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

28. Oktober 2020

Erstprüfer: Prof. Dr. Pedro José Marrón
Zweitprüfer: Prof. Dr. Gregor Schiele
Zeitraum: 1.May 2020 - 1.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 addition to interacting with people, robots also need to learn from the surrounding environment to schedule their activities effectively. In this research, the knowledge of room occupancy is applied to task scheduling by the robot system. Room occupation means the probability of someone in the office. In other words, in a typical office environment, if at least one person is in the office, the office is considered occupied. However, the perspective of individual robots cannot cover the entire office environment. But the knowledge of room occupation can be established by the Internet of Things (IoT) technologies.

There are several requirements for obtaining room occupation for the entire office environment. First, it needs to operate for a long time. Second, it needs to keep the information as current as possible. Based on these requirements, I designed a method to obtain room occupation information. I considered installing BLE sensors on all doors. The robot needs to exchange information with sensors while performing navigation tasks and find sensors when there is no navigation task. To achieve this goal, I have designed a component called the centralized pool. The centralized pool is the global controller in the multi-robot system. It can receive room occupation information from the robots. The centralized pool uses this information in two ways. It schedules the robot's navigation tasks according to the information and can also designate the robot to explore the room lacking the information.

Simulation results demonstrate the efficiency and robustness of the multi-robot system. In the simulated working environment, the multi-system operated stably for three days, and autonomously performed more than 5000 navigation tasks. This system can also overcome some failures. For example, the robot drops tasks when it is not able to perform them. Other robots reuse those dropped tasks.

Chapter 1

Introduction

Since the advancement of robotics in the modern society with robots' reliability and high work quality, robots are being used together as a multi-robot system, providing advantage compared to a single robot [ERP20]. Multi-robot systems are widely employed in various applications such as healthcare facilities to assist elderly residents [BMR+17] and industrial plant inspection[LK12]. Thinking of an office building where people have different working schedules. If at least one person is in the office, the office is considered occupied. The robot working at this office environment should not only interact with people but also to avoid waste energy by moving to empty rooms. This requires the robots to learn from the office environment information and schedule their activities accordingly.

For a working robot, environment information is indispensable for efficiently perform activities. There are several types of environment information: (1) Information about its surrounding, such as the position of furniture, doors, walls, and other robots. (2) Information about room occupation which presents the probability of finding a person in the room. The latter may change during long-time operating. Therefore, the long-term room occupation measuring should be conducted to keep its value up to date. However, current sensing technology, especially for the case of robots equipped with external sensors, is not good enough to gather room occupation information of all rooms satisfactorily. Although the camera is able to identify whether there is someone in the office, it is susceptible to changes in lighting conditions and the appearance of objects. Moreover, the field of view is too narrow. Besides, if I install sensors on robots, the vibrations of a robot body can cause damage to the sensors or incorrect measurement results [PNK+15]. Especially, sensors on robot are not able to capture the background changes, which can cause an incomplete measurement result table.

Therefore, I have considered fixed sensors in the office environment. The fixed sensors are not only stable but also can accurately perform long-term environmental measurements. In order to perform long-term measurement, the sensors use low-power and wireless communication technology and without connected to the centralized pool. To be more precise, its measurement results are stored in its local memory and acquired by the passing robots. In addition, to keep the room occupation information in the multi-robot system and in the sensors synchronized, the centralized pool may assign the robot to some unsynchronized sensors to gather information.

I have developed a multi-robot system able to share the information among sensors, charging stations, robots and the centralized pool. The robot can start five kinds of interactions: (1) As long as the robot enters the range of the sensor, they conduct a rapid exchange of room occupation information. (2) The robot interacts with a charging station at the beginning and at the end of charging process. (3) When the robot is idle, it autonomously requests a task from the centralized pool. The centralized pool then responds robot with a command of a simple task (task contains one target position) or a complex task (task which can be decomposed into multiple simple tasks). This command contains the when and where to perform activity(s). These activities include "navigation task", "charging","gather information" etc. (Figure 3.1). (4) While performing activities, the robot sends acquired room occupation data to the centralized pool. (5) Once the robot finished all activities, the robot notifies the central pool that the task is succeeded and idle state again.

I also have implemented a multi-robot task scheduling architecture in the centralized pool. The goal of multi-robot task scheduling is to find the optimal task for the robot in order to minimize the total cost. According to the application requirements, three cost function are designed to calculate cost for "gather information task", "navigation task" and "charging task". The cost consider decision variable such as energy consumption, time, room occupation, priority etc.

I also pay particular attention to error handling to robust against robot failures. Two failure cases are considered here. First, sometimes the robot may not able to handle the assigned 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 processed and reused for task scheduling. Second, sometimes the robot may have low battery level, but it fails recharging since all charging station are occupied or the charging station does not respond to the robot. In this case, the deficient robot sends failure detail to the centralized pool. This failure will then be observed by the user and the deficient robot waits for the user to manually control it [SM07].

In addition to the architecture I proposed herein, our system uses existing platform called Robot Operating System (ROS [ROS]) 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 station is simple and straightforward [SM07].

1.1 Motivation

Given an office building where people have different working schedules. If at least one person is in the office, the office is considered occupied. When multiple robot working in this environment, an efficient task scheduling algorithm is required, in order to minimize the completion time while decreasing the total power consumption [LK12]. In this case, room occupation becomes a valuable decision variables for task scheduling, which describes the probability that the robot can enter the office. Although there are increasing amounts of research have been conducted in the area of task scheduling for multi-robot system [SM07], unfortunately, current research work mainly on multi-robot cooperation dynamic environments and rarely address on adoption of Internet of Things technologies. Therefore, I consider adding more fixed sensors to in the whole office environment [CV10], and implement a multi-robot system. This multi-robot system should be able to acquire the room occupation information, and use it as one of decision variables to schedule tasks.

1.2 Problem definition

In order to develop a multi-robot system, first I need to set up some working environment as well as the assumptions within the environment [SM07]. The environment is shown in Figure 5.2, which contains a corridor along the central x-axis and 16 rooms located around the corridor. There is no robot limitation in these rooms, as long as the door of the room is opened, it can be entered by any robot. In addition, there are charging stations in the corridor to charge the robots (Figure **??**). Each room has one or multiple doors and each door is attached with a sensor. It is assumed that the people have different working schedule, which cause the room occupied and unoccupied regularly. Consider that there are a set of robots and a set of different tasks randomly distributed in different rooms. Here robot is responsible for moving in 2-dimensional physical space as well as gathering measurement result from sensors. It has a rechargeable battery, and its level drops as robot moves and rotates. The task requires one robot to traverse a path in the workspace and carry out certain activities, such as gather information, charging or transportation [GMS17]. The tasks are classified into two types: simple task (task contains one target position) and complex task (task which can be decomposed into multiple simple tasks). It is assumed that the robot can localize itself within the office environment. The robots are expected to move to the positions where the tasks are located and complete the tasks.

I split the problem into three sub-problems:

1. The first problem is the task scheduling problem. An efficient centralized task scheduling algorithm should be implemented for multi-robot system using the

knowledge of room occupation, in order to complete all tasks as soon as possible.

2. The second problem is the gather information problem, in which the robots are scheduled to gather room occupation information from fixed sensors, in order to keep room occupation information up to date.
3. The third problem is the communication problem, an efficient communication protocol should be developed to allow share the information between components in multi-robot system.

1.3 Thesis Structure

Next chapter briefly introduces the background information and the related work in both task scheduling and cost function. In Chapter 3 I formally present the architecture of the multi-robot system as well as the approaches to schedule tasks. In Chapter 4 I describe the implementation of communication protocol and components in the multi-robot system. Chapter 5 introduces the experiments performed on the implementation. Finally, in Chapter 6 I interpret the results of the performed work.

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

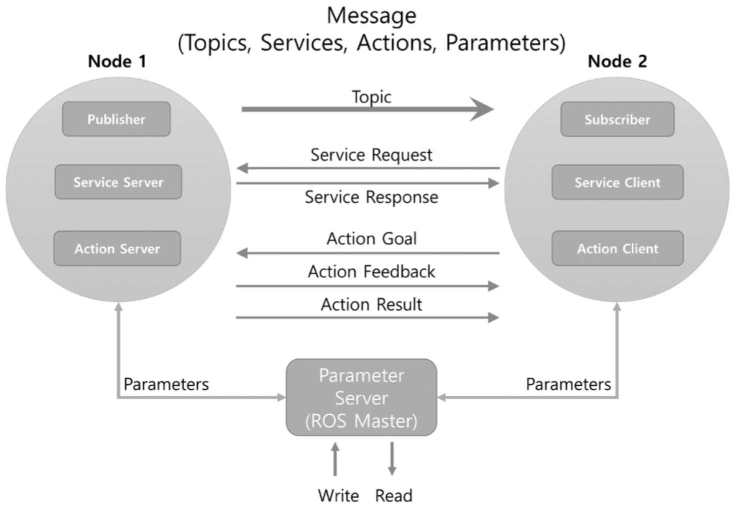


Figure 2.1: ROS Message Communication [HC17]

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 5.2) 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"

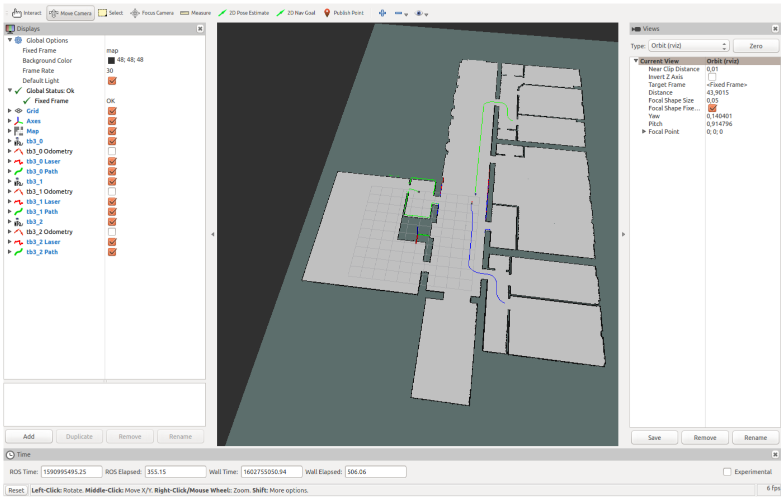


Figure 2.2: Rviz Example: Navigation using TurtleBot 3 and Laser Distance Sensor (LDS)

- **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.

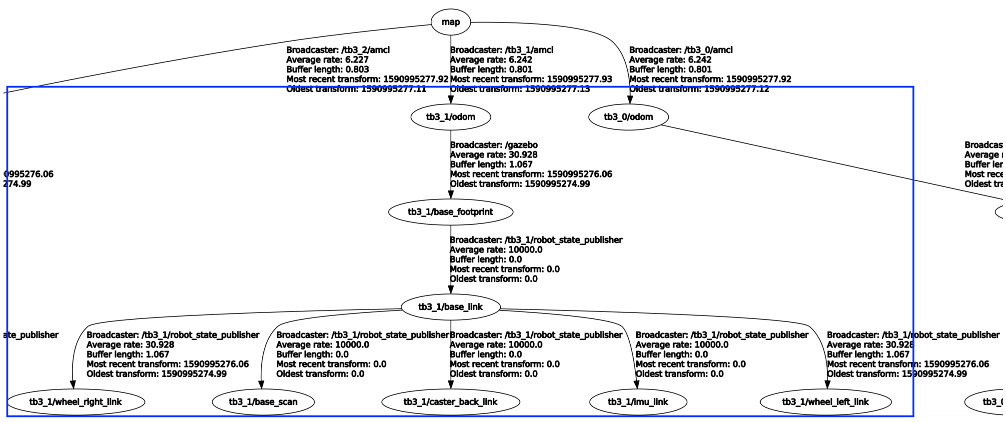


Figure 2.3: One Part of rqt_tf_tree of the multi-robot system. Each robot has a subtree (like the blue box) and is connected to the map server node

- **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.

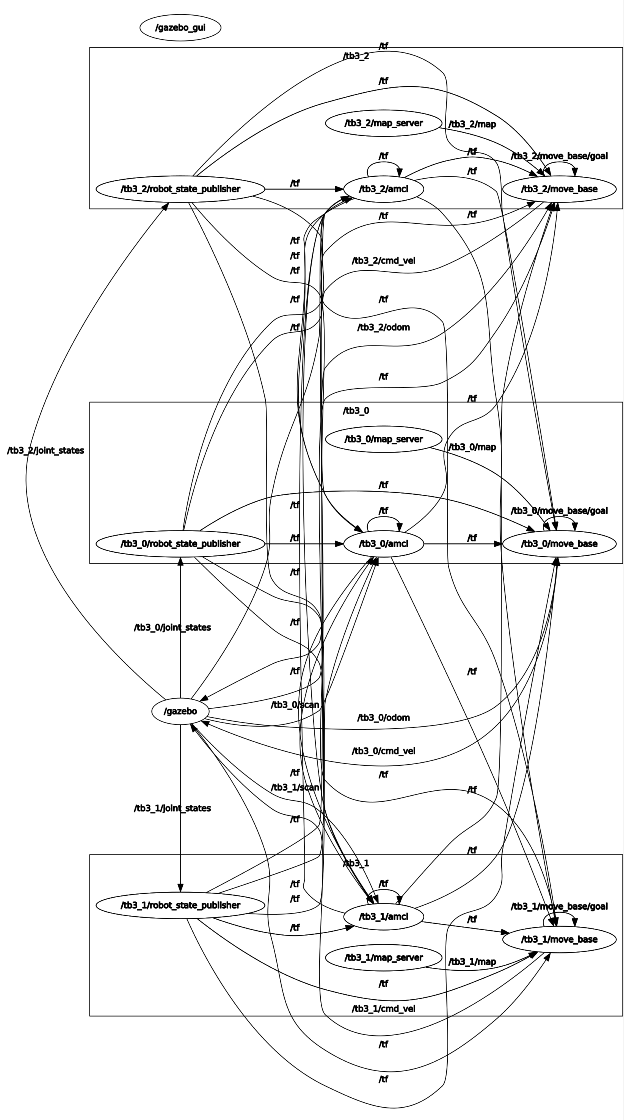


Figure 2.4: rqt_graph of multi-robot system

- **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:

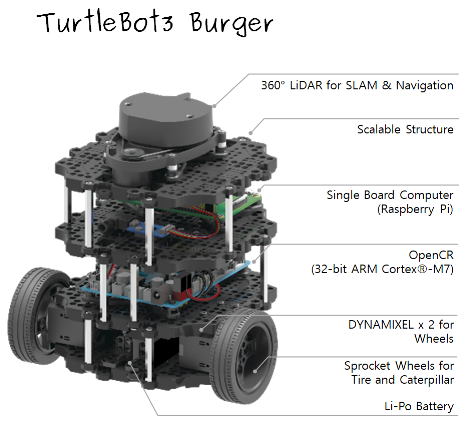


Figure 2.5: Robot Hardware Configuration [HC17]

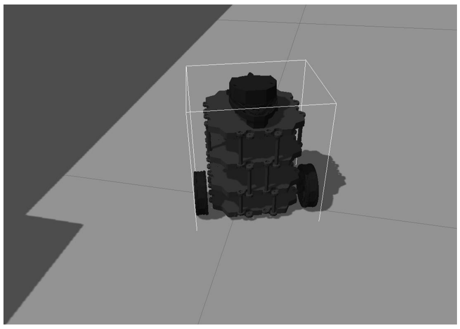


Figure 2.6: Robot 3D Modeling in Gazebo

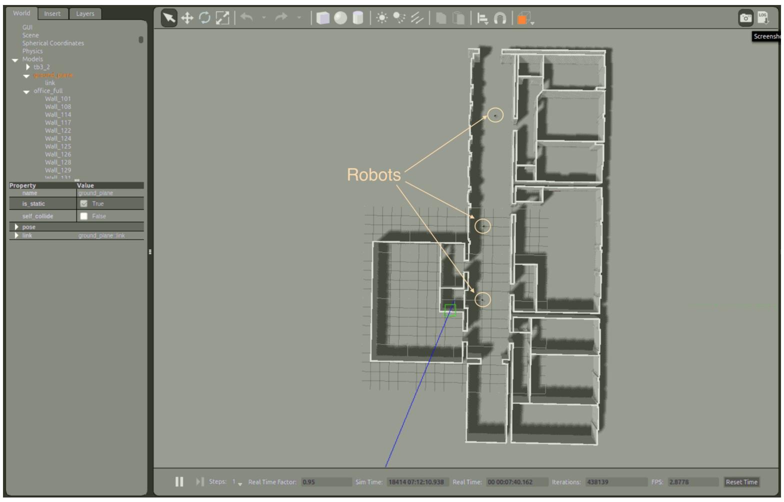


Figure 2.7: 3D Modeling of Indoor Environment in Gazebo

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

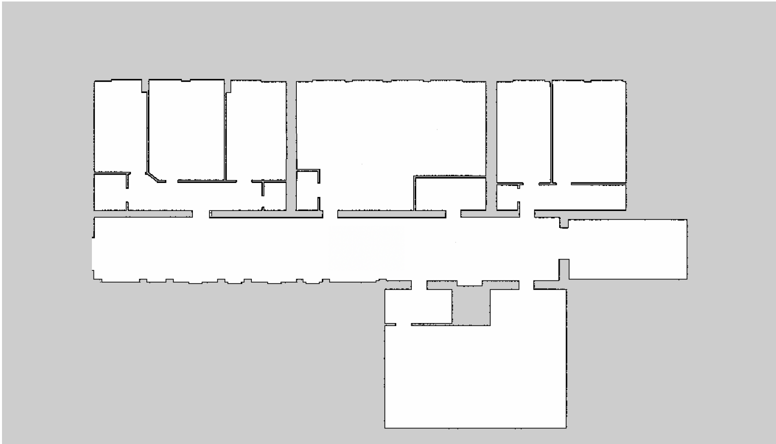


Figure 2.8: Occupancy Grid Map (OGM). White presents the free area in which robots can move, black presents the occupied area in which robots can not move, and gray is the unknown area.

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

This chapter introduces the important concepts used in the task scheduling system. Chapter 3.1 introduces the architecture of this multi-robot system. Chapter 3.2 describes the indoor office environment as well as how robot gather room occupation information. Chapter 3.3 explains the definition of task as well as its composition and decomposition. Chapter 3.4 introduces the multi-robot task scheduling approach applied in the system architecture.

3.1 Architecture Design

The architecture of the system consist 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 designate the robot 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 measurement result. 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 schedule tasks to robots.

- **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 centralized pool and decides when and which task the robot should run. The robot activity module run tasks in local task queue when receives decision from execute module and interacts with environment and its 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 robot move to desired positions using the navigation stack. Its advantages include optionally performing recovery behaviors when the robot perceives itself as stuck[MOV].

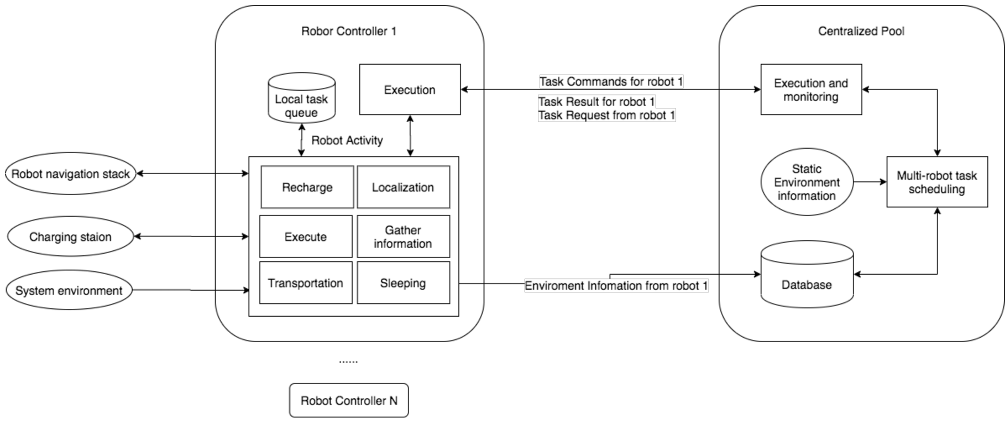


Figure 3.1: Multi-robot Task Scheduling Architecture.

3.2 Environment and Gather Information Approach

The goal of this project is to schedule tasks to robots based on the information gathered about room occupation. However, current robot sensing technology, including robots equipped with external sensors, is not good enough to gather information of whole office environment satisfactorily. For example, compared with office environment, the field within sensing range is rather narrow. Therefore, use of external environment information sources is essential to bridge the local knowledge gap. In this project, since distributed IoT network cost higher than robot and its network connections consume much energy, the fixed sensors capable of short-distance communication are installed in the environment. They can share their environment information with robots, thus robots don't need to equip with numerous sensors [PNK+15]. Robots interact with sensors while moving in the environment and report this environment information to centralized pool. The details about centralized pool storing and studying from environment information are discussed in section 4.3.

3.3 Tasks and Task Composition and Decomposition Approach

3.3.1 Task Specification

Robots should navigation tasks in order to achieve the overall system goals: gather information and environmental information continuously for a long time and schedule tasks to robots based on the environmental information. Therefore, three task names are defined: "gather information task" asks a robot gathers environment information from sensors, "navigation task" asks a robot moves to a point and "charging task" asks robot to refill its battery at charging station. The task specifications are stored in "task table" in database (Table 4.7).

3.3.2 Task Composition and Decomposition

Tasks can be distinguished to "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. 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 battery of robot below 10%, a charging task will be created and sent to robot.
2. When the battery of robot above 10%, firstly, "simple navigation tasks" according to the task table in database are created. Secondly, "complex navigation tasks" will be composed and one of them will be selected and sent to robot. To ensure consistency, a "complex task" composed by only one "simple task" is allowed.
3. If there are no "navigation tasks" in database or after "navigation tasks scheduling", the cost of tasks exceeds the threshold, a "gather environment task" will be created and sent to robot.

3.4.1 Execute Task

As discussed in Chapter 3.4, one of the “complex navigation tasks” should be selected for requesting robot. In order to select an “navigation task”, the decision variables and 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

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

Decision variables

- **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.

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 Environment Task

As is discussed in section 3.4, once there are no suitable tasks in centralized pool, the task scheduling module should create a "gather environment information task" to gather more measurement results and further more 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.

W: Weight
n: Number of doors on trajectory

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

Decision variables

- **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 robot to the front of the door. Equation 3.2 can be used to calculate battery consumption.

3.4.3 Charging Task

As is discussed in section 3.4, once a robot sends task request to the centralized pool, the centralized pool should figure out whether this robot need charging, if yes it should create a "charging task" for requesting 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.

$$\begin{aligned}&\text{W: Weight}\\&\text{Cost}_{\text{charging station}} = \frac{W_{\text{battery}} \times \text{battery consumtion}}{n} + W_{\text{time}} \times T_{\text{remain}}\end{aligned} \tag{3.4}$$

Decision variables

- **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.

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 5.2) 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).

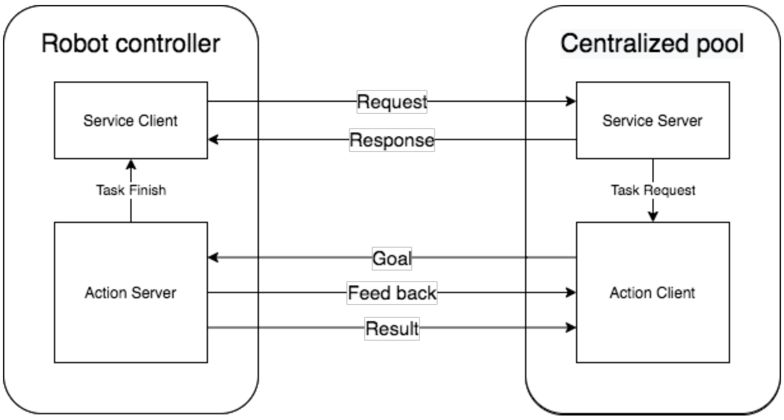


Figure 4.1: Communication between Robot and Centralized Pool

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 5.2).
- **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

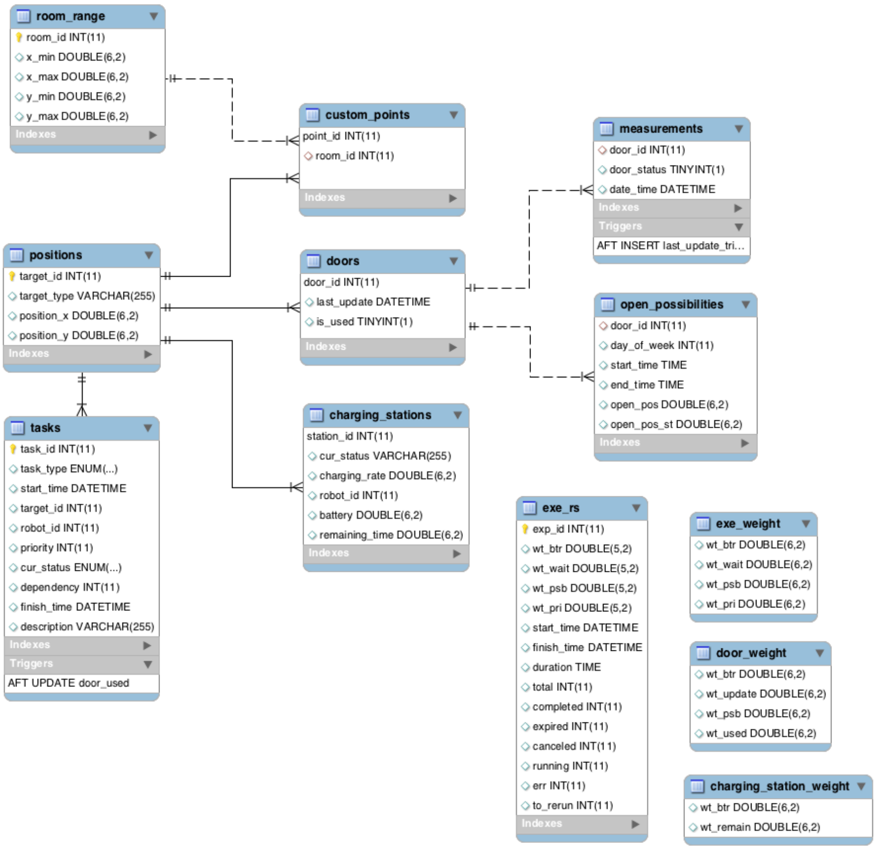


Figure 4.2: Database Entity Relationship Diagram

(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

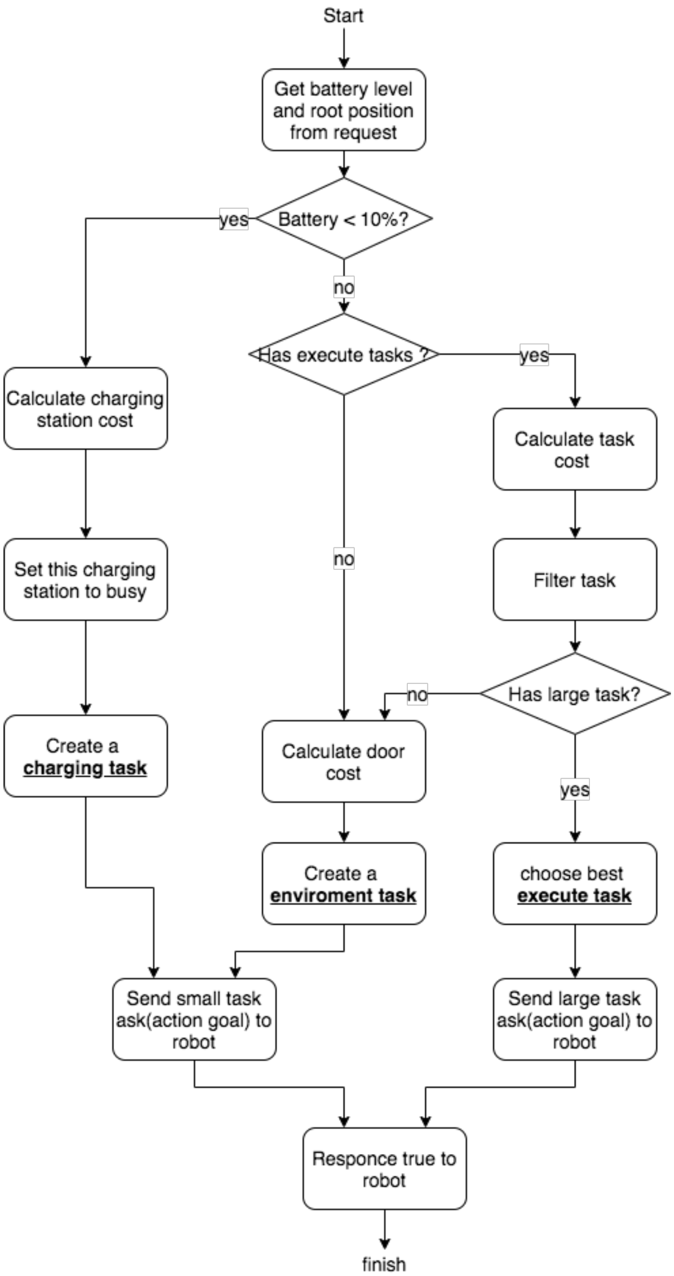


Figure 4.3: Centralized Pool Task Allocation

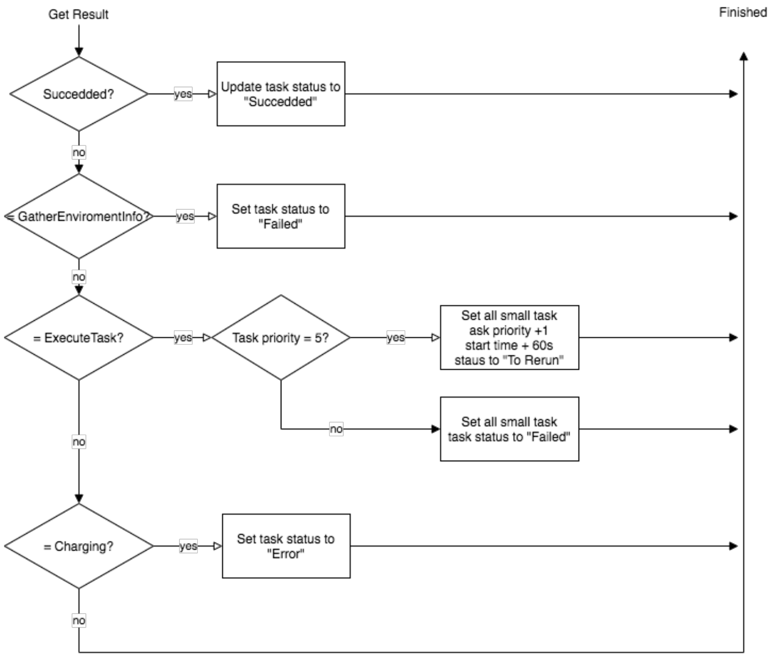


Figure 4.4: Centralized Pool Handle Task Result

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.

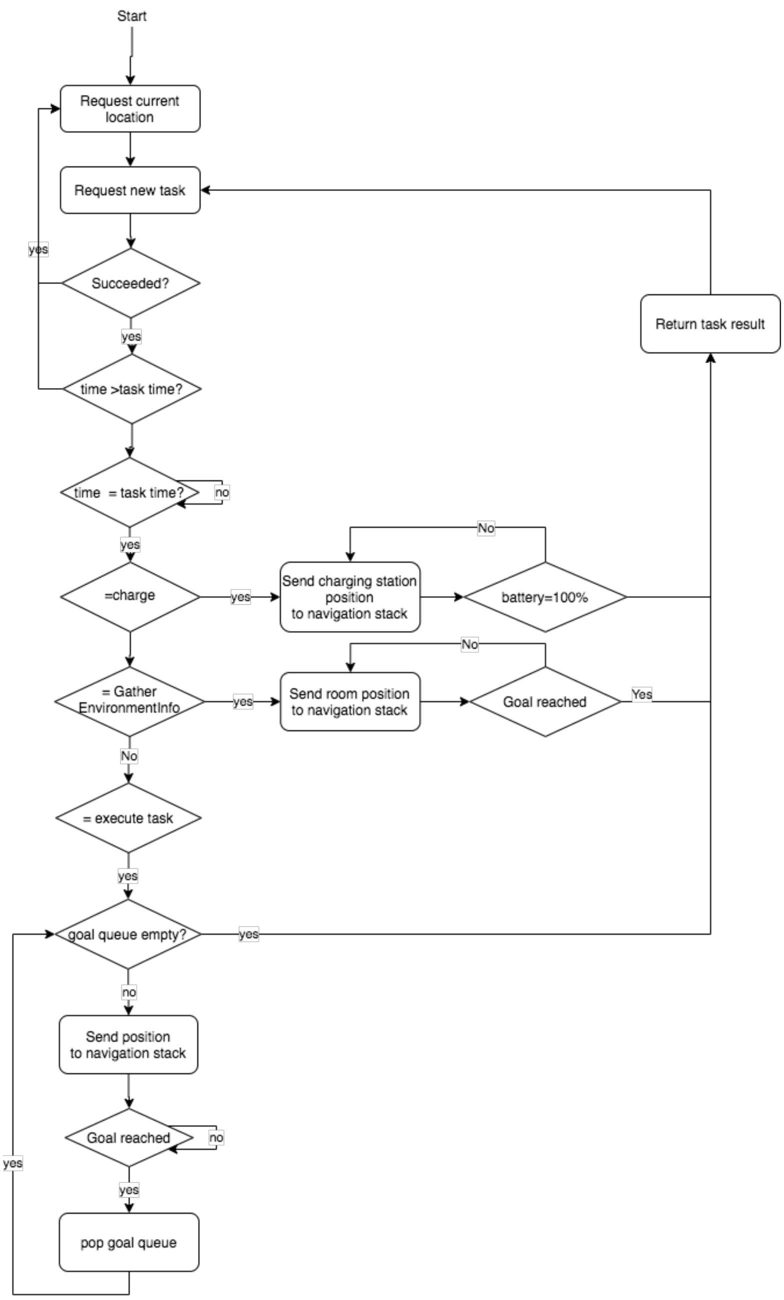


Figure 4.5: Robot Process Task

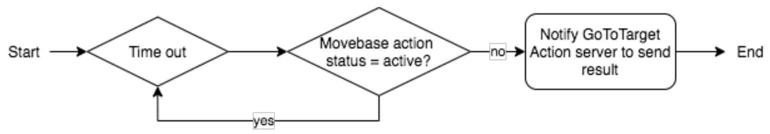


Figure 4.6: Robot Timer

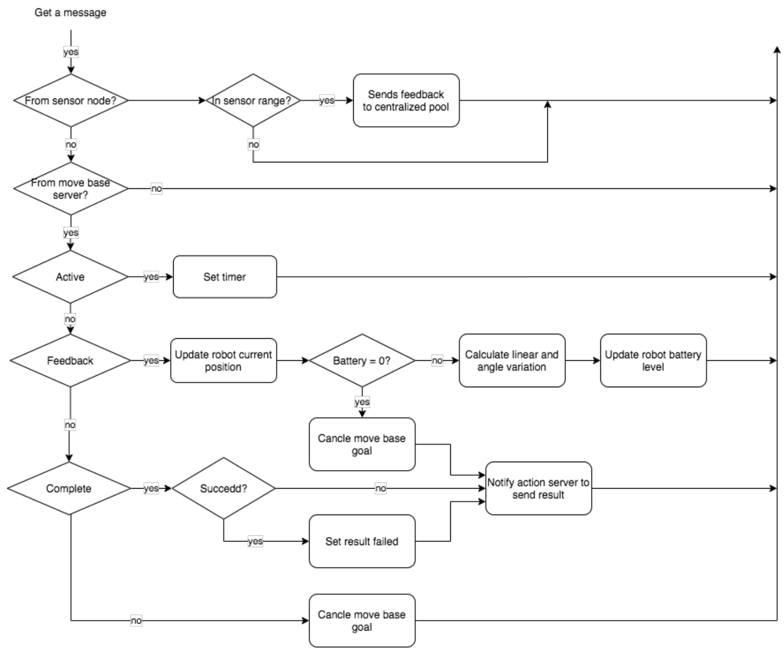


Figure 4.7: Robot Handle Message

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".

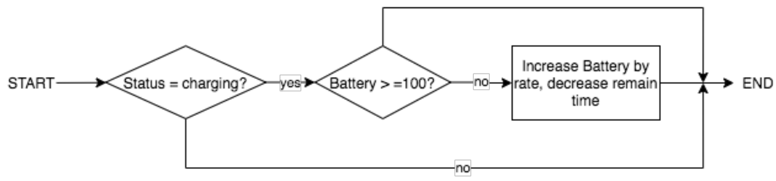


Figure 4.8: Charging Station Scheduled Charging Event in Database

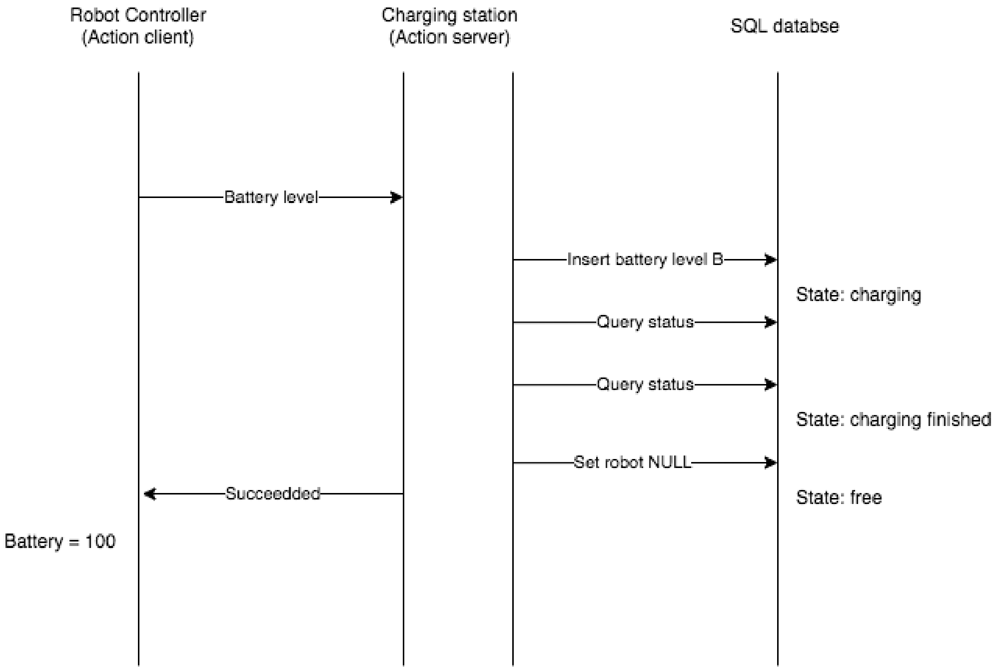


Figure 4.9: Charging Station Message

Chapter 5

Evaluation

This chapter introduces the experiments performed on the implementation. Chapter 5.3 presents precondition of experiments. Chapter 5.2 introduces the simulated sensors. Chapter 5.4 presents the experiments in the case that the centralized pool assign only "gather information tasks" to robots. Chapter 5.5 presents the experiment in the case that centralized pool assign only "navigation tasks" to robots.

5.1 Office Environment Simulation

An office environment is 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 5.2).

Each room is assumed to have one or multiple doors and each door is attached with a sensor. The sensors are simulated using ROS door simulator node (Chapter 5.2), which broadcasts positions and door status periodically. There is no robot limitation in these rooms, as long as the door of the room is opened, it can be entered by any robot. There are no dynamic obstacles in this environment. Before each experiment, tasks are stored in table. The robots are expected to move to the positions where the tasks are located and complete the tasks. In this office environment, it is assumed that the people have different working schedule, which cause the room occupied and unoccupied regularly.

Robot has a rechargeable battery, and its level drops as robot moves and rotates. Therefore, charging stations are designed in the corridor. The positions of charging stations are used by ROS charging station nodes. For details please refer to Chapter4.3.3.

5.2 Sensor Simulation

Sensor Message Publish In this project, a sensor simulator node is used to publish measurement results. The door simulator publishes one message for each door periodi-

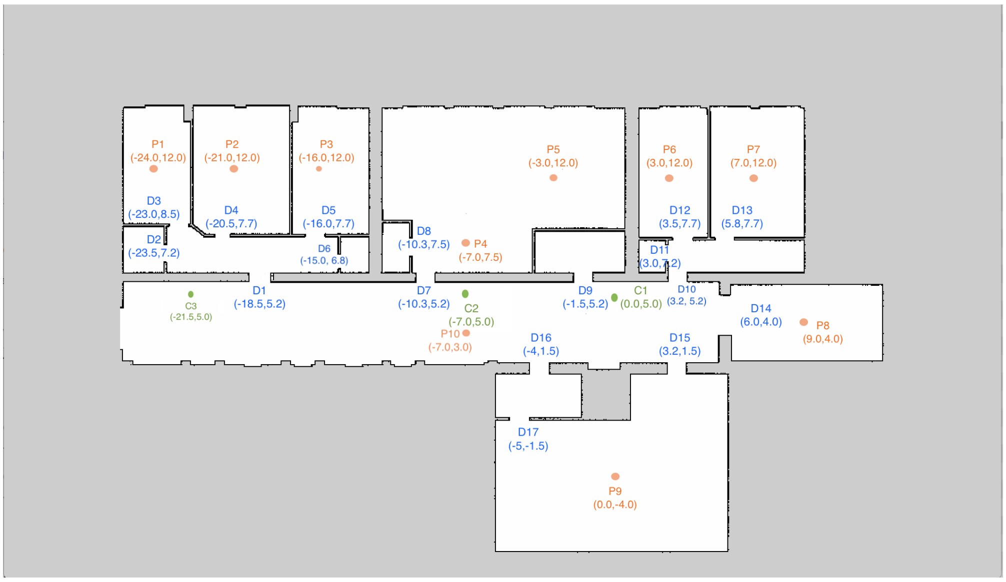


Figure 5.1: Experiment Map. C1-C3 are charging stations and D1-D17 are doors.

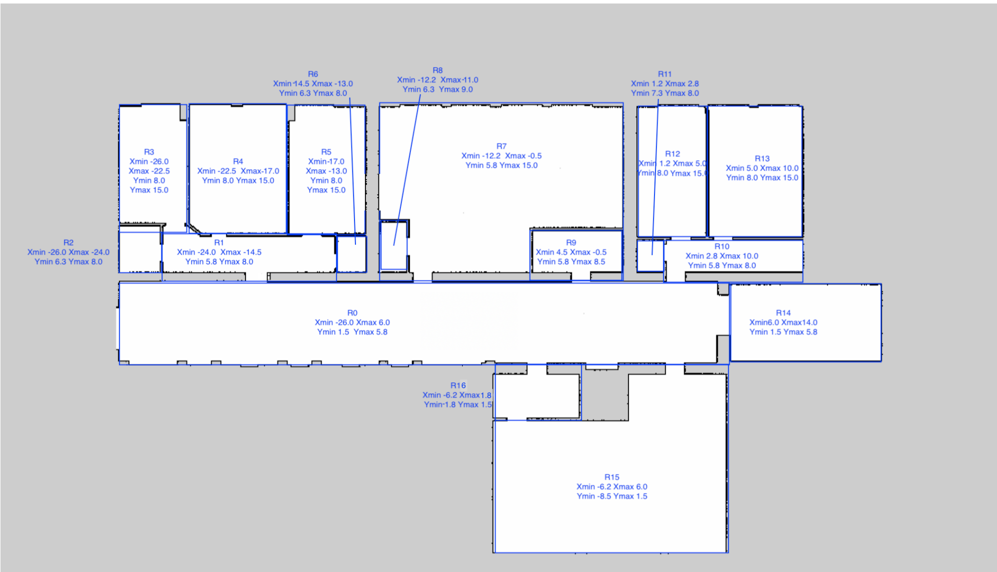


Figure 5.2: Room division. The environment is divided into regions that represent rooms in the facility (Figure 5.2). If the coordinate of a point is in a region, it can be judged that the point is located in the corresponding room.

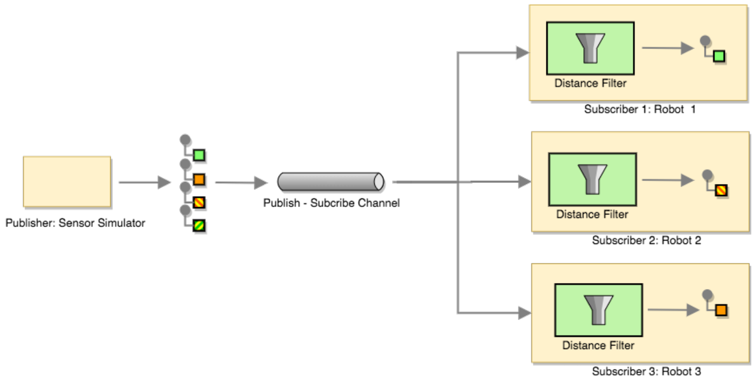


Figure 5.3: Sensor Simulator

cally to a ROS topic "sensor data". The message contains door id, sensor position and door status(Table 4.1.1). The door status are created according to open possibilities table (Table 4.9).

Generate Measuring Data For example, on monday (day of week is 2) at simulation time "10:30:00" (on time slot 10:00:00-10:59:59), in 80% probability a "door open" is generated and in 20% a "door closed" is generated (Initialized Open Possibility is 0.80).

Sensor Message Subscribe The process of sensor simulation is shown in Figure 5.2. 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 at the same time. Their distance filters filter sensor messages with a position outside the communication range and keep sensor messages within the communication range. With this process, the sensor simulator sends the instant measurement result to robots within the communication range. However, if the system is applied to the real world, instead of sending instant measurement results, the real-world sensor could send a record with history measurement.

5.3 Experiment Setup

There following are some preparation before performing experiments.

1. **Create a launch file for simulation** Create a launch file. Firstly this file loaded a gazebo world model with an office environment (Figure 5.4). In this environment, the start time should set to "2020-06-01 9:00:00". Secondly, it loaded three robot

models as well as robot state publishers to simulates robot activity such as LDR sensor measurement.

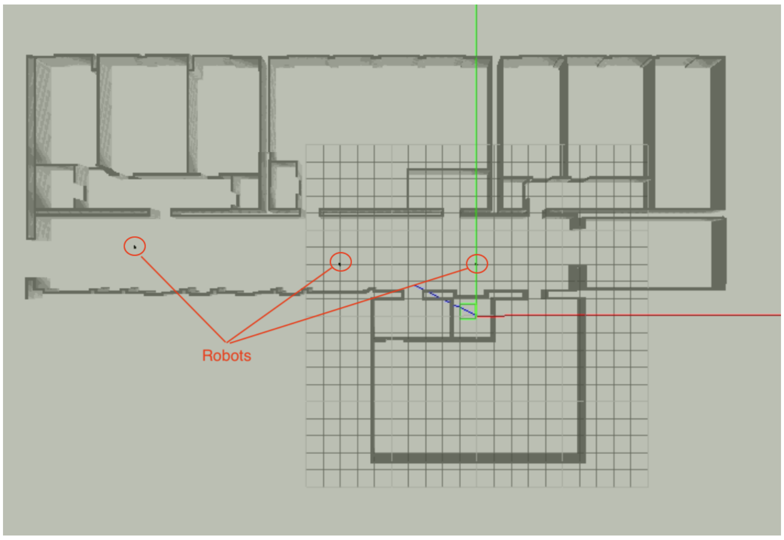


Figure 5.4: Gazebo Simulation

2. **Create a map using SLAM** In stall and execute SLAM packages, use one simulated robot explore the spaces in 3D office environment model. The detailed steps of executing virtual SLAM with TurtleBot3 are shown on the website [T3S]. The map 5.1 The map created by SLAM is shown in Figure 5.1.

3. **Create a launch file for navigation stack.** Create the second launch file. This file started the navigation stack (move_base node). It also started map server to load the map (Figure 5.1) created by SLAM[T3S] and open the GUI of Rviz.

4. **Create a launch file for ROS nodes.** Create the third launch file to start ROS node "centralized pool","robot controller", "sensor simulator" and "charging station".

5. **Configure MySQL database.** Start MySQL database server on the computer and start the charging event in the database.

6. **Design the database script to gather information experiment.** The script of gather information experiment is shown in Figure 5.5. This script had the following advantages. The first advantage was that this script prevented the initial position from affecting the experiment results. At the beginning of each experiment, all robots started from their initial position, which is the charging station. The second advantage was that the robot would charge after each experiment. This script ensures that the robot would not stop moving due to a lack of energy

during the experiment. The third advantage was that each robot gathered the information for time T in each experiment. Also, after an experiment with T set to 15 minutes, about 200 measurement data and 40 task results would be gathered and stored in the database.

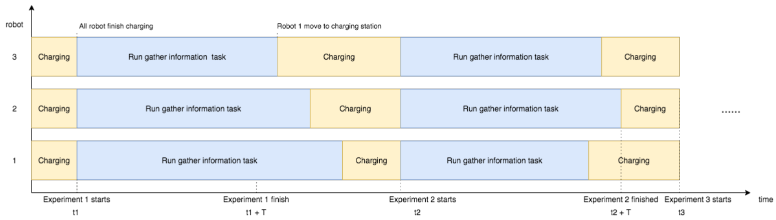


Figure 5.5: The script of gather information experiment.

7. **Design the database script to navigation experiment.** The script of gather information experiment is shown in Figure 5.6. This script had the following advantages. The first advantage was that this script prevented the initial position from affecting the experiment results. At the beginning of each experiment, all robots started from their initial position, which was the charging station. The second advantage was that the robot would charge after each experiment. This script ensured that the robot would not stop moving due to a lack of energy during the experiment. The third feature is that 15 navigation tasks with equal time intervals in each experiment will be generated. The number of successful tasks and the completion time of all tasks will be used to evaluate the performance of task scheduling. Table 5.3 shows "task table" in database. At this point, robots finished task 4-21 in experiment 1 and finished 21-39 in experiment 2.

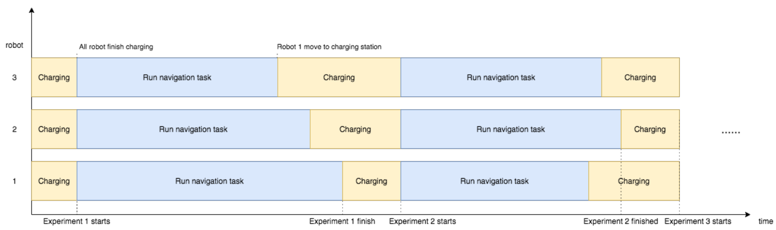


Figure 5.6: The script of navigation experiment.

8. **Connect the corresponding database script to the centralized pool.** When the centralized pool found that the current experiment should end, it should run the database script to automatically start the next experiment.

9. **Set up experiment condition.** The weight values should be import to "experiment result" in database

10. **Run the launch file.** After setting up, the user should run the three launch file. The experiment would keep running until all entities in "experiment result" table was filled.

5.4 Gather Information Task Experiment Result

5.4.1 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.

Analysis design The experiment start time was when all robot finished charging and started request task (Figure 5.5). The experiment duration is a constant value T. The experiment finish time $T_{\text{finish time}} = T_{\text{start time}} + T$. Some important factors are evaluated.

- **Last Update.** The "last update" factor is the time difference between experiment start time and minimal value in "last update" column in door table (Table 4.10) when an experiment finished. For example, "last update" factor in experiment 1 (Figure 5.7) is "00:02:57". It means that the door in the worst case not be measured since 2 minutes 57 seconds after experiment start.
- **Average Update Interval** The "Average update interval" means the average interval of door update. For example, "Average Update Interval" factor in experiment 1 (Figure 5.7) is "00:01:00". It means that on average, every door is updated every minute.
- **Succeeded Task.** The "Number of task" factor means the number of succeeded "gather information" task.

Experiment Result Figure 5.7 represent the experiment result.

Experiment Analysis As shown in the experiment result, all “Minimal Last Update” value is from 1 min to 5 min, which is much less than the experiment duration (10 min). It means some doors were not timely updated information. One possible reason is that the velocity of the robot is 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 is that the robot’s route is partially duplicated. For example, when the system started, robot 1 was in charging station 1 and got a task to door 3, while robot 2 was in charging station 2 and got a task to door 4. As shown in Figure 5.1, 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.

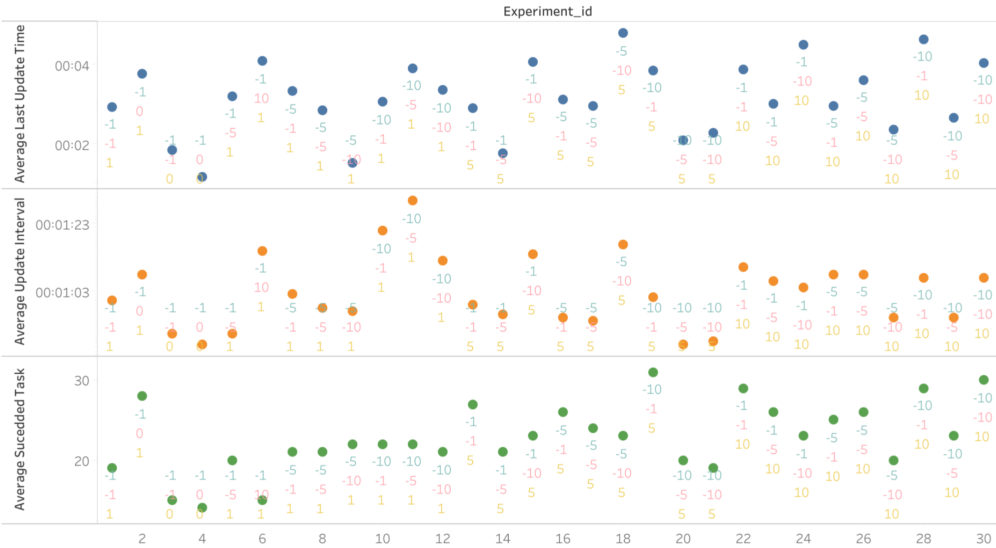


Figure 5.7: Use enumeration method to find best weight combination. The marks are labeled by weight combinations (energy consumption, waiting time, probability, priority)
Green: W_{battery} Pink: $W_{\text{probability}}$ Yellow: W_{update}

5.4.2 Use Analysis to Find the Best Weight Combinations

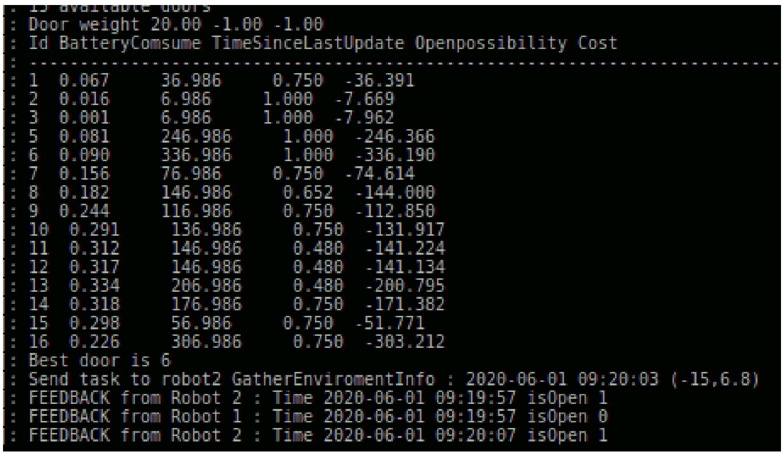
Experiment Introduction As is discussed in the last experiment (Chapter 5.4.1), the experiment time was too short to allow the robot to explore each door. Therefore, the experiment duration in this experiment was increased to 20 min. Firstly, we should find the best weight combination for “weight battery” 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 5.8), 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 (column 4). Therefore, 18 experiments are created with $W_{\text{battery}} = 0.1$ and $W_{\text{battery}} \in \{1, 5, 10, 15, 20, 25\}$.

Analysis design The experiment start time was when all robot finished charging and started request task (Figure 5.5). The experiment duration is a constant value T. The experiment finish time $T_{\text{finish time}} = T_{\text{start time}} + T$. Some important factors are evaluated.

- **Last update.** The "last update" factor is the time difference between experiment start time and minimal value in "last update" column in door table (Table 4.10) when an experiment finished. For example, "last update" factor in experiment 1 (Figure 5.7) is "00:02:57". It means that the door in the worst case not be measured since 2 minutes 57 seconds after experiment start.

- **Average Update Interval** The "Average update interval" means the average interval of door update. For example, "Average Update interval" factor in experiment 1 is "00:01:00". It means that on average, every door is updated every 1 minute.

- **Succeeded task.** The "Number of task" factor means the number of succeeded "gather information" task.



```
: Door weight 20.00 -1.00 -1.00
: Id BatteryComsume TimeSinceLastUpdate Openpossibility Cost
: ----------------------------------------------------------------------
: 1  0.067     36.986     0.750  -36.391
: 2  0.016     6.986     1.000  -7.669
: 3  0.001     6.986     1.000  -7.962
: 5  0.081     246.986     1.000  -246.366
: 6  0.090     336.986     1.000  -336.190
: 7  0.156     76.986     0.750  -74.614
: 8  0.182     146.986     0.652  -144.000
: 9  0.244     116.986     0.750  -112.850
: 10  0.291     136.986     0.750  -131.917
: 11  0.312     146.986     0.480  -141.224
: 12  0.317     146.986     0.480  -141.134
: 13  0.334     206.986     0.480  -200.795
: 14  0.318     176.986     0.750  -171.382
: 15  0.298     56.986     0.750  -51.771
: 16  0.226     306.986     0.750  -303.212
: Best door is 6
: Send task to robot2 GatherEnviromentInfo : 2020-06-01 09:20:03 (-15,6.8)
: FEEDBACK from Robot 2 : Time 2020-06-01 09:19:57 isOpen 1
: FEEDBACK from Robot 1 : Time 2020-06-01 09:19:57 isOpen 0
: FEEDBACK from Robot 2 : Time 2020-06-01 09:20:07 isOpen 1
```

Figure 5.8: Centralized Pool Cost Table

Experiment Result The experiment results are shown in Figure 5.9 and Figure 5.10

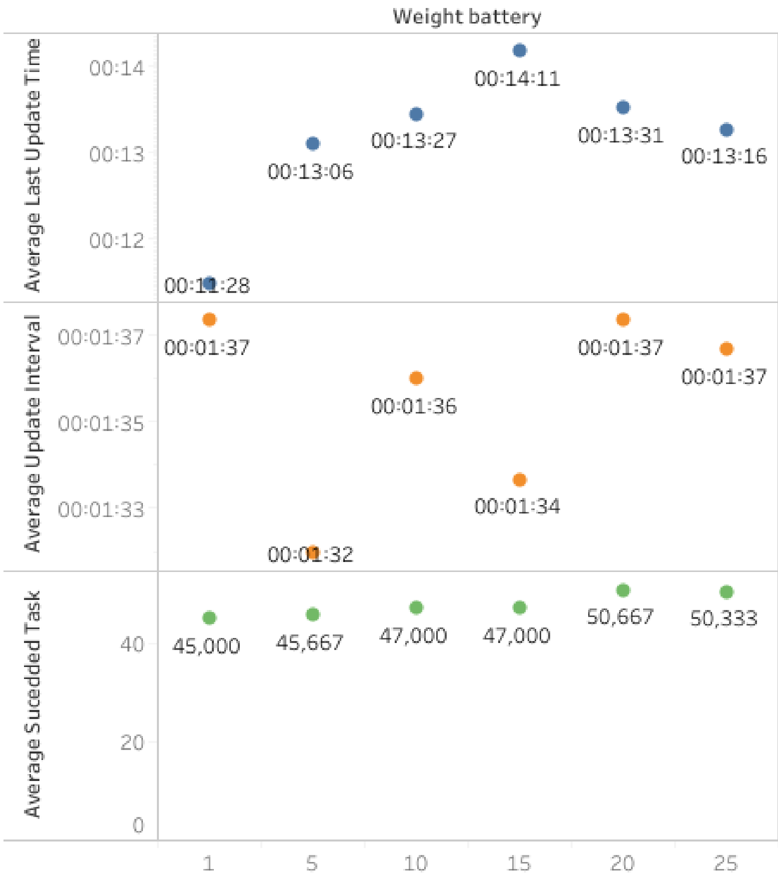


Figure 5.9: Change W_{battery} under condition $W_{\text{update}} = 0.1$ and $W_{\text{probability}} = 0$

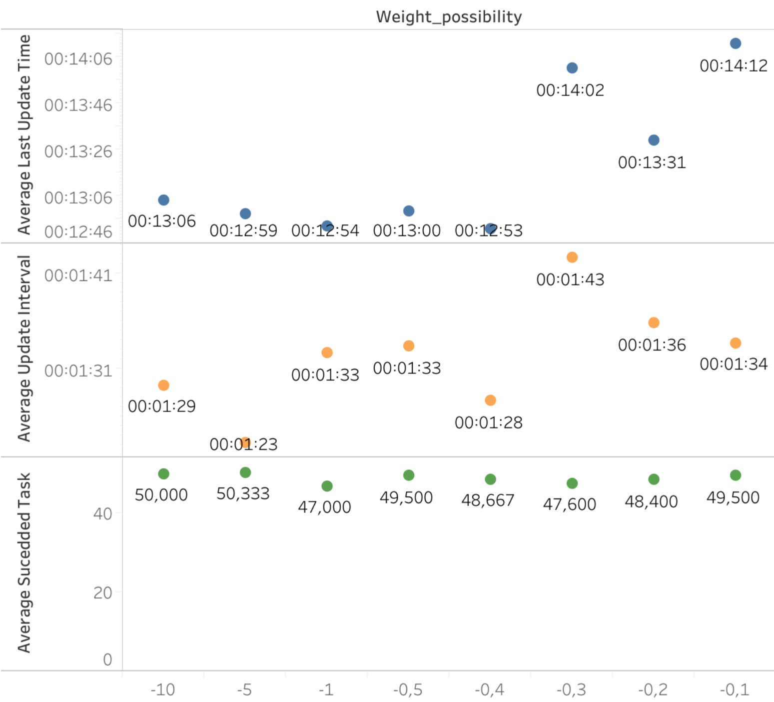


Figure 5.10: Change $W_{\text{probability}}$ under condition $W_{\text{battery}} = 15$ and $W_{\text{update}} = 0.1$

Experiment Analysis As shown in experiment result Figure 5.9, "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 5.10). 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.

5.5 Navigation Task Evaluation Experiment

5.5.1 Find the Best Weight Combinations

Experiment Introduction In addition to the cost function, the task time interval may affect experiment duration and experiment succeeded rate. Therefore, three sets of experiments are generated to find the best weight combinations.

Analysis design In each experiment, 3 robots need to finish 15 "navigation tasks" and 3 "charging tasks". The experiment start time was when all robot finished charging and started request task (Figure 5.15), and the experiment finish time was when the latest task is finished. The experiment duration was an important evaluation factor, 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" column counted the tasks ended with "Succeeded" state, the "Expired" column counted the tasks ended with "Succeeded" state, the "Failed" column counted the tasks ended with "Running", "Error", "Canceled", "To rerun" states. Table 5.2 is an example of experiment result.

Experiment Result Figures 5.11, 5.12 and 5.13 show experiment results when the task time interval is 30, 45 and 65 seconds separately. In addition, when the task time interval is 30 seconds, the task success rate is about 0.67. When the task time interval is 45 seconds, the task success rate is about 0.76. When the task time interval is 65 seconds, the task success rate is 1.0. The maximal and minimal value of y-axis are labeled.

Analysis.

1. **The relationship between task time interval and task success rate.** Firstly, it was concluded that the longer the task time interval, the higher the task success rate. This is because when the task time interval is longer, when the centralized pool select task to robot, there are fewer expired tasks that can not be selected in the task table. This leads to fewer expired tasks when an experiment is finished. Besides, in the simulation experiment, most task can be successfully finished, unless the robot collide with each other, which is time-consuming and cause task failure. Therefore, the fewer expired task results in a higher task success rate.

2. **The best weight combinations.** When an experiment result has the most successful tasks and lowest experiment duration, the corresponding task combination can be seen as one of the best weight combinations. As shown in experiment result, the best weight combinations for different task time interval are different. The best weight combination is labeled in Figure 5.11 and Figure 5.12 and listed in Table 5.1. However, the experiment result with most successful tasks and lowest experiment duration doesn't exist in 5.13. This may because the task time interval is too long, causing the robot to wait after a while before starting the next task after current task is finished. The experiment duration is mainly determined by the last task. Therefore, it is very likely that the experiment duration of many experiments are the same.

W_{battery}	$W_{\text{waiting time}}$	$W_{\text{door probability}}$	W_{priority}
15	15	15	-1
1	1	-30	-1
1	20	-20	-1
1	25	-1	-1
5	1	-5	-5
50	30	0	0

Table 5.1: Best weight combinations

It was concluded that the longer the task time interval, the higher the task success rate.

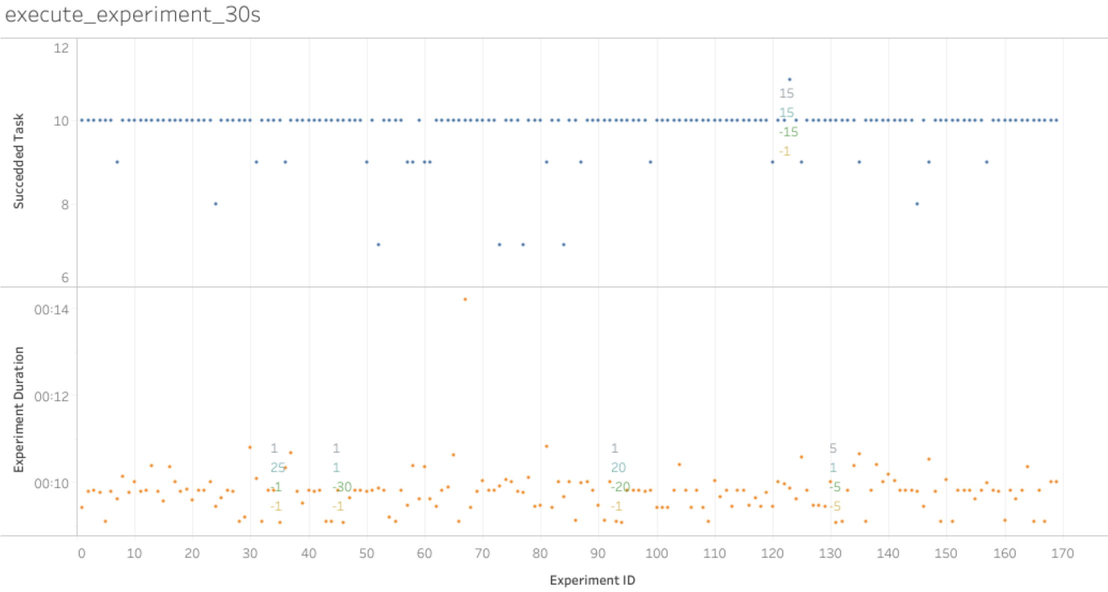


Figure 5.11: In each experiment, 15 “navigation tasks” were created with the interval of 30 seconds. The marks are labeled by weight combinations (energy consumption, waiting time, probability, priority).

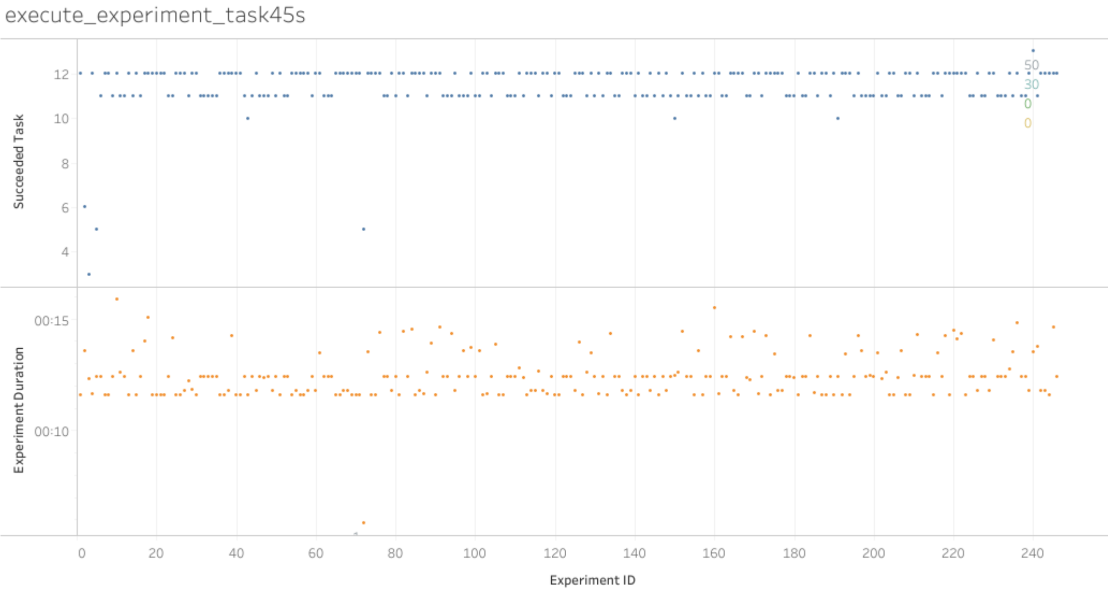


Figure 5.12: In each experiment, 15 “navigation tasks” were created with the interval of 45 seconds. The marks are labeled by weight combinations (energy consumption, waiting time, probability, priority).

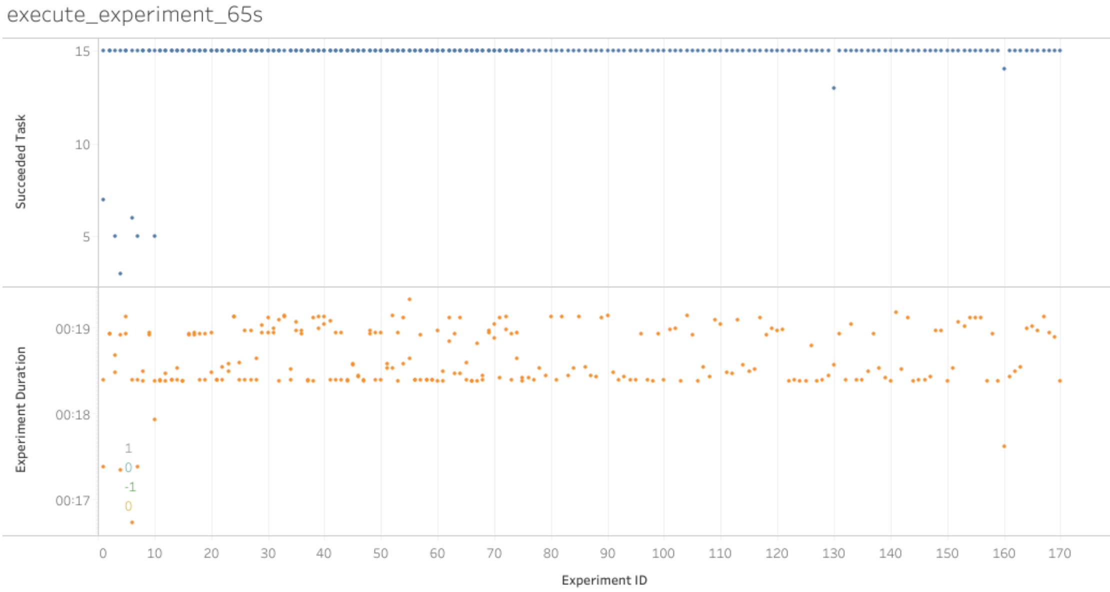


Figure 5.13: In each experiment, 15 “navigation tasks” were created with the interval of 65 seconds. The marks are labeled by weight combinations (energy consumption, waiting time, probability, priority).

5.5.2 How the Number of Robots impact task scheduling

Experiment Introduction In this set of experiments, how the number of robots affected task scheduling was evaluated. One of the best weight combination was selected: $W_{\text{battery}} = 10, W_{\text{waiting time}} = 1, W_{\text{probability}} = -1, W_{\text{priority}} = -10$.

Analysis design As is discussed in the last paragraph, in each experiment, robots need to finish 15 “navigation tasks” and 3 “charging tasks”. The experiment start time was when all robot finished charging and started request task (Figure 5.15), and the experiment finish time was when the latest task is finished. The experiment duration was an important evaluation factor, 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” column counted the tasks ended with “Succeeded” state, the “Expired” column counted the tasks ended with “Succeeded” state, the “Failed” column counted the tasks ended with “Running”, “Error”, “Canceled”, “To rerun” states. Table 5.2 is an example of experiment result.

Result The experiment result is shown in Figure 5.14.

Analysis It is concluded that as the number of robots increased, the task success rate and experiment duration increased. As discussed in Chapter 4.3. This was because the robots requested tasks from the centralized pool separately. The more robots there were, the more tasks could be finished in an experiment. On the other hand, on average, one robot could finish 7 tasks in 17 minutes and 40 seconds, two robots could finish 11 tasks in 18 minutes and 24 seconds, and 3 robots could finish 15 tasks in 18 minutes and 52 seconds. It could be inferred that the more robots, the higher the efficiency of task scheduling.

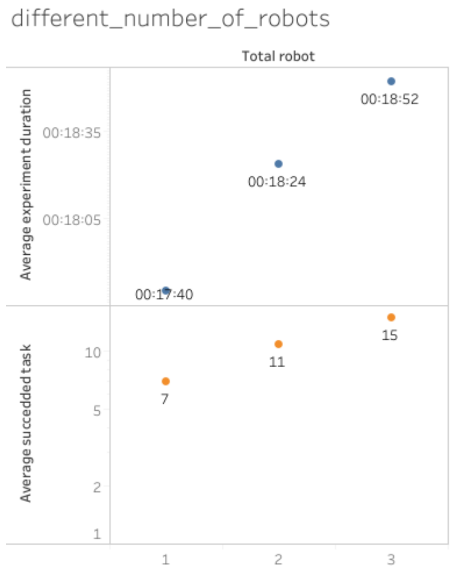


Figure 5.14: How the number of robots impacts task scheduling

5.5.3 Evaluation of how the algorithm improves navigation tasks scheduling

Experiment Introduction This set of experiments evaluated how the algorithm improves navigation task scheduling. Some experiments did not use the algorithm, and others used the designed algorithm and used the best weight combination. The results of the experiment will be analyzed to determine how much task scheduling can be improved by the designed algorithm.

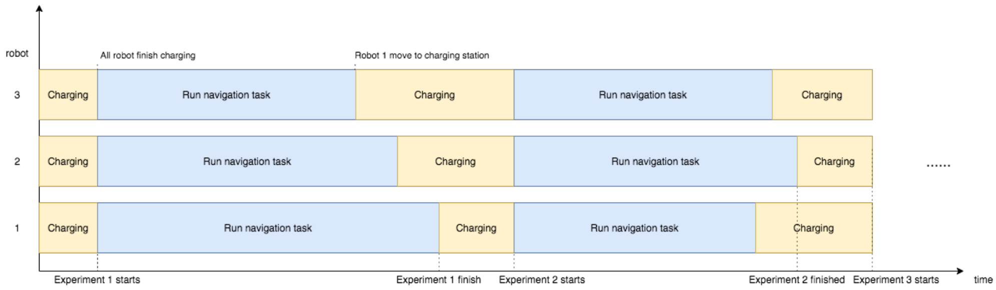


Figure 5.15: Navigation Task Experiment Timeline

Analysis design As is discussed in the last paragraph, in each experiment, robots need to finish 15 "navigation tasks" and 3 "charging tasks". The experiment start time was when all robot finished charging and started request task (Figure 5.15), and the experiment finish time was when the latest task is finished. The experiment duration was an important evaluation factor, 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" column counted the tasks ended with "Succeeded" state, the "Expired" column counted the tasks ended with "Succeeded" state, the "Failed" column counted the tasks ended with "Running", "Error", "Canceled", "To rerun" states. Table 5.2 is an example of experiment result.

Experiment Result The experiment result is shown in Figure 5.16. In the first four experiments, the algorithm was not used because only one decision variable is considered. In the last experiment, the algorithm was used with the best weight combination. This experimental result shows the difference between using the algorithm and without using the algorithm. Compared with the case that only considered the task priority, the number of successful tasks was significant increased after using the algorithm. Compared with the situation that only waiting time are considered, all tasks were finished in both situation, but the duration of the experiment decreased. The experiment 6-24 (Figure 5.17) evaluated each decision variable separately.

Experiment Analysis. The experimental results can preliminarily judge that this algorithm has a certain effect, but this effect is not particularly obvious, because the simulation experiment is limited. Following are the explanation of each decision variables:

- **Door Open probability.** The door open probability is used in the algorithm to make the robot more likely to go to an occupied room. As discussed in Chapter 6.2, the robot can enter any room, even if its door is closed. This limitation

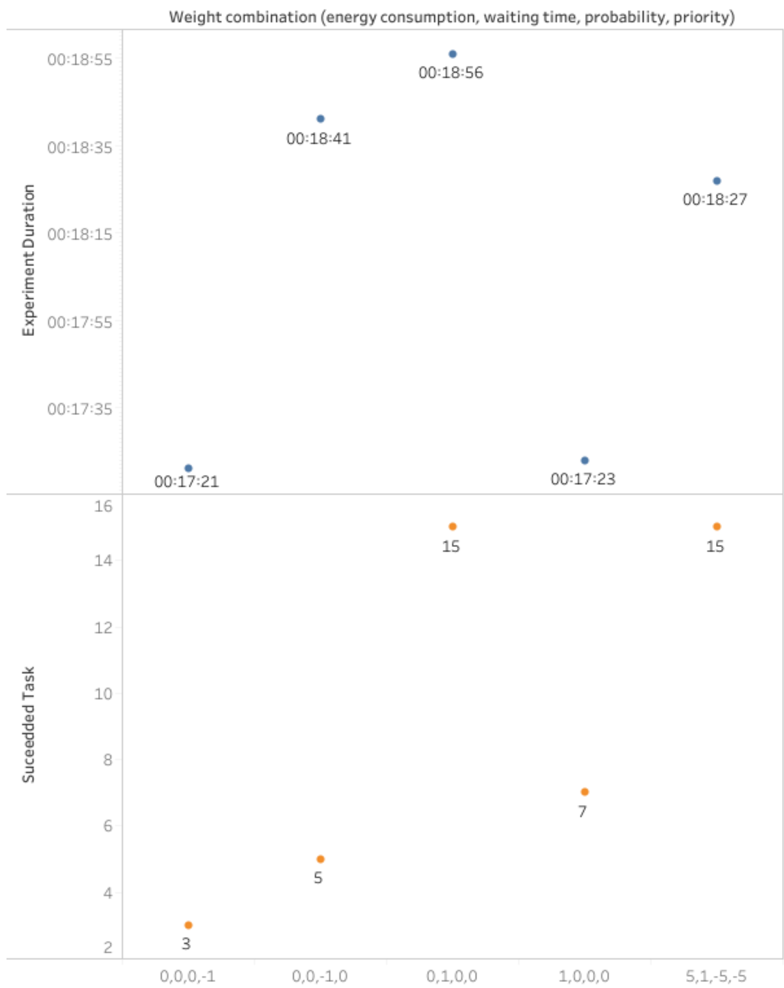


Figure 5.16: How the algorithm improves navigation task scheduling. In each experiment, 15 "navigation tasks" with the task interval of 65 seconds were created.

caused the room occupancy rate to not affect the number of successful tasks and the experiment duration.

- **Task Priority.** Similarly, the priority of the task is applied by the algorithm in order to allow the higher priority tasks to be executed first. However, the task priority had little effect on the two evaluation criteria of the experiment: the number of successful tasks and the experiment duration. If further experiments are conducted, such as evaluating the completion of high-priority tasks, then how the priority affects task scheduling can be evaluated.
- **Energy consumption.** The energy consumption of the robot is considered to allow low-energy tasks to be executed first. But energy consumption had little effect on the two evaluation criteria of the experiment: the number of successful tasks and the experiment duration. As discussed in Chapter 6.2, the energy consumption of robots in real experiments may be accurately estimated. Then further experiments can be conducted. For example, evaluating the energy consumption of each experiment, then how energy consumption affects task scheduling can be evaluated.
- **Waiting Time.** The waiting time of the robot is used in the algorithm to let the earliest tasks be executed first. As shown in Figure 5.16, the waiting time significantly optimizes task scheduling. In detail, after considering this decision variable, more tasks were successfully finished, and the experiment duration was decreased, in other words, the time required to complete all tasks was reduced. Therefore, the waiting time is an indispensable decision variable in the algorithm.

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 experiment result.

Task ID	Task Name	Start Time	Target ID	Robot ID	Priority	Task Status	Dependency	Finish Time	Description
1	Charging	NULL	18	1	5	Succedded	0	2020-06-02 11:03:12	Succeeded
2	Charging	NULL	19	2	5	Succedded	0	2020-06-02 11:03:12	Succeeded
3	Charging	NULL	20	3	5	Succedded	0	2020-06-02 11:03:11	Succeeded
4	Navigation	2020-06-02 11:04:17	21	3	2	Succedded	0	2020-06-02 11:05:16	Succeeded
5	Navigation	2020-06-02 11:05:22	22	1	2	Succedded	0	2020-06-02 11:07:29	Succeeded
6	Navigation	2020-06-02 11:06:27	23	2	2	Succedded	0	2020-06-02 11:08:06	Succeeded
7	Navigation	2020-06-02 11:07:32	24	3	2	Succedded	0	2020-06-02 11:09:25	Succeeded
8	Navigation	2020-06-02 11:08:37	25	1	2	Succedded	0	2020-06-02 11:10:46	Succeeded
9	Navigation	2020-06-02 11:09:42	26	2	2	Succedded	0	2020-06-02 11:12:42	Succeeded
10	Navigation	2020-06-02 11:10:47	27	3	2	Succedded	0	2020-06-02 11:12:56	Succeeded
11	Navigation	2020-06-02 11:11:52	28	1	2	Succedded	0	2020-06-02 11:14:13	Succeeded
12	Navigation	2020-06-02 11:12:57	29	2	2	Succedded	0	2020-06-02 11:14:20	Succeeded
13	Navigation	2020-06-02 11:14:02	30	3	2	Succedded	0	2020-06-02 11:15:36	Succeeded
14	Navigation	2020-06-02 11:15:07	21	1	2	Succedded	0	2020-06-02 11:18:07	Succeeded
15	Navigation	2020-06-02 11:16:12	22	2	2	Succedded	0	2020-06-02 11:18:49	Succeeded
16	Navigation	2020-06-02 11:17:17	23	3	2	Succedded	0	2020-06-02 11:18:58	Succeeded
17	Navigation	2020-06-02 11:18:22	24	1	2	Succedded	0	2020-06-02 11:20:14	Succeeded
18	Navigation	2020-06-02 11:19:27	25	2	2	Succedded	0	2020-06-02 11:21:36	Succeeded
19	Charging	NULL	18	1	5	Succedded	0	2020-06-02 11:23:08	Succeeded
20	Charging	NULL	19	2	5	Succedded	0	2020-06-02 11:23:08	Succeeded
21	Charging	NULL	20	3	5	Succedded	0	2020-06-02 11:23:08	Succeeded
22	Navigation	2020-06-02 11:24:13	21	3	2	Succedded	0	2020-06-02 11:25:12	Succeeded
23	Navigation	2020-06-02 11:25:18	22	2	2	Succedded	0	2020-06-02 11:26:54	Succeeded
24	Navigation	2020-06-02 11:26:23	23	1	2	Succedded	0	2020-06-02 11:28:35	Succeeded
25	Navigation	2020-06-02 11:27:28	24	3	2	Succedded	0	2020-06-02 11:29:23	Succeeded
26	Navigation	2020-06-02 11:28:33	25	2	2	Succedded	0	2020-06-02 11:30:41	Succeeded
27	Navigation	2020-06-02 11:29:38	26	1	2	Succedded	0	2020-06-02 11:32:37	Succeeded
28	Navigation	2020-06-02 11:30:43	27	3	2	Succedded	0	2020-06-02 11:32:54	Succeeded
29	Navigation	2020-06-02 11:31:48	28	2	2	Succedded	0	2020-06-02 11:34:09	Succeeded
30	Navigation	2020-06-02 11:32:53	29	1	2	Succedded	0	2020-06-02 11:34:15	Succeeded
31	Navigation	2020-06-02 11:33:58	30	3	2	Succedded	0	2020-06-02 11:35:32	Succeeded
32	Navigation	2020-06-02 11:35:03	21	2	2	Succedded	0	2020-06-02 11:38:03	Succeeded
33	Navigation	2020-06-02 11:36:08	22	1	2	Succedded	0	2020-06-02 11:38:45	Succeeded
34	Navigation	2020-06-02 11:37:13	23	3	2	Succedded	0	2020-06-02 11:38:54	Succeeded
35	Navigation	2020-06-02 11:38:18	24	2	2	Succedded	0	2020-06-02 11:40:11	Succeeded
36	Navigation	2020-06-02 11:39:23	25	1	2	Succedded	0	2020-06-02 11:41:40	Succeeded
37	Charging	NULL	18	1	5	Succedded	0	2020-06-02 11:43:45	Succeeded
38	Charging	NULL	19	2	5	Succedded	0	2020-06-02 11:43:45	Succeeded
39	Charging	NULL	20	3	5	Succedded	0	2020-06-02 11:43:45	Succeeded

Table 5.3: Task Table

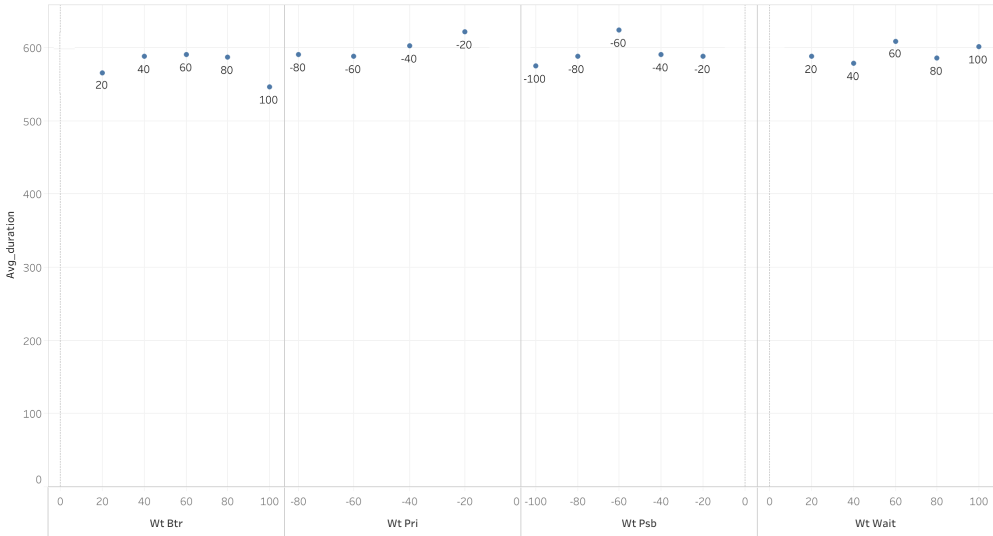


Figure 5.17: Only one decision variable changed others unchanged.

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 5.3, 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

[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.

[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.

[CV10] Coltin, B. and M. Veloso: *Mobile robot task allocation in hybrid wireless sensor networks.* In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2932–2937, 2010.

[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.

[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.

[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.

[ROS] *ROS Website.* `https://www.ros.org`. accessed 2020-10-14.

[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.

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 28, 2020
(Ort, Datum)

Xuanjiao Zhu