UNIVERSITÄT DUISBURG-ESSEN
FAKULTÄT FÜR WIRTSCHAFTSWISSENSCHAFTEN
INSTITUT FÜR INFORMATIK UND WIRTSCHAFTSINFORMATIK
LEHRSTUHL FÜR PERVASIVE COMPUTING

Masterarbeit

Exploiting Knowledge of Room Occupation for the Scheduling of Navigation Tasks of a Fleet of Robots in Office Environments

Xuanjiao Zhu
Matrikelnummer: 3038674

Networked Embedded Systems Group
Institut für Informatik und Wirtschaftsinformatik
Fakultät für Wirtschaftswissenschaften
Universität Duisburg-Essen

29. Oktober 2020

Erstprüfer: Prof. Dr. Pedro José Marrón
Zweitprüfer: Prof. Dr. Gregor Schiele
Zeitraum: 13.May 2020 - 31.October 2020

Contents

Abstract

Since the advancement of robotics in modern society, multi-robot systems are used in office environments to help people accomplish their goals. In a typical office environment, if at least one person is in the office, the office is considered occupied. This thesis proposed a multi-robot system for an office environment. In this system, room occupation becomes an important evaluation parameter for task scheduling. However, compared with an office environment, the robot's perspective is limited, and the field within the sensing range is relatively narrow.

The knowledge of room occupation can be established by the Internet of Things (IoT) technologies. I have considered adding fixed sensors to each room in an office environment so that the robots can gather room occupation information while performing tasks. As long as the robot enters the sensor range, they conduct a rapid exchange of room occupation information. The robot then sends acquired room occupation data to the centralized pool. The centralized pool is the global controller that receives information about the robots and the environment and make decisions base on the information. Moreover, the centralized pool can acquire room occupation information and use it as one evaluation parameter to schedule tasks, together with common evaluation parameters such as priority, battery consumption, etc. Besides, when a robot cannot perform its current task, it hands over tasks to other robots.

Simulation results demonstrate the efficiency and robustness of the multi-robot system. In the simulated office environment, the multi-system operated stably for days and autonomously performed thousands of navigation tasks.

Chapter 1

Introduction

Since the development of robot technology in modern society, the robot's reliability and efficiency increase rapidly. Therefore, robots have been used together as a multi-robot system, which has advantages over single robots [ERP20]. Multi-robot systems are widely used in office environments to help people accomplish their goals. In addition to interacting with people, robots also need to learn from the surrounding environment to schedule their activities effectively.

For a working robot, environmental information is indispensable for efficiently perform activities. There are two types of environmental information. The first type is information about its surroundings, such as furniture, doors, walls, and other robots. The second type is the room occupation, which means the probability of someone in the office. In other words, in a typical office environment, people have different working schedules. If at least one person is in the office, the office is considered occupied. In this research, the knowledge of room occupancy is applied to task scheduling by the robot system. Finding a way to gather room occupancy information is a goal of this thesis. The robot can be equipped with external sensors and tools such as an infrared sensor, Laser Distance Sensor (LDS), and Camera. These modules can detect obstacles. However, compared with an office environment, the robot's perspective is limited, and the field within the sensing range is relatively narrow. To be more precise, robots cannot provide heterogeneous information relevant to infer room occupation over long periods.

1.1 Motivation

This project's primary motivation is to improve the scheduling of tasks. Task scheduling determines when, where, and what order tasks need to be done by multiple robots. One of the most crucial research areas in multi-robot systems is the task scheduling problem. Task scheduling is an actual problem in many real-life applications, e.g., healthcare applications to assist elderly residents [BMR+17] inspection applications for industrial

plants [LK12]. A report about the delivery robot in the hospital shows that a multi-robot system saves 2.8 full-time equivalent employees, as the robot works two shifts and seven days per week [JL17].

It then becomes essential to improve task scheduling in office environments. Task scheduling can be improved in several aspects. The first aspect is to reduce the time required to complete a set of tasks. The second aspect is to minimize the energy consumption of the robots. The knowledge of room occupation can improve task scheduling in office environments. With knowledge of room occupation, robots can find people in the environment and avoid wasting time and energy in empty rooms [BMR+17]. However, it is almost impossible to sense all of the necessary information using only individual robots and sensors attached to it due to their limited perspective.

Therefore, it is necessary to adopt the Internet of Things (IoT) devices in office environments [PNK+15]. Although increasing amounts of research have been conducted in task scheduling for multi-robot systems [SM07], unfortunately, current research mainly focuses on multi-robot cooperation dynamic environments. Rarely research addresses applying the Internet of Things technologies in task scheduling because many people often think about the Internet of Things (IoT) and robotics technology as separate fields. One of the most critical research fields of IoT is the wireless sensor networks (WSN), which are easy to deploy and widely use in indoor environments [LS18].

1.2 Problem definition

Our problem concerns robots that must perform various tasks in office environments. The office environment should consist of multiple rooms. In this project, I assume no robot limitation in these rooms. As long as the door of the room is opened, any robot can enter it. The people can have their working schedules, which cause the room occupied and unoccupied regularly. I assume each room has one or multiple doors, and each door is attached to a sensor. I consider charging stations for robots because the robot energy level drops as it moves and rotates.

If a set of different navigation tasks randomly distributed in different rooms. The system should be able to schedule these tasks. The navigation task requires one robot to traverse a path in the office environment. Particularly, a navigation task can have one or multiple target positions. In other words, the robot can navigate to a destination or pass through several positions and finally reach the destination. An efficient centralized task scheduling algorithm should be implemented for the multi-robot system using room occupation knowledge.

Besides performing navigation tasks, robots should also gather room occupation information from the office environment. Robots should gather room occupation information while performing tasks. The system should be able to share the information among

sensors, robots, and the central pool. The centralized pool is the global controller that receives information about the robots and the environment and make decisions base on the information. Also, the robots with low-energy should go to a nearby charging station so that it has sufficient power to keep performing tasks. An example of office environment is shown in Figure 1.1.

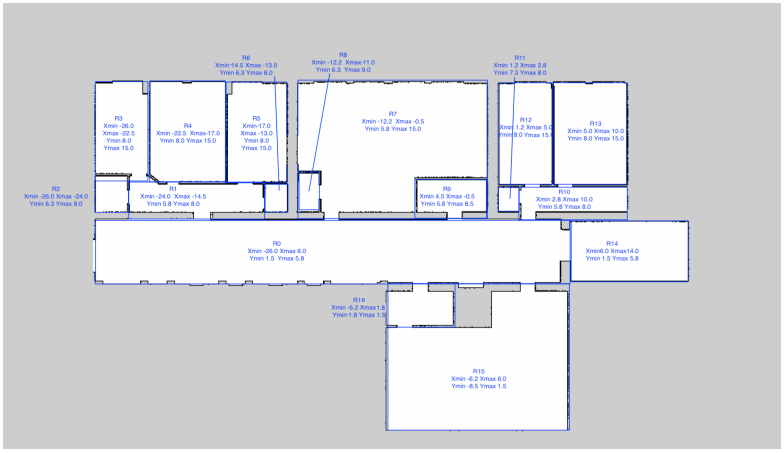

Figure 1.1: An example of office environment. There are 16 Rooms in this office environment.

1.3 Main Contributtions

I have designed a method for robots to gather room occupation information while performing tasks. As long as the robot enters the sensor range, they conduct a rapid exchange of room occupation information. The robot then sends acquired room occupation data to the centralized pool. The centralized pool is the global controller that receives information about the robots and the environment and make decisions base on the information. The centralized pool uses this information in two ways. It assigns robots the navigation tasks according to the information and can assign robots to explore the room lacking the information. Once the robot finished current tasks, it notifies the central pool that the task is succeeded and requests a task from the centralized pool.

I also have implemented a multi-robot task scheduling method in the centralized pool. The goal of multi-robot task scheduling is to decrease the time of completion and energy consumption. Moreover, the centralized pool can use room occupation information as one evaluation parameter to schedule tasks, together with common evaluation parameters such as priority, waiting time, energy consumption, etc. Task priority means the importance of tasks. The waiting time is the difference between the current time and

task start time. Energy consumption is the amount of energy expected to be reduced by performing a task.

I also paid attention to increasing the number of tasks complete. When a robot cannot perform its current task, it hands over tasks to other robots. To be more precise, sometimes, the robot can not perform the current task within a fixed deadline since the target position is temporarily unreachable (e.g., blocked by a closed-door). In this case, the blocked robot sends failure detail to the centralized pool and request another task. The failed task will then be reused for task scheduling.

In addition to the architecture I proposed herein, our system uses the Robot Operating System (ROS) platform to archive core autonomous robot function (navigation, localization, and mapping). Thanks to the ROS, I was able to develop a flexible and efficient system. Adding or removing modules, including robots, sensors, or charging stations, is simple and straightforward.

1.4 Thesis Structure

The next chapter briefly introduces background information. The essential concept about the Robot Operating System (ROS) and ROS Tools are introduced, along with a discussion of previous task scheduling methods.

Chapter 3 formally presents the architecture of the multi-robot system and gives information about tasks. The approaches to schedule tasks are also introduced.

Chapter 4 focuses on the implementation of each component in the multi-robot system. It presents the way robots gather room occupation information while performing tasks.

Chapter 5 introduces the approach to the evaluation, along with the simulated office environment. It also demonstrates the simulation results acquired using the approach. Besides, a quantitative experiment analysis was performed.

Finally, chapter 6 discussed the results of the performed work and future work. The impact of limitations of simulation on the experimental results is also discussed.

Chapter 2

Materials and Methods

This chapter introduced the background and state of the art on multi-robot task scheduling. Chapter 2.1 introduces important concepts of ROS. Chapter 2.2 introduces the 3D modeling of robot and environment in Gazebo. Chapter 2.3 introduces robot navigation. Chapter 2.4 introduces previous task scheduling methods. Chapter 2.5 introduces exist implementation of cost function.

2.1 Important Concepts of ROS

ROS is a highly flexible open-source operating system for the robot. It contains various development tools and libraries for developing robot software programs[HC17]. It aims to simplify the difficulty and complexity of the process of creating software programs across robot platforms. In this project, ROS is used to archive core autonomous robot functions, including navigation, localization, and mapping.

2.1.1 Node

A ROS node is one executable program. In this project, some existing nodes are used. For instance, the " move base" node provides a ROS interface for configuring, running, and interacting with the robot's navigation stack. There are some nodes created for this project, and each node is created for different purposes. For example, one "Robot controller" node controls one robot.

2.1.2 Communication Infrastructure

ROS has a built-in and well-tested messaging system. There are different methods of exchanging messages. ROS topic is a unidirectional anonymous communication. It is used when exchange data continuously. The subscriber receives messages of publisher

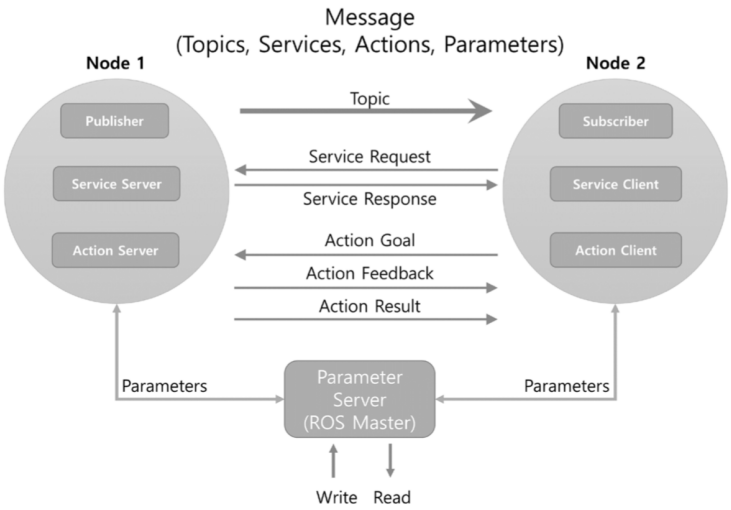

Figure 2.1: ROS Message Communication [HC17]

node only when both of them registered the same topic name. ROS service is bidirectional synchronous communication. The service client requests a service, and the service server responds to the request. The ROS action is a bidirectional asynchronous communication. It is mainly used in two cases. The first case is when it is difficult to use the service due to long response times after the request. The second case is when an intermediate feedback value is needed. The diagram of message communication is shown in Figure 2.1. The messaging system has been applied to this project. For instance, the sensor simulation scenario (Chapter **??**) use ROS Topic method. The communication protocols (Chapter 4.1) are designed based on the ROS Service method and the ROS Action method.

2.1.3 ROS Tools

ROS's core functionality is enhanced by a variety of tools and packages:

- **Rviz.** Rviz[RVI] is the 3D visualization tool of ROS. It can visualize information like the distance from a Laser Distance Sensor (LDS) to an obstacle, image value obtained from a camera, Point Cloud Data (PCD) of the 3D distance sensor such as RealSense, Kinect, or Xtion. As is shown in Figure 2.2, multiple robot model and its path and laser data can be displayed.

- **rqt.** rqt is an integration of more than 30 Qt-based ROS GUI development tool. It has plugins such as "rqt_tf_tree", "rqt_plot" and "rqt_graph"

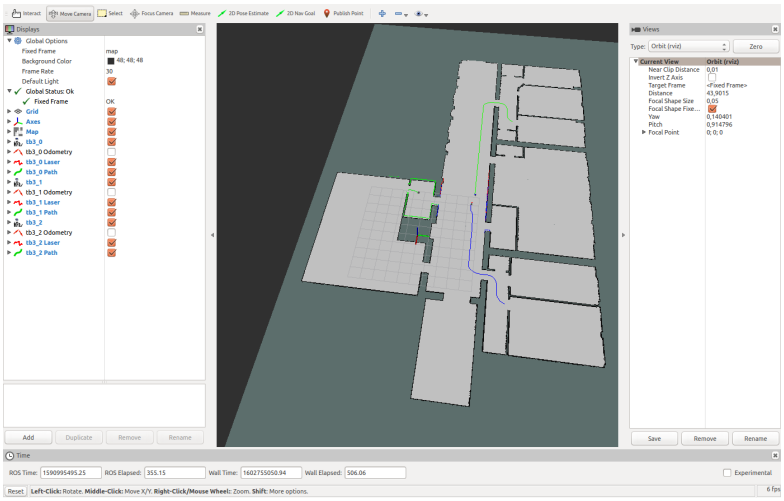

Figure 2.2: Rviz Example: Navigation using TurtleBot 3 and Laser Distance Sensor (LDS)

- **rqt_tf_tree.** rqt_tf_tree is a type of rqt. It is a tool for visualizing the tree of topics being broadcast over ROS. Figure 2.3 presents the relative coordinate transformation (tf) of multi-robot system. If the poses of Laser Distance Sensor (LDS) are considered as the poses of the robots, the pose information of each robot is connected in the order of odom → base_footprint → base_link → base_scan.

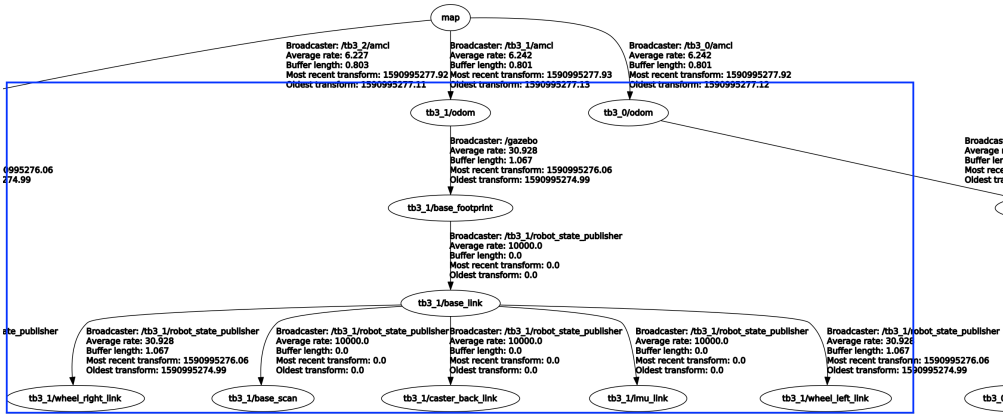

Figure 2.3: One Part of rqt_tf_tree of the multi-robot system. Each robot has a subtree (like the blue box) and is connected to the map server node

- **rqt_graph.** rqt_graph is a type of rqt. It is a graphical tool that presents the status of nodes and topics. Figure 2.4 presents relation of nodes in multi-robot system.

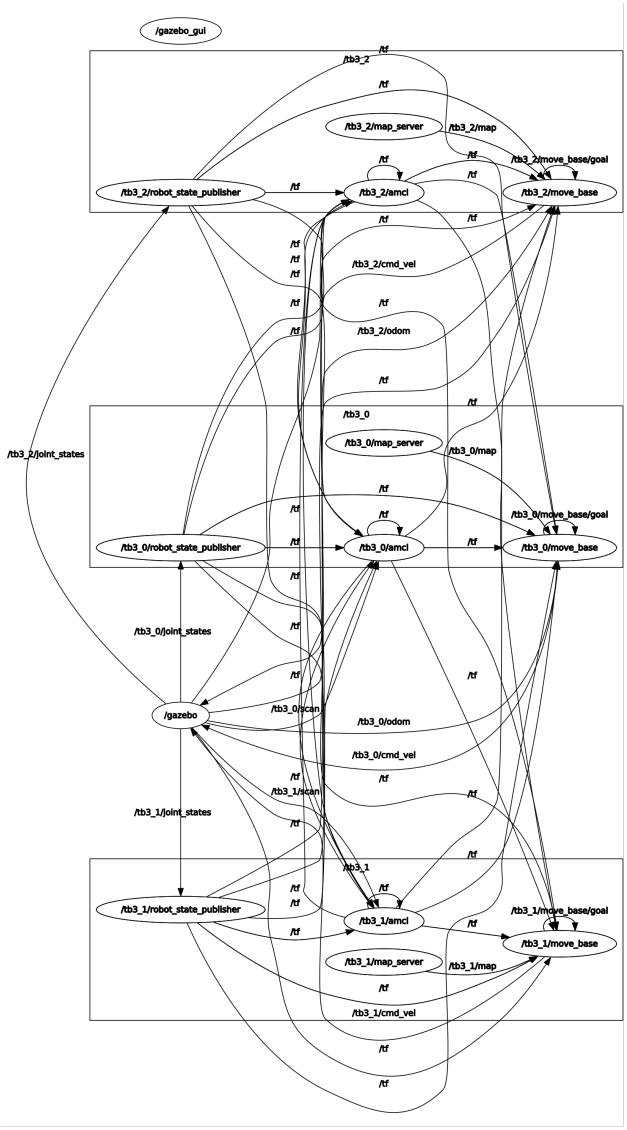

Figure 2.4: rqt_graph of multi-robot system

- **Gazebo.** Gazebo [GZ] is the 3D simulation tool integrated with ROS. The details of Gezebo are introduced in Chapter 2.2.

2.2 TurtleBot3 Simulation Using Gazebo

2.2.1 Gazebo Simulator

Robot simulation is essential for robotics research because it can pre-estimate algorithms' performance before applying it to a real robot [ASM15]. The simulator used in this project is Gazebo. Gazebo[GZ] is a 3D robot simulator software based on physics simulation. It is used to simulate the movement of one or multiple robots in complex indoor and outdoor environments. Using Gazebo, users can create a new 3D model with geometrical primitive or import existing simulated robots and environments.

2.2.2 TurtleBot3 Robot

The robot model used in this project is TurtleBot3 Burger. TurtleBot3 Burger is a small programmable mobile robot based on ROS. It is widely used in robotics research and education. As is shown in Figure 2.5, the basic components are actuators, an SBC (Single-Board Computer) for operating ROS, an LDS sensor for SLAM (Chapter 2.3) and navigation, restructure mechanism, an OpenCR embedded board used as a sub-controller, sprocket wheels that can be used with tire and caterpillar, and a three-cell lithium-poly battery. The simulated robot (Figure 2.6) has a similar outfit. Besides, Gazebo simulates the robot locomotion and sensor measurements used for localization and navigation and exports the simulation results to ROS.

2.2.3 3D Modeling of the Indoor Environment

In this project, we selected a model the same as the department's floor as a trial 3D model (Figure 2.7). This model is a typical office environment that contains a corridor along the central x-axis and 16 rooms located around the corridor.

2.3 Robot Navigation and Virtual SLAM

There are essential technologies to realize autonomous robot navigation:

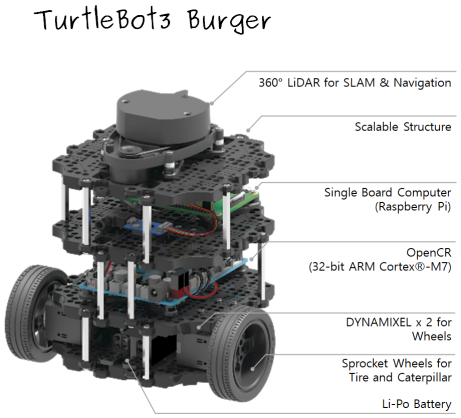

Figure 2.5: Robot Hardware Configuration [HC17]

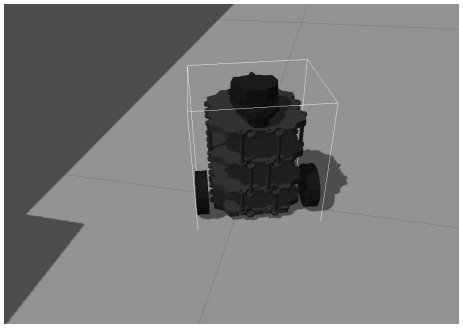

Figure 2.6: Robot 3D Modeling in Gazebo

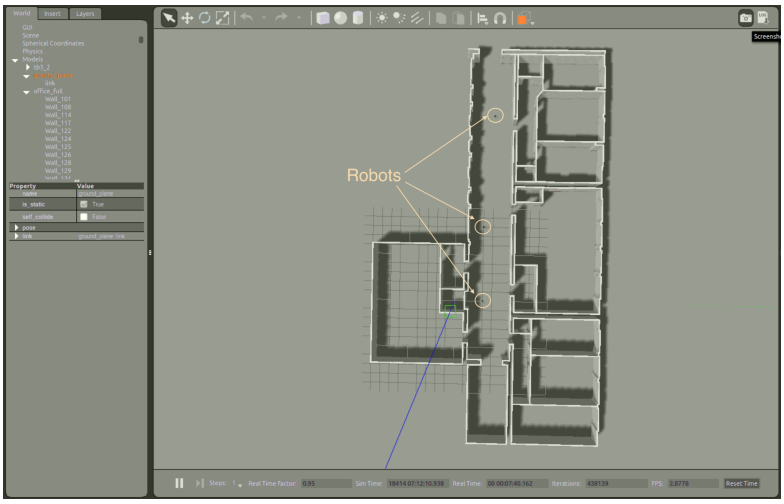

Figure 2.7: 3D Modeling of Indoor Environment in Gazebo

1. **Having a map of the given environment.** In this project, SLAM(Simultaneous Localization And Mapping) is used to create a map of the given environment. Using SLAM, the robot explores the unknown spaces, detects its surrounding areas, estimates its current location, and creates a map. The steps of executing virtual SLAM with TurtleBot3 are shown on the website [T3S]. Once the robot finishes exploring the indoor environment, an occupancy grid map (OCM) is generated (Figure 2.8).

2. **Measuring or estimating the pose of the robot.** The Pose consists of position and orientation. The dead reckoning [DEA] is the most popular indoor pose estimation method for the robot. The amount of movement of the robot is measured with the rotation of the wheel. However, the error between the calculated distance with wheel rotation and the actual travel distance increases over time. Therefore, the inertial measurement unit (IMU) sensor [Seo17] is used to measure tri-axis angular velocities and tri-axis acceleration to estimate the robot's position. This inertial data can compensate for the error of position and orientation between the calculated value and the actual value.

3. **Avoiding obstacles such as walls and furniture.** The laser-based distance sensor on the robot is widely used to determine whether there are obstacles, including walls, furniture, and other robots. The common laser-based distance sensor includes LDS (Laser Distance Sensor), LDF (Laser Doppler Flowmetry), and LiDAR (Light Detection And Ranging) and ultrasonic sensors and infrared distance sensors. The TurtleBot3 equips 360 Laser Distance Sensor LDS-01. The visualization of Laser data in Rviz is shown in Figure 2.2.

4. **Finding the optimal route calculation and driving.** It is important to find the optimized route to the destination. Many algorithms perform path searching and planning, such as A* algorithm[ASE], potential field[POT], particle filte[PAR], and RRT (Rapidly-exploring Random Tree)[RRT].

2.4 Task Scheduling Methods

So far, the background information, such as tools and important concepts of ROS, are introduced. This Chapter discusses some popular methods of task scheduling. Those task scheduling methods can be divided into centralized methods and distributed methods.

2.4.1 Centralized Method vs. Distributed Method

In the case of centralized methods, a centralized schedule collects all task requests and uses resource utilization information to schedule tasks. The centralized method is easy

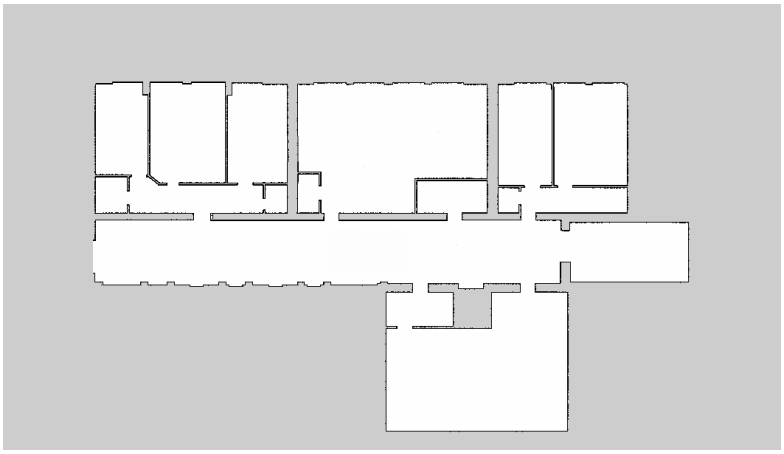

Figure 2.8: Occupancy Grid Map (OGM). White presents the free area in which robots can move, black presents the occupied area in which robots can not move, and gray is the unknown area.

to implement and faster to repair in case of failure. However, the robots' autonomy in a pure centralized method is limited because all robots only execute commands from the centralized scheduler and not determine what tasks to do [NMMG17]. Also, since the centralized scheduler must compute all resources and tasks, the centralized system is difficult to scale to a large-size network [CSMV09]. Chapter 2.4.2 introduces a centralized constraint programming method. Chapter 2.4.3 introduces a centralized method based on A* and Genetic Algorithm.

In the case of distributed methods, multiple distributed schedulers keep tracking the resource availability and use this information to perform task scheduling[CSMV09]. This method distributes the associated computation overhead. As a result, it eliminates the bottleneck caused by the centralized scheduler and improves networks' scalability and reliability. The challenge in distributed methods is the coordination of distributed schedulers and the required control plane overhead [CSMV09]. Chapter 2.4.4 introduces a distributed auction method. Chapter 2.4.5 introduces a communication-efficient distributed method.

2.4.2 Centralized Constraint Programming Method

Booth has proposed a multi-robot system that supports elderly residents in a retirement home setting in [BMR+17]. in the morning, the robots search for elderly residents in the retirement home, eliciting their availability and preferences for activities. The centralized scheduler then uses the constraint programming method to allocate these assistive activities over the day. Those problem-specific constraints include robot energy

consumption, activity priority, robot-user activity synchronization, user location, and user calendars that identify their available and busy intervals. Once this information is attained, the system allocates and schedules robots for the day before executing the plan.

2.4.3 Centralized Method based on A* and Genetic Algorithm

Chun has proposed a centralized task scheduling and path-planning method based on A* and Genetic Algorithm [LK12]. In this research work, the Genetic Algorithm is responsible for task scheduling to find the optimal solution for industrial plant inspection problems. A* Algorithm is used to calculate the traveling cost and path-planning of a robot moving from one position to another. It can consider more detailed path conditions such as obstacles, furniture, and terrain conditions. The traveling cost is one evaluation parameter of the fitness function in the Genetic Algorithm, but it is referred to as the completion time in this research. To be more precise, it is the period between the first robot starting its tasks and the last robot finishing its tasks. The goal of the task scheduling method is to find an ordered set of tasks for robots to minimize the completion time while decreasing the total power consumption. Two Greedy algorithms are proposed for task assignment. The first one is to find one task for each robot that provides for the minimum completion time in each step to minimize the total completion time. The second one is finding only one robot-task pair that takes the minimum time in each step to decrease the total power consumption.

2.4.4 Distributed Auction Method

When the system performs a long-term task allocation process, the communication link between the customer agent and robots may be disconnected. These problems may cause a conflict or failed assignment. Distributed methods are more suitable in this case to distribute the computation to individual agents [NMMG17]. Dong-Hyun Lee has proposed a resource-oriented, distributed auction algorithm [LZK15]. The customer agents and robots with limited communication ranges construct an ad-hoc network tree. The customer agent becomes auctioneer and broadcasts an auction call to the task. The robots become bidders and submit their bid values to the customer agent. The bid values consider local information such as robot task queue, robot's resource levels, and estimated travel distance and time for multiple paths. Since each path consists of different charging stations, the robot's resource levels after completing a task and estimated travel time depend on the path. After receiving all bid values, the agent assigns the task to the robot with the lowest bid value. This scenarios not only avoids unexpected battery drain while robot processing task, but also let robots maintain high energy.

2.4.5 Distributed Method with Global Unit

Kashyap Shah proposed a communication-efficient distributed dynamic task scheduling system with a shared global unit [SM07]. Each robot can make its own decision by communicating with other robots and checking and updating the current task status in the global unit. Besides, this method considers two unexpected situations. Firstly, some robots may be unable to handle their current task because the task environment has changed. In this case, the defective robot checks the global unit and sends a help message to robots that can handle the task. Secondly, some robot may fail in a dynamic environment. This robot failure will be detected by tracking the communication signal, and its status in the global unit will be updated as failed to make sure no more tasks will be allocated to this robot. If the failed robot is running a task, its current task will be reassigned to another robot.

2.5 Cost Function

One of the most important steps when designing a multi-robot task scheduling algorithm is calculating the tasks' costs. Jia summarizes several physical quantities used in the algorithm's cost in [JM13]. In their study, it can be concluded that the most commonly used decision variables are estimated travel distance and time, as proposed in [LZK15]. Other kinds of decision variables involved are the number of traversals and energy consumed. Besides, Korsah proposed a comprehensive taxonomy of multi-robot task scheduling problems that explicitly consider the issues of interrelated utilities and constraints. In this taxonomy, tasks are distinguished by decomposability and multi-agent-allocatability [KSD13]. In the case of Chun's method [LK12], the cost is defined as completion time. A completion time is calculated for the robot-tasks pairs. The completion time is the time span of the first robot starting its tasks and the last robot finishing its tasks.

Chapter 3

Approach

3.1 Architecture Design

The architecture of the system consists of several parts: centralized pool, robot controller, navigation stack, charging station, and system environment(Figure 3.1).

- **Centralized Pool.** The centralized pool is the global controller that receives information about the robots and the environment and make decisions base on the information. To be more precise, the centralized pool uses this information in two ways. It schedules the robot's navigation tasks according to the information and can also assign robots to explore the room lacking information. It consists of several modules: multi-robot task scheduling module, map information, database, execution, and monitoring. The database stores dynamic indoor environment information such as room occupation data. The map information modules contain the static map information(Figure 4.2). The execution and monitoring module interacts with robots. The multi-robot task scheduling module selects tasks in the database based on the robot and environment information and then give the task to the execution and monitoring module.

- **Robot Controller.** A robot controller contains several modules: local task queue, execution, and robot activity. The local task queue stores tasks that the robot needs to complete sequentially. The execution module receives commands from the centralized pool and decides when and which task the robot should run. The robot activity module performs tasks in the local task queue when it receives the execution module's decision. The robot activity module also interacts with the office environment and ROS navigation stack.

- **Navigation stack.** The move_base node provides a ROS interface for configuring, running, and interacting with the navigation stack on a robot. It makes the robot move to desired positions using the navigation stack. Its advantages include optionally performing recovery behaviors when the robot is stuck:

Obstacles outside the area specified by the user on the map will be removed. If possible, the robot will rotate to clear the space. If this fails, the robot will clear its map more aggressively, removing all obstacles outside the rectangular area so that the robot can rotate in place within the rectangle. If all this fails, the robot will consider its goal unreachable and print the abort details to ROS[MOV].

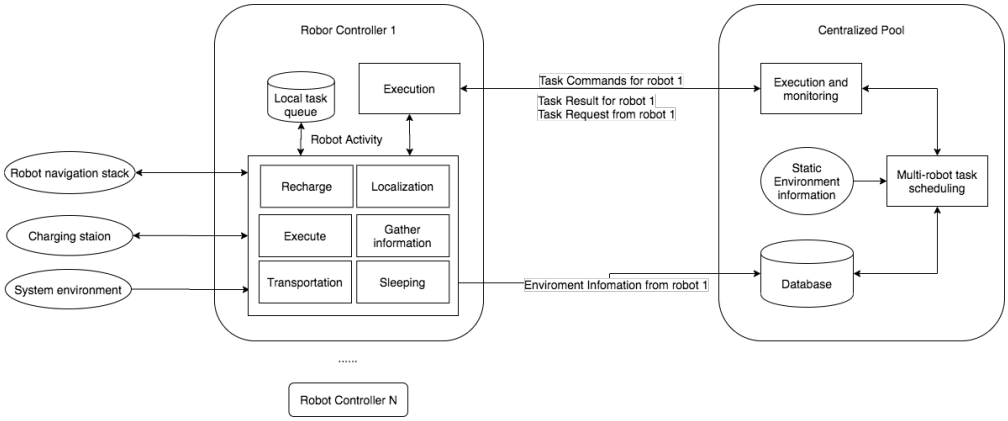

Figure 3.1: Multi-robot Task Scheduling Architecture.

3.2 Gather Information Approach

This project aims to schedule tasks for robots based on the room occupation information gathered about room occupation. Room occupation means the probability of someone in the office. In other words, in a typical office environment, people have different working schedules. If at least one person is in the office, the office is considered occupied. The first is to equip the robot with sensors and tools. For example, robotic tools such as Laser Distance Sensor (LDS) Camera and infrared sensor module can detect obstacles include doors. However, compared with the entire office environment, the robot's perspective is limited, and the field within the sensing range is relatively narrow. Therefore, the use of external environment information sources is essential to bridge the local knowledge gap. The second is to establish a distributed IoT network. But gateways connect sensors, and gateways need to connect to the Internet. They consume much energy. The third method is to install wireless sensors on the doors. The wireless communications include Bluetooth Low Energy (BLE) [BLE], ZigBee [Zig], ANT [ANT] , etc. This experiment uses the third method. A wireless module called SensorTag was used because it has a long battery life and has multiple sensors. The acceleration sensor can be used to infer the opening and closing of the door. An open door means someone in the room, and the room is occupied. A closed-door means that there is no one in the room and the room is not occupied.

Robots interact with sensors while navigating in the office environment and report this environment information to the centralized pool. The centralized pool is the global controller that receives room occupation information about the robots and the environment and make decisions base on the information. The way centralized pool storing and making decisions base on the information is discussed in section 4.3.

3.3 Task Composition and Decomposition Approach

3.3.1 Task Specification

The overall system goal is to gather room occupation information for a long time continuously, and the centralized pool assigns tasks for robots based on the robot and environmental information. The centralized pool is the global controller that receives information about the robots and the environment and make decisions base on the information 3.1.

Task information consists of the following parts: The Task ID is a unique task identification. The task name is a description of the main activity. Three task names are defined: "gather information task" asks a robot to gathers environment information from sensors, "navigation task" asks a robot to move to a point, and "charging task" asks the robot to refill its battery at charging stations. The start time refers to when the robot should move towards the target. A starting time is given when the task is created. This time can be a time in the future or empty (no time limit). The finish time refers to when the robot finishes the task. Targets include doors, points, and charging stations. Target ID is unique target identification. Robot ID is a unique robot identification. Task Priority describes the importance of the task. Once a task failed, its priority will be increased by one and reused in task scheduling. If this task has already the highest priority, the task fails. Task status are "Created", "Succeeded", "Failed", "To rerun", "Error". The difference between task status is discussed in Figure 4.4. Task dependency means the required last task. If task B has a dependency of task A, task A needs to be preceded by tasks B. Those dependent tasks should be composed in the centralized pool. The Task description is one sentence. The description of a succeeded task is "succeeded". The description of a failed task is its failure reason. A example of task specifications are stored in "task table" in database (Table 4.7).

3.3.2 Task Composition and Decomposition

Tasks can be distinguished into "simple tasks" and "Complex tasks". "Simple tasks" comprises a single target position that can be performed by a single robot. A "Complex tasks" can be broken up or decomposed into multiple "small tasks". Those sub-tasks of

a complex task need to be performed by the same robot. A "complex task" can have only one sub-tasks. In this project, the centralized poll not only can create "simple tasks" according to task specifications (Table 4.7) but also can analyze the dependencies of "simple tasks" and form a dependency chain to compose "complex tasks". The robot (robot controller) can decompose a "Complex task" to "simple tasks" and execute "small tasks" according to their dependencies.

3.4 Multi-robot Task Scheduling Approach

The multi-robot task scheduling module in the architecture should perform multi-robot task scheduling. The implementation of task scheduling is shown in Chapter 4.3. There are some general rules for multi-robot task scheduling.

1. When the robot's energy below 10%, the centralized pool's task scheduling module (Chapter 3.1) creates and sends a "charging task" to the robot based on rule discribed in in Chapter 3.4.3.
2. When the robot's energy above 10% and there are "navigation tasks" in the database, the task module composes simple tasks into complex tasks and selects a complex task for the robot according to the rule shown in Chapter 3.4.1.
3. When the robot's energy above 10% and there is no "navigation task" or the cost of all navigation tasks is higher than the threshold, create a "gather information task" for the robot according to the rule discussed in Chapter 3.4.2.

3.4.1 Select Navigation Task

When one of the "complex navigation tasks" should be selected for robot, In order to select an "navigation task", the decision variables are considered.

Decision variables used to select navigation tasks.

- **Task Priority.** The priority is discussed Chapter 4.2.
- **Product of Door Open Possibility.** The product of open possibilities of doors on trajectory: All doors that the robot will pass through when moving from its location to the target point. An example of "measurement result" table is shown in Table 4.8, an example of "open probability" table is shown in Table 4.9.
- **Waiting Time.** The waiting time is the difference between the current simulation time and start time of the first task to be executed. $T_{waiting} = T_{first\_task} - T_{now}$

- **Battery Consumption.** The Battery Consumption is related to robot trajectory. For a Large "navigation task" that contains n simple task, Equation 3.2 can be used to calculate battery consumption. The centralized pool will send the task with the lowest cost to this robot.

Equation 3.1 are used to calculate the cost. The "complex navigation tasks" with the lowest cost will be selected.

W: Weight

n: Number of doors

$$\text{Cost}_{\text{Large navigation task}} = \frac{W_{\text{battery}} \times \text{Battery consumtion}}{n} + W_{\text{waiting}} \times \text{waiting time} + W_{\text{probability}} \times \prod_{i=1}^{n} \text{Door open probability} + W_{\text{priority}} \times \text{Priority} \tag{3.1}$$

B:Battery consumption

W: Weight

m: Number of waypoint

n: Number of simple task

$$\begin{aligned} B_{\text{complex task}} &= \sum_{\text{task}_1}^{\text{task}_n} B_{\text{trajectory}} \\ &= \sum_{t=\text{task}_1}^{\text{task}_n} \sum_{\text{waypoint}_1}^{\text{waypoint}_m} [W_{\text{position}} \times \text{position variation} + W_{\text{angle}} \times \text{angle variation}] \\ &= \sum_{t=\text{task}_1}^{\text{task}_n} \sum_{p=\text{waypoint}_1}^{\text{waypoint}_m} [W_{\text{position}} \times \sqrt{(x_p - x_{p-1})^2 + (y_p - y_{p-1})^2} \\ &+ W_{\text{angle}} \times 2 \times \arccos(w_p)] \end{aligned} \tag{3.2}$$

3.4.2 Create Gather Information Task

Robot can perform "gather environment information task" to gather more measurement results and furthermore improve the accuracy of "open possibilities" table. To create a "gather environment information task", Equation 3.3 and following decision variables

are used to calculate the costs of doors. A "gather environment information task" to the door with the lowest cost will be created.

Decision variables used to create a gather information task

- **Door Last Update Time.** The latest timestamp when the door is measured.
- **Product of Door Open Possibility.** The product of open possibilities of doors on trajectory: All doors that the robot will pass through when moving from its location to the front of the target door.
- **Battery Consumption.** The battery consumption is related to the trajectory from the robot to the front of the door. Equation 3.2 can be used to calculate battery consumption.

W: Weight
n: Number of doors on trajectory

$$\begin{aligned}\text{Cost}_{\text{door}} = &\frac{W_{\text{battery}} \times \text{Battery consumtion}}{n} + W_{\text{time}} \times (T_{\text{last update}} - T_{\text{now}}) \\ &+ W_{\text{probability}} \times \prod_{i=1}^{n} \text{Door open probability}\end{aligned} \tag{3.3}$$

3.4.3 Create Charging Task

Once a robot sends task requests to the centralized pool, the centralized pool should determine whether this robot needs charging. If yes, it should create a "charging task" for the robot. To create a "charging task", Equation 3.3 and following decision variables are used to calculate the costs of charging station. A "charging task" to the charging station with the lowest cost will be created.

Decision variables used to create a charging task.

- **Remain Time.** It describes how long will a charging station be free.
- **Battery Consumption.** Similar to "navigation task" scheduling, the battery consumption is related to the trajectory from robot to the charging station. Equation 3.2 can be used to calculate battery consumption.

W: Weight

$$\text{Cost}_{\text{charging station}} = \frac{W_{\text{battery}} \times \text{battery consumtion}}{n} + W_{\text{time}} \times T_{\text{remain}} \tag{3.4}$$

Chapter 4

Implementation

This chapter introduces the implementation of multi-robot system. Chapter 4.1 presents how the communication among fixed sensors, charging stations, robots and the centralized pool. Chapter 4.2 introduces different tables in database. Chapter 4.3 describes the work flow of each componengts in the system

4.1 Communication Protocols

Centralized pool, robots, charging stations and sensors need to share information with each other. To be more precise, the robot needs to exchange information with sensors while performing navigation tasks and find sensors when there is no navigation task. The scheduling of navigation tasks is implemented in a component called the central pool. The centralized pool is the global controller in the multi-robot system. It can receive and learn room occupation from the robot. Also, it can designate the robot to explore a room with too little information.

To improve the communication efficiency, communication protocols are designed.

4.1.1 Message about Measurement

When a robot passes by a door, it should receive messages from a sensor. In this project, we use a ROS node "sensor simulator" to simulate door sensors (Chapter **??**) to publish instant measurement result (Table 4.1).

These Communication protocols save unnecessary communication cost by avoiding keep tracking the current position, availability and states of all robots (Figure 4.1).

Door ID	Position	Timestamp	Measurement Result
1	(-18.5,5.2)	2020-06-01 9:00:02	Door opened

Table 4.1: Measurement Message Format and Example

4.1.2 Message about Task

There are some basic requirements for communication between robot and centralized pool: firstly, robot should initiate the communication once task queue becomes empty. Secondly, robot should forward sensor data to centralized pool immediately after interacting with snesors. Four types of message are defined: (1)Task request message(Table4.2); (2) Task goal messages(Table 4.3); (3) Task feedback message (Table 4.4); (4) Task result message (Table 4.5).

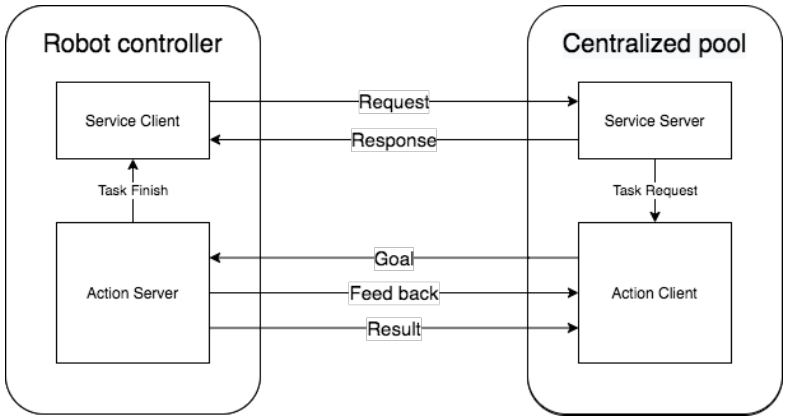

Figure 4.1: Communication between Robot and Centralized Pool

Battery Level	Position	Robot ID
93	(2,4)	1

Table 4.2: Request Message Format and Example

4.1.3 Message about Charging

When a robot arrives charging station's position, it sends a message to Charging station(Figure 4.6). The details of charging station is discussed in Chapter 4.3.3.

Task ID-[]	Task type	Target ID	Goal[]
1	Gather Environment Info	9	(-1.5,5.2) 2020-06-01 9:00:00
[3,4]	Execute task	21, 22	(-24.0,12.0), 2020-06-01 9:02:00 (-21.0,12.0) 2020-06-01 9:02:00
5	Charging	17	(0.0,5.0), 2020-06-01 9:04:00

Table 4.3: Action Goal Message Format and Example

Robot ID	Door ID	Measurement time	Measurement result
1	3	2020-06-01 9:00:03	Door open
1	4	2020-06-01 9:00:13	Door open
1	5	2020-06-01 9:00:23	Door open

Table 4.4: Action Feedback Message Format and Example

4.2 Database

The centralized pool keep environment information in database to make decisions. The structure of database is shown in Figure 4.2.

Table tasks

- **Column Task ID.** A unique task identification.
- **Column Task Name.** Tasks names are “gather enviroment information task ”, “navigation tasks” or “charging task”.
- **Column Start Time.** The start time refers to when the robot should move towards the target. A starting time is given when the task is created. This time can be a time in the future or empty (no time limit).
- **Column Finish Time.** The default value is empty. When the centralized pool receives task result, this column will be updated to the time when the centralized pool received the result.
- **Column Target ID.** Targets include doors, points and charging stations. When a robot run a “gather environment information task”, it moves to the front of a door and interact with a sensor in the door position without entering the door.

Task ID	Task type	Result
1	Gather Environment Info	Success

Table 4.5: Action Result Message Format and Example

Robot ID	Battery Level
1	93

Table 4.6: Message to Charging Station

When robot run an "navigation tasks", the robot moves to a given point ether in corridor or in the room. When robot runs a "charging task", it moves to a charging station and interact with this charging station.

- **Column Robot ID.** A unique robot identification.
- **Column Priority.** Task priorities allow user to easily prioritize tasks to clearly plan what to do next. The "charging tasks" are given the highest priority of 5. The "gather environment information tasks" are given the lowest priority of 1. The "navigation taskss" has priority between 2-4 in the "created" status. Once this task failed once, its priority will be increased by 1 until it exceeds the maximum and is marked as "Failed" (Figure 4.4).
- **Column Task Status.** Task status are "Created", "Succeeded", "Failed", "To rerun", "Error". The difference between task status is discussed in Figure 4.4
- **Column Dependency.** If task B has a dependency of task A, task A needs to be preceded by tasks B. Those dependent tasks should be composed in the centralized pool.
- **Column Description.** The description of a succeeded task is "succeeded". The description of a failed task is its failure reason.

Task ID	Task Type	Start Time	Target ID	Robot ID	Priority	Status	dependency	Finish Time	Description
1	Charging task	2020-06-01 9:00:00	18	1	5	RanToCompletion	0	2020-06-01 9:00:20	Succeeded
2	Execute task	2020-06-01 9:00:50	22	2	2	RanToCompletion	0	2020-06-01 9:01:20	Succeeded
3	Gather environment information task	2020-06-01 9:02:00	2	2	1	Running	0	2020-06-01 9:02:40	Succeeded

Table 4.7: Task Table in Database

Table measurements

- **Column Door ID.** Unique identification of the door.
- **Column Door Status.** Value 0 represent door closed. Value 1 represent door opened.
- **Column Date Time.** Measuring time.

Door ID	Door Status	Date Time
9	1	2020-06-01 09:05:39
1	0	2020-06-01 09:05:49
7	1	2020-06-01 09:05:49
9	1	2020-06-01 09:05:49
1	1	2020-06-01 09:05:59
7	1	2020-06-01 09:05:59
9	1	2020-06-01 09:05:59
7	1	2020-06-01 09:06:09
16	0	2020-06-01 09:06:29
7	1	2020-06-01 09:06:39
16	1	2020-06-01 09:06:39
9	1	2020-06-01 09:06:49
8	0	2020-06-01 09:06:59
...	...	...

Table 4.8: Table measurements

Table door open possibilities

- **Column Door ID.** Unique identification of the door.
- **Column Day of Week** a weekday.
- **Column Start Time and End Time** a time slot between start time and end time.
- **Column Initialized Open Possibility** Predefined value to used to simulate door sensors (Chapter **??**).
- **Column Open Possibility Statistic** Statistics of measurement result in the weekday and time slot.

Door ID	Day Of Week	Start Time	End Time	Initialized Open Possibility	Open Possibility Statistic
1	2	9:00:00	9:59:59	0.90	0.85
1	2	10:00:00	10:59:59	0.90	0.92
1	2	11:00:00	11:59:59	0.10	0.05
...	...	...	...	...	...

Table 4.9: Door Open Possibility.

Table doors

- **Column Door ID** Unique identification of door.
- **Column Last Update** The timestamp of last measurement result on the door.
- **Column Is Used** Value 1 represent at least one other robot is moving to this door. Value 0 represents no robot is moving to this door.

Door ID	Last Update	Is Used
1	2020-06-01 15:15:26	0
2	2020-06-01 15:15:06	0
3	2020-06-01 15:12:36	0
4	2020-06-01 15:15:16	0
5	2020-06-01 15:11:46	0
6	2020-06-01 15:11:36	1
7	2020-06-01 15:14:26	0
...	...	...

Table 4.10: Doors Table.

4.3 Procedure

As stated in the Chapter 3, the goal of task scheduling is finishing all tasks as soon as possible while keep the cost as low as possible. The task assignment and execution has at two level. [GMS17] the task and the path planner solves a planning problem. It takes and occupancy grid, a specific robot and a set of task specifications, and generates trajectories for each task under the assumption that there are no dynamic obstacles (include other robots). According to those trajectories and task specifications, the task with the lowest cost will be assigned to robot. At the dynamic level, after each robot receive a task, it runs a navigation stack to execute this task stepwise. Each robot computes a local trajectory but takes into account dynamic obstacles. The process of the robot task scheduling system is as follows.

4.3.1 Centralized Pool

Handle task Request With robot status such as positions and available battery provided by robot, the multi-robot task scheduling module in the architecture should perform multi-robot task scheduling. When the centralized pool receives a task request

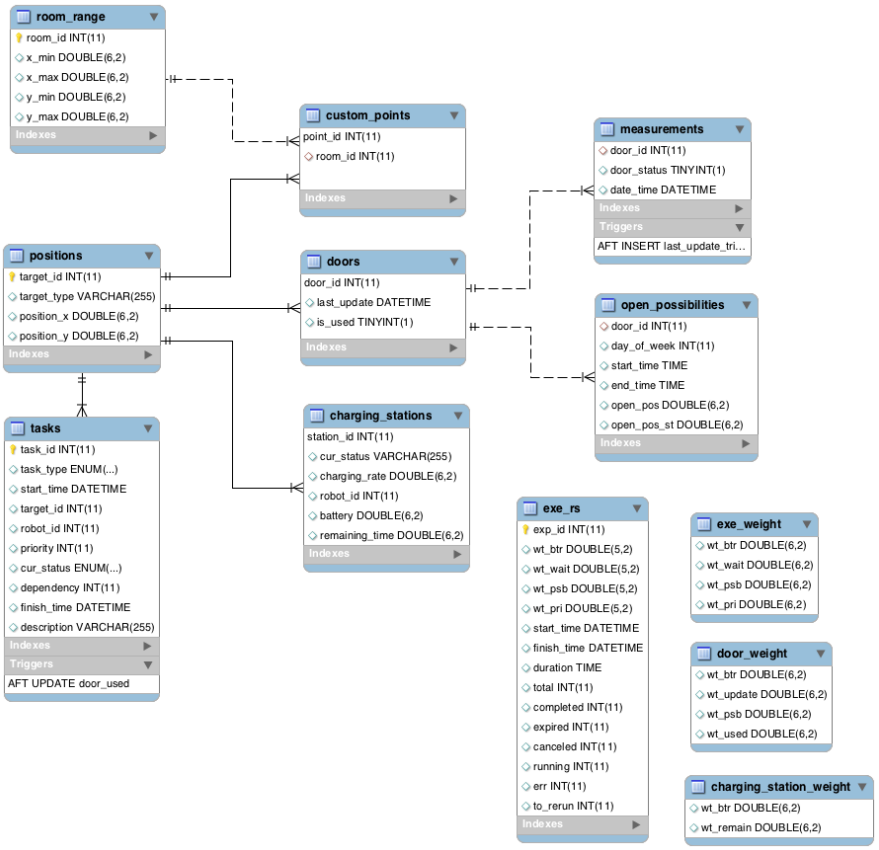

Figure 4.2: Database Entity Relationship Diagram

(Table 4.2) from robot, the multi-robot task scheduling module in the architecture. The implementation of task scheduling is shown in Figure 4.3.

Handle Task Feedback. When the centralized pool receives a task feedback (Table 4.4) that contains a new measurement result from robot, it will add a record in "measurement table" and update "open possibilities table" in database (Table 4.9).

Handle Task Result. When the centralized pool receives a task result (Table 4.5), it updates status column in "tasks" table in database. In order to make the robot complete the task as much as possible, every time when an "navigation tasks" failed, its start time will be delayed and its priority will be increased by 1 until it exceeds the maximum and is marked as "Failed" (Figure 4.4).

4.3.2 Robot

Robot Process Tasks When the task queue(Figure 3.1) in a robot is empty, the robot requests a new task. If the robot gets a "charging task", it will move to the position of charging station(Figure **??**) and interact with charging station node (Chapter 4.3.3). When a robot gets an "navigation tasks" which is a complex task, it will move to all goals in order. When a robot gets a "gather environment information" task, it will move to the door's position. During task processing, the timer checks periodically the status of navigation stack. If any errors occurs, the robot send a "failed" result with description to the centralized pool. When all tasks are completed without error, the robot will send "Succeeded" result to the centralized pool.

Robot Handle Messages While a robot is processing a task, it listens to door sensors and forwards measurement result to the centralized pool. Besides messages from sensor, it also receives messages from "move_base" node. The details of robot message handling is shown in Figure 4.7.

4.3.3 Charging Station

The charging station consists of a charging station node and "charging station" table in database (Table 4.2).

A charging station has four states : "Free", "Charging" and "Charging finished". Its initial state is "Free". When a robot arrives the charging station, it will start interacting with charging station node (Figure 4.9). Once the charging station receives robot information, its state will be changed to "Charging" and its "battery level" will be increased

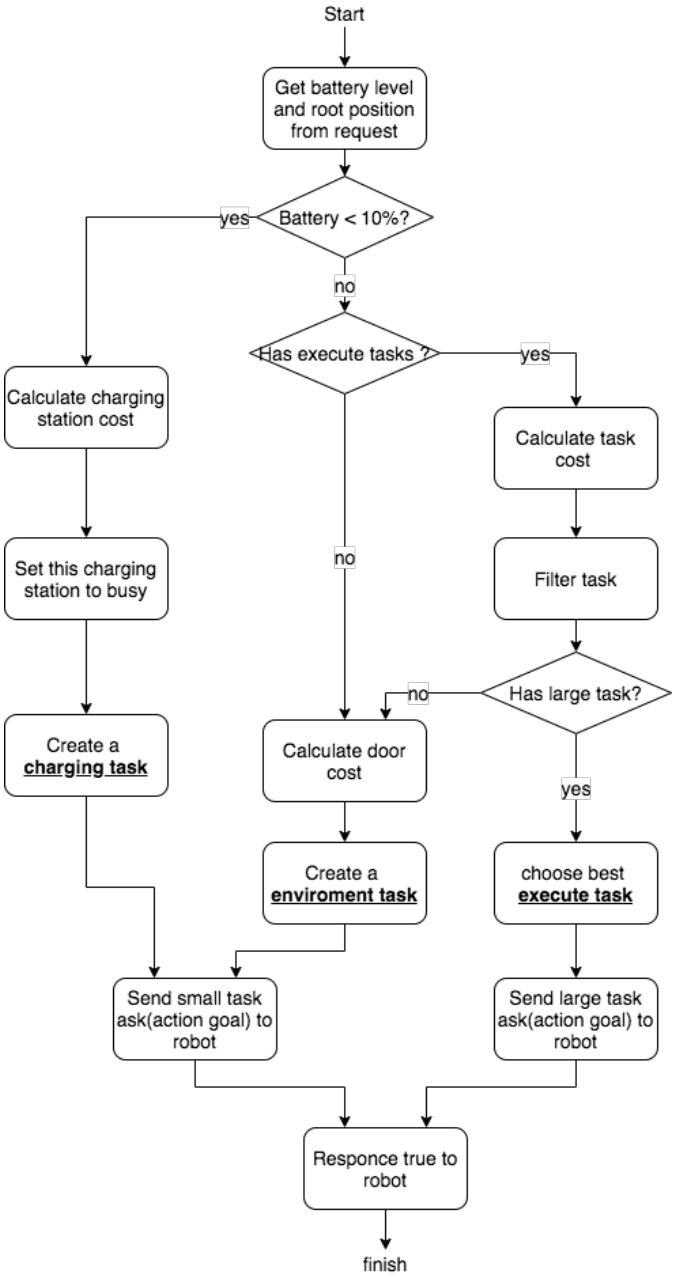

Figure 4.3: Centralized Pool Task Allocation

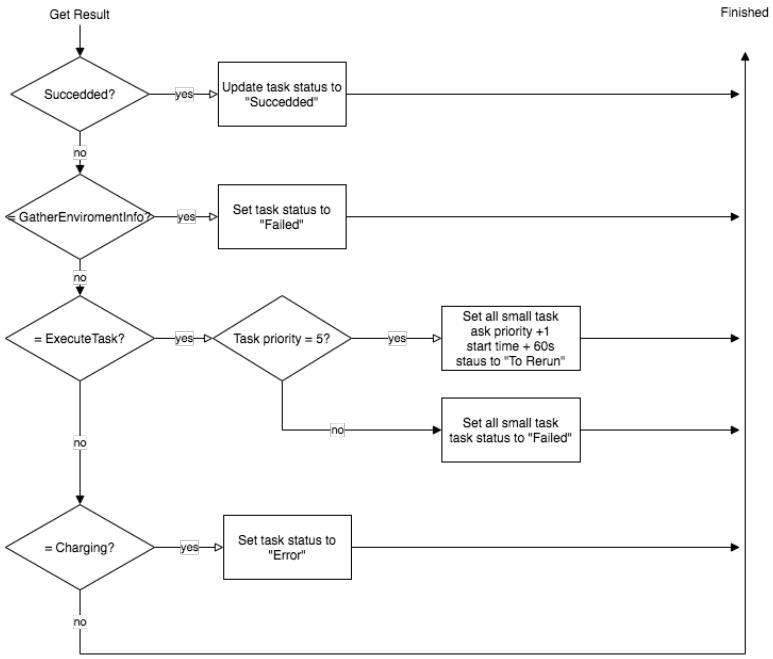

Figure 4.4: Centralized Pool Handle Task Result

Task Status Explanation

- **Succeeded.** Robot successfully moved to the goal position and completed the task.
- **Error.** An error that cannot be corrected by itself occurred and the system requires manual restart. For example, a robot failed charging.
- **Failed.** A Task failed. Reasons of task failure includes: The robot was not able to move to goal positions or process robot action(3.1).
- **To rerun.** If an "navigation tasks" failed, its priority was increased. The task is marked as "to rerun task" for a future scheduling by task scheduling module.

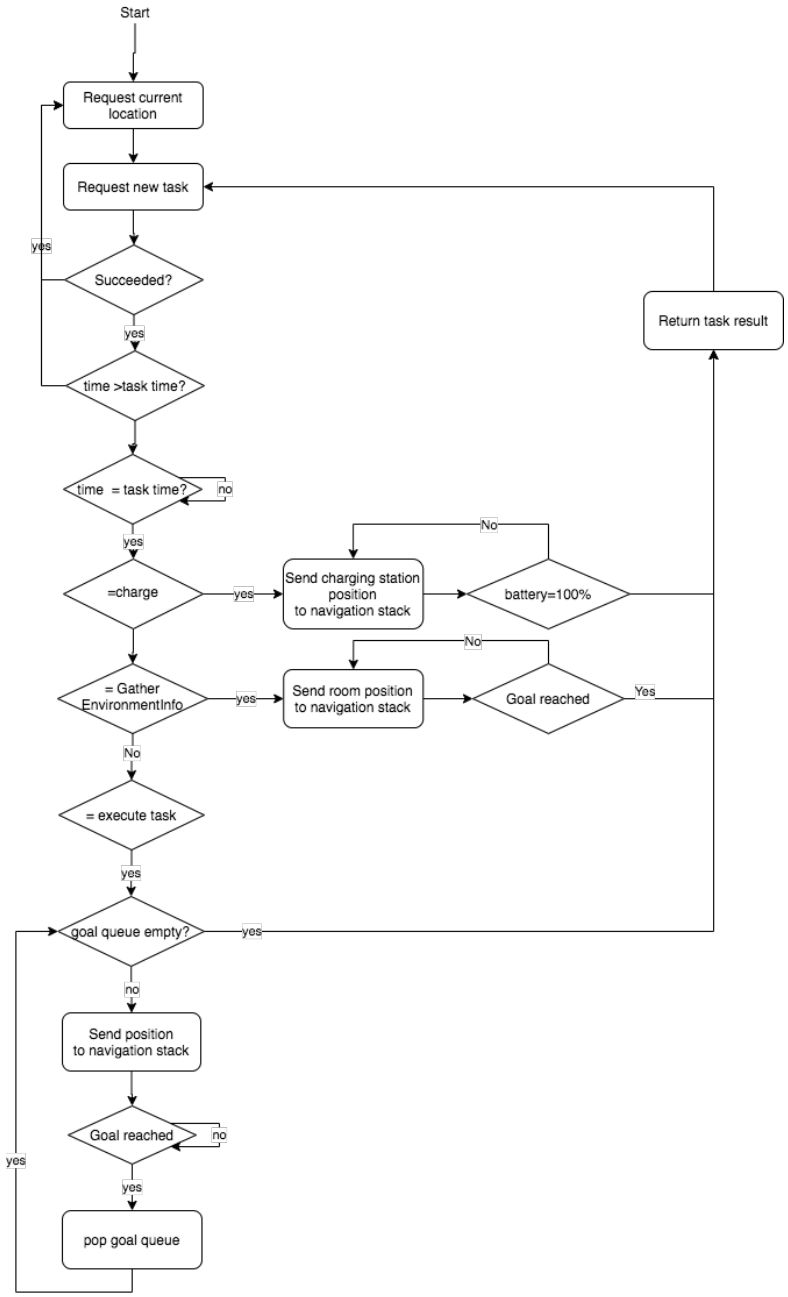

Figure 4.5: Robot Process Task

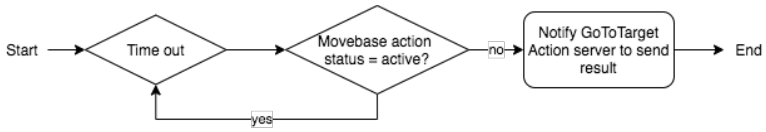

Figure 4.6: Robot Timer

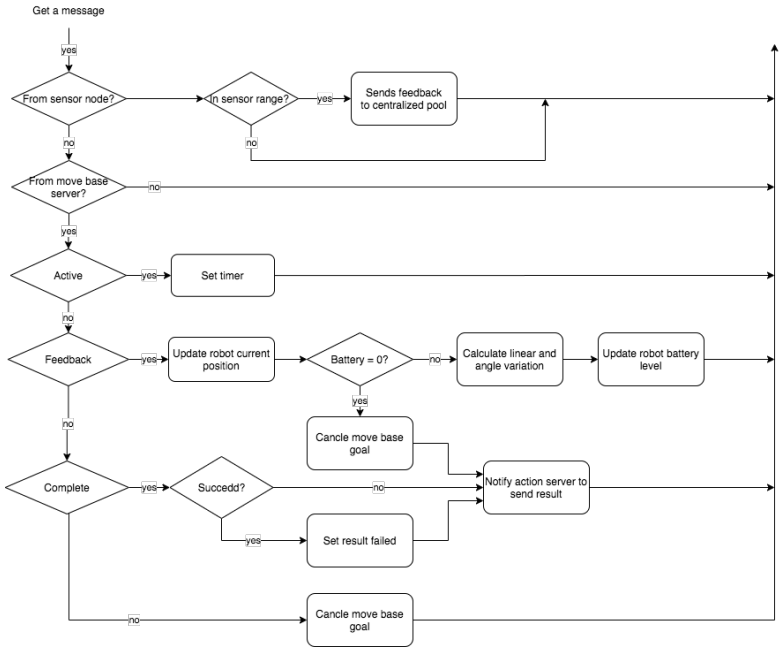

Figure 4.7: Robot Handle Message

and its "remaining time" will be decreased (Figure 4.8). Once finishing charging, its status will be set to "Charging finished". When robot leaves charging station, its status will be set to "Free".

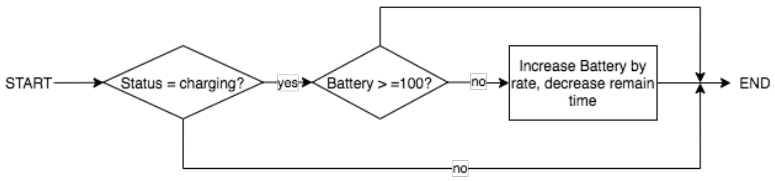

Figure 4.8: Charging Station Scheduled Charging Event in Database

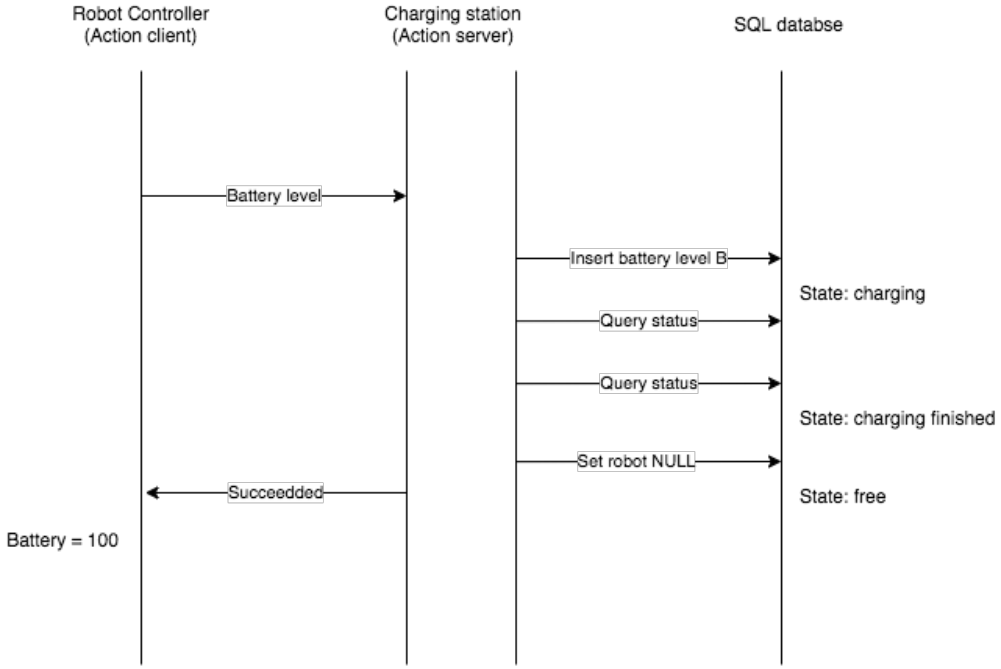

Figure 4.9: Charging Station Message

Chapter 5

Evaluation

To evaluate this multi-robot system, we used Gazebo to create a simulated office environment. There were no obstacles to this office environment. Simulation time was used in Gazebo. The simulation time flowed fluctuates, and its flow rate was about 10%-20% slower than in real-time. In the experiment, the sensor measured the opening and closing of the door. An open door meant the office was occupied, and a closed-door meant the office was empty and not occupied. The robot gathered measurement data from sensors while running tasks and sent task results and measurement data to the centralized pool. The centralized pool was the global controller that received information about the robots and the environment and made decisions base on the information. The task results and measurement data were counted and analyzed. Two kinds of experiments were designed: gathering information experiments and navigation experiments. Each gathering information experiment ran for 15 simulated minutes. The centralized pool could get about 30 task results and 90-140 measurement data. In each navigation experiment, robots completed 15 navigation tasks. The centralized pool could get 30-60 measurement data. The multi-robot system ran 600 experiments, took about 10,800 simulated minutes, completed 9,000 tasks, and gathered 50,000 measurement data.

5.1 Simulation

Simulation is essential because it can pre-estimate the performance of the multi-robot system before applying it to a real office environment. Gazebo is a 3D robot simulator software based on physics simulation. To evaluate this multi-robot system, Gazebo was used to create a simulated office environment. The sensors are simulated using a ROS node called "sensor simulator" because ROS nodes can use ROS build-in messaging system.

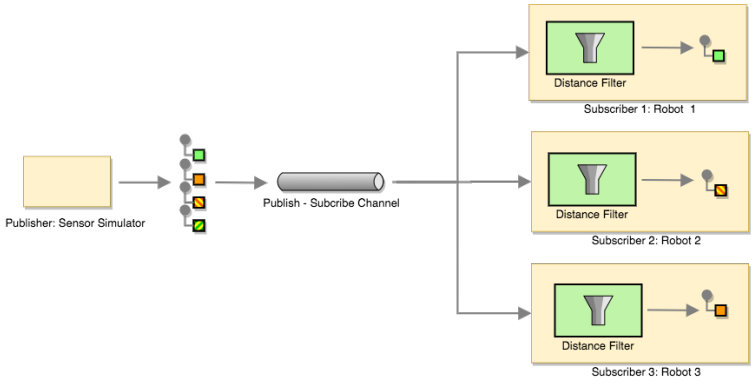

Figure 5.1: The sensor simulator.

5.1.1 Sensor Simulation

Sensor simulator generated measurement data. The measurement data about door status was an integer that could be set to 0 (door closed)or 1 (door opened). The sensor simulator first uses the current simulated timestamp to find the door open probability in the door open probability table. Then generate this number based on this probability. For example, on Monday (day of week value is 2) at simulation time "10:30:00" (on time slot 10:00:00-10:59:59), in 80% probability a "door open" was generated and in 20% a "door closed" was generated (Initialized probability of door open was 0.80).

Sensor simulator published sensor messages. In this project, a sensor simulator node was used to publish sensor messages. It published one message for each door periodically to a ROS topic "sensor data". The sensor message contains door id, sensor position, timestamp, and the generated measurement data. An example of a sensor message is shown in Table 4.1.1. Figure 5.1 shows that the sensor simulator published sensor messages and the robot subscribe to sensor messages.

Robots subscribed to sensor messages. The process of sensor simulation is shown in Figure **??**. The robots (robot controller nodes) subscribe to the same ROS topic "sensor data". Every time the sensor simulator sends a message, all robots will receive this message simultaneously. Their distance filters filter sensor messages with a position outside the communication range and keep sensor messages within the communication range. In all experiments communication range was set to 1 meter. The working principle of filter is: the robot first reads its position information, then reads the position information in the sensor message, and finally calculates the linear distance between the two positions and compares it with the communication range. If the linear distance was greater than

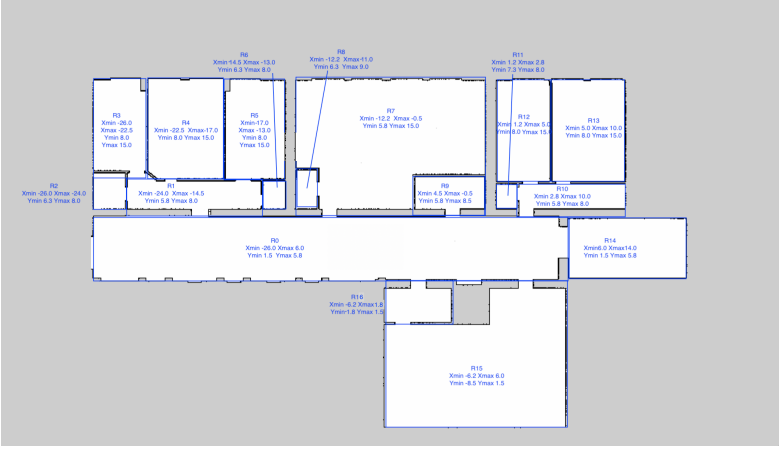

Figure 5.2: The simulated office environment. There are 16 Rooms in this office environment.

the communication range, this message was filtered out. As a result, the robot simulator sends instant sensor messages to robots in the range.

Limitations of the sensor simulator There were two main limitations of the sensor simulator. The sensor simulator published measurement data while robots immediately received and filtered these data by its position. It was a one-way process: once the measuring data are received by the robot and filtered out, they were deleted. This limitation caused the robot to gather different amounts of data from the sensors. A map with door positions is shown in (Figure 5.2). For example, the robot must pass through room 7 to go to room 8. Therefore, the robot may gather more data from room 7 than from room 8.

5.1.2 Office Environment Simulation

An office environment was used to evaluate the multi-robot system. This office environment contains a corridor along the central x-axis and 16 rooms located around the corridor (Figure 5.4). Each room was assumed to have one or multiple doors, and each door was attached to a sensor. The sensor simulator (Chapter 5.1.1) was used to broadcasted sensor messages periodically. There were no dynamic obstacles in this environment.

Simulation time was used in this office environment in Gazebo. All experiments used the default simulation time. The simulation time flowed fluctuates, and its flow rate was about 10%-20% slower than in real-time. The start and pause bottom of the Gazebo

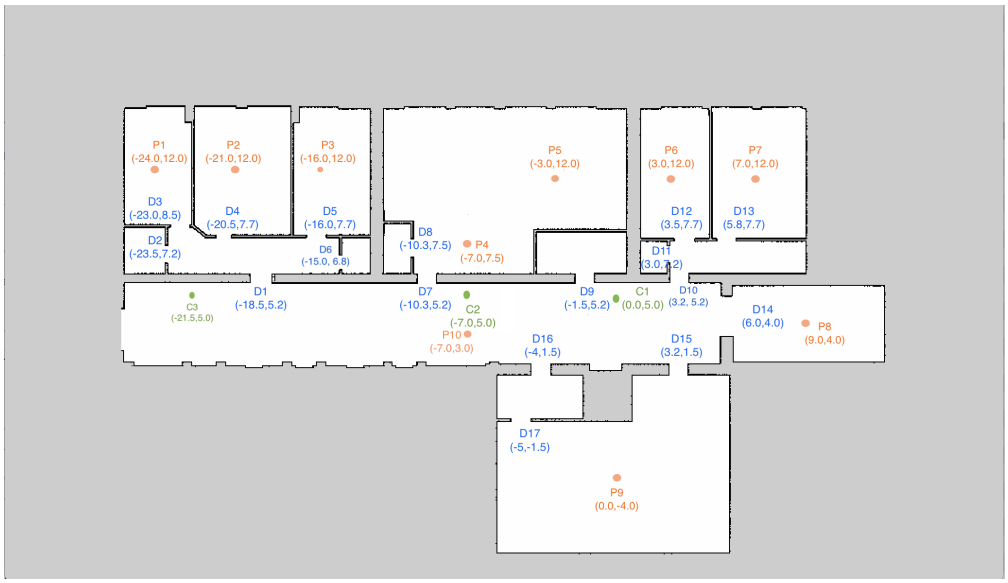

Figure 5.3: The position of doors and charging stations. C1-C3 are charging stations and D1-D17 are doors. The sensor attached to the door has the same position with the door. P1-P10 are target positions used in navigation experiments

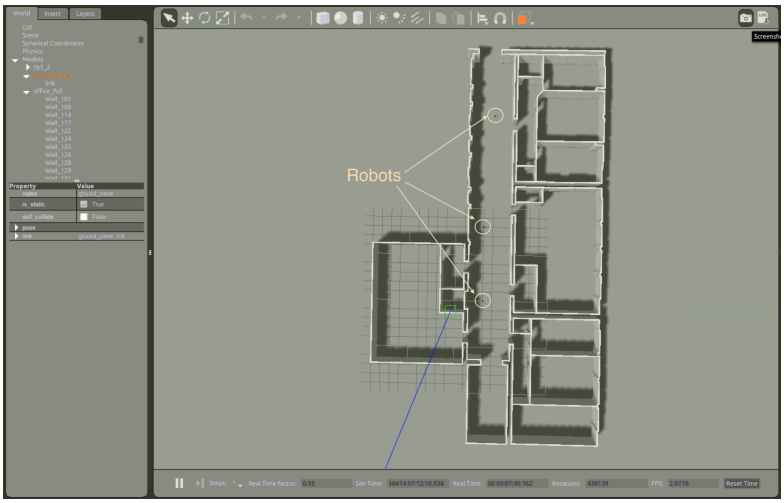

Figure 5.4: The simutated office environment in Gazebo.

GUI interface can speed up the flow of simulation time. But speeding up by five times would cause Gazebo to crash.

Limitations of the office environment. As shown in Figure 5.2, even though the simulated working environment was very similar to the real working environment, there was one limitation: There was no door installed at the exit in the simulated working environment. As a result, even if the instant measuring data was "door closed", the robot can enter the rooms and successfully finish the task. However, this task needed to be canceled or done again because a door was closed. This limitation affected the number of successful tasks.

5.2 Experiment Procedure

Two kinds of experiments were designed: gathering information experiments and navigation experiments. Charging was required between any two experiments to conduct multiple experiments one after another. In other words, at the beginning of each experiment, all robots started from their initial position, which was the charging station. After each experiment, the robot returned to the charging station to charge. For example, robot 1 charged at charging station 1, robot 2 charged at charging station 2, robot 3 charged at charging station 3. In this case, the robot would not stop moving due to a lack of energy during the experiment.

5.2.1 Gather Information Experiment

After the simulation starts, it went to the charging station to charge. In each gather information experiment, robots need to complete navigation tasks continuously. Fifteen minutes later, the robot first completed the current task and then returned to the charging station. The task completion status and measurement results in these 15 minutes will be statistically analyzed. When all robots were charged, the next experiment starts.

When all robots were charged, the first experiment began. Figure 5.5). The experiment duration was a constant value T. The experiment finish time $T_{\text{finish time}} = T_{\text{start time}} + T$.

The task scheduling approach (Chapter 3.4) make decisions base on information about robots and the environment. In this decision-making process, it was necessary to calculate the cost of robot computing tasks. At the beginning of each experiment, the weight of the cost function parameters would be changed according to the experiment result table (Table 5.1). The followings are columns of the Table:

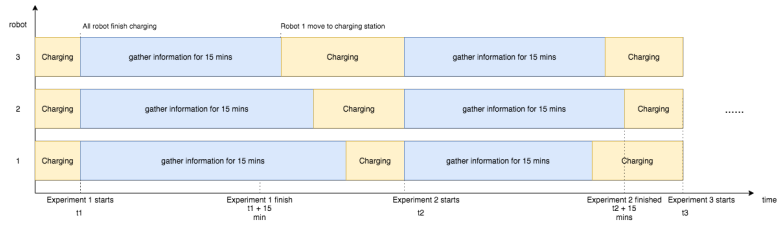

Figure 5.5: The procedure of gather information experiment. In each gather information experiment, robots need to complete navigation tasks continuously for 15 minutes.

W_{energy}	$W_{\text{last update}}$	$W_{\text{probability}}$	Average Last update Time	Average Update Interval	Succeeded Task	Failed
15.00	-0.10	-0.20	00:13:18	00:01:29	52	0
15.00	-0.10	-0.20	00:14:30	00:01:33	47	0
15.00	-0.10	-0.20	00:12:52	00:01:36	49	0
15.00	-0.10	-0.30	00:14:44	00:01:55	50	0
15.00	-0.10	-0.30	00:13:38	00:01:57	50	0

Table 5.1: Example result of gather information experiment. The values of the first three columns (weight parameters) were set before the simulation experiment was started, and the values of other columns were filled after the simulation experiment.

The "Average last update time" factor was the time difference between experiment start time and minimal value in "Last update" column in the door table (Table 4.10) when an experiment finished. For example, "Last update" factor in experiment 1 (Figure 5.6) was"00:02:57". It means that the door in the worst case not be measured since 2 minutes 57 seconds after the experiment starts. The "Average update interval" means the average interval of door update. For example, "Average Update Interval" factor in experiment 1 (Figure 5.6) was "00:01:00". It means that, on average, every door was updated every minute. The "Succeeded Task" factor means the number of succeeded "gather information" task. The "Failed Task" factor means the number of failed "gather information" task.

Use Enumeration Method to Find the Best Weight Combinations

Experiment Introduction To find the best weight combinations, 30 experiment are created with $W_{\text{energy consumption}} \in \{0, 1, 5, 10\}, W_{\text{update}} \in \{-10, -5, -1\}, W_{\text{probability}} \in \{-10, -5, -1, 0\}$. In addition, the conditions $W_{\text{update}} = 0$ was not testable, because during the experiment centralized pool generate the same "gather information tasks", which let robots not moving and measuring their nearest door. Experiment Duration T = 10 min.

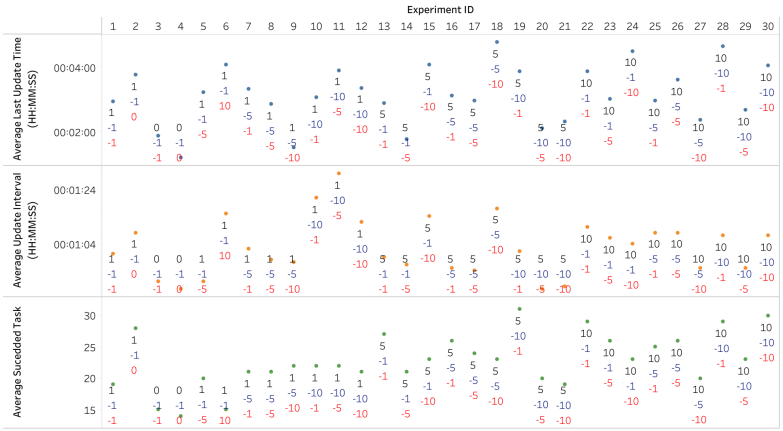

Figure 5.6: Use enumeration method to find best weight combination. The marks are labeled by weight combinations (energy, last update, probability).

Experiment Result Figure 5.6 represent the experiment result.

Experiment Analysis As shown in the experiment result, all "Minimal Last Update" value was from 1 min to 5 min, much less than the experiment duration (10 min). It means some doors were not timely updated information. One possible reason was that the velocity of the robot was small (about 0.2 per second). The experiment duration (10min) was not long enough to let three robots pass to all doors. Another possible reason was that the robot's route was partially duplicated. For example, when the system started, robot 1 was at charging station 1 and got a task to door 3, while robot 2 was at charging station 2 and got a task to door 4. As shown in Figure 5.2, these two route are partially duplicated, both of them pass through door 1 and entered room 1 (Figure 5.2). The ideal solution was to give both tasks to one robot.

5.2.2 Use Analysis to Find the Best Weight Combinations

Experiment Introduction As is discussed in the last experiment (Chapter 5.2.1), the experiment time was too short to allow the robot to explore each door. Therefore, the experiment duration in this experiment was increased to 20 min. Firstly, we should find the best weight combination for "weight battery" and "weight last update", then find the third weight "weight probability". Finally, the best combination of weight will be concluded. According to the cost table (Figure 5.7), the update time value has a minimal value of 6.986 and a maximal value of 336.986 (column 3), which are much larger than energy consumption (column 2) and product of door open possibilities

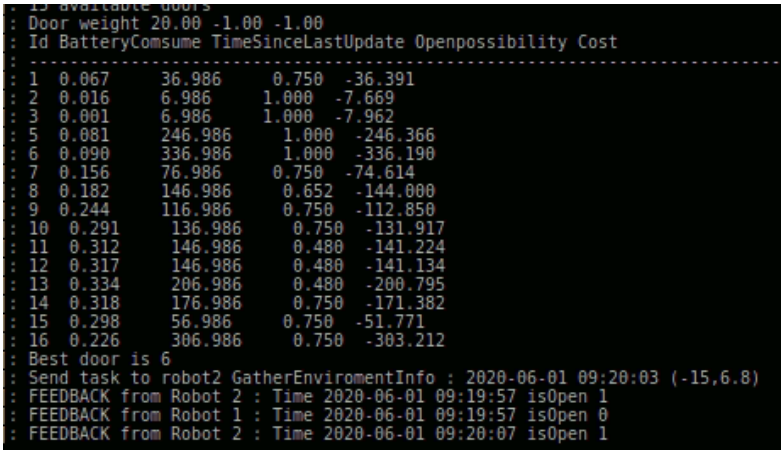

Figure 5.7: The cost table in the centralized pool.

(column 4). Therefore, 18 experiments are created with $W_{\text{battery}} = 0.1$ and $W_{\text{battery}} \in \{1, 5, 10, 15, 20, 25\}$.

Experiment Result The experiment results are shown in Figure 5.8 and Figure 5.9

Experiment Analysis As shown in experiment result Figure 5.8, "weight battery" increased from 0 to 25, the "average succeeded experiment task number" slightly increased from 45 to 50, but "average update interval" were floated in a small range around "00:01:35". Especially, as "weight battery" increased from 0 to 15, "average last update" showed an upward trend, and as "weight battery" increased from 15 to 25, "average last update" showed a downward trend, therefore "weight battery" 15 and "weight update" 0.1 was the best combination. This combination was used in next experiment set (Figure 5.9). From this experiment set, it was concluded that "weight battery" 15, "weight update" 0.1, "weight probability" 0.1 and "weight battery" 15, "weight update" 0.1, "weight probability" -0.3 were two best weight combinations for the cost function.

5.2.3 Navigation Experiment

After the simulation started, the robot went to the charging station to charge. After all the robots were charged, the first experiment began. The robot returned to the charging station to charge after completing 15 navigation tasks. The completion status and measurement results of these 15 tasks will be statistically analyzed. When all robots are charged, the next experiment starts. In other words, robots need to finish 15 navigation tasks and 3 charging tasks in each navigation experiment. The experiment start time

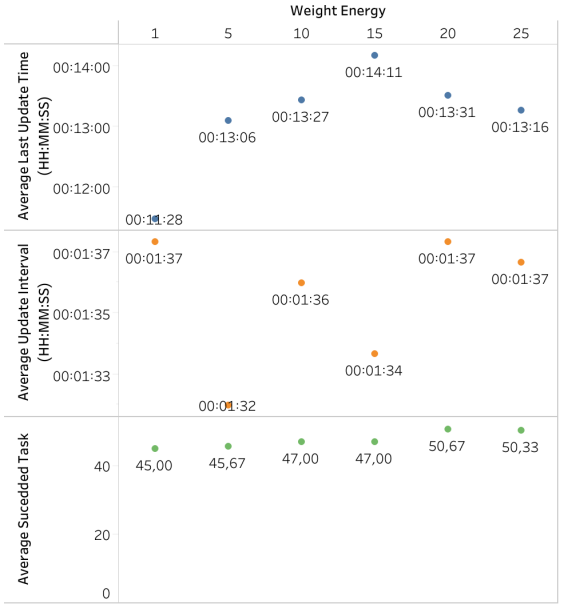

Figure 5.8: Experiment: Change W_{battery} under condition $W_{\text{update}} = 0.1$ and $W_{\text{probability}} = 0$

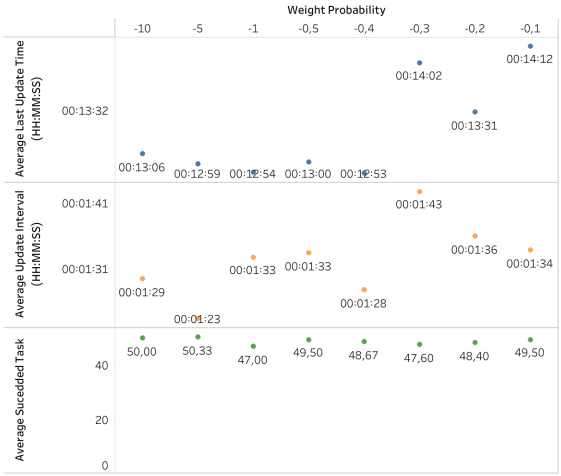

Figure 5.9: Experiment: Change $W_{\text{probability}}$ under condition $W_{\text{battery}} = 15$ and $W_{\text{update}} = 0.1$

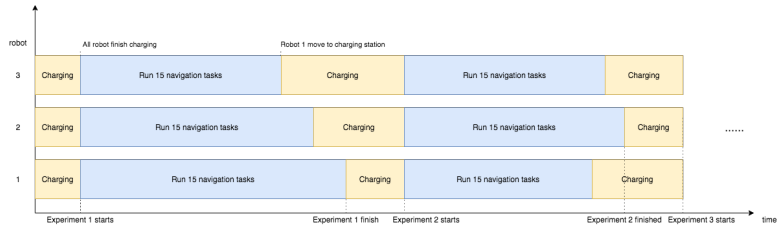

Figure 5.10: The procedure of navigation experiments. In each experiment, 3 robots need to finish 15 navigation tasks and 3 charging tasks.

Experiment	W_{battery}	$W_{\text{waiting time}}$	$W_{\text{door open probability}}$	W_{priority}	Experiment Duration	Total Task	Succedded Task	Expired Task	Failed Task
1	1.00	1.00	-1.00	-1.00	00:18:23	15	15	0	0
2	1.00	0.00	0.00	0.00	00:17:23	15	7	8	0
3	0.00	1.00	0.00	0.00	00:18:56	15	15	0	0
4	0.00	0.00	-1.00	0.00	00:18:41	15	5	10	0
5	0.00	0.00	0.00	-1.00	00:17:21	15	3	12	0

Table 5.2: Example result of navigation experiment. The values of the first four columns (weight parameters) were set before the simulation experiment was started, and the values of other columns were filled after the simulation experiment.

was when all robots finished charging and started request tasks, and the experiment finish time was when the latest task was finished. This experiment process is shown in Figure 5.10.

The task scheduling approach (Chapter 3.4) make decisions base on information about robots and the environment. In this decision-making process, it was necessary to calculate the cost of robot computing tasks. At the beginning of each navigation experiment, the weight of the cost function parameters would also be changed according to the experiment result table (Table 5.2). The followings are columns in the table:

The experiment duration was a critical evaluation factor in this table, which was the time difference between the experiment start time and experiment finished time. The end state of navigation tasks was evaluated. In an experiment result table, the “Succeeded Task” column counted the tasks ended with the succeeded state, the “Expired Task” column counted the tasks ended with the created state, the “Failed” column counted the tasks ended with states including running, error or canceled states.

5.3 Experiment Summery

Chapter 6

Discussion

6.1 Result

Task scheduling is a significant research area in the multi-robot system. This research focuses on exploiting knowledge of room occupation for the scheduling of navigation tasks of robots in office environments, decreasing the time of completion and energy consumption. With the information of room occupation, on the one hand, it can prevent the robot from wasting energy to go to empty rooms, and on the other hand, it can make the robot try to find people to interact with.

I have implemented a multi-robot system based on the simulated environment in Gazebo. The following functions are successfully implemented: (1) The fixed sensors can conduct the long-term room occupation measuring to keep the "room occupation" information up to date. (2) The robots can initialize task scheduling procedures by requesting tasks in the centralized pool and then perform received tasks sequentially. (3) The centralized pool, which is a control system that can store global information and schedule tasks to robots according to the task scheduling algorithm. In the simulated working environment, the multi-system operated stably for three days, and autonomously performed more than 5000 navigation tasks. This system can handle some unexpected problems. For example, failed tasks are reassigned to other robots.

Also, a centralized task scheduling algorithm has been designed for the multi-robot system. This algorithm considers the robot status, the task specification as well as room occupation. As discussed in Chapter 5, the algorithm can improve the scheduling of tasks taking into account the number of tasks to be performed, decreasing the time of completion. In theory, this algorithm can reduce energy consumption. But current experiments cannot prove this idea. Instead, real experiments are needed.

6.2 Limitations

The simulation has the following problems, and these problems should be solved in future realistic experiments.

Sensor Problem

The simulated sensors are not specific objects in the Gazebo simulation scene, instead, there are simulated by a ROS node called "sensor simulator", because ROS nodes can use ROS build-in messaging system. As shown in Figure 5.1, the "sensor simulator" publishes measurement data while robots immediately received and filter these data by its position. This is a one-way process: once the measuring data are received by the robot and filtered out, they are deleted. As a result, the robot simulator sends instant measuring data to robots in the range. Besides, these instant measuring data are generated according to a given schedule in the database. However, these simulated data cannot perfectly imitate the real room occupancy situation.

In real experiments, an IoT device should replace the simulated sensor. This device should contain the following components:

- An accelerometer sensor to measure door open/close status. Optional are other sensors, such as light sensors and infrared sensors. Those sensors are useful in measuring room occupation.
- Enough RAM or an SD card module to store the background measuring data.
- A wireless communication module such as BLE or ZigBee. Once the IoT device finds a robot in its communication range, it should send the robot measuring data over a while.

Door Problem

The doors in the simulated environment are always open. As a result, even if the instant measuring data is "door closed", the robot can successfully enter the rooms and run the task. The task will almost only fail due to timeouts. For example, the robot is not able to communicate with its navigation stack, or several robots are entangled for a long time. However, in real experiments, The reason for the failure of the task includes that the robot hits an obstacle, the robot is damaged, and the battery runs out.

Battery Problem

The estimation of the power consumption is not accurate enough. On the other hand, the robot will change its battery level constantly. The change in battery level is calculated from the distance of travel and the angle of rotation. The traveling distance and the rotation angle are calculated based on the continuous acquisition of its current location. In the real world, the robot has a real battery, which will drop according to many factors, such as movement, the flatness of the ground, battery life, etc. According to these factors, the task scheduling module also needs an energy consumption estimation algorithm.

6.3 Future Work

This research can be further explored in several aspects.

- A similar multi-robot system in the real world could be implemented. The setup of IoT devices and doors and the estimation of robot energy consumption are discussed in Chapter 6.2.
- More types of tasks can be considered, such as delivery tasks. The delivery task includes picking up multiple goods and putting down the same amount of goods.
- Tasks can be given with different task duration, to be more precise, the difference between the estimated end timestamp and the estimated start timestamp. In this case, the task duration could be one of the decision variables, because tasks can be finished in a shorter time should be executed first. Furthermore, robots can go to the nearby charging station during the task execution. In this case, when calculating the task duration or task energy consumption of the path, the time or energy consumption required for charging should be included in the task duration and task energy consumption. The advantage of this method is that it allows the robot to maintain sufficient energy to execute the task.
- There are other methods to enhance this algorithm, such as using machine learning. Machine learning can learn discrete measurement data about room occupation, and finally, obtain a room occupation table for continuous-time in each working day. Besides, a mobile or web user interface could be developed. With this interface, the user can specify that some rooms as occupied at specific times according to their working schedule. The user can also monitor the status of the robot, its current position, and battery level and receive an alert of the robot failure.

Chapter 7

Acknowledgements

I would like to begin by thanking my supervisor Dr. Matteo Zella and Carlos Medina SÃ¡nchez. Their strict attitude and enthusiasm to research has encouraged me. Many thanks for their reading and examining on my thesis. I can not finish my thesis without their guidance.

I also would like to thank all the people who have supported me during writing. I am so luck to study in the University Duisburg-Essen.

Bibliography

[ANT] *ANT Website*. `https://www.thisisant.com`. accessed 2020-010-26.

[ASE] *A* search algorithm*. `https://en.wikipedia.org/wiki/A*_search_algorithm`. accessed 2020-10-17.

[ASM15] Afanasyev, Ilya, Artur Sagitov and Evgeni Magid: *ROS-Based SLAM for a Gazebo-Simulated Mobile Robot in Image-Based 3D Model of Indoor Environment*. In Battiato, Sebastiano, Jacques Blanc-Talon, Giovanni Gallo, Wilfried Philips, Dan Popescu and Paul Scheunders (editors): *Advanced Concepts for Intelligent Vision Systems*, pages 273–283, Cham, 2015. Springer International Publishing.

[BLE] *BLE Website*. `https://en.wikipedia.org/wiki/Bluetooth_Low_Energy`. accessed 2020-10-26.

[BMR+17] Booth, K. E. C., S. C. Mohamed, S. Rajaratnam, G. Nejat and J. C. Beck: *Robots in Retirement Homes: Person Search and Task Planning for a Group of Residents by a Team of Assistive Robots*. IEEE Intelligent Systems, 32(6):14–21, 2017.

[CSMV09] Christodoulopoulos, K., V. Sourlas, I. Mpakolas and E. Varvarigos: *A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in Grid networks*. Computer Communications, 32(7):1172 – 1184, 2009.

[DEA] *Dead reckoning*. `https://en.wikipedia.org/wiki/Dead_reckoning`. accessed 2020-10-14.

[ERP20] Eijyne, Tan, G Rishwaraj and G PonnambalamS: *Development of a task-oriented, auction-based task allocation framework for a heterogeneous multirobot system*. Sadhana, 45:1–13, 2020.

[GMS17] Gavran, Ivan, Rupak Majumdar and Indranil Saha: *Antlab: A Multi-Robot Task Server*. ACM Trans. Embed. Comput. Syst., 16(5s), September 2017.

[GZ] *Gazebo Main Page*. `http://gazebosim.org/`. accessed 2020-10-14.

[HC17] Hancheol Cho, Leon Jung, Darby Lim: *ROS Robot Programming (English).* ROBOTIS, 12 2017.

[JL17] Jeon, S. and J. Lee: *Performance analysis of scheduling multiple robots for hospital logistics.* In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 937–940, 2017.

[JM13] Jia, Xiao and M. Meng: *A survey and analysis of task allocation algorithms in multi-robot systems.* 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 2280–2285, 2013.

[KSD13] Korsah, G. Ayorkor, Anthony Stentz and M. Bernardine Dias: *A comprehensive taxonomy for multi-robot task allocation.* The International Journal of Robotics Research, 32(12):1495–1512, 2013.

[LK12] Liu, Chun and Andreas Kroll: *A Centralized Multi-Robot Task Allocation for Industrial Plant Inspection by Using A* and Genetic Algorithms.* In Rutkowski, Leszek, Marcin Korytkowski, Rafał Scherer, Ryszard Tadeusiewicz, Lotfi A. Zadeh and Jacek M. Zurada (editors): *Artificial Intelligence and Soft Computing*, pages 466–474, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[LS18] Li, H. and A. V. Savkin: *Wireless Sensor Network Based Navigation of Micro Flying Robots in the Industrial Internet of Things.* IEEE Transactions on Industrial Informatics, 14(8):3524–3533, 2018.

[LZK15] Lee, D., S. A. Zaheer and J. Kim: *A Resource-Oriented, Decentralized Auction Algorithm for Multirobot Task Allocation.* IEEE Transactions on Automation Science and Engineering, 12(4):1469–1481, 2015.

[MOV] *Move Base Node.* `http://wiki.ros.org/move_base`. accessed 2020-09-25.

[NMMG17] Nunes, Ernesto, Marie Manner, Hakim Mitiche and Maria Gini: *A taxonomy for task allocation problems with temporal and ordering constraints.* Robotics and Autonomous Systems, 90:55 – 70, 2017. Special Issue on New Research Frontiers for Intelligent Autonomous Systems.

[PAR] *Potential field.* `https://en.wikipedia.org/wiki/Particle_filter`. accessed 2020-10-17.

[PNK+15] Pyo, Yoonseok, Kouhei Nakashima, Shunya Kuwahata, Ryo Kurazume, Tokuo Tsuji, KenÂ€™ichi Morooka and Tsutomu Hasegawa: *Service robot system with an informationally structured environment.* Robotics and Autonomous Systems, 74:148 – 165, 2015.

[POT] *potential field.* `http://www.cs.cmu.edu/~./motionplanning/lecture/Chap4-Potential-Field_howie.pdf`. accessed 2020-10-17.

[RRT] *RRT (Rapidly-exploring Random Tree).* `https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree`. accessed 2020-10-17.

[RVI] *Rviz.* `http://wiki.ros.org/rviz`. accessed 2020-10-17.

[Seo17] Seo, Woojin: *Indoor Dead Reckoning Localization Using Ultrasonic Anemometer with IMU.* Journal of Sensors, 2017:12, 2017.

[SM07] Shah, K. and Y. Meng: *Communication-Efficient Dynamic Task Scheduling for Heterogeneous Multi-Robot Systems.* In *2007 International Symposium on Computational Intelligence in Robotics and Automation*, pages 230–235, 2007.

[T3S] *turtlebot3 SLAM.* `https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/`. accessed 2020-10-17.

[Zig] *ZigBee Website.* `https://en.wikipedia.org/wiki/Zigbee`. accessed 2020-010-26.

Versicherung an Eides Statt

German

Hiermit versichere ich, dass ich die vorliegende Bachelor(Master)arbeit selbständig - mit Ausnahme der Anleitung durch die Betreuer - verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe. Ich versichere dass ich alle entsprechenden Angaben nach bestem Wissen und Gewissen vorgenommen habe, dass sie der Wahrheit entspechen und dass ich nichts verschwiegen habe. Mir ist bekannt, dass eine falsche Versicherung an Eides Statt nach §156 und nach §163 Abs. 1 des Strafgesetzbuches mit Freiheitsstrafe oder Geldstrafe bestraft wird.

English

I hereby declare that I have written this Bachelor (Master) thesis - except for the guidance of the supervisor - independently, using no other than the specified sources and resources, and that all quotations have been indicated. I declare that I have reported to the best of my knowledge all the relevant information, that it is true and that I concealed nothing. I am aware that a false declaration will be punished according to §156 and §163 par. 1 of the criminal code with a prison sentence or a monetary penalty.

Essen, October 29, 2020
(Ort, Datum)

Xuanjiao Zhu