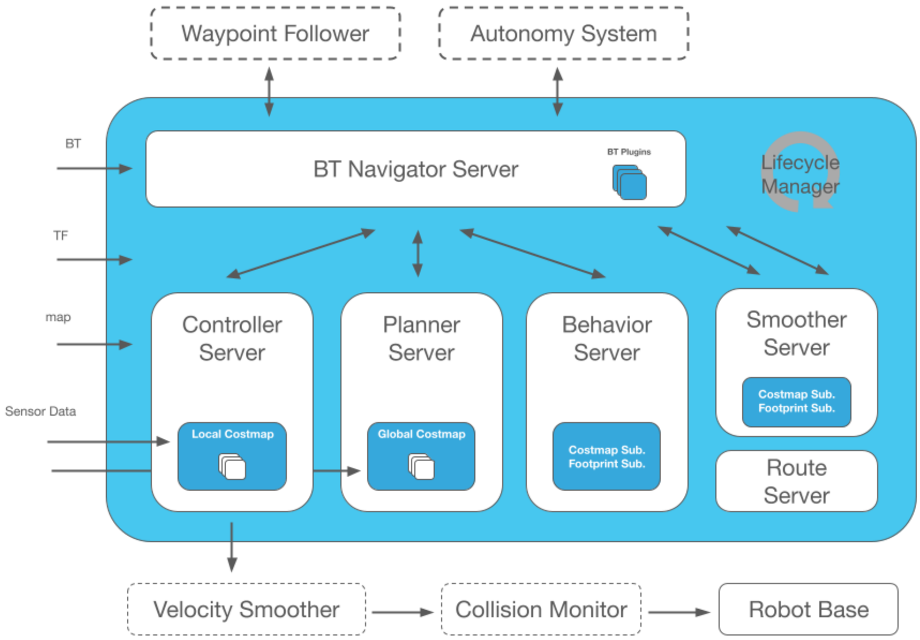

Figure 3.17: Nav2 stack architecture. Source: `https://docs.nav2.org/`

The primary objective was to enable the robot to navigate complex environments autonomously, including maneuvering through corridors, avoiding obstacles, and using stairs for inter-floor transport. The Nav2 stack's modularity allowed for customization and fine-tuning of each component to meet the specific requirements of our robot and its operational environment. By leveraging the capabilities of the Nav2 stack, the robot achieved high levels of autonomy and reliability, crucial for its intended application in object grasping and delivery tasks across multiple floors in a building.

Localization was achieved through the Adaptive Monte Carlo Localization (AMCL) algorithm[33]. AMCL uses a particle filter to estimate the robot's pose within a given map. The map was created using simultaneous localization and mapping (SLAM) techniques with the help of the Cartographer package. During the navigation process, sensor data from the LIDAR, IMU and wheel encoders were continuously fed into the AMCL module to refine the robot's position.

For path planning, the Nav2 stack utilizes a set of optimized path planning algorithms[11] for global path planning and the Dynamic Window Approach(DWA) for local path planning. The global planner computes the optimal path from the robot's current location to the target position, ensuring collision-free movement. The local planner dynamically adjusts the robot's path to navigate around unforeseen obstacles in real-time. The combination of these two planning strategies allows for efficient and safe navigation in dynamic environments.

The robot's movement control was handled by the Nav2 Controller Server, which generates velocity commands based on the planned path and real-time sensor feedback. This server ensures smooth and accurate execution of the planned trajectories, considering the robot's kinematic constraints and ensuring stability during operation on various terrains.

A critical aspect of the Nav2 stack's navigation functionality is the use of Behavior Trees(BTs) for task coordination and decision-making. Behavior Trees provide a structured and flexible way to define complex robot behaviors through a combination of simple tasks and conditions. In the context of the Nav2 stack, BTs are used to manage the sequence of navigation actions, such as localization, goal setting, path planning, obstacle avoidance, and movement control. Considering that the process of navigating between different floors performed by UDOn requires the task of climbing stairs, which needs to be separated from the navigation task so that it can be done independently. The BT framework in Nav2 allows for clear separation of different navigation task. Each node in the BT represents a specific action or condition, and these nodes are arranged in a tree structure that dictates the order and conditions under which tasks are executed. This modular approach enhances the robot's ability to handle dynamic and unpredictable environments by allowing it to react to changes in real-time and switch between different behaviors as needed.

In this study, we utilized only the localization, path planning, and navigation control modules from the Nav2 stack, without integrating UDOn's stair climbing tasks. Instead, we employed separate control nodes for managing UDOn's inter-floor movement and map switching. Future research will focus on a deeper exploration of the Nav2 stack, aiming to integrate UDOn's navigation, obstacle avoidance, and stair utilization modules using the BT framework.

3.2.4 Implementation

The deployment of the Cartographer algorithm and Nav2 stack on the UDOn robot was carried out in an Ubuntu 20.04 environment using ROS-Foxy. Each module was encapsulated as a separate node, and each node published its corresponding data topics, as shown in Fig.3.18. The Cartographer algorithm was used to create detailed 2D maps of each floor by integrating data from the IMU, 2D LiDAR, and the robot's servo encoders. The IMU node provided orientation data, the 2D LiDAR node captured distance measurements, and the servo encoder node supplied wheel odometry information. These data streams were published to relevant topics and fused in realtime by the Cartographer node to construct accurate 2D maps of the surrounding environment and generate predicted odometry data for UDOn.

We conducted separate tests for UDOn's SLAM and navigation functionalities. For SLAM we focused on the mapping functionality with the goal of creating high-quality 2D maps. we

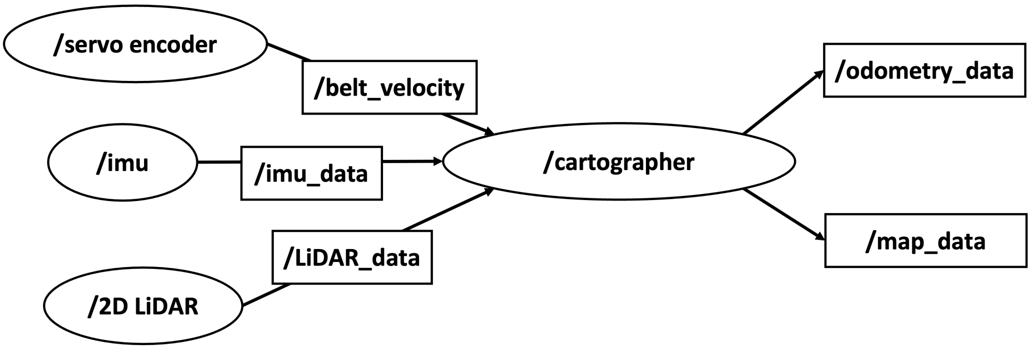

Figure 3.18: UDOn SLAM system overview

used a solution that relied solely on 2D LIDAR. However, due to the repetitive features in the corridor environment and the accuracy limitations of the LIDAR, the Cartographer algorithm struggled to compute precise odometry information based on LIDAR data alone, resulting in suboptimal mapping performance. It was only after we integrated IMU and encoder data that we achieved the current high-quality mapping results. Through continuous testing and parameter adjustments in real-world environments.

Once the maps were generated, they were utilized for navigation tasks. The localization node within the Nav2 stack used the generated maps to determine the robot's position, while the path planning node computed optimal path for navigation task. The navigation control node executed these paths, ensuring efficient movement and obstacle avoidance. To enhance the accuracy of robot localization during actual deployment, we manually set the initial pose estimate. This approach facilitated subsequent path planning and navigation experiments. The Nav2 stack also utilized the maps to predict the robot's position and generate cost maps for efficient path planning. The navigation system overview of UDOn was shown in Fig.3.19.

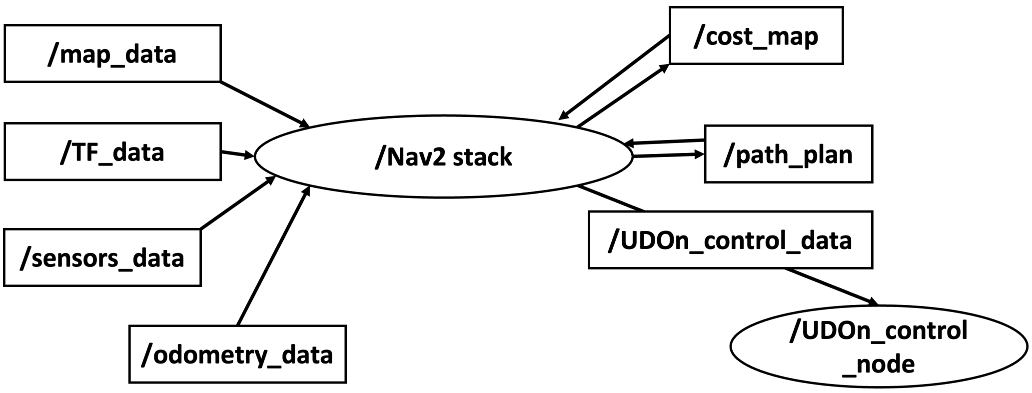

Figure 3.19: UDOn navigation system overview

Extensive testing validated the system's performance. The robot successfully navigated through various floors, demonstrating the effectiveness of the Cartographer-generated maps and the robustness of the Nav2 stack in performing autonomous navigation tasks. This deployment provided a robust framework for the UDOn robot, combining advanced SLAM capabilities with reliable navigation control, and facilitating efficient and accurate movement across different floors.

Chapter 4

Experiments

4.1 Grasping Performence

Utilizing the PyTorch framework, we deployed a trained model to capture real-time images through the implementa- tion of an RGB camera affixed to the UDOn robot, and after the real-time processing of each frame, the actual detection results was presented in Fig.4.1.

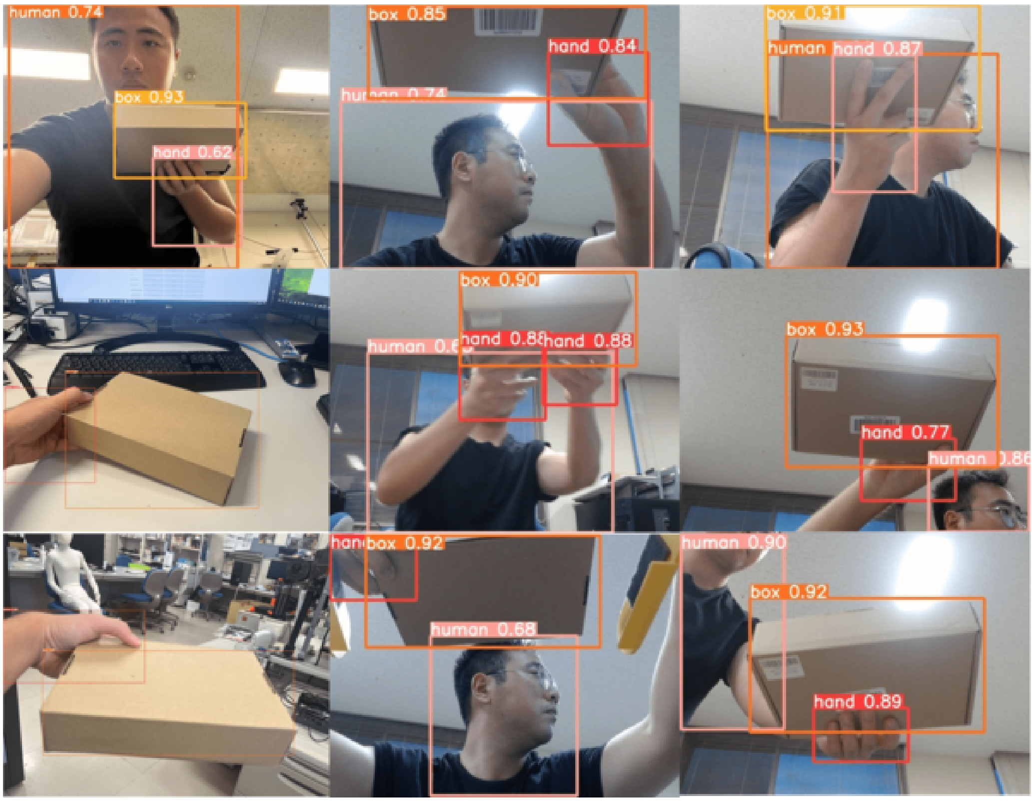

Figure 4.1: Actual detection results

From the derived test results, note that while the dataset employed in this study remains limited in scale, the focus of the detection task has effectively aligned with the experimental requisites for evaluating the capabilities of the robot arm's grasping functionality. The model demonstrated a remarkable aptitude for swiftly identifying target objects present within each frame and exhibited commendable accuracy levels. In this experiment, the activation process of the robot arm is shown in Fig.3.10.

In pursuit of enhancing the robustness of the control system, a strategic adjustment was implemented in the activation criteria governing the actions of UDOn arm. Specifically, activation timing of the UDOn arm in the grasping action was recalibrated to occur when a predetermined count of frames within a specified temporal interval. This modification was introduced with the overarching goal of mitigating potential risks, such as erroneous triggering of the robotic arm grasping program due to computational errors occurring in specific image frames. Given that the camera sensor in this research is at a rate of 30fps, the revised activation condition was established to activate the grasping program once the tally of frames featuring the target object detected attains 15 frames within a single second. ensuring that the control system is adept at target object presence and activating the UDOn arm when warranted. Subsequent to the successful gripping of the target object, the move pattern of the UDOn arm is visually detailed in Fig.4.2.

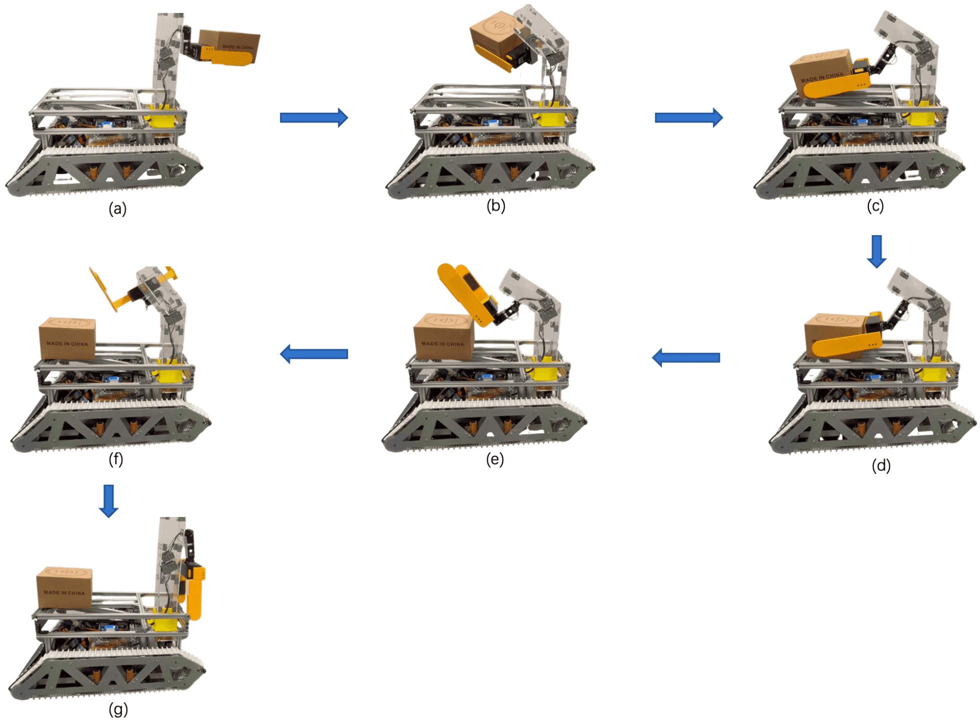

Figure 4.2: Move pattern of UDOn arm after gripping the target

4.2 Map Building

In this section, we detailed the experimentation of the Cartographer algorithm in an Ubuntu 20.04 environment using the ROS-Foxy distribution. The process involved several key steps, including setting up the ROS environment, configuring the necessary nodes, and ensuring effective communication between these nodes. The first step was to install ROS-Foxy on Ubuntu 20.04. This provided the foundational infrastructure needed to run the Cartographer algorithm and other ROS nodes. Cartographer and its dependencies were installed from source to ensure compatibility with our specific system setup.

After completing the environment setup and algorithm preparation, we wrote the corresponding driver and data publishing nodes for different sensor modules in the ROS2 system environment. ROS2 nodes for modules such as IMU and LIDAR were obtained from the open-source code libraries released by the manufacturers. For the UDOn robot's servo speed encoder, real-time data was read to obtain the necessary information.

To visualize the communication between nodes, we used ROS2 built-in rqt-graph tool to generate the communication graph as shown in Fig4.3. This graph illustrates how data flows between the IMU, 2D LIDAR, encoder nodes, and the cartographer node, which integrates these data streams to generate 2D maps. We initially constructed the floor plans for the 5th and 6th floors of our building. The maps generated directly by the algorithm are displayed in Fig.4.4 and Fig.4.5.

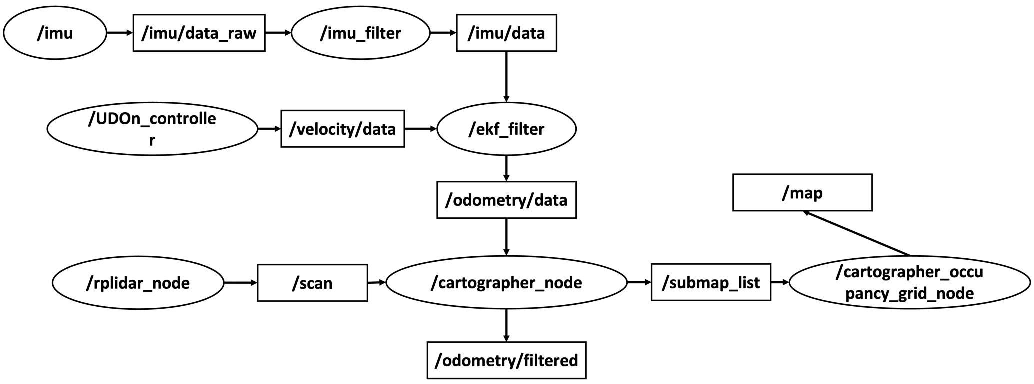

Figure 4.3: Rqt graph of SLAM system

In addition to Cartographer, we conducted experiments using the SLAM Toolbox[12] to compare their performance. The SLAM Toolbox was also deployed in the same Ubuntu 20.04 and ROS-Foxy environment, with same sensor configurations. SLAM Toolbox is another powerful tool for Simultaneous Localization and Mapping in ROS environments, which offers a simpler setup process compared to Cartographer, making it easier to integrate and deploy for various applications. While Cartographer can handle both 2D and 3D mapping, SLAM Toolbox is optimized for 2D environments, making it highly efficient for floor mapping in indoor settings.

Cartographer integrates IMU and encoder data, generally produces more accurate maps.

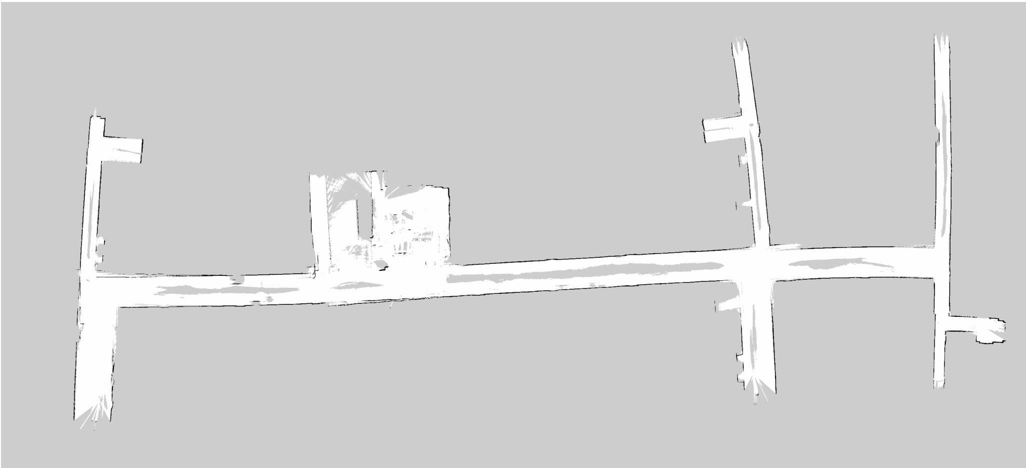

Figure 4.4: 5th Floor Map

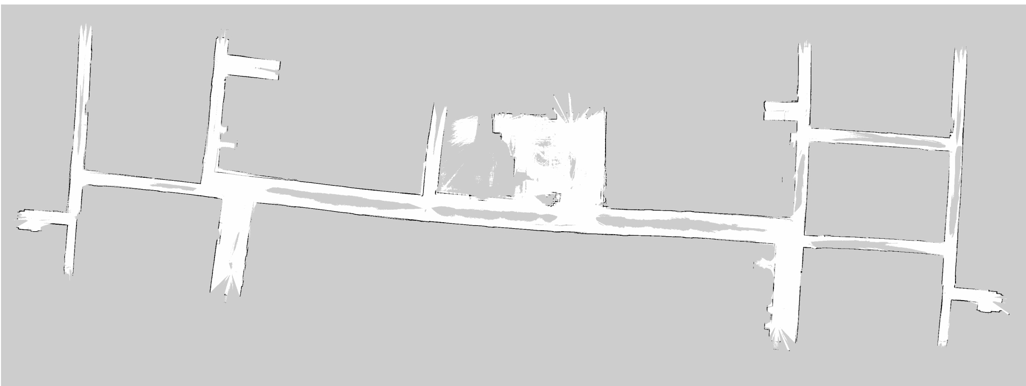

Figure 4.5: 6th Floor Map

While SLAM Toolbox is more efficient, it also relies heavily on accurate odometry data. Inaccuracies in odometry can significantly affect the quality of the generated maps, it encountered loop detection failures during our tests, resulting in map overlaps (Fig4.6). It is possible that our limited understanding of the SLAM Toolbox has resulted in incorrect parameter settings. In future research, we plan to conduct a more in-depth study to address these issues. In contrast, Cartographer's integration of IMU and encoder data helps mitigate such issues, providing more robust performance in environments with less reliable odometry. Fig.4.7 illustrate the Rviz visualization during SLAM.

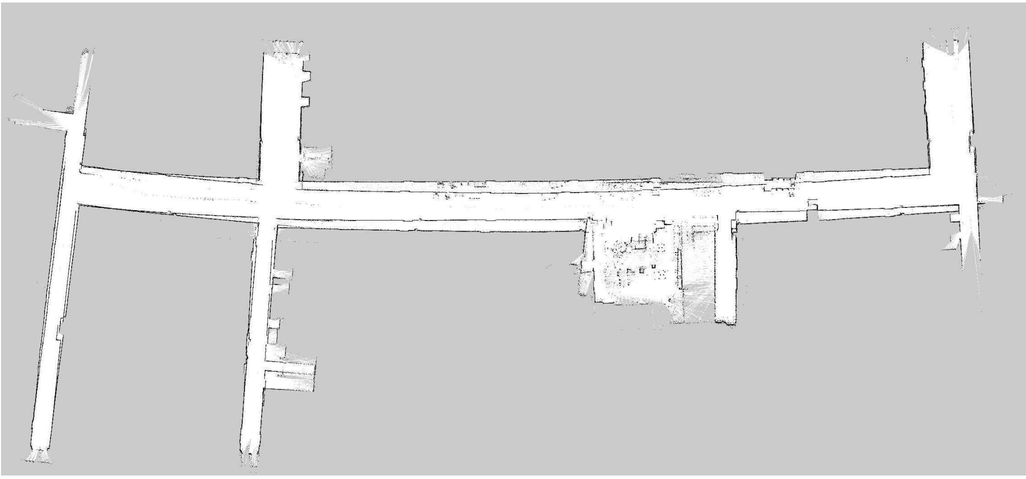

Figure 4.6: Overlaped map using SLAM Toolbox

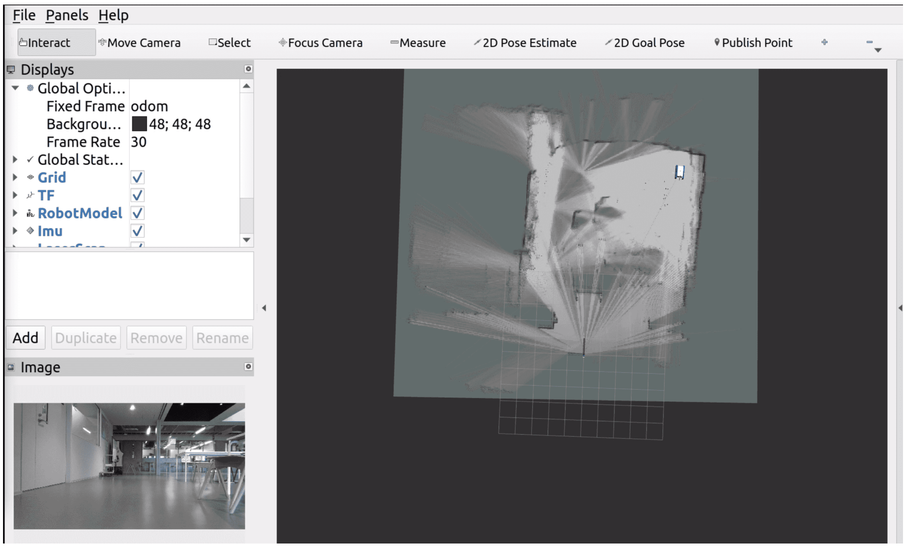

Figure 4.7: Rviz visualization during SLAM

4.3 Autonomous Navigation

After generating high-quality maps with the Cartographer algorithm, we tested the navigation functionalities using the Nav2 stack. The Nav2 stack was deployed in the same Ubuntu 20.04 and ROS-Foxy environment. Its architecture is highly modular, with various functionalities implemented as plugins. These plugins can be swapped or reconfigured, allowing for easy customization of navigation behavior for different robots and environments.

The primary categories of plugins in the Nav2 stack include global and local planners, controllers, behavior trees, recovery behaviors, and costmap layers. This modularity ensures that each component can be independently developed, tested, and optimized, resulting in a robust and flexible navigation system. In our study, we utilized several key plugins in the Nav2 stack to enhance the navigation capabilities of the UDOn robot, as summarized in Table4.1.

Table 4.1: Plugins

Plugin Category	Plugin Name
Costmap Layers	Voxy Layer
	Static Layers
	Obstacle Layers
	Inflation Layers
Controllers	DWB Controller
	Rotation Shim Controller
Planners	NavFn Planner
Behaviors	Spin, Back Up and wait

Detailed instructions for these plugins can be found on source: `https://docs.nav2.org/`.

When selecting the FollowPath plugins, we opted for the Rotation Shim controller instead of the default DWB controller. This choice was made because the UDOn robot has limited turning capabilities and a large turning radius, resulting in low movement efficiency and frequent control failures with the DWB controller. The Rotation Shim controller allows the UDOn robot to rotate in place to face the planned path direction before moving, significantly enhancing its movement efficiency.

During the navigation experiments between different floors, the UDOn's navigation logic is illustrated in Fig.4.8. When the destination is on a different floor, the navigation system first guides the robot to the location of the stairs. The UDOn robot then activates the stair-climbing behavior node to execute the stair-climbing procedure. After reaching the new floor, the navigation module is restarted. The system will load the 2D map of new floor, sets the starting position, and finally navigates to the destination on the new floor. Fig.4.9 illustrate the Rviz visualization graphs of the UDOn during experiments. Fig.4.9-(a) illustrate the navigation node startup and map loading. After setting the starting point and the target point, the navigation algorithm calculates an optimal path, represented by the green line in Fig.4.9-(b). Fig.4.9-(c) and (d) show the process of UDOn moving along the planned path and reaching the target point.

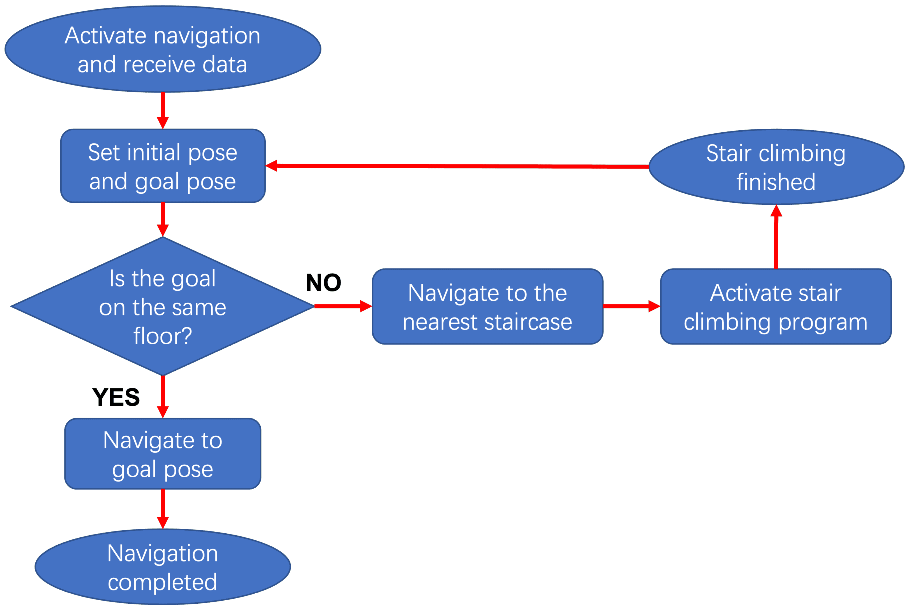

Figure 4.8: UDOn navigation logic

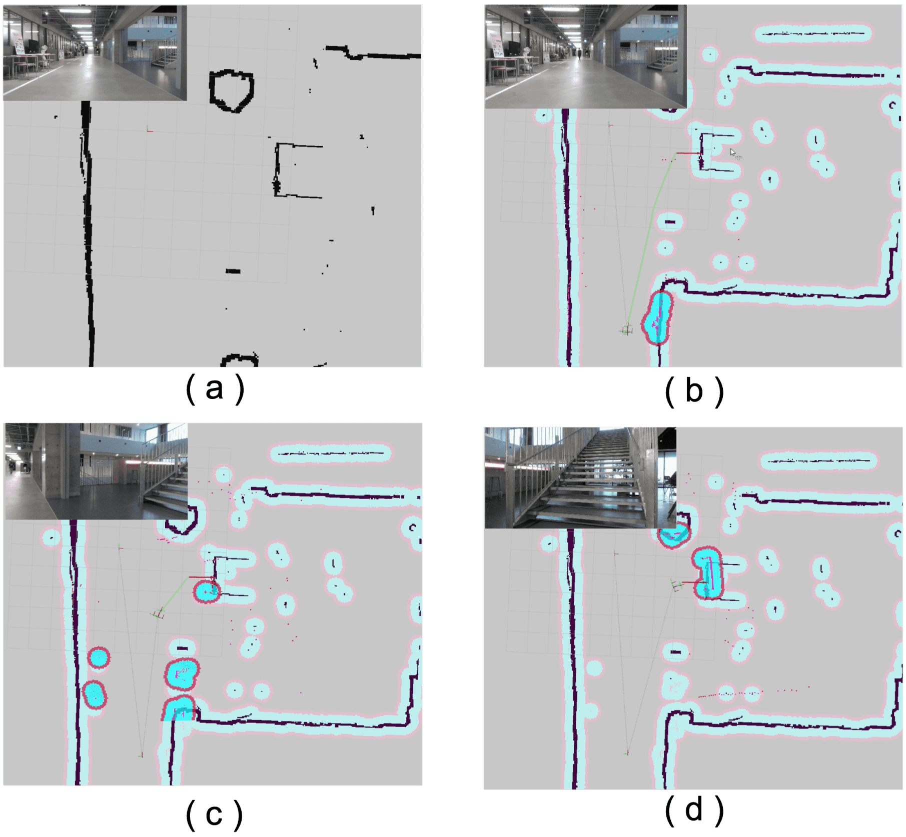

Figure 4.9: Rviz visualization graphs of the navigation.

Chapter 5

Conclusion

5.1 Review and Summarize

In this paper, we presented the implementation of an autonomous system for the UDOn (Ubiquitous Delivery On-demand) robot, a tracked mobile robot designed to navigate multi-floor environments using stairs and perform object grasping and transport tasks. We initially designed and developed a 3DoF robotic arm to assist the UDOn robot in performing simple cargo grasping tasks. By integrating an RGB-D camera and the YOLO object detection open-source framework, UDOn gained the capability to identify and grasp target objects in front of it. Although the current functionality of the robotic arm is relatively basic, the ability to recognize and grasp objects signifies that UDOn's capabilities and application scenarios have the potential for further expansion in the future.

For UDOn's autonomous mobility, there are two key aspects: the SLAM algorithm and the navigation algorithm. For SLAM, we used the industry-standard Cartographer framework combined with a 2D LiDAR, IMU, and servo encoder to achieve real-time localization and mapping. This setup successfully produced high-quality 2D grid maps of different floors. For navigation, we utilized the open-source Nav2 stack, which includes various algorithms for localization, path planning, and motion control, successfully enabling UDOn's navigation capabilities. Given UDOn's ability to climb stairs, we devised a method for autonomous movement and navigation between different floors. Ultimately, we integrated all of UDOn's functionalities within the ROS2 system environment and deployed them on the UDOn robot, conducting a series of experiments and tests in real-world settings.

Regarding the SLAM algorithm, our hardware combination of 2D LiDAR, IMU, and encoder with the Cartographer framework effectively handled real-time localization and mapping tasks in building corridors. Experiments conducted after deploying the Nav2 navigation stack showed that the UDOn robot can autonomously navigate indoor spaces, including climbing stairs and transporting objects between floors. These results validate the effectiveness of our system in real-world scenarios and highlight its potential applications in logistics, healthcare, and facility management.

5.2 Future Directions

For the future research directions of the UDOn robot, I propose the following points:

1. **Improvements in UDOn structure and servo motors to enhance movement speed**: Through a series of experiments in real environments, we observed that while the current UDOn robot possesses the ability to autonomously navigate and transport goods, its movement speed on level surfaces is significantly limited by its heavy body and the limited power provided by the servo motors, with a maximum speed of only about 0.15 m/s. Future research could focus on optimizing UDOn's structural framework and chassis design, or replacing the current motors with higher power drive motors to improve its movement speed.

2. **Utilization of visual SLAM system**: The current UDOn achieves effective real-time localization and mapping using a 2D LiDAR, IMU, and servo encoder. However, relying solely on 2D LiDAR for environmental perception has inherent limitations, as UDOn operates in a three-dimensional space where real-world scenarios are more complex than our experimental setups. Integrating visual data into UDOn's existing SLAM system will be a crucial research topic moving forward, providing richer environmental perception capabilities.

3. **Integration of all functions and behaviors into the Nav2 stack framework**: In this study, we used the Nav2 stack to achieve UDOn's navigation capabilities. However, other behaviors such as target object grasping and inter-floor movement using stairs were handled separately. As we deepen our understanding of the Nav2 Stack, particularly its strategy of managing different tasks using behavior trees, a key future research direction will be to integrate all of UDOn's functions and behaviors into this framework. This will aim to achieve fully automated execution of all behaviors.

References

[1] Carlos Campos et al. "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam". In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[2] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[3] Ross Girshick et al. "Region-based convolutional networks for accurate object detection and segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2015), pp. 142–158.

[4] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improved techniques for grid mapping with rao-blackwellized particle filters". In: *IEEE transactions on Robotics* 23.1 (2007), pp. 34–46.

[5] Wolfgang Hess et al. "Real-time loop closure in 2D LIDAR SLAM". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 1271–1278.

[6] Junji Hirasawa. "Improvement of the mobility on the step-field for a stair climbable robot with passive crawlers". In: *Journal of Robotics and Mechatronics* 32.4 (2020), pp. 780–788.

[7] Naoki Igo et al. "Robots climbing up and down a steep stairs and robots retrieving objects from high places". In: *Journal of Robotics and Mechatronics* 34.3 (2022), pp. 509–522.

[8] G. Jocher. *YOLOv5 by Ultralytics*. `https://github.com/ultralytics/yolov5`. Accessed: 2023-02-30. 2020.

[9] Stefan Kohlbrecher et al. "A flexible and scalable SLAM system with full 3D motion estimation". In: *2011 IEEE international symposium on safety, security, and rescue robotics*. IEEE. 2011, pp. 155–160.

[10] Wei Liu et al. "Ssd: Single shot multibox detector". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37.

[11] Steve Macenski, Matthew Booker, and Josh Wallace. "Open-Source, Cost-Aware Kinematically Feasible Planning for Mobile and Surface Robotics". In: *Arxiv* (2024).

[12] Steve Macenski and Ivona Jambrecic. "SLAM Toolbox: SLAM for the dynamic world". In: *Journal of Open Source Software* 6.61 (2021), p. 2783.

[13] Steve Macenski et al. “The marathon 2: A navigation system”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2718–2725.

[14] Jorge L Martínez et al. “Kinematic modelling of tracked vehicles by experimental identification”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 2. IEEE. 2004, pp. 1487–1492.

[15] Ben Mildenhall et al. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[16] S Ali A Moosavian and Arash Kalantari. “Experimental slip estimation for exact kinematics modeling and control of a tracked mobile robot”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 95–100.

[17] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[18] Yasuaki Orita, Kiyotsugu Takaba, and Takanori Fukao. “Human tracking of a crawler robot in climbing stairs”. In: *Journal of Robotics and Mechatronics* 33.6 (2021), pp. 1338–1348.

[19] Tong Qin, Peiliang Li, and Shaojie Shen. “Vins-mono: A robust and versatile monocular visual-inertial state estimator”. In: *IEEE transactions on robotics* 34.4 (2018), pp. 1004–1020.

[20] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[21] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[22] Antoni Rosinol, John J Leonard, and Luca Carlone. “Nerf-slam: Real-time dense monocular slam with neural radiance fields”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 3437–3444.

[23] Taewon Seo et al. “Stair-climbing robots: A review on mechanism, sensing, and performance evaluation”. In: *IEEE Access* 11 (2023), pp. 60539–60561.

[24] Tixiao Shan et al. “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping”. In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2020, pp. 5135–5142.

[25] Maxim Sokolov, Oleg Bulichev, and Ilya Afanasyev. “Analysis of ROS-based Visual and Lidar Odometry for a Teleoperated Crawler-type Robot in Indoor Environment.” In: *ICINCO (2)*. 2017, pp. 316–321.

[26] Yi Kiat Tee and Yi Chiew Han. “Lidar-based 2D SLAM for mobile robot in an indoor environment: A review”. In: *2021 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*. IEEE. 2021, pp. 1–7.

[27] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas". In: *Machine Learning and Knowledge Extraction* 5.4 (2023), pp. 1680–1716.

[28] Angelo Ugenti et al. "Kinematic modelling of a high mobility tracked robot". In: *The International Conference of IFToMM ITALY*. Springer. 2022, pp. 37–44.

[29] Tianmiao Wang et al. "Analysis and experimental kinematics of a skid-steering wheeled robot based on a laser scanner sensor". In: *Sensors* 15.5 (2015), pp. 9681–9702.

[30] Wei Xu and Fu Zhang. "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3317–3324.

[31] Shichao Yang and Sebastian Scherer. "Cubeslam: Monocular 3-d object slam". In: *IEEE Transactions on Robotics* 35.4 (2019), pp. 925–938.

[32] Ji Zhang, Sanjiv Singh, et al. "LOAM: Lidar odometry and mapping in real-time." In: *Robotics: Science and systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9.

[33] Lei Zhang, Rene Zapata, and Pascal Lepinay. "Self-adaptive Monte Carlo localization for mobile robots using range finders". In: *Robotica* 30.2 (2012), pp. 229–244.

[34] Xuexi Zhang et al. "2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots". In: *Journal of Advanced Transportation* 2020.1 (2020), p. 8867937.

[35] Qin Zou et al. "A comparative analysis of LiDAR SLAM-based indoor navigation for autonomous vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2021), pp. 6907–6921.

[36] 康益赫, 李周浩, et al. "室内宅配ロボットのためのアーム機構の開発". In: ロボティクス・メカトロニクス講演会講演概要集 *2023*. 一般社団法人 日本機械学会. 2023, 1A1–H24.

[37] 楊俊彦, 高橋邦光, 李周浩, et al. "階段昇降が可能なコンパクトクローラ型移動ロボットの機構". In: ロボティクス・メカトロニクス講演会講演概要集 *2021*. 一般社団法人 日本機械学会. 2021, 2A1–L09.

[38] 高橋邦光, 楊俊彦, 李周浩, et al. "複層建物におけるクローラ型宅配サービスロボットの自律移動". In: ロボティクス・メカトロニクス講演会講演概要集 *2021*. 一般社団法人 日本機械学会. 2021, 2A1–L08.