Grid Based Navigation and Robust Control of Multiple Networked Mobile Robots

A Dissertation Presented

by

Doug Kim

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

May 2010

Stony Brook University

The Graduate School

Doug Kim

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree,
hereby recommend acceptance of this dissertation.

Sangjin Hong, Dissertation Advisor
Associate Professor, Department of Electrical & Computer Engineering

Dmitri Donetski, Chairperson of Defense
Assistant Professor, Department of Electrical & Computer Engineering

Milutin Stanacevic,
Assistant Professor, Department of Electrical and Computer Engineering

Hongshik Ahn,
Professor, Department of Applied Mathematics and Statistics

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Grid Based Navigation and Robust Control of Multiple Networked Mobile Robots

by

Doug Kim

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2010

The field of robotics has been drawing much interest from academia and industry alike. However its applications have been limited to static models such as factory assembly lines where it performs repetitive actions that are precisely programmed. This is to minimize the human interaction and possibilities of any deviation from prescribed actions such as navigation errors in case of mobile robots. The robots being used in many different application also have very limited interactions between robots themselves. The collaboration between the robots can increase the efficiency of the job they are programmed to do as well as assisting in situations where a robot is in recoverable and non-recoverable failure mode. The collaboration between robots necessitates a reliable communication network as well as a robust control mechanism.

In this paper, first, the grid based navigation is presented where the area robots assigned to navigate is divided into grids. The grid based navigation approach simplifies the navigation algorithm with a certain degree of tolerance. The size of grid depends on the resolution of data robots are collecting. A numerical analysis of an error tolerance for an application of RF path loss data collection is studied to show the effect of grid sizes. We also take into account a range of the communication network between robots in calculating the size of grids to ensure the reliability of the communication network. The navigation algorithm is based on using laser range finders for estimating and deciding the current grid locations that the robots are in. The path planning methods for robots are also presented with precise timing as required in a multi-robot collaboration. The script-based path planning reduces risk of robots interfere with each other while navigating to their target grids. The scripts are pre-generated based on the grid configuration and the number of robots being utilized. Secondly, A novel robust robot control algorithm is presented so that when there is any failure in robots, the robots can reconfigure themselves to complete the task on hand. A robust control algorithm is devised to prevent catastrophic failure at the system level as the algorithm can determine severity of the problem and take an action to isolate and mitigate the problems caused by failed robots. The robust control algorithm also aids the navigation where the navigation scripts are not executed properly by robots. Since the scripts are pre-generated, we use the robust control algorithm to reconfigure and regenerate scripts for the robots. Finally, the simulation and experimental results are presented using the grid based navigation and robust control algorithm. The simulation shows how the robots estimate the current grid location using boundary information from the rangefinder measurement data. The simulation and experiment data also show how the navigation method responds to different building configurations and properly estimate robots' locations.

The paper finds that the grid based navigation and robust control algorithm can be applied to multiple mobile robots to enable collaboration between them while minimizing the probability of system level catastrophic failures.

Contents

List of Figures

Chapter 1

Introduction

Networked multiple robots control and communication are the interesting topic to many researchers in recent years. A 'Network robot' is defined as a robotic device communicating through the internet or LAN [1]. This has many advantages as compared with the single robot based system controlled by a human supervisor in terms of safety and efficiency. One application is the large scale warehouse [2]. Most of tasks in the warehouse are to distribute or stack goods. Since they are deterministic and repetitive, the multiple robots based system is easily applied. This will improve the job efficiency tremendously. Another possible application is the bomb disposal. Although a few robots are used to dispose bombs, the humans usually search them first at high risk. If the automated networked multiple robot system is deployed, it can dispose bombs more safely and efficiently. Moreover, this system can be applied to hospital, airport, and other public places.

The common problems in the network multiple robots system are robot synchronization and control. Suppose that the multiple robots system is designed to measure the indoor wireless measurements. The similar problem is described in [3] but the system is operated in semi-automatic mode with a single robot. When the multiple robots are used, the scheme is required to synchronize between robots generating the signal and robots measuring the signal strength. It is important to plan how the system dispatches the robots to the proper places. If they are not synchronized, the system cannot obtain the accurate measurement information when a robot has problems in physical operation or with obstacles.

The remainder of this report has 4 chapters. In chapter 2, we investigate the problems related self-localization and navigation of robots. The chapter presents also a navigation algorithm using rangefinders. Chapter 3 discusses synchronization problems and a scheme to ensure synchronization between the robots. We also present failure detection, recovery, and reconfiguration problems and our approaches to solve the problems. In chapter 4, we present an application utilizing the proposed navigation algorithm and robust control of networked multiple mobile robots. The application is to collect RF path loss data for indoor wireless access point placement. In addition, a data acquisition strategy and an approach to handle missing data based on the size of grid. Some numerical analysis is performed to present a relationship between data characteristics versus grid sizes. Finally, our contribution is summarized in chapter 5 along with future works.

Chapter 2

Map Based Indoor Robot Navigation using Range-finders

2.1 Introduction

Robot control is related to how a robot moves from one place to another place as dealing the obstacles. In navigation, it is also important to know the current orientation and position for properly executing intended tasks. When a robot navigates in the outdoor environment, the GPS is usually utilized. However, in the indoor environment, it is not guaranteed for a robot to receive the GPS data even with the indoor GPS device.

There are a number of methods for self-localization of the robots such as using RFID [4] [5] or using known landmarks as guides [6] [7]. Using RFIDs can produce highly accurate self-localization results as navigation system on robots know exactly where those RFIDs are located especially point-to-point navigation. The landmark recognition schemes can be tricky as the robots need to be trained or preprogrammed to properly identify the landmarks from the similarly shaped objects. Once robots can properly differentiate the landmarks from the noise generating object, it also can yield good accuracy navigation system. These methods however require pre-navigation preparations. When the same mobile robot system is to navigate a different site, the preparation has to be done again to custom fit the locations of the RFIDs or landmarks.

In this chapter, we propose the novel algorithm using the range finders to navigate in the indoor environment. We assume that the synchronization between multiple robots is perfect and focus on the navigation scheme as to how a robot moves exactly to the directed destination from the main server.

The remainder of this chapter has 5 sections. In Section 2.2, we present overview of the application model and background. Also, the problem is described. Section 2.3 discusses the navigation algorithm using the range finders. In Section 2.4, we investigate the effect of non-ideal parameters in the algorithm. In Section 2.7, we provide comprehensive simulation for evaluating the navigation algorithm. Finally, our contribution is summarized in Section 2.8 along with future work.

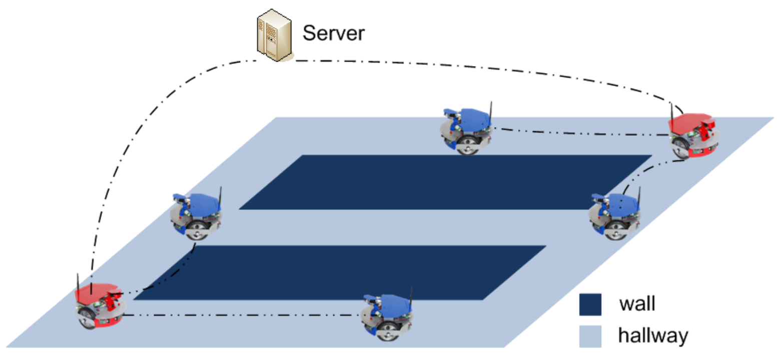


Figure 2-1: Illustration of an application model with the networked multiple robots system.

2.2 System Model and Problem Description

2.2.1 Application Model

If robots can be localized in indoor environments, there are various applications with the networked multi-robots system. Known positions of robots facilitate path planning and management of the robots. Fig. 2-1 illustrates an application model of two layer network with the multi-robot system. A set of robots act as masters to transmit path or task information from a main server to other slave robots. The coverage of a master robot can be defined by a network coverage. Slave robots move to specified destinations and execute tasks. Two layer network can easily manage robots in groups. The networked multi-robot system can be implemented in airports, factories, hospitals, and warehouses with various network topologies.

2.2.2 Map Based Navigation

In order to create the automated wireless signal measurement system by networked robots, each robot should be self-localized and move according to the received path from the server. Since the global localization system (i.e. the GPS system) is not generally available in the indoor environments, the relative position should be used. Without any map information and absolute reference, it is hard to plan the path for a robot. The robot navigation is also not trivial to implement due to imperfect mechanical components and their controllers. Thus, we divide the building into the certain size of settions. This map based approach with the grid enables us to do path planning for robots in a concise and accurate manner. The path is easily represented by the sequential list of the grids to be visited.

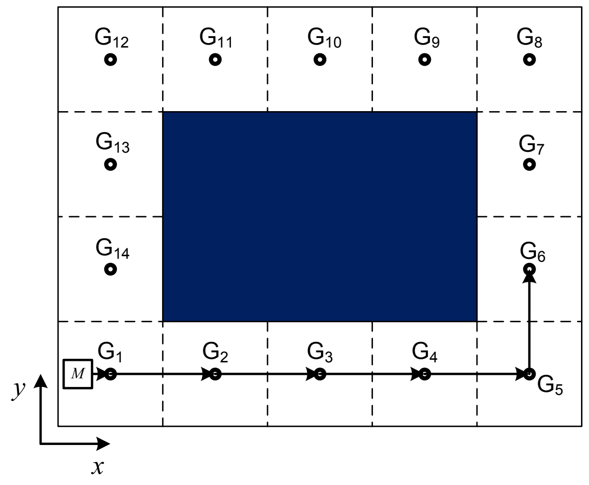


Figure 2-2: Illustration of the grid based path planning. Each grid has pre-defined coordinate point.

Fig. 2-2 shows the predefined map with the grids. When a robot moves accurately to the center of each grid, it executes a task at that point. The number of grids in one region is directly related to the accuracy of the data being collected because the data collected at the center of the grid would represent the entire grid not just the center point of the grid. In terms of the robot navigation, it enables a robot to have the relative position with the grid number. Also, it is easy for a robot to move to the next grid because it is always a straight path. The path can be planned by using the grid numbers (the path : $P_{G_1} \rightarrow P_{G_2} \rightarrow P_{G_3} \rightarrow .P_{G_4} \rightarrow P_{G_5} \rightarrow P_{G_6}$). P_{G_j} is the center point of the grid G_j and R_i denotes the robot i. We assume that the path is planned without the possibility of collisions at any time and that the robots are perfectly synchronized. Even with a path plan, the robot requires peripheral sensing devices for the self-localization so that the robot can recognize the current grid or the position.

Index	Coordinate	x_l	x_f	x_r	x_b
1	$(1, 1)$	1	7	9	1
2	$(3, 1)$	3	1	7	1
3	$(5, 1)$	5	1	5	1
4	$(7, 1)$	7	1	3	1

Table 2.1: The example of the boundary information at each grid.

The advantage of the map based navigation is that it provide a robot with the information on its surroundings. Table 2.1 shows the predetermined boundary information of grids assuming the grid size of 2×2 as in Fig. 2-2. x_l is the distance to the left boundary from the center point of the grid when the orientation is the y-axis direction. $\mathbf{X}^i$ denotes the boundary information of $\{x_l, x_f, x_r, x_b\}$ at grid i. This information helps a robot to correct its orientation or to estimate the current position. When the robot is at P_{G_2} assuming the orientation of the table is the same as the robot, the range data of the four range finders attached on the robot should be close to the data in the table (Index 2).

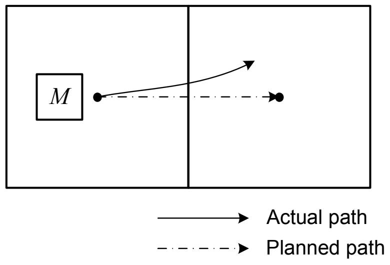


Figure 2-3: Illustration of the deviated path because of the mechanical problems.

2.2.3 Problem Description

A robot has imperfect navigation capability because of the friction on the ground as well as the inaccurate motors. These imperfect components cause the robot to miscalculate the orientation, the moving distance and the velocity. Fig. 2-3 shows the example of the deviated path because of the mechanical imperfection. When a robot moves from one point to another point, it calculates the distance and orientation first. Although the calculation for the navigation may be correct given two points, it is not guaranteed that the robot moves to the destination with the accuracy required by the system due aforementioned issues.

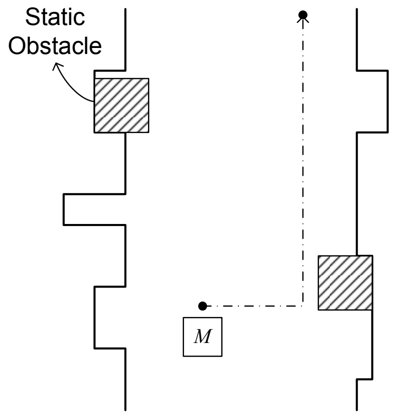


Figure 2-4: Illustration of the possible problems in the complex environment.

The other problems may also rise due to the environment such as the uneven boundary and the static or dynamic obstacles as shown in Fig. 2-4. Since a robot needs the relative reference obtained from the map information, the proper map representation in the robot system is required. The map information consists of the grid information. When a grid is defined on a map, the representation should be simple and have enough information so that the robot can be easily self-localized. However, it is not trivial to represent all uneven boundaries and static obstacles on the map in a simple format. Even when multiple robots move around the application environment, they are considered as dynamic obstacles to each other. These conditions severely influence the navigation performance because they are considered as noise to the sensing devices.

Fig. 2-5 illustrates the peripheral sensing device configuration of a robot. We attach four range finders to the each side of the robot. RF^f is the range finder at the front of the robot, RF^l at the left, RF^r at the right and RF^b at the back. Since the range finder measures the surroundings as the range with the sensing angle, we can obtain the distance to the boundaries as well as the distance to the obstacles. The obtained distances are used to localize the robot with the predefined boundary table of grids.

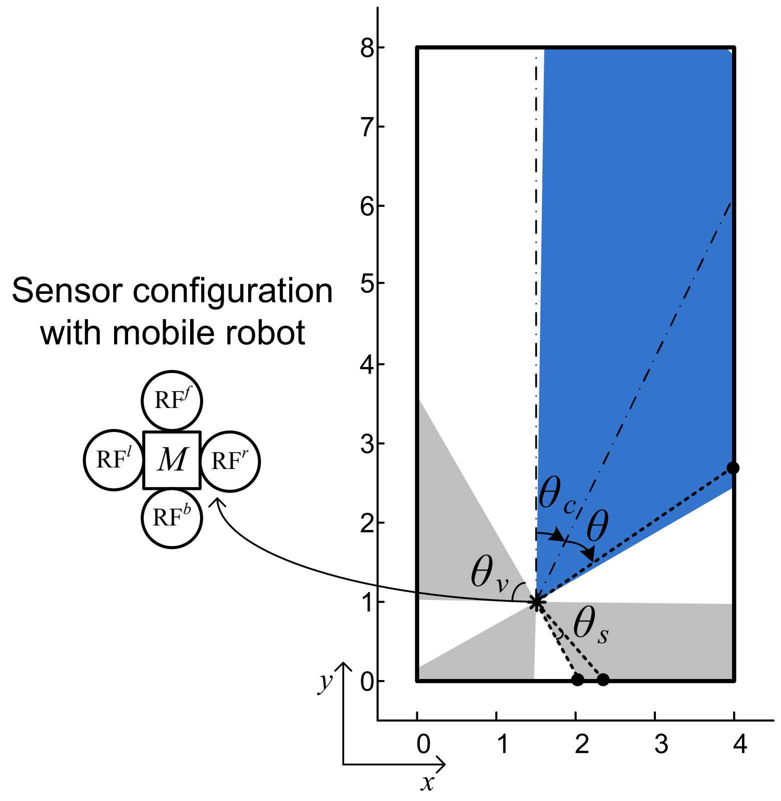


Figure 2-5: Illustration of the configuration of the robot with range finders.

2.3 Map Based Navigation Using Range Finders

2.3.1 Sensor Characterization and Measurement

Signal Characteristics

Fig. 2-6 shows sample range data plotted from the range finders. Each sensor measures the ranges with the signal speed and the traveling time. Since the signal can be corrupted by the external noise the measurement errors are unavoidable. The measurement errors produce irregular boundaries. Thus, the irregular boundaries are filtered to be smooth and straight. The estimated boundaries by minimizing the mean square error is are shown in Fig. 2-7.

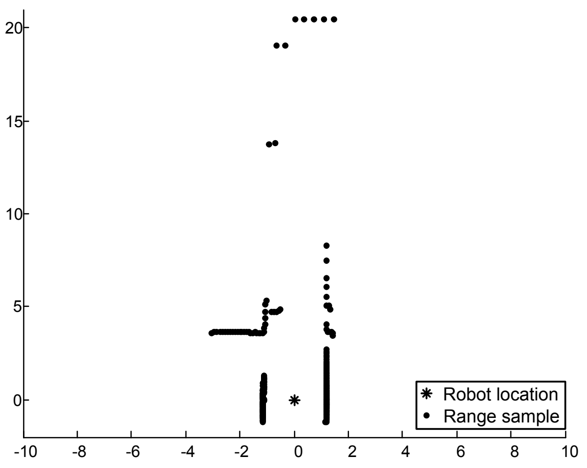


Figure 2-6: Real range samples obtained by a laser range finder.

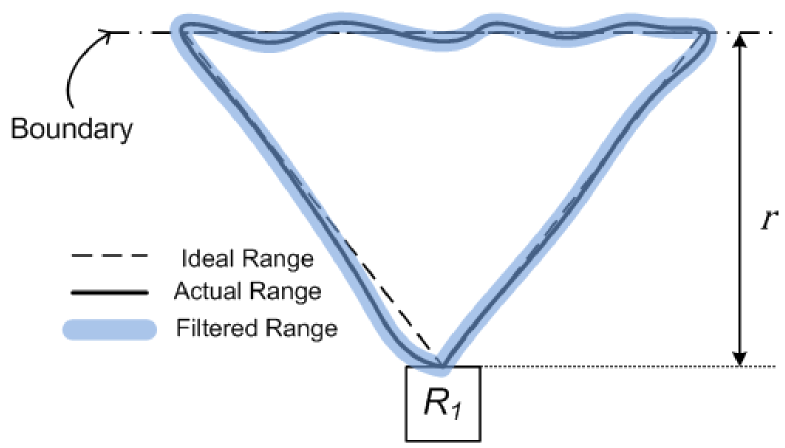


Figure 2-7: Illustration of the filtered range for detecting the boundary in the actual range image.

Surroundings Construction with Range Data

Fig. 2-8 shows the basic concept to construct the surroundings based on the range data. After the robot activates the range finders, each sensor obtains range data of shaded regions. We define that θ_v is the searching angle of the range finder and θ_s is the sampling angle. θ_v/θ_s gives the number of range samples for each range finder. θ_c denotes a robot orientation defined by a path in terms of y-axis and its range is

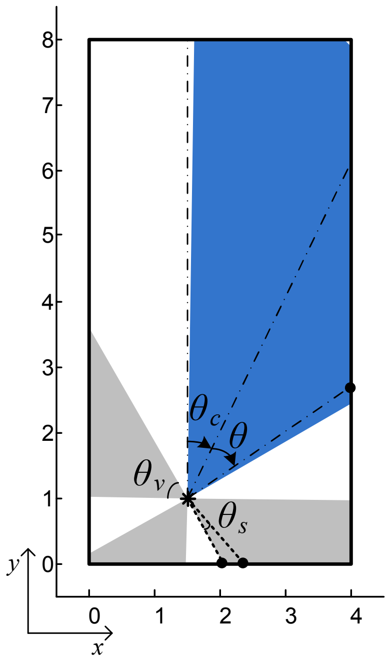


Figure 2-8: Illustration of the surroundings construction based on range data in the simple environment.

$-\pi \leq \theta_c \leq \pi$. Each sampled data is defined by.

$$(r_\theta, \theta_c + \theta),$$

where r_θ denotes the sampled range at the angle θ ($-\pi \leq \theta \leq \pi$). Since angles of range data are measured in terms of the heading of a robot, θ_c is added to represent the sampled data in the global coordinate.

Various techniques are available for line extraction with 2D range data [8]. As shown in Fig. 2-9, an extracted line is represented by

$$xcos\alpha + ysin\alpha = r, \tag{2.1}$$

where $-\pi < \alpha \leq \pi$ is the angle between the x axis and the normal of the line; $r \geq 0$ is

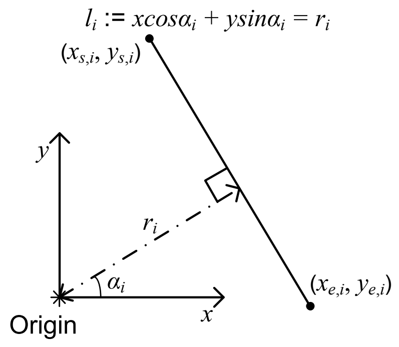


Figure 2-9: Illustration of a line in the *Polar* coordinates.

the perpendicular distance of the line to the origin; (x, y) is the *Cartesian* coordinates of a point to the line. We utilize *Split-and-Merge* algorithm for line extraction because of its superior speed and correctness.

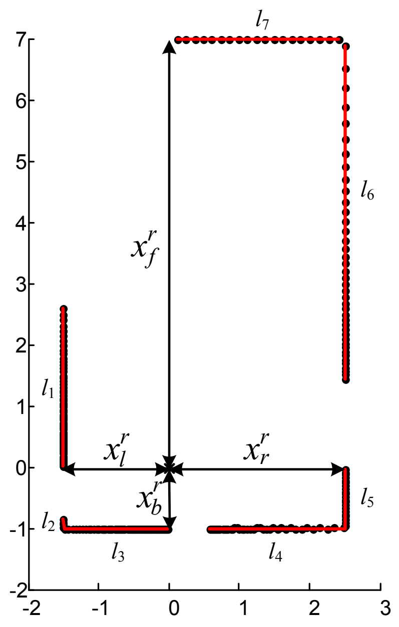


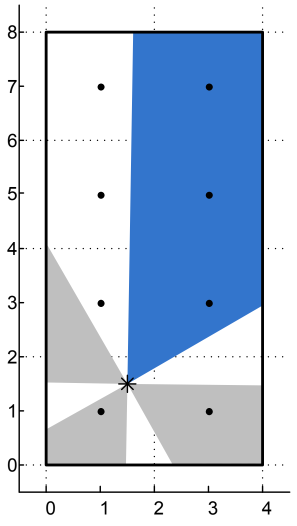
Figure 2-10: Illustration of extracted lines by *Split-and-Merge* algorithm.

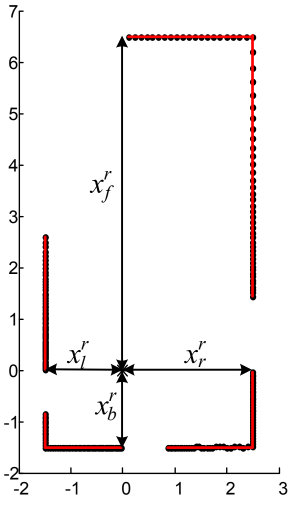
Fig. 2-10 illustrates extracted lines by Split-and-Merge algorithm. l_i denotes an extracted line i. Since we assume rectangular hallways, there are ideally four candidate borders to be detected. Angles of four candidate borders are $\alpha = \pm 0^o, \pm 90^o, \pm 180^o$

and each angle indicates which boundary lines are originated from. Perpendicular distances defining lines are distances to boundaries. For example, an angle 0^o means that a line is originated from a right side boundary and a distance r of a line having an angle 0^o is the distance to the right side boundary. Similarly, distances to other boundaries are obtained. We denote a distance to each boundary as $x_l^r, x_f^r, x_r^r, x_b^r$ to be differentiated with the distance data in the predefined table. $\mathbf{X}^r$ denotes the measured boundary information ($\{x_l^r, x_f^r, x_r^r, x_b^r\}$) of a robot.

2.3.2 Navigation with Information Matching

Localization


(a) Robot position



(b) Measured range data

Figure 2-11: Illustration of robot self-localization by using the grid table information.

Fig. 2-11 shows an example for self-localization. After range data is measured, $\mathbf{X}^r$ is obtained. If a robot is at the center of each grid, the current position is easily obtained by comparing the grid list on the table. Ideally, these values are the same as those in the predefined table of grid information. However, when a robot is at other positions beside the center of each grid, $\mathbf{X}^r$ does not match with any $\mathbf{X}^i$. Thus, the system needs to determine which $\mathbf{X}^i$ corresponds to $\mathbf{X}^r$. This is checked by the following equation.

$$|x_j^r - x_j^i| \leq s_i + s_{err}, j \in \{l, f, r, b\} \tag{2.2}$$

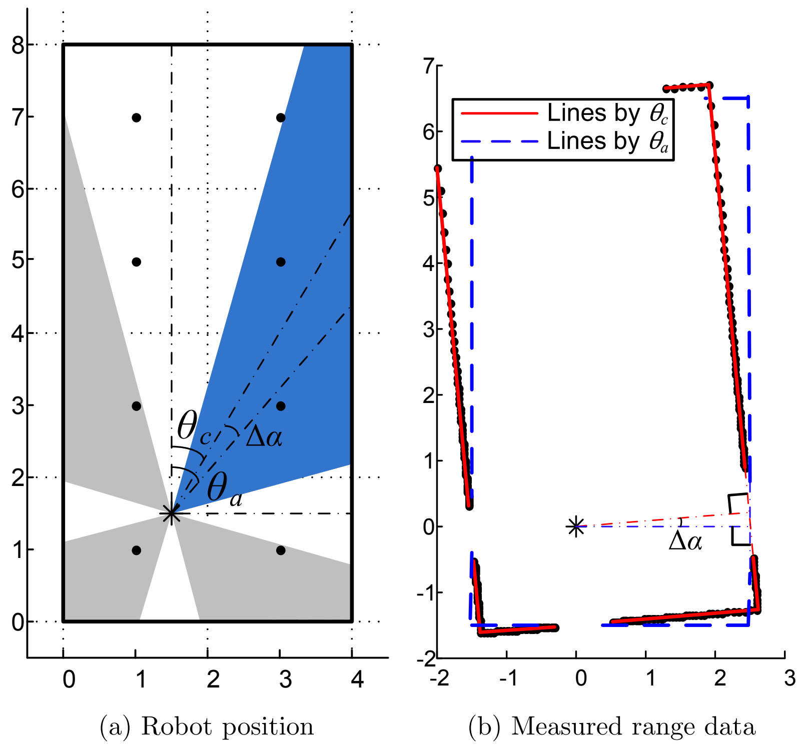
where s_i denotes the size of the grid i defined by the shortest distance between the center and the boundary of a grid and s_{err} is tolerable error caused by the uncertainty of the edge detection. Then, the location is calculated by,

$$\begin{aligned} r_x &= c_x^i + (x_l^r - x_l^i), \\ r_y &= c_y^i + (x_b^r - x_b^i), \end{aligned} \tag{2.3}$$

where (r_x, r_y) is the coordinates of a robot and (c_x^i, c_y^i) is the coordinates of the center of the grid i. Since the robot has the redundant information (*i.e.*, x_f^r, x_r^r), the location is also calculated by,

$$\begin{aligned} r_x &= c_x^i + (x_f^i - x_f^r), \\ r_y &= c_y^i + (x_r^i - x_r^r). \end{aligned} \tag{2.4}$$

Finding Orientation



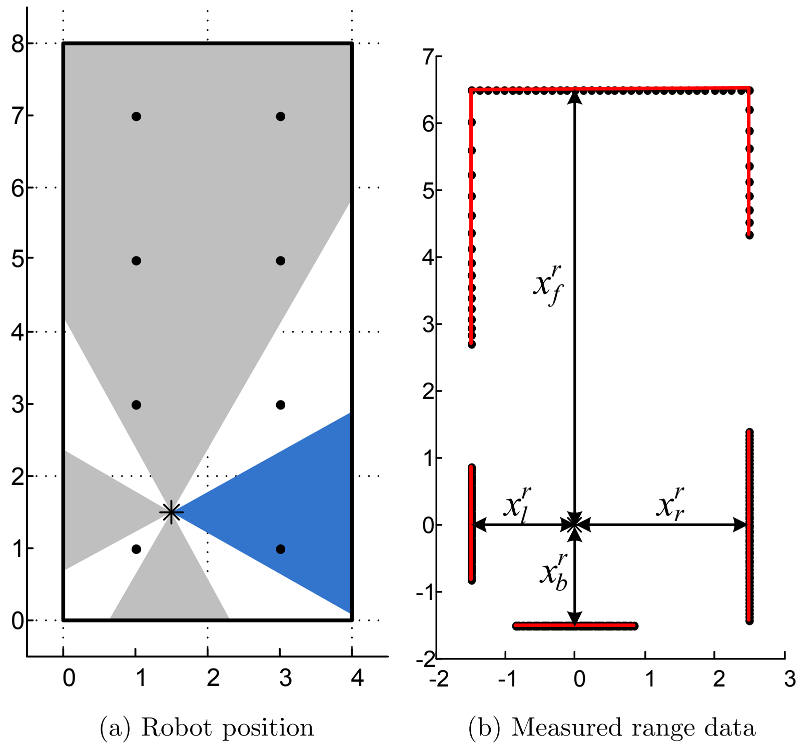
(a) Robot position (b) Measured range data

Figure 2-12: Illustration of range data when the actual orientation of a robot is different from the planned orientation of a robot.

In previous sections, we assumed that the orientation of the robot is the same as θ_c, which is the planned orientation of the robot defined by the path. However, the actual orientation of the robot can become deviated from the planned orientation of the robot because of the issues such as non-ideal mechanical components. as shown in Fig. 2-12. θ_a denotes the actual orientation of the robot. When θ_c is different from θ_a, angles of extracted lines are not the same as possible angles (i.e., $\pm 0^o, \pm 90^o, \pm 180^o$) anymore. Thus, an angle error between the actual orientation and the planned orientation is obtained by finding the minimum angle to rotate lines to be matched with possible angles. Then, the actual orientation is obtained by

$$\theta_a = \theta_c + \Delta\alpha. \tag{2.5}$$

Localization in Rotated Situation



(a) Robot position (b) Measured range data

Figure 2-13: Illustration of the robot self-localization when the robot orientation is different from the table orientation.

Fig. 2-13 illustrates the localization case that the robot orientation is different from the table orientation. The table orientation is always the the direction of y-axis. When a robot is rotated, there are four possible rotation cases to match with boundary information of the rectangular map. $\mathbf{X}^r$ is rotatable according to the robot orientation.

Fig. 2-14 illustrates possible rotated surroundings. Each image of the surround-

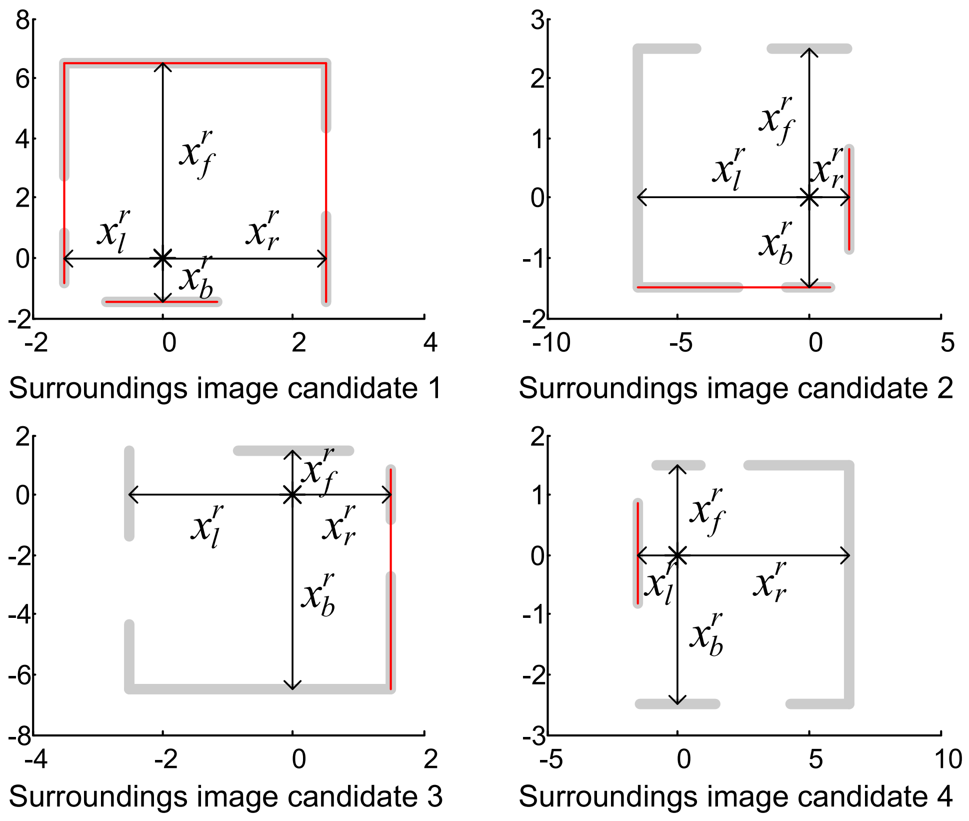


Figure 2-14: Illustration of candidates of constructed surroundings when the robot orientation is different from the table orientation.

ings has its own $\mathbf{X}^r$. The initial filtering is done by comparing with $\mathbf{X}^i$ in the table. If a candidate does not satisfy (2.2), that image of the surroundings is not considered anymore. Although some of them may be removed in the initial filtering, the robot may still have multiple candidates because of the symmetric nature of the surroundings. For example, in addition to the correct surroundings of the candidate 1, the candidate 3 is also satisfied by (2.2) because the robot is regarded as locating at $(2.5, 6.5)$. The correct $\mathbf{X}^r$ among them can be selected by finding a candidate minimizing the distance difference from the previous position. The angle information also helps to find a correct surrounding boundaries with (2.2). There are four angles associated with the four surroundings candidates (i.e., $\theta_a, \theta_a + \pi/2, \theta_a - \pi/2, \theta_a \pm \pi$). Since the robot knows the planned orientation, the angle having the minimum difference from the planned orientation is selected.

2.4 Algorithm Design and Analysis

2.4.1 T - Shape Map

Unlimited Range

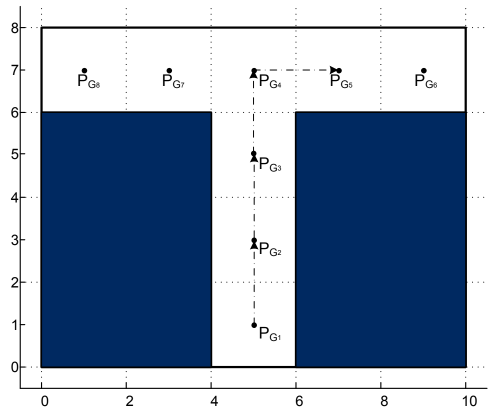
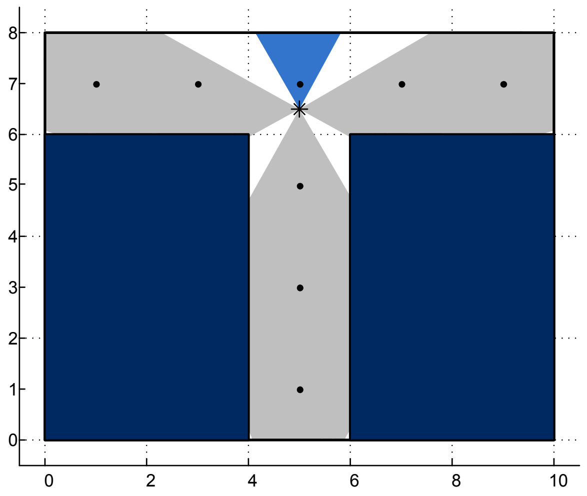


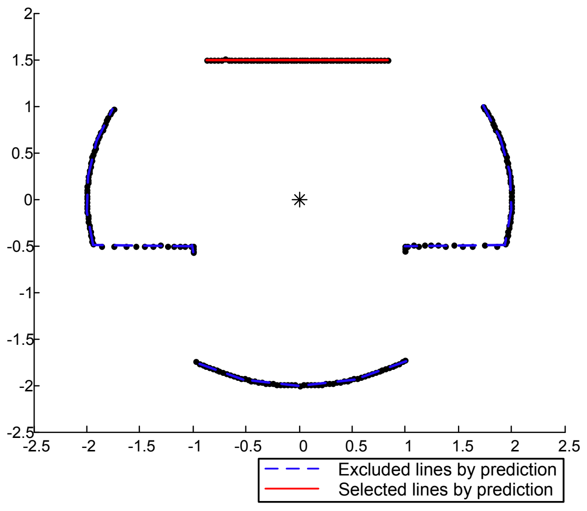
Figure 2-15: Illustration of a T - Shape Map.

In the previous section, the target environment is easy for robot navigation since $\mathbf{X}^r$ is available in most cases with the wide viewing angle and unlimited range of range finders. However, when the target environment is not closed as shown in Fig. 2-15, $\mathbf{X}^r$ is partially available in some cases. For example, a robot moves along a path $P_{G_1} \rightarrow P_{G_2} \rightarrow P_{G_3} \rightarrow .P_{G_4} \rightarrow P_{G_5} \rightarrow P_{G_6}$. The strategy for robot navigation until P_{G_3} is the same as that on the map with the straight path. When multiple boundary information (i.e., x_f^r or x_b^r, x_l^r or x_r^r) is available, the robot selects range data expected to be closest to the boundaries. For instance, at grid P_{G_1}, distance data x_l^r and x_r^r

have equal weight for localization and x_b^r is more preferable than x_f^r. A critical issue rises due to unnecessary boundary information when a robot moves from P_{G_3} to P_{G_5}.



(a) Robot position



(b) Measured range data

Figure 2-16: Illustration of measured range data with extracted lines when a robot is at $(5, 6.5)$ and selection of extracted lines by predicted boundary information.

Fig. 2-16 illustrates extracted lines from range data when a robot is at $(5, 6.5)$. While some lines are compatible with boundary information in the predefined table, others are unrelated to the predefined boundary information. In order to filter out the extracted lines, the robot needs to predict expected boundary information for the next destination. Since the path is already given to the robot, the next location is predicted within the movement error of the robot. Then, the expected boundary information is extracted from the predefined table with the predicted location. Suppose $\mathbf{X}^p = \{x_l^p, x_f^p, x_r^p, x_b^p\}$ denotes the predicted ranges. They are based on the predicted position (r_x^p, r_y^p) and calculated by,

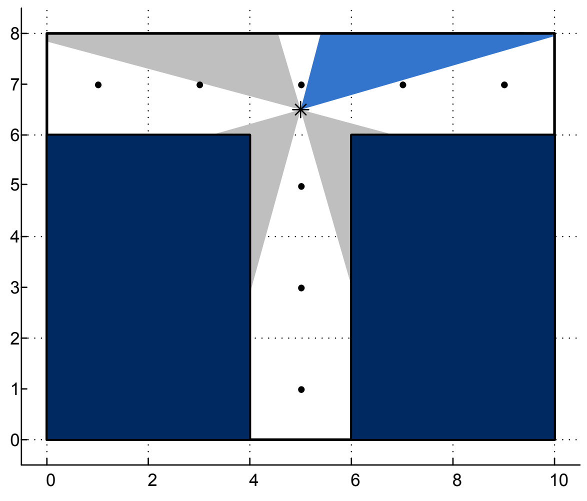
$$\begin{aligned} x_l^p &= x_l^i + (r_x^p - c_x^i), \\ x_f^p &= x_f^i + (c_y^i - r_y^p), \\ x_r^p &= x_r^i + (c_x^i - r_x^p), \\ x_b^p &= x_b^i + (r_y^p - c_y^i), \end{aligned} \tag{2.6}$$

where (r_x^p, r_y^p) belongs to the grid i. Then, the effectiveness of the measured range is defined by,

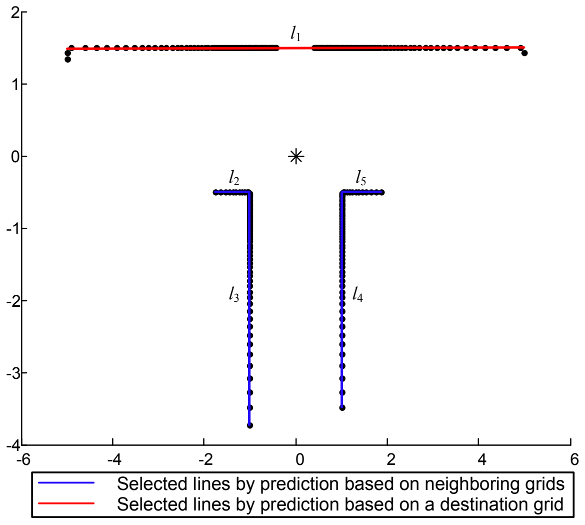
$$|x_j^p - x_j^r| \leq p_{err}, j \in \{l, f, r, b\}, \tag{2.7}$$

where p_{err} denotes the prediction error caused by the range uncertainty. Since the predefined boundary information consists of distances to boundaries at the center of each grid, selected lines are l_1, l_2, l_3, l_4, l_5, and l_{10}. If lines close to the robot have more credibility, the robot is localized with lines l_1 or l_5 and l_3.

Although the robot has unlimited range finders, measured range data can be insufficient for localization due to robot rotation. In Fig. 2-17, the robot with $\theta_a = 45^o$ is at $(5, 6.5)$. Suppose that a predicted position is the same as the current position. Then, the predicted ranges are $x_l^p = 4.5$, $x_f^p = 1.5$, $x_r^p = 4.5$, $x_b^p = 6.5$ by (2.6). After they are filtered out by (2.7), only x_f^p is available and the robot can be accurately localized in terms of only the y-coordinate. However, if the robot can utilize the excluded lines, it can calculate the x-coordinate position as well as the y-coordinate



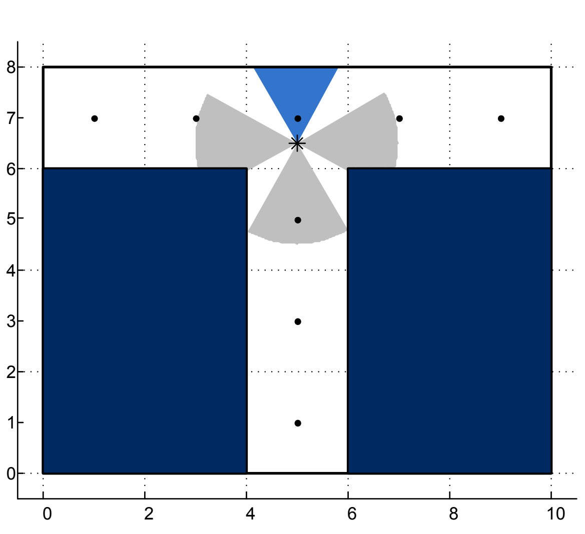
(a) Robot position



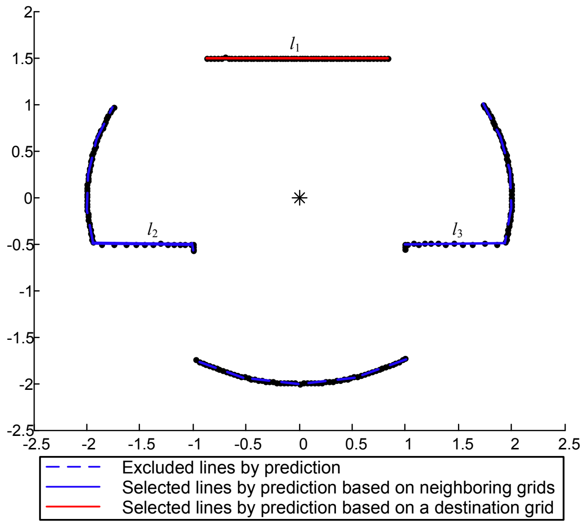
(b) Measured range data

Figure 2-17: Illustration of insufficient measured range data for localization due to robot rotation when a robot is at $(5, 6.5)$.

position. For example, when the robot moves to an open grid in three ways such as P_{G_4}, it can expect that lines constructed from the boundaries of neighboring grids are still valid for localization. Since the robot predicts a next robot orientation and the viewing angles of range finders are known, the robot can check which grids are reached by range finders. If close grids to the destination grid have more credibility than other grids, the robot utilizes boundaries of the neighboring grids P_{G_3}, P_{G_5}, P_{G_7}. Then, lines l_2 and l_5 can be used to calculate the y-coordinate position of the robot and lines l_3 and l_4 can be used to calculate the x-coordinate position of a robot. Therefore, a robot needs to predict ranges for not only the grid having a next destination but also the neighboring grids for possible rotation errors.



(a) Robot position

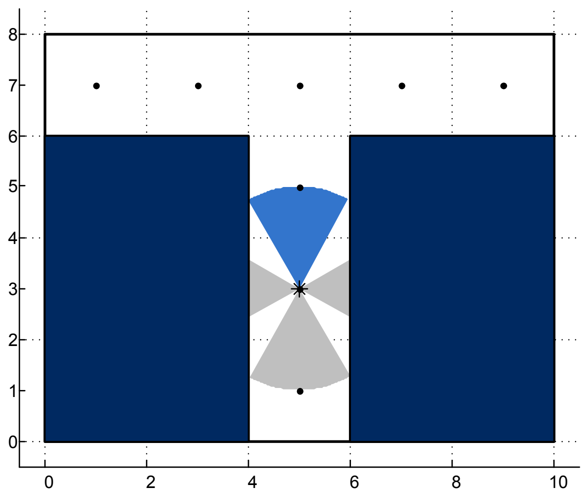


(b) Measured range data

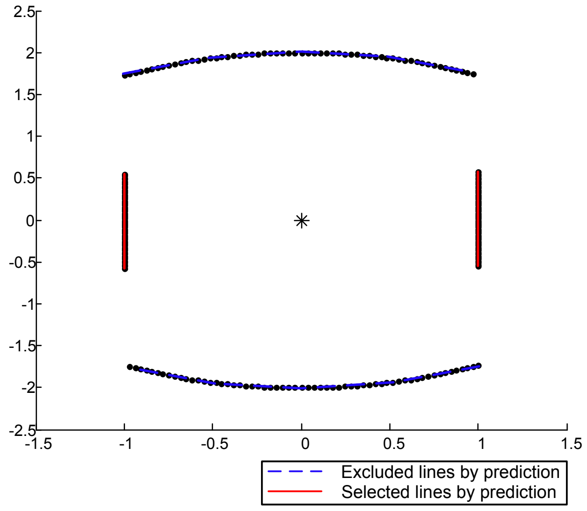
Figure 2-18: Illustration of insufficient measured range data for localization due to limited range finders ($R = 2$) when a robot is at $(5, 6.5)$.

Limited Range

In the previous section, it is assumed that the range finders have ranges of unlimited distance to measure boundaries of surroundings. However, in practical situations, range finders have usually limited ranges due to the available output power as shown in Fig. 2-18. Let us denote R a measurable range of range finders. Since range finders need to measure boundaries at each grid, a grid size of s_i cannot be greater than R. The critical problem with limited range range finders is that the robot a lot of times has insufficient range data for localization. Especially, when the robot moves to an open grid far from the boundaries, available range data is very limited. Hence, it is critical for the robot to predict range data based on the destination grid and the neighboring grids as explained in the previous section. Also, range data smaller than R is valid. When four grids (i.e., P_{G_3}, P_{G_4}, P_{G_5}, P_{G_7}) are considered for predicting range data, three extracted lines (i.e., l_1, l_2, l_3) are utilized for localization.



(a) Robot position



(b) Measured range data

Figure 2-19: Illustration of insufficient measured range data for localization due to limited range finders ($R = 2$) when a robot is at $(5, 3)$.

Fig. 2-19 illustrates a case that range data is partially available for only the x-coordinate position or the y-coordinate position. Since the range data alone are not sufficient for the localization in this case, the robot is localized based on the previous location. There are two possible approaches to calculate navigation distances using its odometer or the orientation of the robot. Fig. 2-20 illustrates key parameters in this situation.

d_m denotes the actual navigation distance of the robot and θ_a denotes the orientation of the robot. The value y is simply obtained by *Pythagorean theorem* with d_m and an absolute value of the difference between the previous boundary information and the current boundary information. If the orientation of the robot can be successfully extracted from the extracted lines, the value y is obtained by a *tangent* function

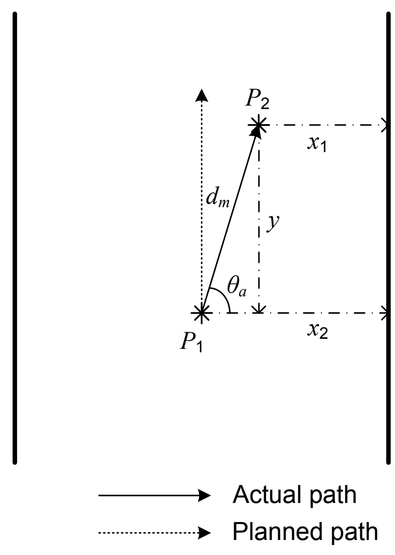
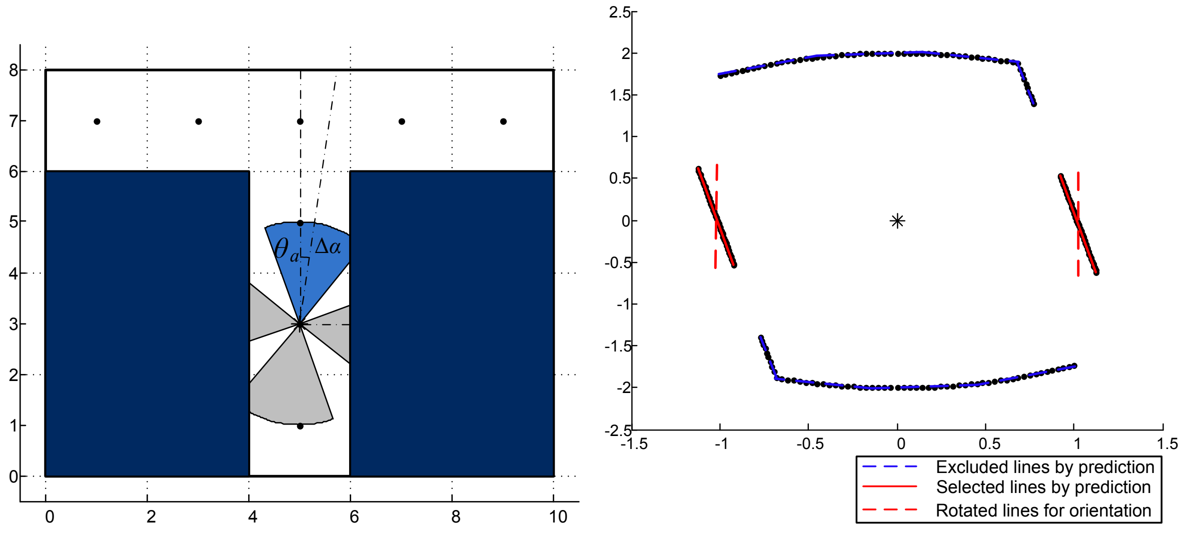


Figure 2-20: Illustration of estimation for location with partial information.

with the absolute value of the difference between the previous boundary information and the current boundary information. On the other hand, if the previous location is unknown, the robot needs to navigate until a sequence of extracted features are matched with the corresponding sequence of predefined features in the table.

The importance of prediction for range data can also be demonstrated in finding the orientation of the robot. In Fig. 2-21, θ_c is 0^o and θ_a is 10^o. The actual robot orientation is obtained by aligning the extracted lines to have possible angles (i.e., $\pm 0^o, \pm 90^o, \pm 180^o$) on a rectangular map. The amount of the angle changes is the difference between the actual orientation and the planned orientation. When unexpected lines such as excluded lines by prediction are considered, the obtained orientation of the robot may have an error. Thus, the robot needs to predict which lines should be considered to find a robot orientation.



(a) Robot position (b) Measured range data

Figure 2-21: Illustration of filtering measured range data for a robot orientation due to limited range finders ($R = 2$) when a robot is at $(5, 3)$.

Effect of Grid Size

The grid information where the robot is currently located in is important in terms of navigation certainty. When the robot knows the grid information of the current location, it is easy to compensate for the possible navigation error in the next navigation. If the robot is localized at the center of each grid, the navigation error is also corrected at the same time. Hence, the size of a grid is proportional to the probability that the robot is deviated from the destination as shown in Fig. 2-22. However, it is not necessary for the grid size to be smaller than the sum of the size of the robot and the minimum navigation distance. On the other hand, as the grid size becomes smaller, it increases the probability that the robot may navigate to the neighboring grids instead of the destination grid. Since several grids can exist between the boundaries, the robot may check more grids within R for the boundary prediction. The grid

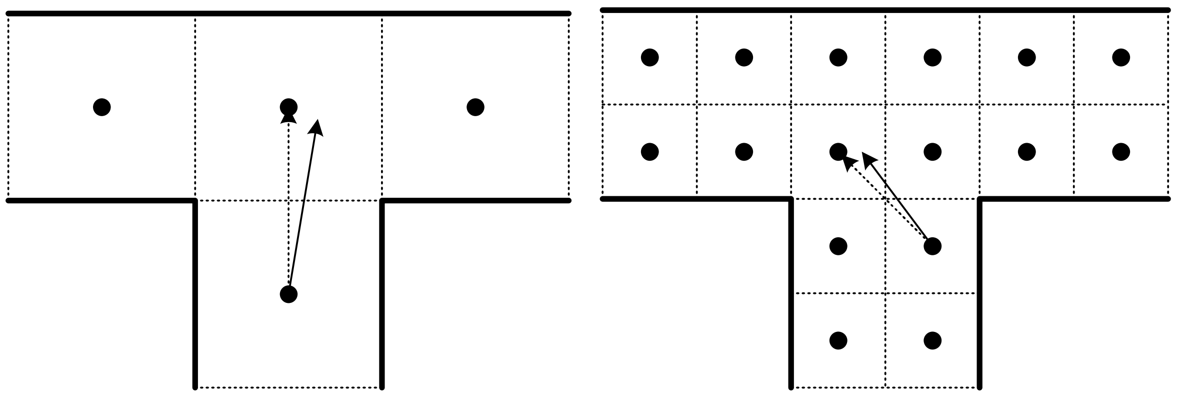

(a) $s_i = 1$ (b) $s_i = 0.5$

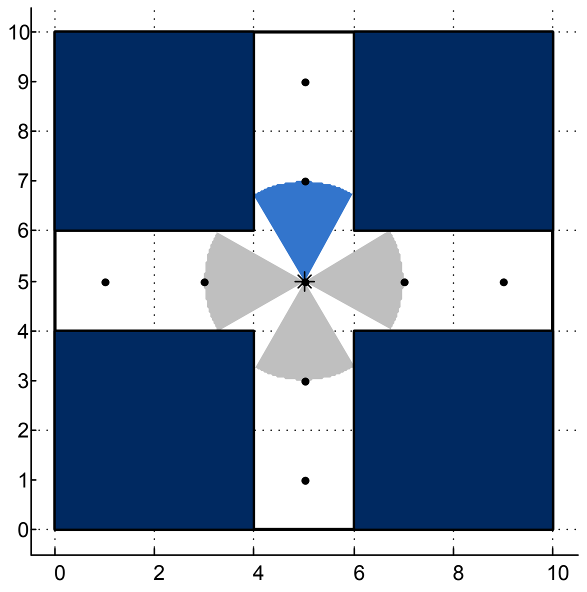
Figure 2-22: Illustration of navigation error in terms of grid size.

information is confirmed by localization with the predicted boundary information. Thus, the smaller grid size requires to investigate more boundary information.

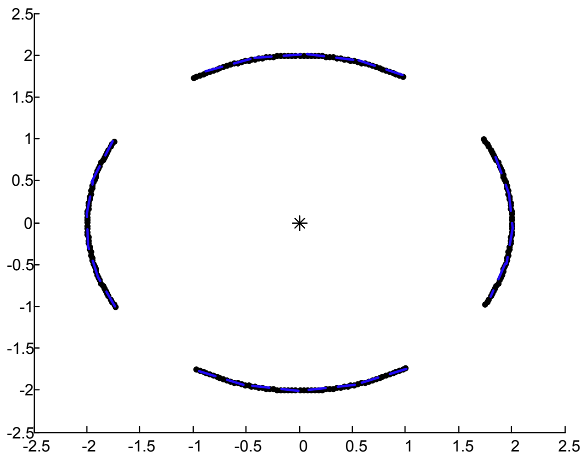
2.4.2 Cross - Shape Map

Similar to T - shape map, there exists a case when the required distances to boundaries are not available due to the limited sensing angle as shown in Fig. 2-23. Since the grid at the cross is open in four ways, there exists a case that sufficient range data may not be available at all depending on R and θ_v. In this case, the robot may navigate until boundary information is available. The other approach is to navigate closely to boundaries or to decrease grid size so that the robot can measure boundary information. Fig. 2-24 illustrates the case that θ_v is increased and Fig. 2-25 illustrates the case that grid size is decreased at the cross section.

When the grid information is constructed or the robot moves towards the cross section, it needs to check if the grid size is appropriate for R and θ_v of the range



(a) Robot position



(b) Measured range data

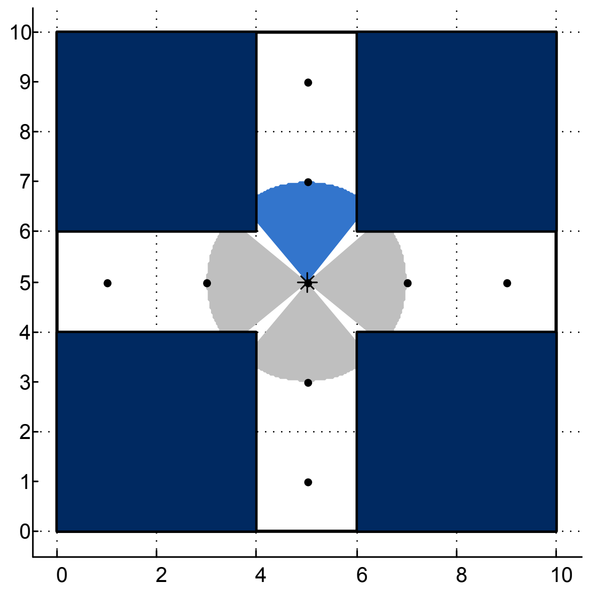
Figure 2-23: Illustration of localization problem in a cross - shape map.

finders. the proper grid size means that lines can be extracted at the center of a grid. In Fig. 2-26, n denotes the number of consecutive samples to be extracted as a line. Assuming θ_c is 0^o, the maximum range R with $\theta_f - n\theta_s$ should reach a boundary. θ_f denotes an angle of the last range in terms of the axis parallel to the corresponding boundary. This is checked by

$$Rsin(\theta_f - n\theta_s) \leq s_i. \tag{2.8}$$

2.4.3 Navigation Algorithm

The important issue in navigation is to predict range data based on the predefined table for the grid information. The tolerance of the predicted range data is determined by navigation error. The robot predicts them by using which grid it belongs to at the


(a) Robot position

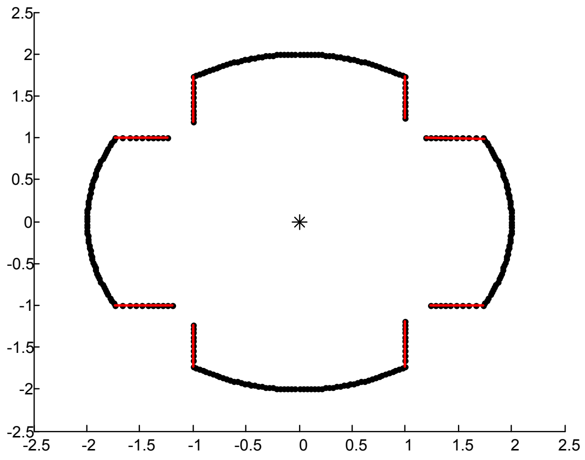

(b) Measured range data

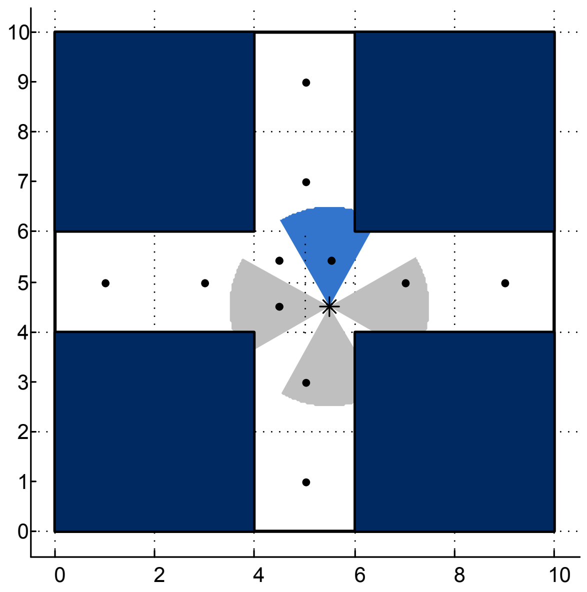
Figure 2-24: Illustration of increased viewing angles of range finders in a cross - shape map.

next location. However, some grids do not have boundaries which can be reached by range finders of the robot. Thus, the robot needs to check if the boundary information of the neighboring grids can be utilized. In order to make this possible, each grid information in the table should also have the information about neighboring grids.

Idx	Coord.	size	x_l	x_f	x_r	x_b	n_l	n_f	n_r	n_b

Table 2.2: The template of a proposed boundary information at each grid.

Table 2.2 illustrates the template of the proposed boundary information at each grid. x indicates the distance to each side, and n indicates the index of a neighboring grid to each side. If n is a zero, it does not have a neighboring grid but a boundary. The robot checks if boundaries of a grid can be reached by the range finders with R, the predicted next location, and coordinates of the neighboring grids. Once they are


(a) Robot position

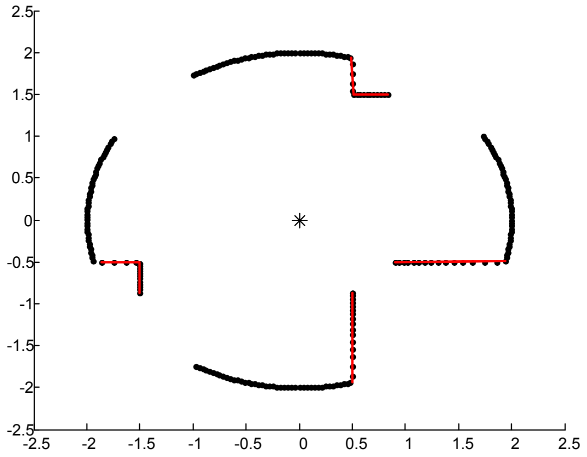

(b) Measured range data

Figure 2-25: Illustration of increased viewing angles of range finders in a cross - shape map.

determined, the robot predicts all possible range data with the boundary information of the reachable grids.

Fig. 2-27 shows the robot navigation flowchart with range finders. In the initial state, the robot does not know its orientation and position. In order to move to the destinations, these should be known to the robot. The necessary information to initially localize the robot is the current grid. Since the predefined distances to boundaries are known from the grid information table, the correct surroundings is selected from four candidates with the predicted range $\mathbf{X}^p$. When (2.7) is used to find the correct surroundings, p_{err} is set to be the size of the grid because the robot should be within the initial grid.

The state of the robot is defined by,

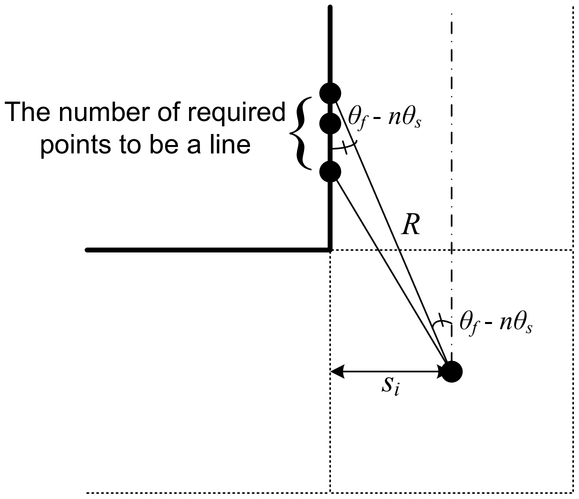


Figure 2-26: The illustration of a method to check if grid size is compatible with R and θ_v of range finders.

$$\mathbf{X}_t = \begin{pmatrix} r_{x,t} \\ r_{y,t} \end{pmatrix},$$

where t denotes a time for state update. The system equation is given by,

$$\mathbf{X}_{t+1} = \mathbf{X}_t + \mathbf{U}_t + \mathbf{w}_t,$$

where $\mathbf{U}_t$ is the system input by the next destination and $\mathbf{W}_t$ is the noise process.

$$\mathbf{U}_t = f(\mathbf{X}_t, \mathbf{P}_t^n) = \begin{pmatrix} \Delta x_t \\ \Delta y_t \end{pmatrix},$$

where $\mathbf{P}_t^n$ is the coordinate of the next destination and the function $f(\cdot)$ calculates the distance to the next destination in terms of x-coordinate and y-coordinate. $\mathbf{W}_t$ is the zero-mean Gaussian distribution with the variance due to the odometer uncertainty

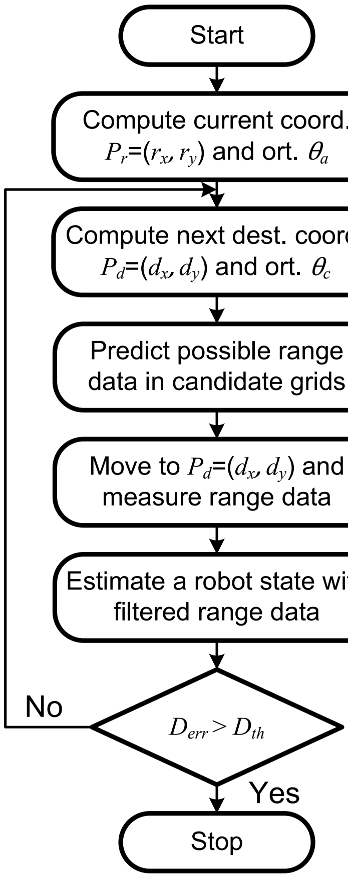

Start
Compute current coord. $P_r=(r_x, r_y)$ and ort. θ_a
Compute next dest. coord. $P_d=(d_x, d_y)$ and ort. θ_c
Predict possible range data in candidate grids
Move to $P_d=(d_x, d_y)$ and measure range data
Estimate a robot state with filtered range data
$D_{err} > D_{th}$
No
Yes
Stop

Figure 2-27: The illustration of robot navigation flowchart.

and wheel slipping.

$$\mathbf{w}_t \sim (0, \mathbf{Q}_t),$$

where σ_x^2 and σ_y^2 is the variance of each coordinate. We assume that they are independent and $\mathbf{Q}_k$ is defined as,

$$\mathbf{Q}_t = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix},$$

The measurement equation is given by,

$$\mathbf{Y}_t = \mathbf{X}_t + v_t,$$

where v_t is noise vector of range finder measurement and R_t is the variance. R_t is defined as,

$$\mathbf{R}_t = \begin{pmatrix} \sigma_{r,}^2 & 0 \\ 0 & \sigma_{r,y}^2 \end{pmatrix},$$

where $\sigma_{r,x}^2$ and $\sigma_{r,y}^2$ is the measurement variances of each coordinate.

Since the system equation evolves linearly, we can apply the Kalman Filtering to update the robot state. [26] The filter is realized in the following way.

$$\hat{\mathbf{X}}_{t+1|t} = \hat{\mathbf{X}}_{t|t} + \mathbf{U}_t, \tag{2.9}$$

$$\mathbf{P}_{t+1|t} = \mathbf{P}_{t|t} + \mathbf{Q}_t, \tag{2.10}$$

$$\mathbf{K}_t = \mathbf{P}_{t+1|t}(\mathbf{P}_{t+1|t} + \mathbf{R}_t)^{-1}, \tag{2.11}$$

$$\hat{\mathbf{X}}_{t+1|t+1} = \hat{\mathbf{X}}_{t+1|t} + \mathbf{K}_t(\mathbf{Y}_t - \hat{\mathbf{X}}_{t+1|t}), \tag{2.12}$$

$$\mathbf{P}_{t+1|t+1} = \mathbf{P}_{t+1|t} - \mathbf{K}_t\mathbf{P}_{t+1|t}, \tag{2.13}$$

where $\hat{\mathbf{X}}_{t+1|t}$ and $\mathbf{P}_{t+1|t}$ are a priori state estimate and its covariance, $\hat{\mathbf{X}}_{t+1|t+1}$ and $\mathbf{P}_{t+1|t+1}$ are a posteriori state estimate and its covariance after the measurement $\mathbf{Y}_t$ is processed, $\mathbf{K}_t$ is Kalman gain.

The robot moves distance d_r at a time. The next destination is defined by the given path and the distance d_r. If the rest of distance to the center of a grid is smaller than d_r, the robot moves the rest of distance. Once the robot is believed to be at the destination coordinate, the robot computes the coordinate error D_{err} between the current coordinate and the destination coordinate. Based on the coordinate error, the robot autonomously tries to go to the final destination. This is done by recomputing the current coordinate. The robot repeats this process until the destination coordinate is reached within the acceptable error range D_{th}. d_r is related to how many times the range finders are activated for localization. As d_r is shorter, it minimizes collision with the wall and unnecessary rotations but requires more range finder activation and computation.

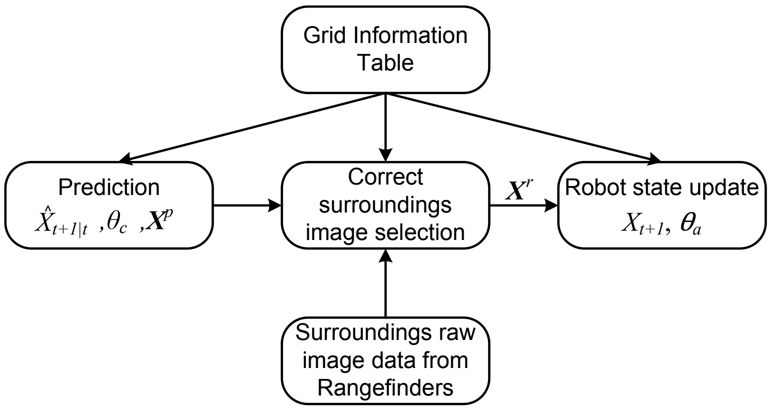

Grid Information Table
Prediction $\hat{X}_{t+1|t}$,θ_c ,$\boldsymbol{X}^p$
Correct surroundings image selection
$\boldsymbol{X}^r$
Robot state update X_{t+1}, θ_a
Surroundings raw image data from Rangefinders

Figure 2-28: The illustration of the information flow for robot state estimation.

Fig. 2-28 summarizes necessary information flow for correct localization. Predefined grid information is given to the robot. Before the robot moves to the next destination, the next position $\hat{X}_{t+1|t}$ is predicted in the process of Kalman Filtering. Based on the predicted position, the predicted boundary $\mathbf{X}^p$ and the planned orientation θ_c are calculated. After the range finders are activated, the correct surroundings information is selected with $\mathbf{X}^p$ and θ_c. Then, the measured position of the robot is used to estimate the position of the robot by Kalman Filtering.

2.5 Evaluation and Analysis

2.5.1 Effect of Non-ideal Range

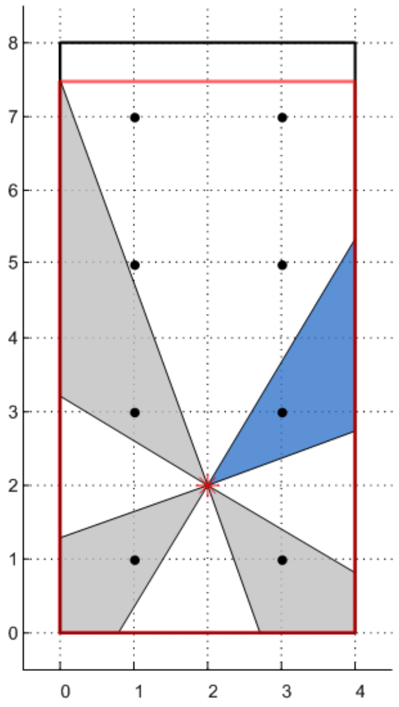


Figure 2-29: Illustration of the problem when the sensing angle of a range finder is small.

Fig. 2-29 shows the problem in the map construction when the sensing angle of a range finder is small. The sensing angle represents how wide a range finder searches the surroundings. The problem occurs when the range finder can not reach the boundary of the map. This makes finding the exact position unreliable. There are are possible solutions to this problem. The first solution is to increase the sensing angle if the constructed map does not match with the true map. The other solution is to change the orientation of the robot to ensure the enough view to obtain the overall map. The third solution is to have the robot proceed until the robot obtains the constructed map similar to the true map.

2.5.2 Effect of Complex Boundary

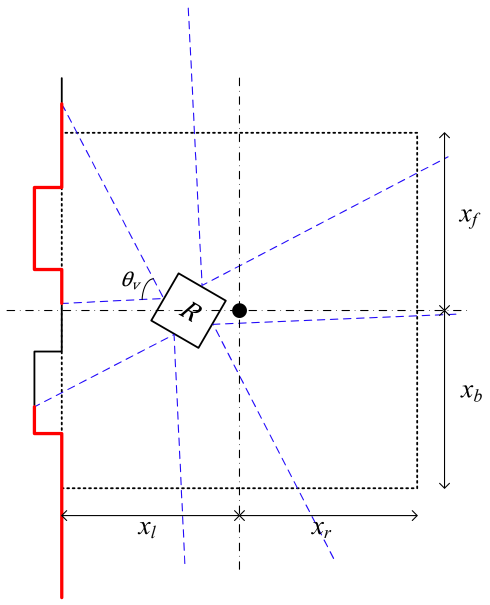
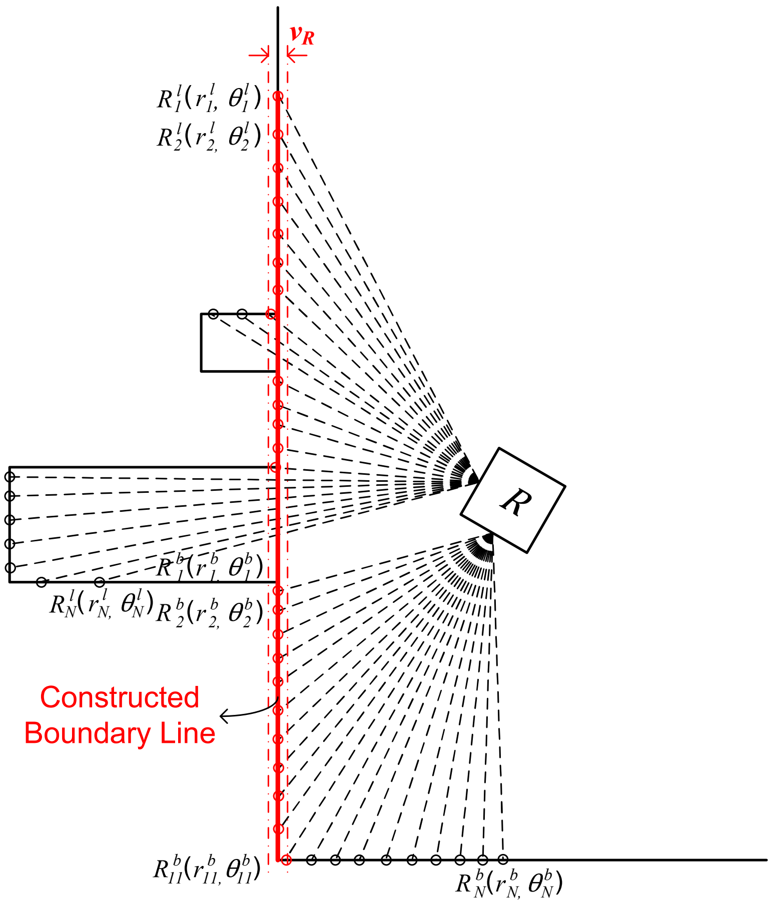


Figure 2-30: The illustration of the uneven boundary environment.

The uneven boundaries exist in most buildings due to such structures as doors, fountains, and windows, etc. Fig. 2-30 shows the environment which boundary is not even. This problem is similar to T-shape map but it is not clear to the robot whether the indented region is the other hallway or not. Moreover, the robot may have the several boundary lines as shown in the figure. This may cause localization errors. One approach for this problem is to have all boundary information in the map table. However, this increases the table size tremendously and the grid shape becomes too irregular because the grid size is not flexible anymore. The second approach is to average several boundary lines within the boundary variance. The map table needs to have the boundary variance information for each grid. However, this method is only feasible for the small boundary variance, and a large boundary variance introduces a larger localization error. Also, it is not easy to define the boundary variance for each grid. The last method is to use the predicted boundary information. Assuming that the correct candidate image is chosen, the irrelevant boundary line is filtered out by using the variance of the range finder.



v_R
$R_1^l(r_1^l, \theta_1^l)$
$R_2^l(r_2^l, \theta_2^l)$
R
$R_1^b(r_1^b, \theta_1^b)$
$R_N^l(r_N^l, \theta_N^l)$
$R_2^b(r_2^b, \theta_2^b)$
Constructed Boundary Line
$R_{11}^b(r_{11}^b, \theta_{11}^b)$
$R_N^b(r_N^b, \theta_N^b)$

Figure 2-31: The illustration of the range utilization method with the range finder variance in the uneven boundary environment.

Fig. 2-31 shows the range utilization example with the predicted boundary information. For simplicity, R^l and R^b are shown in the figure. Since the robot knows the predicted range to the boundary at the predicted position by (2.6), the left side range data is filtered out by the following equation.

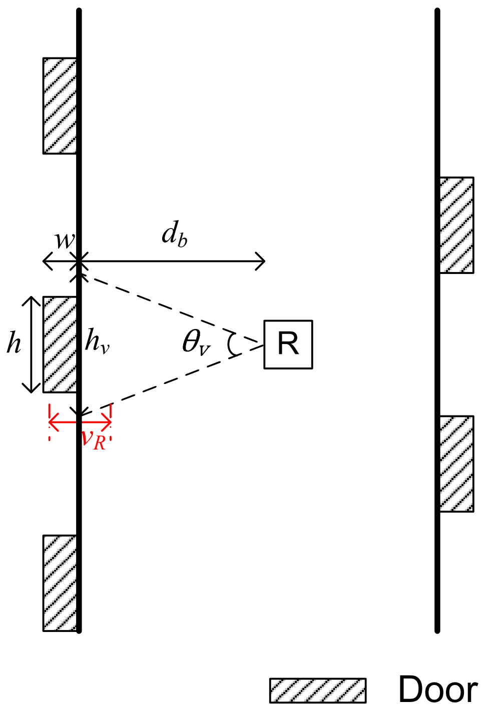
$$|x_l^p - r_n^l \cdot \cos(\theta_n^l)| \leq v_r, \; 1 \leq n \leq N, \tag{2.14}$$

where v_r is the range finder variance and N is the number of range samples obtained by θ_v/θ_s. In the figure, the range sample of the red circle satisfies (2.14) and the range sample of the black circle is filtered out by (2.14). Suppose that $\mathbf{S}^l = \{s_1^l, s_2^l, ..., s_M^l\}$ is the set of the values of $r_n^l \cdot \cos(\theta_n^l)$ satisfying (2.14) for the left side boundary. M is the number of range samples satisfying the equation. Then, the distance from the robot to left side boundary is estimated by,

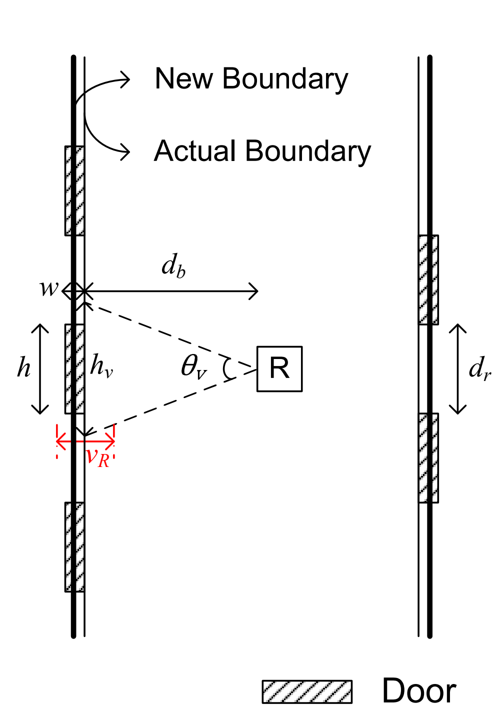
$$x_l^r = \frac{1}{M}\sum_{m=1}^{M} s_m^l. \tag{2.15}$$

The distances to other boundaries are similarly obtained. On the other hand, this restricts the viewing angle of the range finder, θ_v. If this is too small to obtain the range sample from the boundary matched with the map table, all range samples are filtered out by the (2.14). The robot deals with this problem by the redundant boundary or the appropriate viewing angle.

Fig. 2-32(a) shows how the uneven boundary restricts the viewing angle of the range finder. If a uneven boundary exists regularly such as doors, we can expect how wide the viewing angle requires to be. The thick line is the boundary in the map



(a) The viewing angle of a range finder



(b) The map table construction

Figure 2-32: The effect of the uneven boundary in terms of the viewing angle of the range finder and the map table construction.

table. The shaded region is the door with the width w and the height h. Suppose that w is larger than the measurement variance v_R. In order for the robot to self-localize accurately, the viewing angle should be wide enough to sample the range from the boundary known to the robot. The amount of the sampled region from the boundary is defined by,

$$h_v = 2\tan(\frac{\theta_v}{2})d_b, \tag{2.16}$$

where h_v is the possible sampled region from the boundary when the robot moves d_b away from the boundary. If h_v of a range finder is smaller than h at some position, the sampled range is filtered out by (2.14) and becomes useless.

This also affects how the boundary is represented in the map. d_r denotes the distance between the patterned region in Fig. 2-32(b). When $w > v_R$ and the enough viewing angle is ensured, the actual boundary does not cause the localization error. However, the measurement variance can introduce the localization error. Suppose $w < v_R$, $d_r < h_v$ and the enough viewing angle is ensured. Then, the range samples from the door also contribute to localize the robot by (2.14). This deviates the localization of the robot because this information is not included in the map table. Thus, the represented boundary in the map table should consider the pattered region in this case. When the multiple boundary lines are detected, they are estimated by (2.15). If there are multiple boundaries satisfying the measurement variance, the average of them should be represented in the map table to minimize the localization error.

Fig. 2-33 shows the simulation model for the relationship between the uneven boundary and the viewing angle. The thick solid line is the boundary in the map table. We assume that w and h of the uneven region is distributed by,

$$w \sim (0, v_w), h \sim (0, v_h), \tag{2.17}$$

where, v_w and v_h are the variance of w and h respectively. If w is negative, the uneven region is generated outward. Otherwise, it is generated inward. In the simulation, only w greater than the measurement variance is used. The distance between the uneven regions follows,

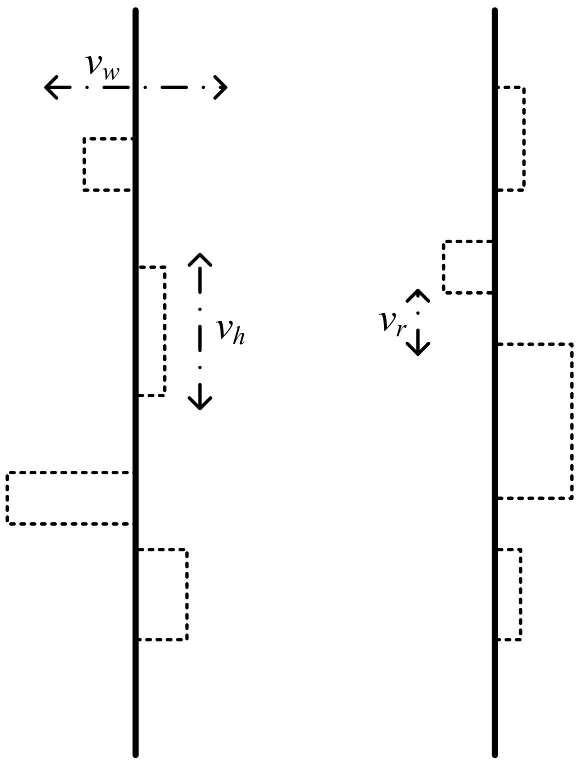


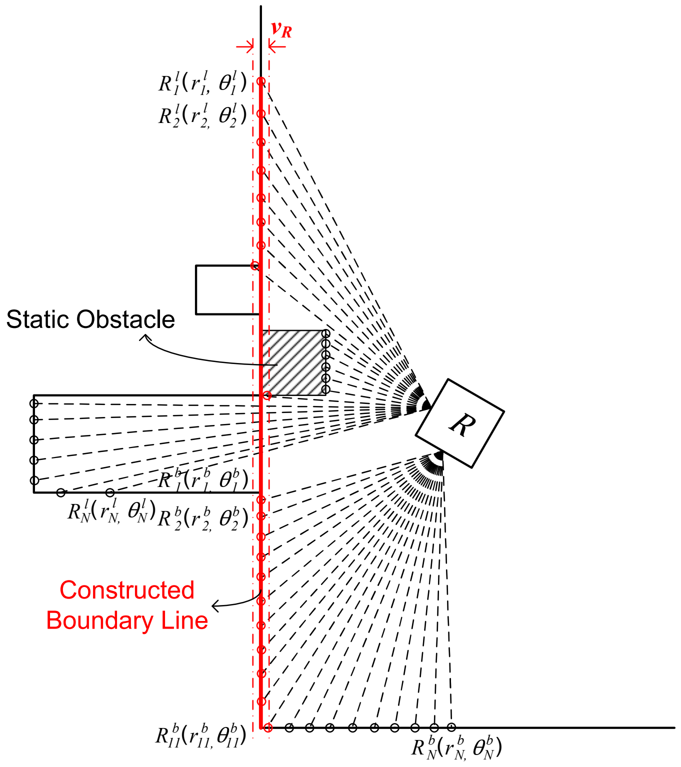
Figure 2-33: The simulation model for the relationship between the uneven boundary and the viewing angle.

$$d_r \sim (d_{i,r}, v_r), \tag{2.18}$$

where $d_{i,r}$ is the mean of the distance between the uneven regions.

2.5.3 Map with Static Obstacles

Fig. 2-34 shows the environment with the static obstacles. The same approach for the uneven boundary environment is feasible for this case. In this case, the possibility that all range samples are filtered out by the range finder variance increases because the obstacle blocks the range signal to the boundary. By using the number of range samples satisfying (2.14), the robot selects the measured boundary information for



v_R
$R_1^l(r_1^l, \theta_1^l)$
$R_2^l(r_2^l, \theta_2^l)$
Static Obstacle
R
$R_1^b(r_1^b, \theta_1^b)$
$R_N^l(r_N^l, \theta_N^l)$
$R_2^b(r_2^b, \theta_2^b)$
Constructed Boundary Line
$R_{11}^b(r_{11}^b, \theta_{11}^b)$
$R_N^b(r_N^b, \theta_N^b)$

Figure 2-34: The illustration of the range utilization method with the range finder variance in the uneven boundary environment with the static obstacles.

the localization. For example, the robot is localized when either x_l^r or x_r^r and either x_f^r or x_b^r are available. Suppose that all four range data are available. Then, either of x_l^r or x_r^r is chosen based on the number of range samples satisfying (2.14) and either x_f^r or x_b^r are chosen by the same way.

When the obstacle exists further from a robot and closer to the boundary, the possibility that all range samples are filtered out is small. Fig. 2-35 shows the localization failure cases because of closely moving obstacles. When obstacles blocks either the left and right range finder or the front and back range finder at the same time, a robot cannot be localized due to the insufficient information. In this case, the robot state is updated based on the predicted state. Since this happens by the

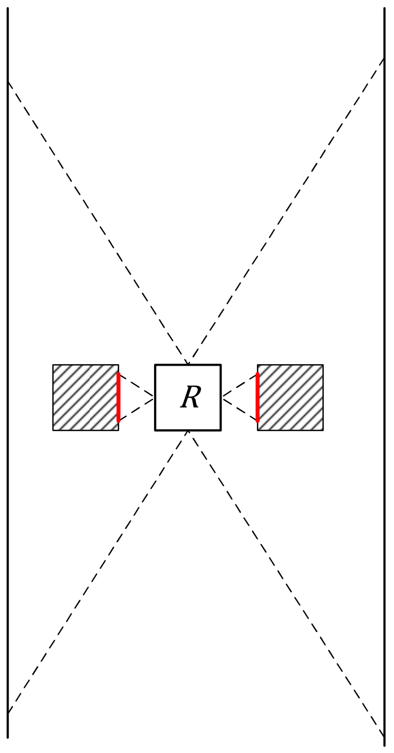


Figure 2-35: The illustration of the impossible case of the localization because of closely moving obstacles.

moving obstacles at an instant, this problem rarely affects the total performance of the navigation.

2.6 Experiments

The mobile robot used for the experiment is shown in Fig. 2-36. The basic platform of the mobile robot is X80 developed by Dr Robot. The laser range finder is UTM-30LX developed by Hokuyo [9] [10]. θ_s, the sampling angle of the laser range finder is 0.25^o. Since θ_v, searching angle of the laser range finder is 270^o, extracted lines from the first and last range data are ignored. Also, r_{max}, the maximum range distance of the laser range finder is set to be $4m$. For the experiment, a limited number of points in the

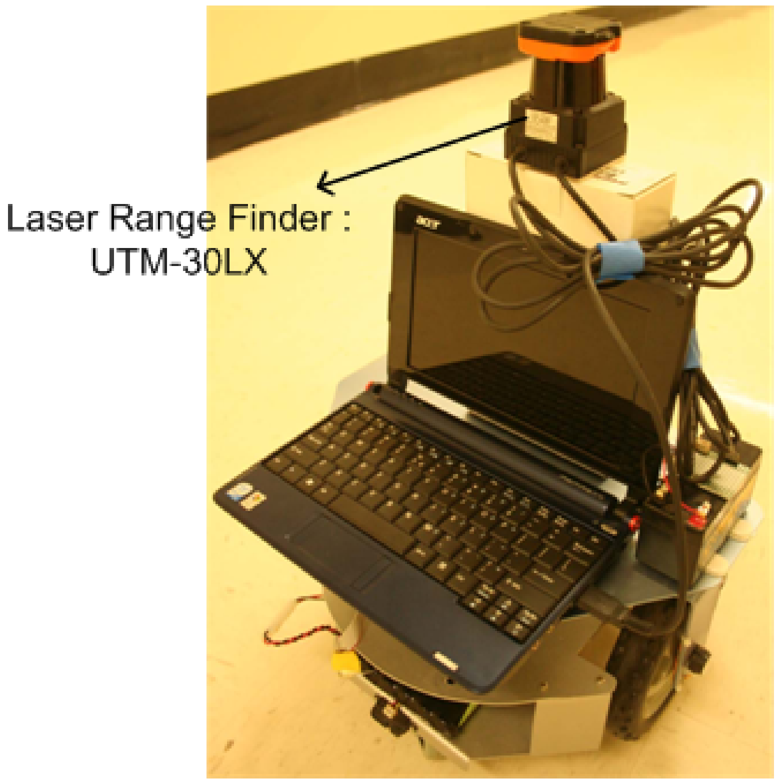


Figure 2-36: The illustration of a mobile robot (Dr Robot : X80) equipped with a laser range finder (Hokuyo : UTM-30LX).

floor map called vertices are used as the boundary information. This is done to make experiment more practical as it would be a time-consuming task to translate with a very small grid size compared to overall area that needs to be covered. The vertices represent points where two edges meet. Each grid have a distance and angle towards all vertices visible from the centers of each grid. We also assume the robot's initial position is at the center of a known grid to create one reliable location information before navigation starts. Knowing the initial grid and boundary information from the grid enables us to estimate where the next vertex may be observed, and thus use it for self-localization.

Fig. 2-37 shows the simulation environment of our department building. The mobile robot moves along with the given path (i.e. $\{G_1, G_2, \ldots, G_{19}\}$). The size of each grid is $1.17m$. The mobile robot stops incorrectly at each grid due to the mechanical problems but the correct location is estimated by the proposed method. The extracted lines from the range data at each grid are shown in Fig. 2-38. At grid G_{18}, the mobile robot is localized twice because it changes the orientation to grid G_{19}. It moves to the next grid based on the estimated location.

Fig. 2-39 shows the estimated orientation and the distance error between the actual location and estimated location.

2.7 Evaluation and Analysis

2.7.1 Effect of Sensor Characteristics

The number of the extracted lines is usually proportional to r_{max} since the range finder can reach more boundaries around the surroundings. On the other hand, the probability of location error for the extracted lines usually increases. The longer range does not mean that the navigation system can localize a robot more accurately. Instead, it may increase the computational time to find the matched pattern. However, the range should be enough to reach the boundary of the corridor at a grid. It is expected that the effect of the range on the navigation performance will be saturated by the performance of the line extraction from the range data. If the range finder cannot scan the surroundings entirely due to the limited searching angle, it will decrease the number of range data points around the surroundings.

According to the specification of the used laser range finder, the variance of the range accuracy is less than $10mm$ when the range is less than $10m$. In order to analyze the effect of the sampling angle and the range, the range error is modeled as Gaussian with $\sigma = 10mm$ and the variance is proportional to the range. The localization error in Grid 12 is the current limitation of the proposed method.

2.7.2 Effect of Grid Size

The grid information where the robot is currently located in is important in terms of navigation certainty. When the robot knows the grid information of the current location, it is easy to compensate for the possible navigation error in the next navigation. If the robot is localized at the center of each grid, the navigation error is also corrected at the same time. Hence, the size of a grid is proportional to the possibility that the robot is deviated from the destination as shown in Fig. 2-22. However, it is not necessary for the grid size to be smaller than the sum of the size of the robot and the minimum navigation distance. On the other hand, as the grid size becomes smaller, it increases the possibility that the robot may navigate to the neighboring grids instead of the destination grid.

2.8 Summary

In this chapter, we showed the robot navigation algorithm using the data extracted from the onboard range finders. BY exploiting the fact that we have pre-populated boundary information along with the range data, we presented the novel navigation algorithm by self-localization. We also showed the navigation algorithm could handle more realistic navigation tasks such as T- and Cross - shaped map. The algorithm also took into account that the range data from the boundaries can contain noise from the boundary not being smooth. We used the MSE (Mean Square Error) and the Kalman filter with the known boundary information to assist the self-localization. We also investigated the case where physical obstacles were present to hinder the self-localization and how the algorithm would cope with this situation. Overall, We demonstrated that the range data from the range finders can be utilized for the self-localization and the navigation of mobile robots. The algorithm provided a way to compensate for the errors caused by the imperfect data and mechanical components. When the self-localization can be realized without the help of any absolute position information, the algorithm can be applied to many systems that use mobile robots.

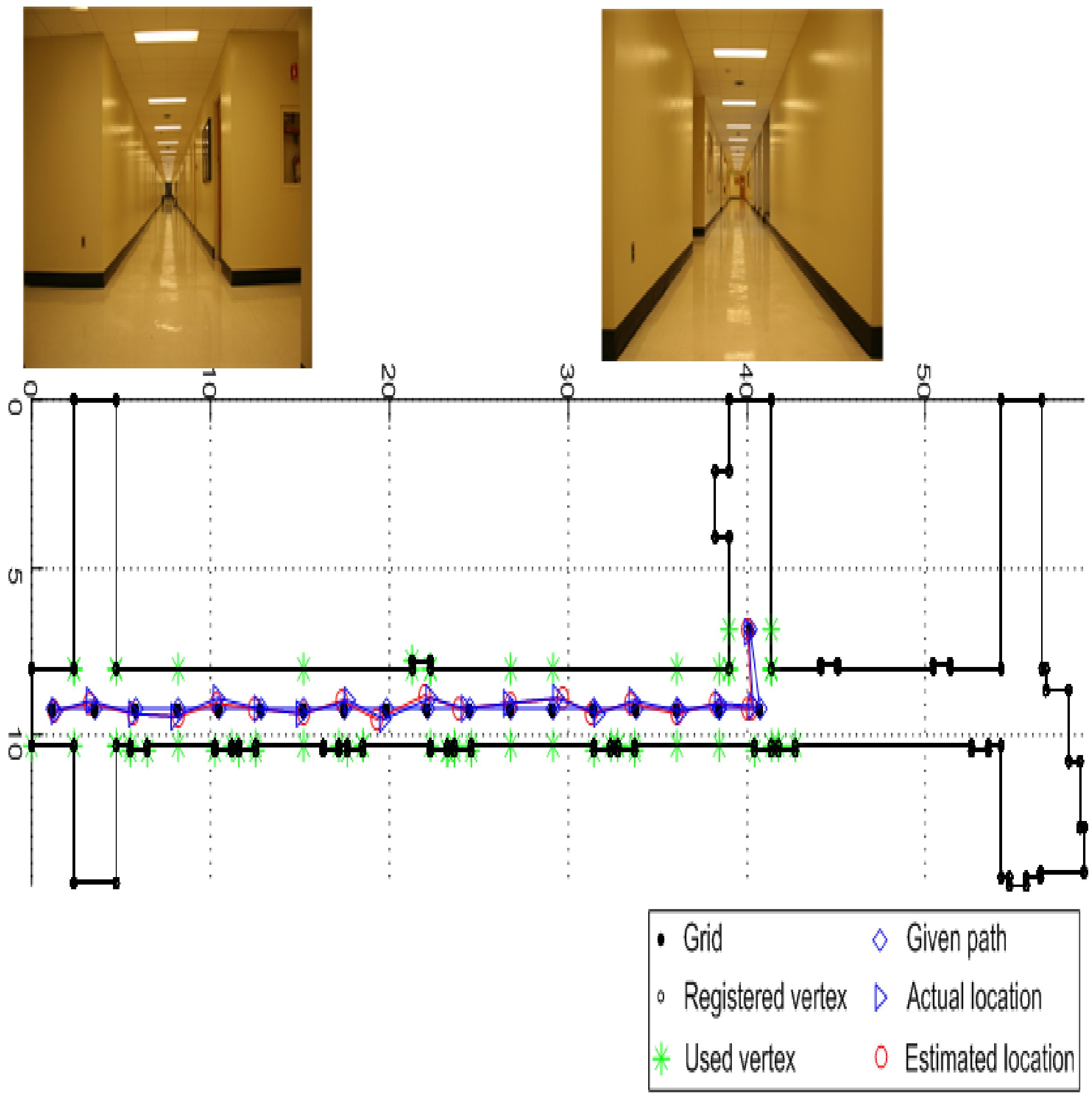


Figure 2-37: The simulation trajectory for the pattern matching based navigation in the department building.

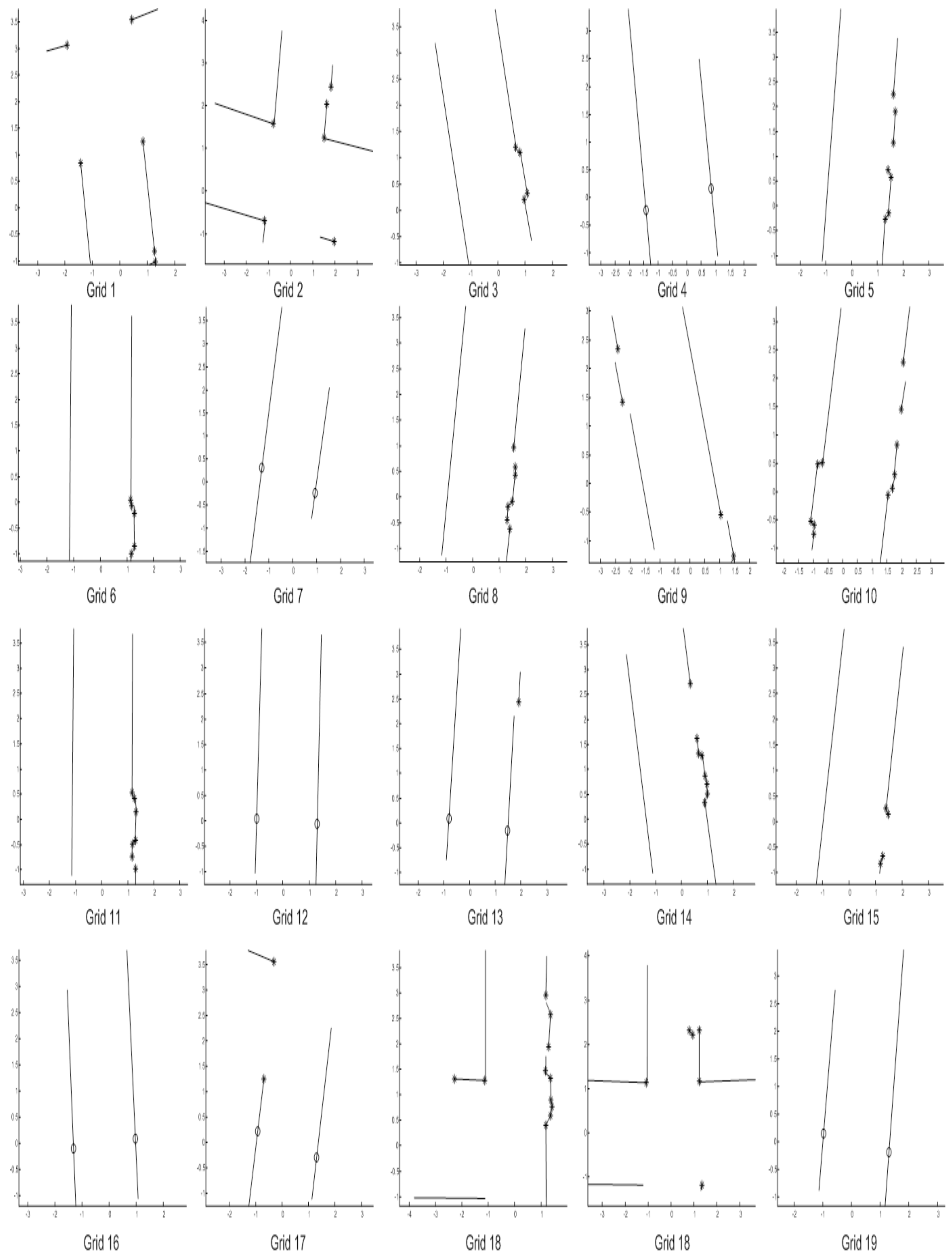
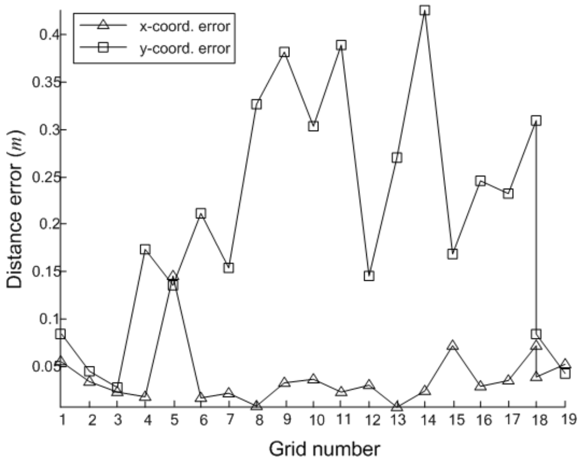
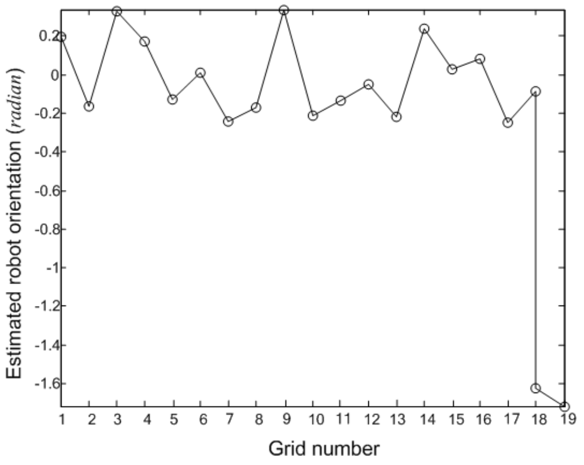


Figure 2-38: The extracted lines from the actual range data at each grid.

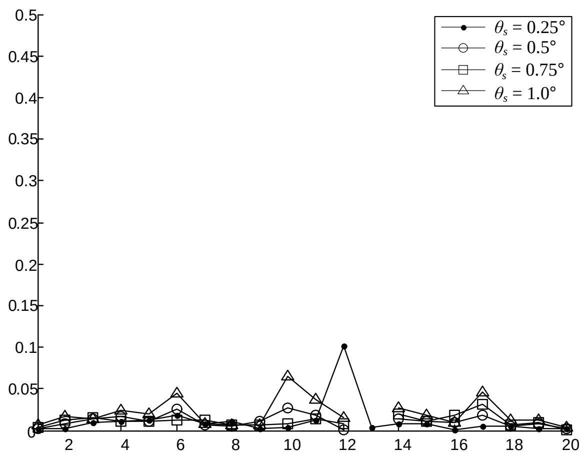


(a) The distance error of estimated location

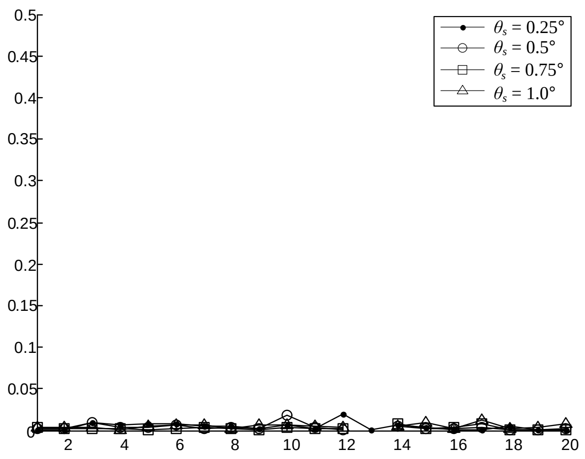


(b) Estimated orientation

Figure 2-39: Localization result for Fig. 2-37.

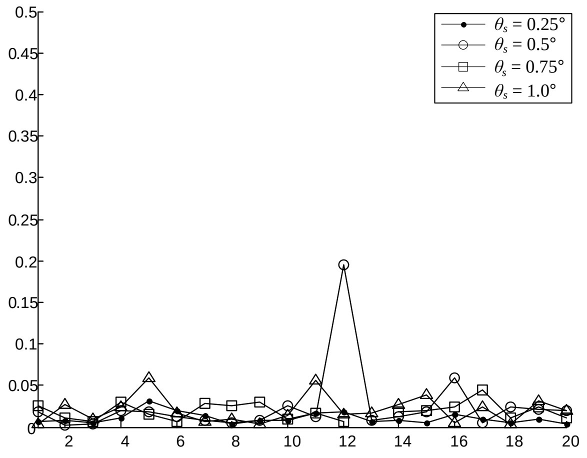


(a) Average location error

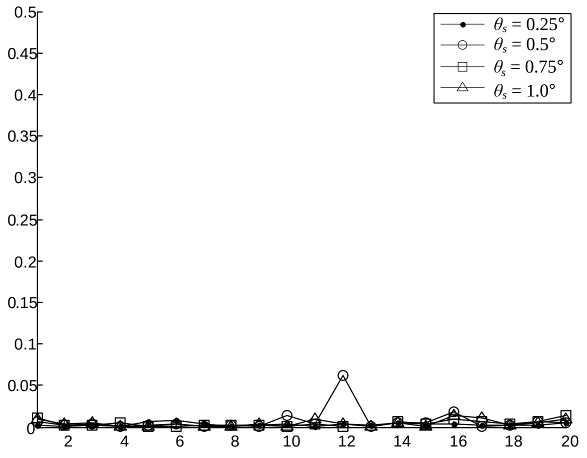


(b) Average orientation error

Figure 2-40: Localization performance and vertex usage according to the sampling angle with $r_{max} = 3m$.

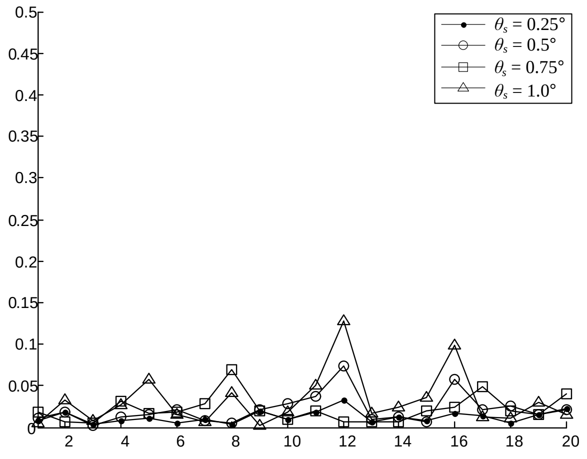


(a) Average location error

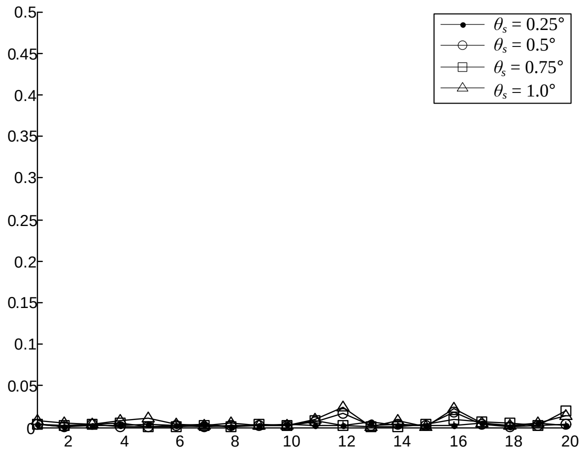


(b) Average orientation error

Figure 2-41: Localization performance and vertex usage according to the sampling angle with $r_{max} = 6m$.



(a) Average location error



(b) Average orientation error

Figure 2-42: Localization performance and vertex usage according to the sampling angle with $r_{max} = 9m$.

Chapter 3

Robust Navigation Control and Synchronization of Networked Robots

3.1 Introduction

Maintaining robust robot control is a critical issue for multi-robot systems. A robust control is one that has a proper scheme to recover from full or partial malfunctions of the robots in events such as communication loss or mechanical breakdown. Robustness must be present for multi-robot collaboration especially in communication loss, robot malfunction and dynamic environment. In this chapter, we present a robust control and synchronization scheme for multi-robot collaboration in which heterogeneous mobile robots perform inter-dependent tasks temporally and spatially to complete a given objective by optimally assigning tasks for available resources whenever resource availability is dynamically changed. A centralized and communication-based scheme is used for task executions and dynamic formation of robot teams when detecting any failure and recovery of robots, and the new addition and removal of robots from the teams. This scheme can generally be applied to a system that utilizes Server-Client or Master-Slave modeling.

3.2 Application Model with Two-Level Hierarchical Network

3.2.1 Application Model

Figure 3-1 illustrates the application model that consists of two layers of network for reliable communication for collaboration. Since the communication network of robots is wireless based, we are required to ensure all robots are under the coverage of the underlaying wireless network. This is a difficult task as areas that robots are planned to cover can be much larger than the coverage area of wireless communication network can reliablty cover. The model proposed in this chapter introduces master robots that relay messages from a central server to their respective slave robots. The master robots not only communicate with the server but also relay messages to another master robot forming an ad hoc network in case when a master robot is not able to establish the communication link with the central server. This model assumes that master robots are always in positions that guarantee at least one master robot having a reliable communication with the server and other master robots have

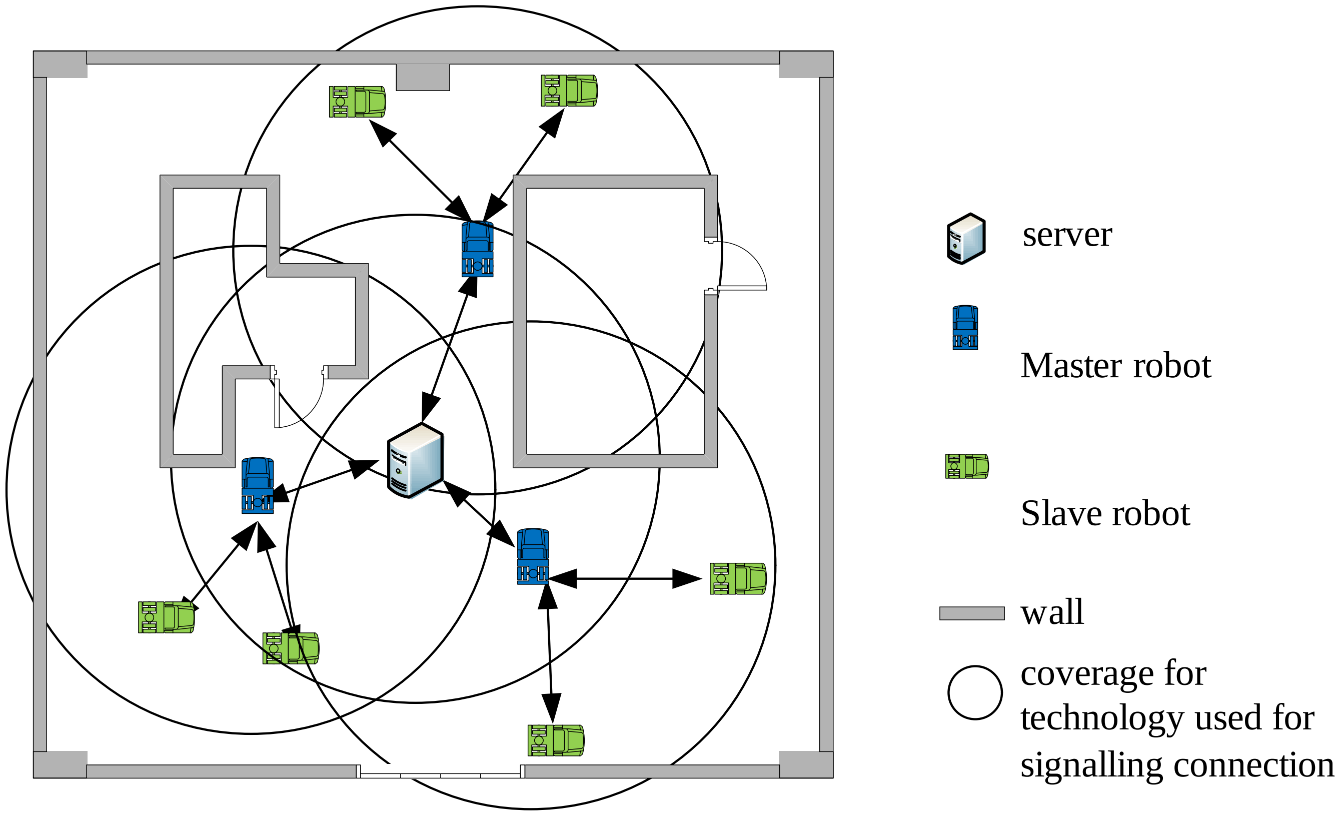


Figure 3-1: Application model

a communication link with at least one other master robot. Slave robots maintain distances with their master robot not to loose their communication links. T With this model, as long as a master robot is within the coverage area of the communication network, the entire fleet of the robots can connect and exchange messages with the central server. The central server provides the actual robust control processing to the robots as well as assigning tasks. Depending on the task the robots need to perform, the master robots can also execute the tasks in addition to their role as communication repeaters for the wireless network. We denote a master robot *MR* and a slave robot as *SR*. Individual master and slave robots are denoted as MR_i and SR_j respectively. Here, i^{th} represents master robot ID and j^{th} represents slave robot ID. Each robot is wireless communication capable, and all behaviors of the robots are controlled by the server. A group consists of one master robot and multiple slave robots. This is a classical *divide and conquer* method to efficiently cover a large area. The server can also be a robot to provide an additional layer of communication reliability by relocating itself to ensure the reliable network. Since the server has the knowledge of exact grid locations of the robots, it can simply move within the communication range of the nearest master robot. The robot groups navigates such a way that each group can communicate with at least one other group.

Let $\mathbf{MR}^{ij}$ and $\mathbf{SR}^{ij}$ denote as the position of MR and SR with i^{th} row and j^{th} column, where $1 \leq i \leq M$ and $1 \leq j \leq N$. Given a map, the whole area is equally divided into $M \times N$ grids. More specifically, $\mathbf{MR}^{ij}$ and $\mathbf{SR}^{ij}$ are denoted as $(MR^{ij,x}, MR^{ij,y})$ and $(SR^{ij,x}, SR^{ij,y})$ in the 2-D cartesian coordinate.

We model the relationship between of master robot position and slave robot position as a directed graph $G(\mathbf{MR}, L)$, where $\mathbf{MR}$ is the set of master robot position and L is the set of directed links. Each node u $\in$ $\mathbf{MR}$ is assumed to know its own position as well as that of its neighbors. An edge (u,v) $\in$ L iff v satisfies condition, v $\subseteq$ L. If (u,v) $\in$ $\mathbf{MR}$ and (v,u) $\nsubseteq$ L, we say that (u,v) is unidirectional. Otherwise (u,v) is bidirectional. Let L_k denote the set of neighbors of node k.

$$L_k = \{l | \sqrt{(k-l)^2} \leq d, k \in \mathbf{MR}, l \in \mathbf{SR}\} \tag{3.1}$$

where d is the radius of wireless coverage of master robot.

The matrix $task^{MR}$ represents M $\times$ N mutually exclusive tasks for the master robots. Since the application does not consider interference between the base stations, the transmitter is turned on with some time interval between the base station. $task_{ij}^{MR}$ is a task at $\mathbf{MR}^{ij}$.

$$task^{MR} = \begin{pmatrix} task_{11}^{MR} & ... & task_{1N}^{MR} \\ ... & ... & ... \\ task_{M1}^{MR} & ... & task_{MN}^{MR} \end{pmatrix} \tag{3.2}$$

The matrix $task^{SR}$ represents M $\times$ N independent tasks for the slave robots. $task_{ij}^{SR}$ is a task at $\mathbf{SR}^{ij}$.

$$task^{SR} = \begin{pmatrix} task_{11}^{SR} & ... & task_{1N}^{SR} \\ ... & ... & ... \\ task_{M1}^{SR} & ... & task_{MN}^{SR} \end{pmatrix} \tag{3.3}$$

$task_{ij}^{MR}$ is temporally and spatially inter-dependent on $task^{SR}$ such that L_{ij} satisfies because $task_{ij}^{MR}$ at $\mathbf{MR}^{ij}$ is prerequisite to $task^{SR}$ at $\mathbf{SR}^{kl}$.

The following assumptions are made to further investigate the topic on hand.

- There are entirely known tasks and environment(e.g. map) to complete the mission.

- The map to be covered is divided into grids, which is the smallest unit that each robot horizontally and vertically moves between unblocked adjacent grids. The multiple robots can occupy one grid at the same time without blocking each other.

- Presenting a multi-robot coverage algorithm [12] for path planning is beyond the scope in this chapter.
- The system is constituted by cooperative multi-robot teams where a team is composed of a master robot and more than one slave robots.
- The robot for dynamic role assignment can either be a master or a slave robot.
- The robot has a limited wireless radio range, and the communication link may become unreliable.
- The robot can self-localize its position.
- The robot does not know behaviors of other robots.
- A centralized server controls behaviors of multi-robot teams, and a master robot controls all of its slave robots in each robot team according to the centralized planning.

Since there may be events that prevent the robots from executing their tasks, the probability of the events affect the overall system performance. For a robust control and synchronization scheme, we consider failure detection, recovery of the robots from failures, and system reconfiguration for task or role reassignment. In our approach, the goal is to complete the mission with a maximum rate of operation wit available resources and a minimum mission completion time.

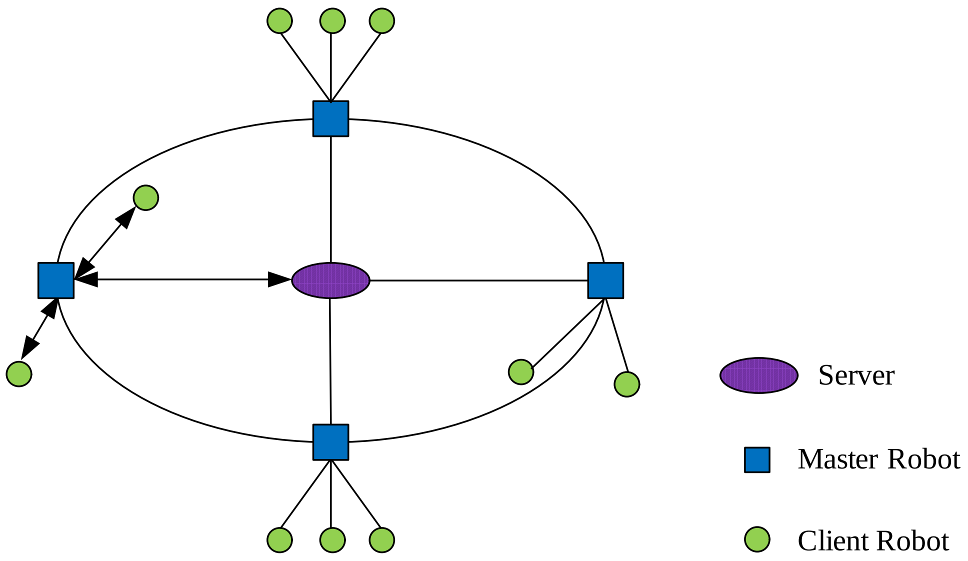


Figure 3-2: two-level network model

3.2.2 Network Model

The network topology under consideration in this chapter is a multi-robot network composed of a server, master robots and slave robots. The multi-robot system adopted here is composed of a double-tier relaying system. In this system, data from server is distributed and relayed to each slave robot via a master robot and data from each slave robot is combined and relayed to the server via a master robot as well. Figure 3-2 illustrates the overall network topology. Since every node in this network can have mobility at any given time, the distance between each node needs to be maintained such that the connectivity between the nodes are always present.

3.3 Map Based Navigation and Basic Synchronization

3.3.1 Map Based

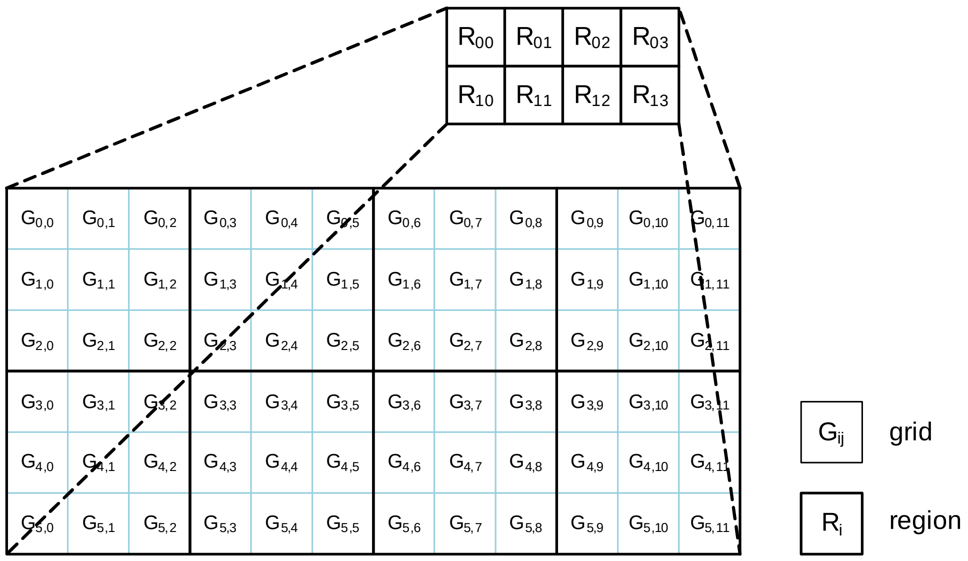


Figure 3-3: floor map

Given a floor map, we divide the area into the subareas. We call a subarea as a region where an allocated robot cluster executes tasks. The map is divided into N_R regions. We denote region R_{ij}, where i and j are the indices of the rows and the columns respectively. The size of a region is to be determined such that a master robots in any grid within the region can communicate with slave robots in any of the grids in the region. A region is represented as a square as shown in figure 3-3. Each region is again divided into a number of grids. Each robot navigates the entire map on a grid by grid basis. Let's denote grid G_{ij}, where i and j again are the indices of the rows and the columns respectively. The grids can be either active G_{ij} or inactive

G'_{ij}. In an active grid, the robot performs the specific tasks such as transmitting or measuring RSSI values as will be discussed in Chapter 4. The robot uses inactive grids only as a path and passes through without performing any task.

If the number of robot clusters is less than the number of regions, the formation of robot clusters is important because tasks in one region may be dependent on tasks being performed in other regions under a collaborative task environment. In this case, we need to move the robot clusters whenever they finish tasks in the assigned region to the regions that have not been visited for tasks. We denote the number of repetition as N_r. We suppose that $N_{MR} = 4$, N_{SR} =6 and N_R=8. Since N_{MR} is less than N_R, we can assign each robot team to an unvisited regions individually. However, this may not always possible as the communication links need to be maintained as the robots teams move to another regions. There may be a time when a robot team needs to stay in a region where they have already completed the tasks assigned to them. In other words, a number of regions each robot team visits may not necessarily be N_R/N_{MR}. Also, if the number of grids within a region is more than N_{SR}, some robot teams may have to wait as each region may present different navigation condition even if the grid size is the same. This would also require careful synchronization from the central server.

3.3.2 Basic Synchronization Scheme

Initialization procedure

Given divided regions and available resources, the robot formation is decided by making each robot cluster with a region assigned. The centralized task planner, *Path Planning block* determines the optimal task scheduling for all robots. We will not discuss the optimal task scheduling in this chapter because it is out of the scope for the topic in this chapter. The format of task scheduling, input script format are shown in table 3.1 and 3.2. The input script format will be used in one region if there is no change in resources.

Table 3.1: Input Script Format for Master robots

Master robot	n_{SR}	SRID		periodicity	R	N_G^{MR}	P_0	active	...	P_8	active
1	2	1	2	none	R_{00}	8	$G_{0,0}$	1	...		
2	1	3		none	R_{01}	9	$G_{0,3}$	1	...	$G_{2,5}$	1
3	1	4		none	R_{10}	6	$G_{3,0}$	1	...		
4(N_{SR})	2	5	6	none	R_{10}	6	$G_{0,0}$	1	...		

Table 3.2: Input Script Format for Slave Robots

Slave robot	periodicity	R	N_G^{SR}	P_0	active	...	P_5	active
1	acyclic	R_{00}	5	$G_{0,2}$	1	...		
2	cyclic	R_{00}	4	$G_{1,1}$	1	...		
3	cyclic	R_{01}	6	$G_{0,4}$	1	...	$G_{0,5}$	1
4	cyclic	R_{10}	6	$G_{4,0}$	1	...	$G_{5,0}$	1
5	acyclic	R_{11}	5	$G_{3,5}$	1	...		
6(N_{SR})	cyclic	R_{11}	4	$G_{4,4}$	1	...		

Each Master robot has n_{SR} number of associated Slave robots. The SRID are the ID for each slave robot associated. The scheduled tasks for a robot is called a path sequence, $\mathbf{P} = \{P_0, P_1, ..., P_n\}$. The parameter, periodicity is how the path sequence is to be repeated. i.e. periodicity = none/cyclic/acyclic. For a Master robot, periodicity is always *"none"* because the Master robot does not repeat the path sequence in one region. If the parameter, active is 0 for grid G_{ij}, the Master robot instruct the slave robots to do nothing at their current grids. For slave robot related information, the parameter, periodicity specifies how to repeat path sequence whenever the master robot changes its position. For slave robot, periodicity could be cyclic ,acyclic, or none based on the type of tasks they perform. If it is cyclic, the path sequence is repeated in forward direction. If it is acyclic, path sequence is repeated in reverse direction. This is because path sequence for slave robot may be repeated to reduce uncertainty in the result of task execution. If the parameter, active is 0 for grid G_{ij}, the slave robots does not perform any task at this grid. The limitation of this script format is as follows.

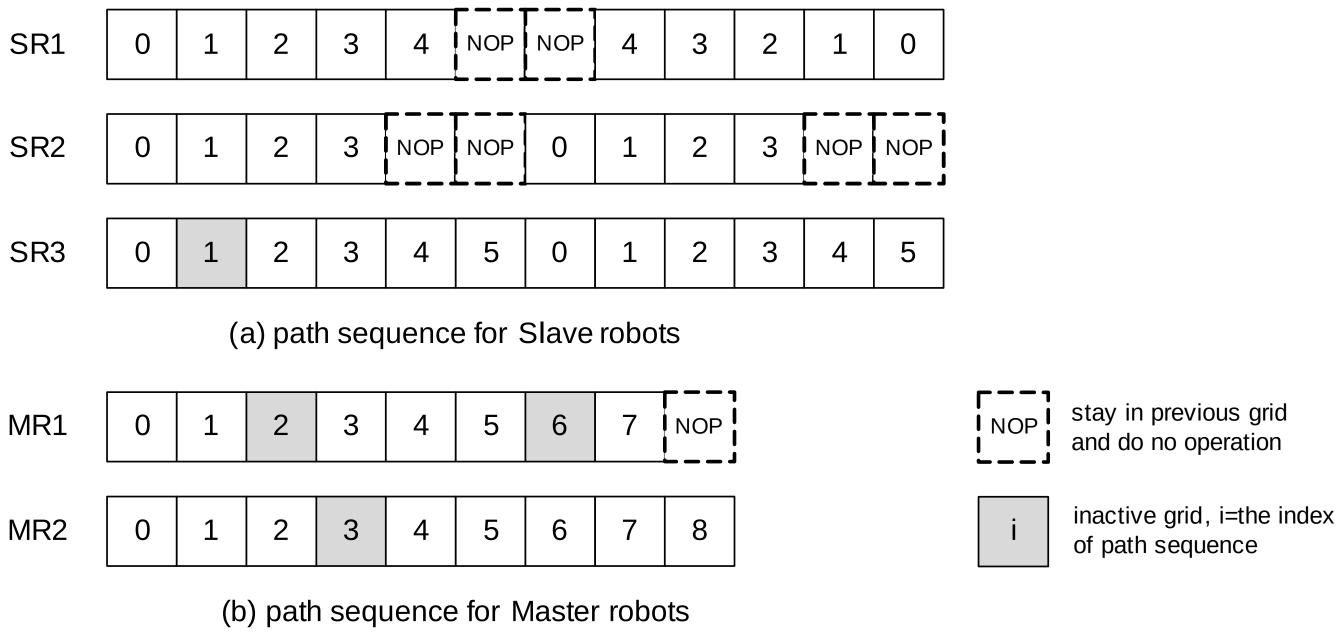
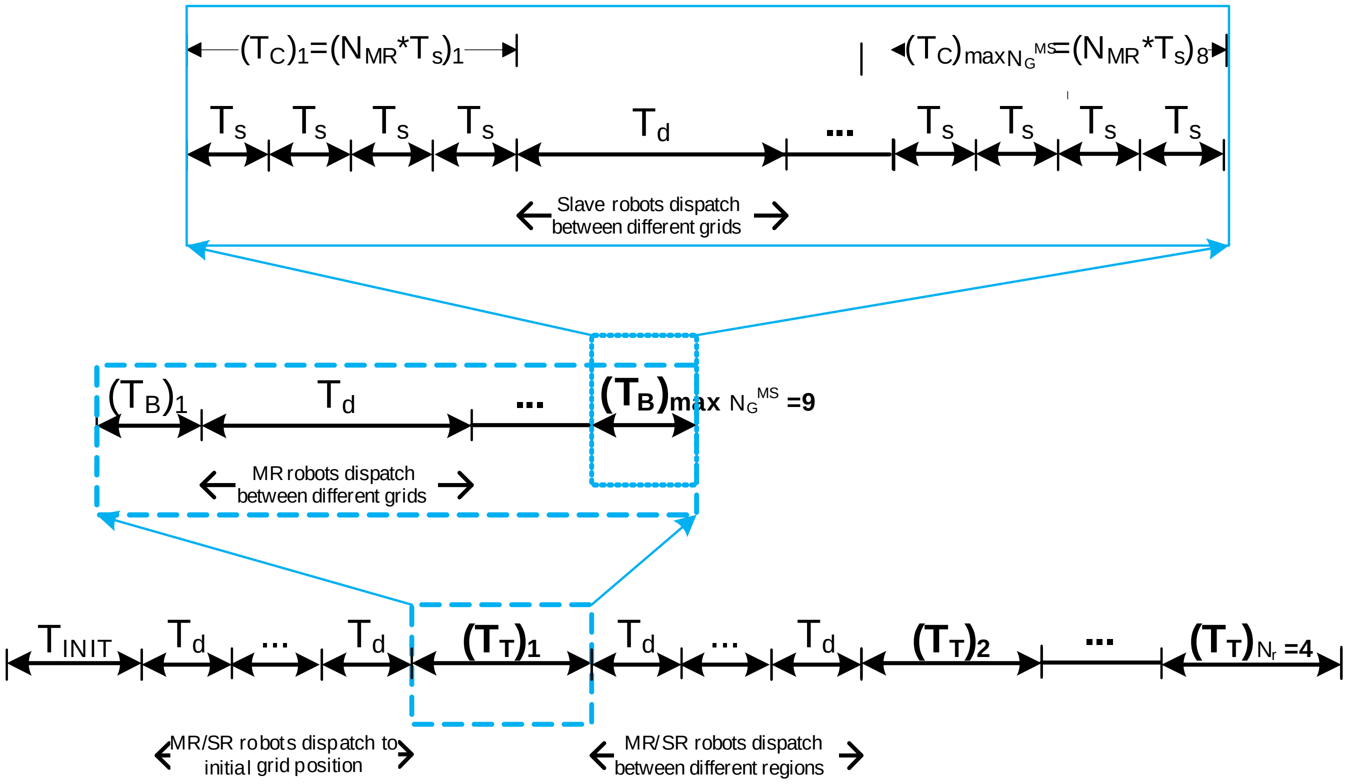

SR1
0 1 2 3 4 NOP NOP 4 3 2 1 0
SR2
0 1 2 3 NOP NOP 0 1 2 3 NOP NOP
SR3
0 1 2 3 4 5 0 1 2 3 4 5
(a) path sequence for Slave robots
MR1
0 1 2 3 4 5 6 7 NOP
MR2
0 1 2 3 4 5 6 7 8
(b) path sequence for Master robots
NOP
stay in previous grid and do no operation
i
inactive grid, i=the index of path sequence

Figure 3-4: different path sequence among robots

Figure 3-4 illustrates how to control different sizes of path sequences assigned to the master and the slave robots. If a robot has a shorter path sequence than other robots, it remains in the previous grid in the path sequence for this robot and does not execute any task.

Timing

Given the input script format in table 3.1 and 3.2, each master robot controls all slave robots in the same cluster according to the scheduled tasks from the server to execute tasks among robot clusters in parallel. Suppose that there are N_{MR} master robots and N_{SR} slave robots. Figure 3-5 illustrates the overall timing diagram. When the system is to begin performing tasks, the system initializes a parameter, T_{INIT} required to execute mission as discussed in section 3.3.2.



$(T_C)_1=(N_{MR}*T_s)_1$
$(T_C)_{maxN_G^{MS}}=(N_{MR}*T_s)_8$
T_s
T_d
...
Slave robots dispatch between different grids
$(T_B)_1$
$(T_B)_{max}$ $N_G^{MS}=9$
MR robots dispatch between different grids
T_{INIT}
$(T_T)_1$
$(T_T)_2$
$(T_T)_{N_r}=4$
MR/SR robots dispatch to initial grid position
MR/SR robots dispatch between different regions

Figure 3-5: timing

Let T_C denote the time duration that all slave robots execute their tasks for entire cluster. T_S represents the time it takes for each slave robots to finish their tasks. Though T_C and T_S are specified separately to generalize the concept of the synchronization strategy used, if tasks can be performed in parallel at all grids at the same time, T_C and T_S will have the same value. The control signalling for each master and slave robot uses the round robin scheduling to prevent any potential signalling conflict.

$$T_C = N_{MR} \times T_S \tag{3.4}$$

where T_S is the time duration for which the master robot instruct the slave robots for tasks. The slave robots perform their tasks and send the results of their tasks to the server. After T_C , all slave robots move to the next grid in the path sequences. The dispatch time for the slave robot from grid index j to j+1 in region i is denoted as $T_{d,SR_i}^{(i,j),(i,j+1)}$. Let T_B denote the time duration for which the slave robots finish perform their tasks at all grids in the region. The master robots doe not move for T_B.

$$T_B = \sum_{j}^{\max N_G^{SR}} \{T_C + \max T_{d,SR}^{(i,j),(i,j+1)}\} \tag{3.5}$$

where $\max N_G^{SR}$ is the maximum number of grid for slave robots, $T_{d,SR}^{(i,j),(i,j+1)}$ is dispatch time for slave robot to move from grid index j to j+1 in region i.

Let T_T denote the time duration that the slave robots finish to execute their tasks by navigating through all grid positions whenever the master robot move to a new grid position in one region.

$$T_T = \sum_{j}^{\max N_G^{MR}} \{T_B + \max T_{d,MR}^{(i,j),(i,j+1)}\} \tag{3.6}$$

where $\max N_G^{MR}$ is the maximum number of grid for the master robots. $\max T_{d,BS}^{(i,j),(i,j+1)}$ is the maximum time for all master robots dispatch between grid j and j+1 in region i.

Let T_{total} denote the time duration to complete the overall procedure.

$$T_{total} = T_{INIT} + \sum^{N_r} \{T_T + \max T_d^{(i,j),(i+1,j)}\} \tag{3.7}$$

where $\max T_d^{(i,j),(i+1,j)}$ is the maximum dispatch time for the master and the slave robots moves between region i and i+1.

Our proposed timing diagram is based on a centralized approach. ”Centralization” means all movements and task executions are directed by and reported to the central point of control. Thus, the information stored in the central place can be quickly analyzed and used for the next course of actions for the robots without needing to collect any additional information from the system components. With the timing diagram, once can optimally design task plan based either on time constraints or a number of robots available. Also, optimal compromise can be obtained by mixing right amount of time for a set of task and a number of robots. The timing diagram can fundamentally be used at the planning phase of multi-robot collaborative tasks. The centralized approach also provides robustness, consistency, failure detection, recovery, reconfiguration, and flexibility.

3.4 Robust Control

3.4.1 Failure detection and handling of robot

Failure detection and handling is extremely critical for robust navigation control and task synchronization. The system needs to detect failures in robots and properly handle the failures to continue on with the tasks. We can categorize three main failure types as communication failure between robots, dispatch failures with time constraint and task execution failures in this model. The decision for slave robot failure is made from the server's point of view. We also assume that the slave robot cannot detect its own failure whether the failure is partial or complete in nature.
The communication failures can result in impairment in the system either temporarily or permanently. They can be caused by link failures or by the failures in the communication module of the robot.

Figure 3-6(a-1)(b-1) illustrates the signalling scheme between the master robots, the slave robots, and the server. There is an acknowledgement for every message exchanged. The slave robot only responds for request from its master robot. To minimize the execution time of task execution and dispatch task, only the signalling is based on round robin scheduling. The signalling has two parts, task request and task result query. Even though 1st or 2nd acknowledgement between the master robot and the slave robot is not recevied by the master robot, the master robot continues to

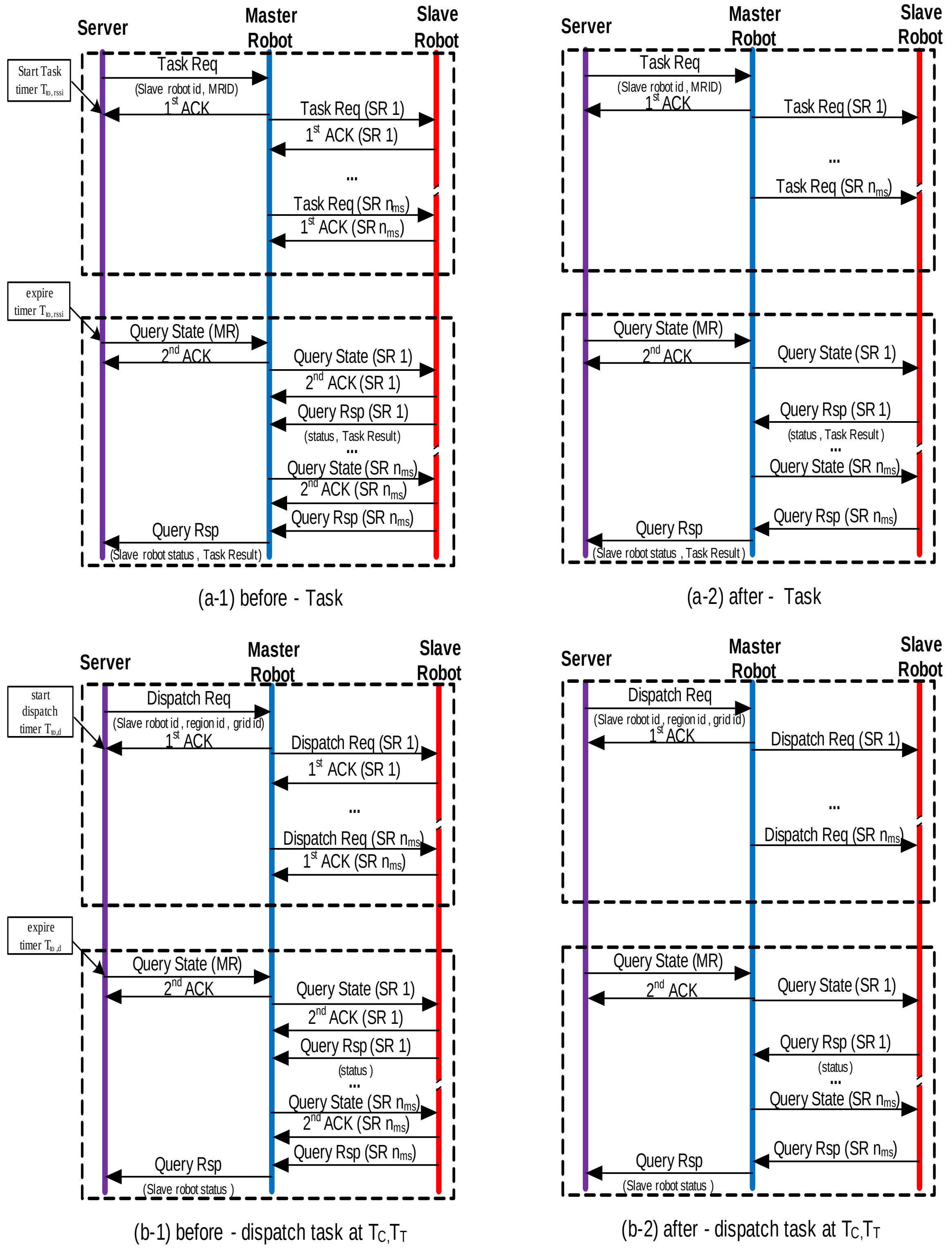


Figure 3-6: Control Signalling Scheme for Master and Slave Robots

signal the slave robot because the acknowledgement failure may be temporary. The temporarily missing acknowledgement does not affect task completion. Thus, we can modify the signalling as in figure 3-6(a-2)(b-2) by removing the acknowledgement such that the slave robot only responds for the requests. By using the modified signalling, the communication failure can be detected by checking response for querying task result. The round robin scheduling for signalling has advantage in that its robustness by reducing signalling conflict in master robot helps minimize undesirable failures. However, it increases task completion time due to a time delay between the task completion time and the task result query time.

Another type of failures is the failure in dispatching the robots to their destinations within a certain time constraint. This failure can occur when there are some obstacles in the path or mechanical breakdowns. The time constraint is set in the server to avoid robots being stuck in dead lock as in figure 3-6(b-2). Since the robot cannot deviate from the path sequence to avoid obstacles, it stops whenever obstacle blocks. The slave robots can localize its position relative to the source and the destination and reports it to the master robot when transmitting task result.

A task execution failure can occur when either the device for executing the task becomes unusable or any condition that prevents the task being completed.

Communication Failure Between Master Robot And The Associated Slave Robots

The time slot is not changed if there is no inter-dependency between the slave robot tasks. If there is inter-dependency between the slave robot tasks after communication

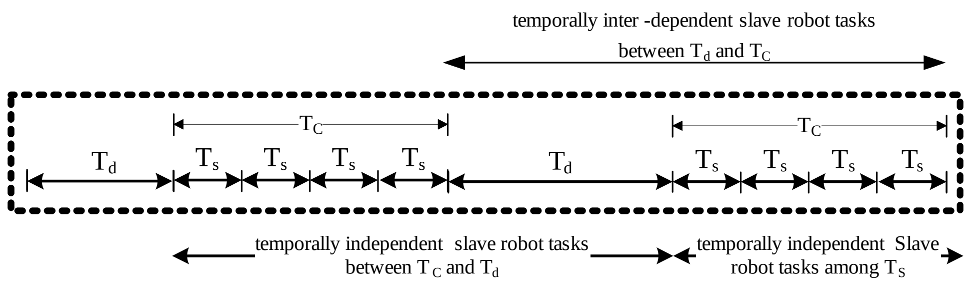

temporally inter -dependent slave robot tasks between T_d and T_C
T_C
T_C
T_d
T_s
T_s
T_s
T_s
T_d
T_s
T_s
T_s
T_s
temporally independent slave robot tasks between T_C and T_d
temporally independent Slave robot tasks among T_S

Figure 3-7: Timing: Temporal Dependency Among Slave Robot Tasks

failure is detected, the slave robot with communication failure is considered as a failed slave robot. The time slot for the failed slave robot remains empty until the communication between the master and the slave robot is re-established. The empty time slot signifies that there is no signalling messages for the slave robot. As shown in figure 3-7, there is no inter-dependency among tasks within T_C and the following dispatch task for T_d. However, there is inter-dependency among ensuing dispatch tasks for T_d and the following tasks within T_C.

For all failure detection or the empty time slots for T_S, the missing tasks and the inter-dependent tasks are recorded for error-handling. Also, the slave robot considered as failed at T_d is recorded for resource re-allocation. The task of the slave robot in one grid becomes temporally and spatially inter-dependent on task of the master robot when the slave robot is within the master robot coverage as shown in figure 3-8.

It is obvious that minimizing the possibility of communication failure should be the primary goal when designing a network system to minimized the number of missed tasks. To minimize the possibility of the link failures, we consider tight coordination of the robot formation while executing the mission. For example, by using multiple routing path between the master and the slave robots such that there are multiple

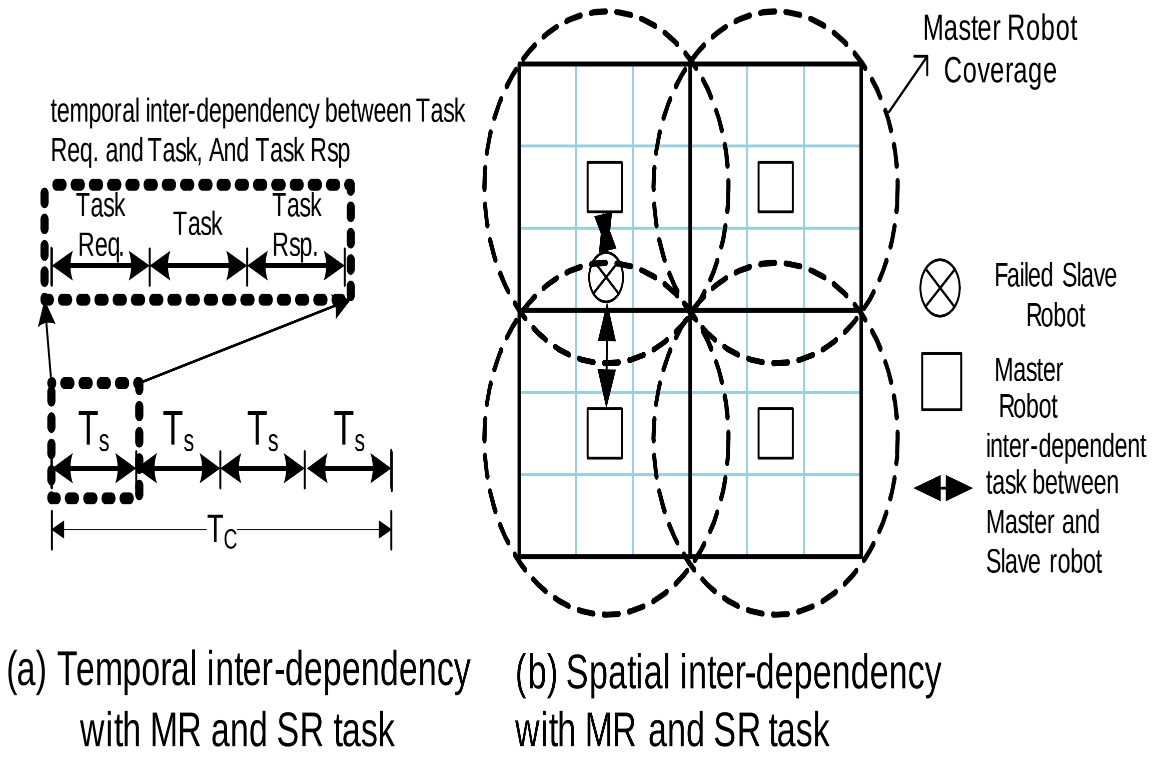

temporal inter-dependency between Task Req. and Task, And Task Rsp
Task Req.
Task
Task Rsp.
T_s
T_s
T_s
T_s
T_C
Master Robot Coverage
Failed Slave Robot
Master Robot
inter-dependent task between Master and Slave robot
(a) Temporal inter-dependency with MR and SR task
(b) Spatial inter-dependency with MR and SR task

Figure 3-8: Temporal and Spatial Inter-dependency between Failed Slave robot and Master robot task such that Slave robot is within Master Robot Coverage

slave robots in the same cluster, we decrease communication failure rates by retransmitting packets through different routes. Even with the very careful communication link design, we can not guarantee the fail-safe communication links. With a certain rate of communication failures, we need to effectively handle the failure cases.

Dispatch Failure With A Time Constraint

If dispatch task is considered as failed, the time slot for the slave robot is empty after the failure detection because of the inter-dependency between the dispatch task and the following task. Even though the dispatch task failed, the slave robot is not considered as a failed slave robot because the robot is still available for reassignment for another task. Every empty time slot for T_S, the task and inter-dependent task for the master robots need to be recorded as missing tasks and is rescheduled. To increase the operation rate of the resources, the task rescheduling needs to be performed immediately after the dispatch failure is detected. The task within T_C is completed because of inter-dependency among dispatch task and the following tasks. If the slave robot fails to dispatch until T_T, it is considered as failed because it is not an available resource anymore and the slave robot is no longer in the active measurement cluster where the tasks are being executed.

Even though the progress of dispatch task is known by monitoring the relative position of the slave robot between the origin and the destination, it is difficult to calculate the arrival time at the destination when the robot arrives at the destination due to the non-ideal navigation modules on the robot as well as the path conditions. It is very difficult to come up with an optimum solution for minimizing dispatch failures

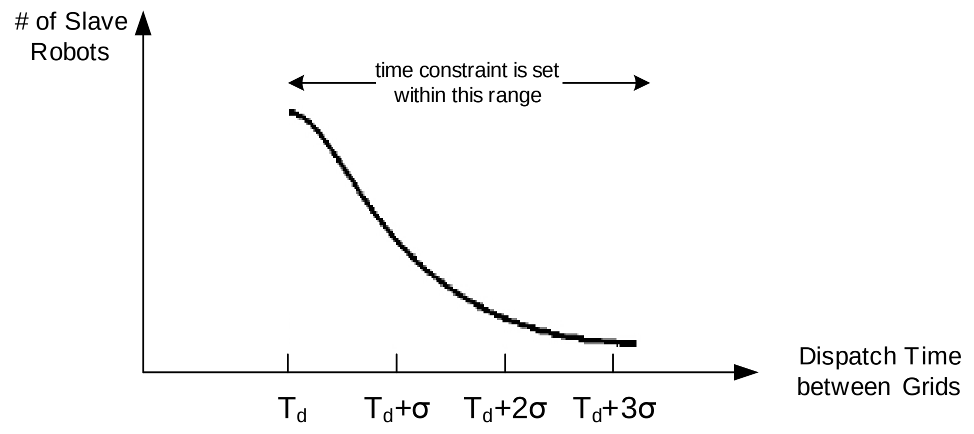


Figure 3-9: Normalized Gaussian Distribution of Dispatch Time between Grids

with respect to the total completion time required. Thus, there is a tradeoff between minimizing dispatch failures by increasing the time constraint and increasing the total completion time. We can increase the time constraint, by repeating the additional signaling to check task results n times according to the progress of dispatch task. This is illustrated in figure 3-10. The time constraint is also dependent on the grid size and the speed of the robots.

Task Execution Failure

If task execution fails, the time slot within T_C is not changed because the task within T_C is independent in figure 3-7. When the slave robot is considered as failed after T_C, the following time slot is empty until its functionality to execute a task is recovered.

To manage resources, it is required to have information such as robot id, failure type and the current position whenever a slave robot is considered as failed at T_d. To handle missed tasks, we need to store information such as the grid index of the master robot and the failed slave robot whenever missed tasks occur at T_S. To store

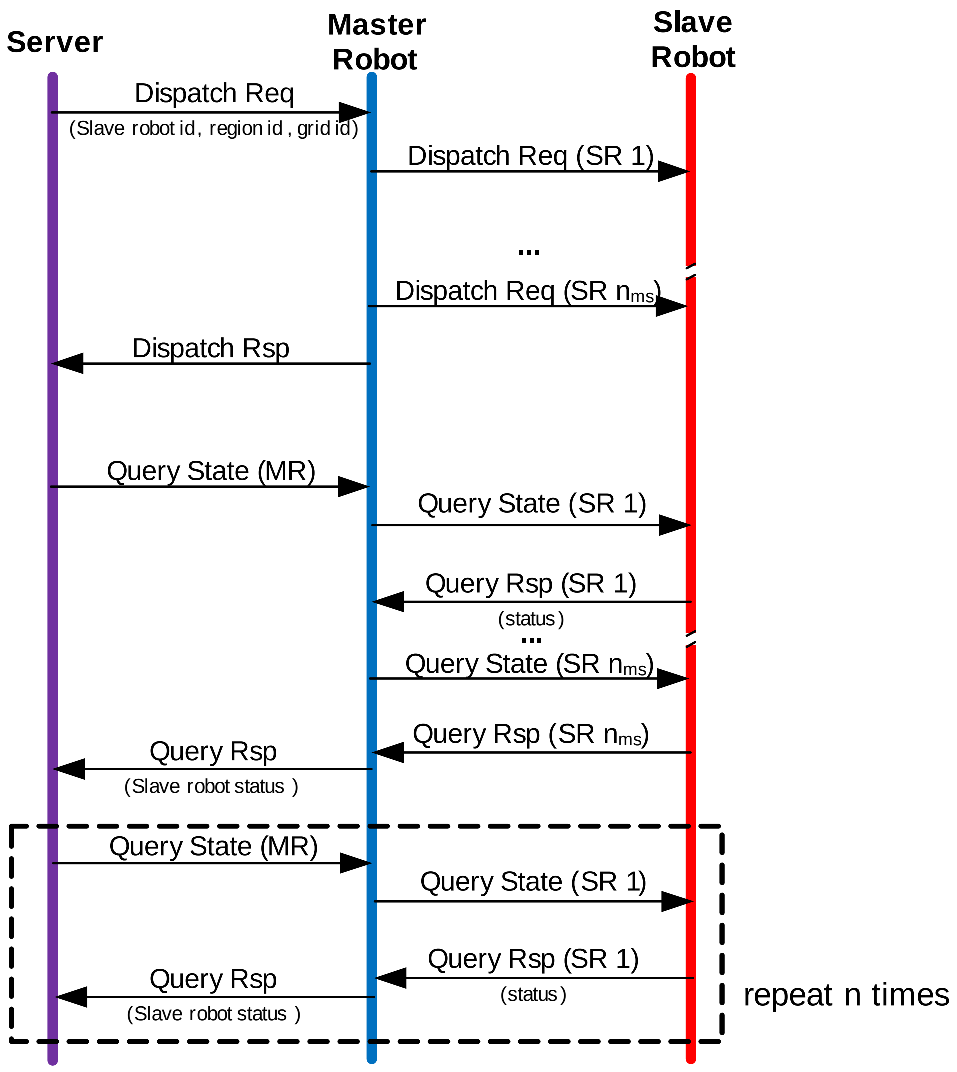


Figure 3-10: Modified Signalling for Dispatch Fail with Time Constraint

the missed tasks, the input scripts for the master and the slave robots in table 3.1 and 3.2 are duplicated. The missed tasks for the master and the slave robot are recorded as active and completed tasks as inactive. Since the input script represents the scheduled tasks in T_B time unit by repeating it for T_T, if any missed task exists, the duplicated script is generated every T_B for the missed tasks, which show random distribution within T_T. Our mechanism can handle the slave robot failure within T_S time unit because we use different path sequences among the master and the slave robots in 3-4.

Whenever any robot is considered as failed, the server checks for the minimum resource requirement in the system. If the minimum resource requirement in system is not met, the system stops the operation. Under this condition, the server keeps the connectivity between itself and the remaining functional master and slave robots by polling their statuses. This is illustrated in figure 3-11.

3.4.2 Recovery of Robot from Failure

Failed robots need to be recovered if possible when it significantly degrades overall system performance or when it causes the system to stop its operation due to the minimum resource requirement as discussed in previous section. Thus, we need to have a recovery mechanism for robust and flexible resource usage.

As discussed in previous section, since there is inter-dependency between T_d and the following T_C, and the time slots within T_C is not changed even when a failure detected, it is appropriate to use the minimum recovery duration at the end of T_C.

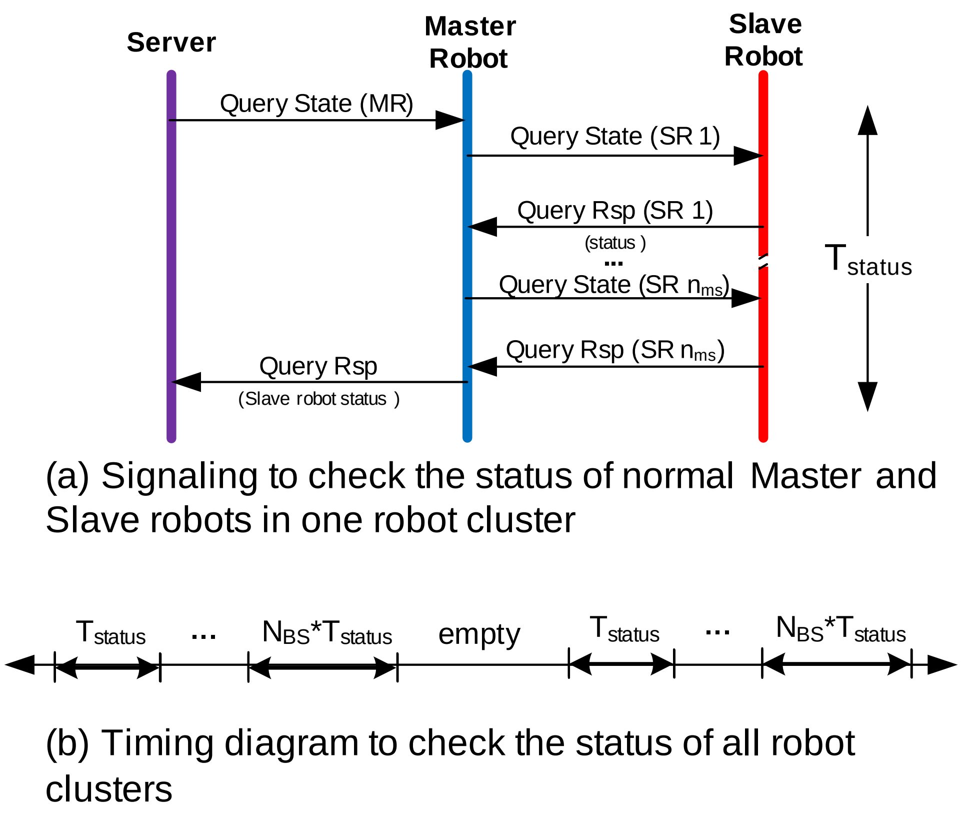


Figure 3-11: check the status of all robot clusters to keep the connectivity

The recovery procedures at the end of every T_C, T_B and T_T do not increase total signalling time compared to no failure cases as illustrated in figure 3-12 and 3-13. We decide to use recovery procedures at every T_C because T_C is the minimum time that the slave robot dispatches between the grids. It also does not degrade overall system performance while increasing the rate of operation of resource.

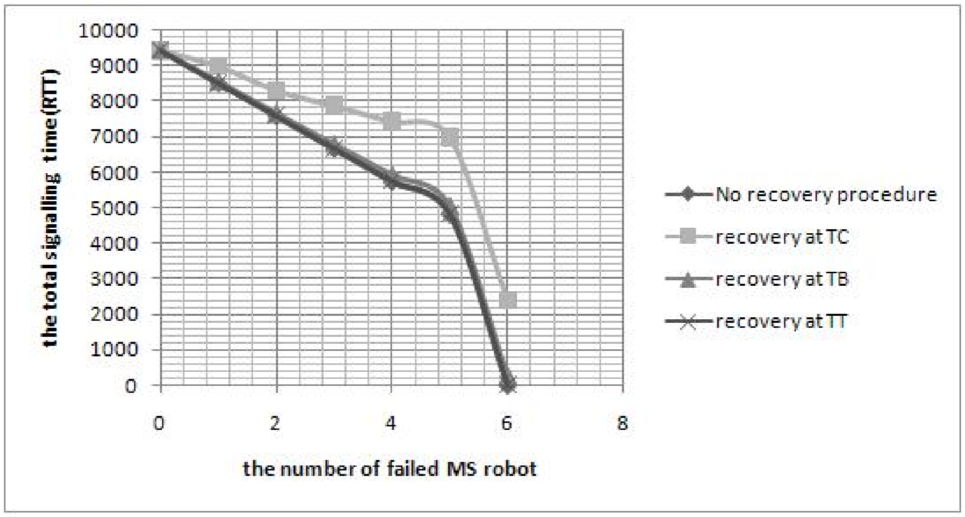


Figure 3-12: Total Signalling Time as Number of Failed Slave Robot Increases without Recovery Mechanism

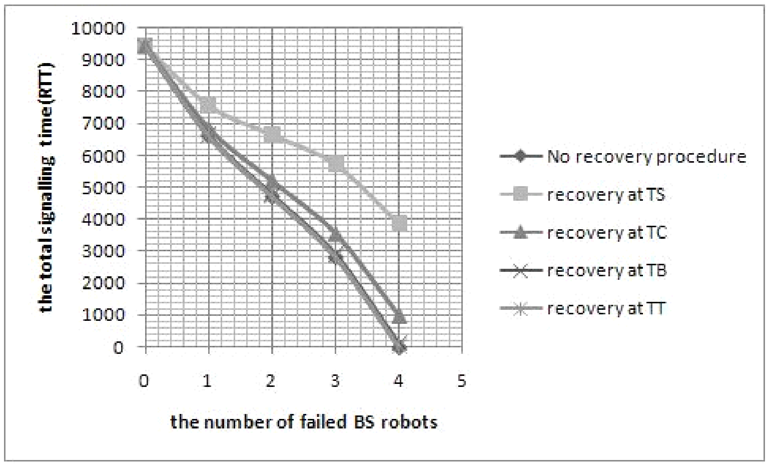


Figure 3-13: Total Signalling Time as Number of Failed Master Robot Increases without Recovery Mechanism

If there are more than one failed slave robot within T_T after a communication failure or a task execution failure is detected, the recovery procedure is initiated at the end of every T_C. The signalling scheme is shown in figure 3-14. We denote the recovery time as $T_{recovery}$. We assume that there is no failure related to the master robot. Since each cluster is independently controlled by a master robot in that cluster, we only consider recovery of failed slave robots in the cluster.

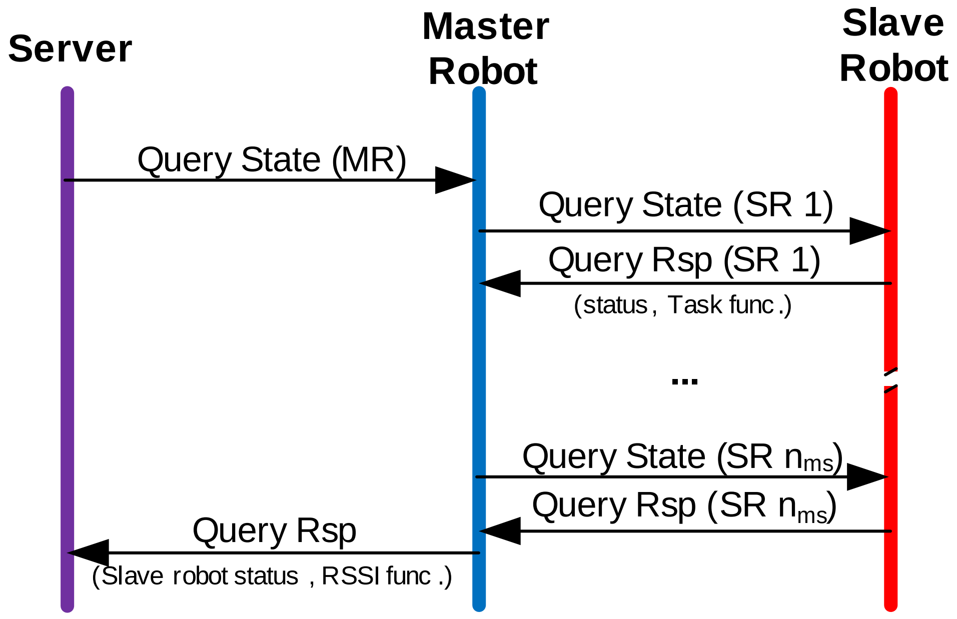


Figure 3-14: Signalling to Recover Multiple Failed Slave Robots in The Same Cluster for $T_{recovery}$

Figure 3-15 shows the timing diagram to continue or abandon the task for recovered robot within T_T in case the failure type is a communication or a task execution failure. If the failed slave robot is considered as recovered by checking the status of the Slave robot for $T_{recovery}$, the decision to continue or abandon the task is made. If the decision is made to continue with the task for the recovered slave robot, to synchronize the tasks with the active slave robots, the recovered slave robot executes the dispatch task to get to the grid where the current grid index indicates. After the dispatch task is completed, the server resumes sending the tasks for the recovered slave robot. If the server decides to abandon the task for the recovered slave robot, its status is updated as available resource while the time slot is kept empty until the task rescheduling for the recovered slave robot is performed. If more than one slave robot are recovered while system is not operational due to minimum resource requirement, the system resumes using the recovered slave robot if the minimum operational resource condition is met with the recovered slave robots.

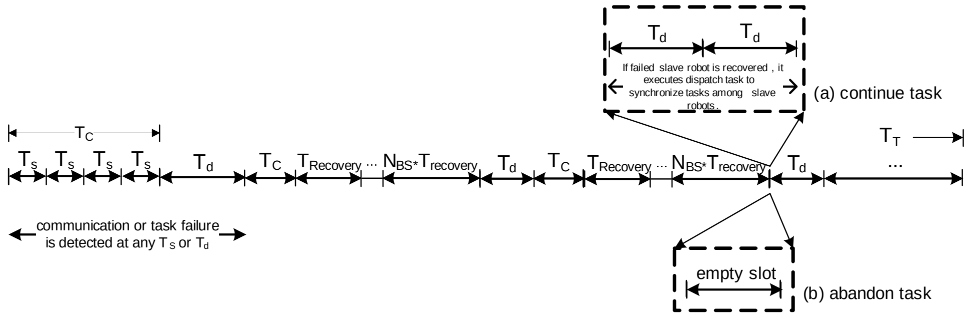

T_d
T_d
If failed slave robot is recovered , it executes dispatch task to synchronize tasks among slave robots .
(a) continue task
T_C
T_T
T_s T_s T_s T_s T_d T_C T_Recovery ··· N_BS*T_recovery T_d T_C T_Recovery ··· N_BS*T_recovery T_d ···
communication or task failure is detected at any T_S or T_d
empty slot
(b) abandon task

Figure 3-15: Timing Diagram to Continue/Abandon Task for Recovered Robots within T_T

If the failed robot is not recovered within T_T, the failed slave robot is considered a member of the robot cluster it used to belong because the scripts are generated at every T_T. Since the failed slave robot is an individual agent, the recovery procedure for the failed slave robot can be controlled by any master robot which can signal slave robots by considering signalling coverage as shown in figure 3-16(a). Otherwise, master robots need some coordination to make routing path between the server and the failed slave robot as shown in figure 3-16(b). We call master robots that makes routing path to the failed slave robot as scouts. While the slave robots act as scouts, the slave robots remain in the active regions because there is no necessary communication within each robot cluster.

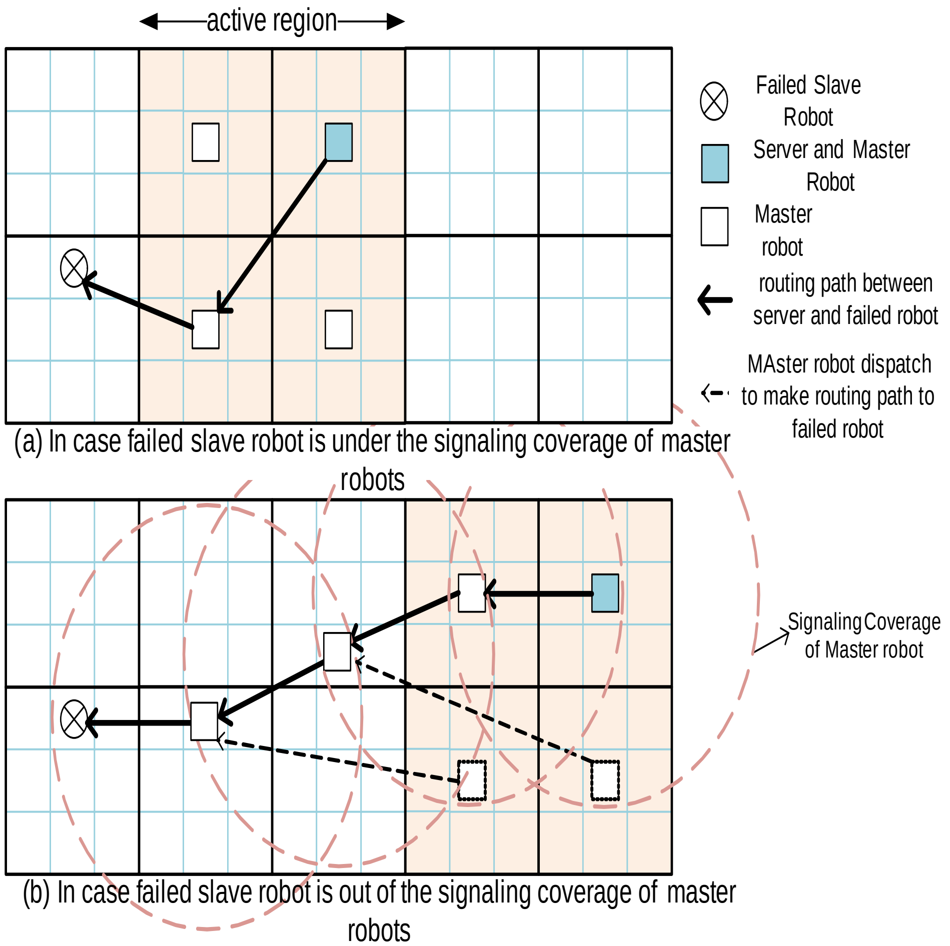

active region
Failed Slave Robot
Server and Master Robot
Master robot
routing path between server and failed robot
MAster robot dispatch to make routing path to failed robot
(a) In case failed slave robot is under the signaling coverage of master robots
Signaling Coverage of Master robot
(b) In case failed slave robot is out of the signaling coverage of master robots

Figure 3-16: recovery scenario beyond T_T

After fulfilling the routing condition to establish the communication link between the server and the failed slave robot, the recovery procedures for the communication and task execution failure are the same in figure 3-14. The recovery procedures for the dispatch failure with a time constraint is different in that we need to check whether the dispatch functionality is recovered by executing a dispatch task. The recovery procedures for the failed slave robot due to the dispatch failure with a time constraint are shown in figure 3-17. If the failed slave robots is recovered within $T_{recovery}$, added to the robot cluster. The task rescheduling for the recovered robot is also needed. Before resuming the basic operations, the recovered slave robot and scouts need to return to their active region. If the recovery procedures for the failed slave robot is not successful, only the scouts return to the active regions to continue with basic operations.

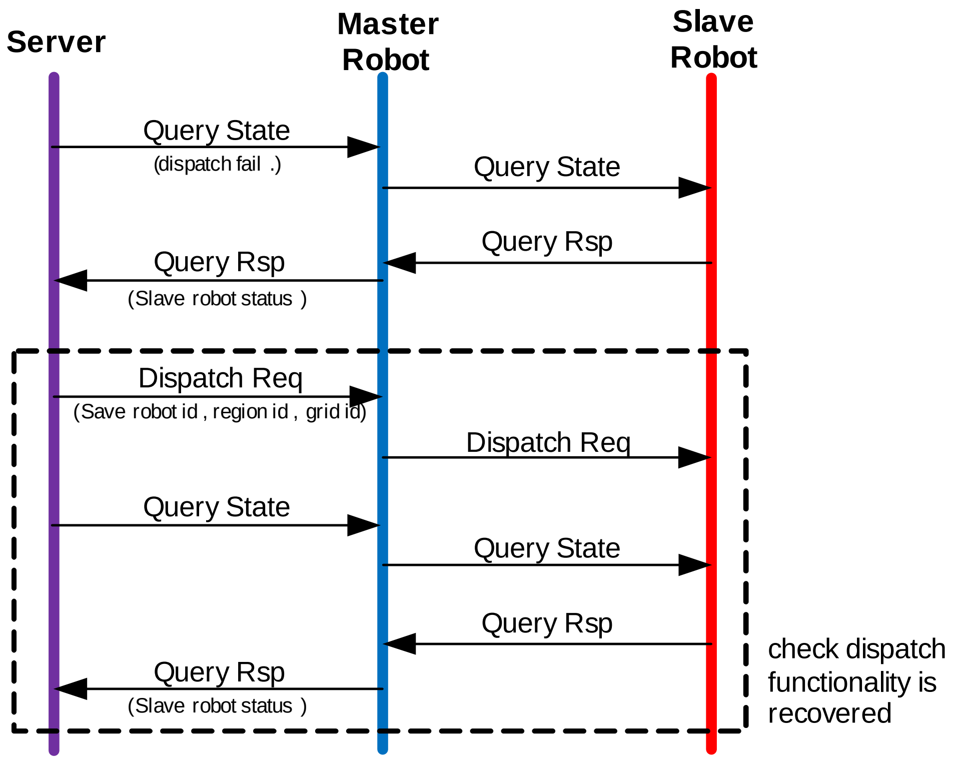


Figure 3-17: Signalling to Recover Failed Slave Robot due to Dispatch Failure with Time Constraint

To use scout robots, it is required that

- The scout robot is to form the signalling coverage of the master robots to relay the recovery message. Normally, more than one scout robot is required to make a routing path to the failed robot.

- We need additional time for the scout robots to form the routing path towards the failed MS robot and also to return to the active regions with the recovered MS robot. This will significantly increase the completion time.

Since the time the master robots spent as the scouts increases the total completion time, it is not possible to perform the recovery procedure at the end of every T_C when the failed slave robot is out of the signalling coverage area of the BS robots. Thus, we need to make a decision to declare the failed slave robot as nonfunctional after T_T or a multiple of T_T.

If the system stops its operation due to the minimum resource requirement with a failed slave robot within T_T after failure detection, the signalling and timing scheme still is the same as in figure 3-11. However, the statuses are checked for all master and slave robots including the failed slave robots periodically. If any of the failed slave robots is recovered with the minimum resource requirement met, the system resumes the operation. If the system is stopped due to the minimum resource requirement and there is failed slave robot beyond T_T after failure detection, master robots immediately acts as scouts for recovery procedure.

3.5 Reconfiguration procedure

In this section, we will explain the reconfiguration procedure in which the dynamic formation of a robot team is performed, and tasks for robots are rescheduled consequently. The reconfiguration can occur when detecting any failure and recovery of robots, and the new addition and removal of robots from teams by human intervention. Also the reconfiguration procedures are performed every T_T because the script including robot tasks is generated at every T_T.

3.5.1 Reconfiguration every T_T for task coverage

Since the task coverage for one master robot is one region, the reconfiguration procedures should be performed whenever the task in one region is finished where the number of regions is greater than the number of the master robots. Whenever the task for each region is finished for each T_T, the path sequence for the next regions needs to be generated for each robot. Suppose N_{MR} is less than N_R. i.e. $N_{MR} = 4$ and $N_R = 9$ in figure 3-3. After the master and the slave robots finish the tasks for the first four regions, the reconfiguration for the next four regions can be started.

3.5.2 Reconfiguration within T_T after resource failure

The objective for reconfiguration due to robot failure is to gracefully degrade overall system performance. i.e. It is to minimize the number of missed tasks and the completion time. If the system stops because the minimum resource requirement decided by application requirement is not met, the reconfiguration is not necessary since even after the reconfiguration the resource shortage will remain.

As discussed in the recovery section, 3.4.2, if the dispatch failure with a time constraint is detected, the immediate reconfiguration to reschedule the tasks for the slave robot needs to occur. The reconfiguration is to reschedule tasks for other slave robots in same cluster as well. If the communication or task execution failure is detected, the reconfiguration at the end of T_B is required to execute the missed tasks whether the failed slaveMS robot is recovered within T_B or not. Since the missed task is inter-dependent on a task request, the reconfiguration for the missed task is performed before the master robot dispatches to the next grid at T_B. To perform the reconfiguration, there should be at least one available slave robot in the robot cluster.

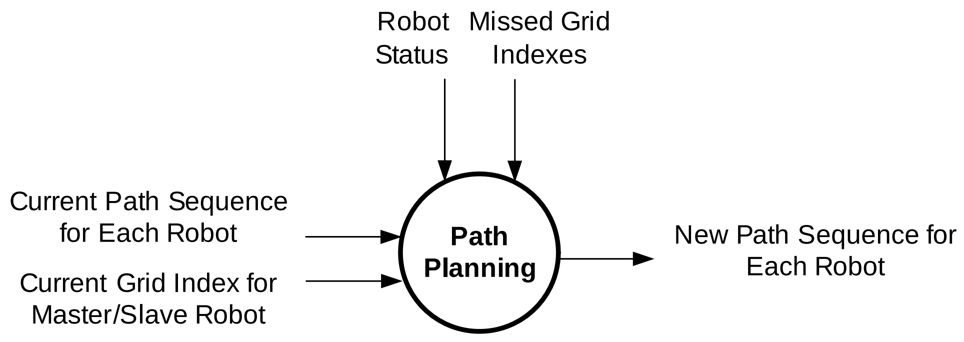

Robot Status
Missed Grid Indexes
Current Path Sequence for Each Robot
Current Grid Index for Master/Slave Robot
Path Planning
New Path Sequence for Each Robot

Figure 3-18: Path Sequence Re-generation

Figure 3-18 illustrates the path planning block which generates new path sequences for available robots based on the current grid indices for the master and the slave robots, the statuses of robots and the duplicated script for missed tasks. Based on these information, the tasks are rescheduled with currently available resources.

If there is no available slave robot, i.e. all slave robots in the cluster have failures, slave robots from other cluster need to reassigned to the cluster. Resource redistribution is discussed in a later section.

3.5.3 Reconfiguration within T_T after resource recovery

The goal of reconfiguration due to robot recovery is to gracefully improve overall system performance. i.e. It is to minimize the number of missed grids and to minimize the completion time. There are two types of recovery to be considered in this section as shown in figure 3-19. If the failed robot is recovered before the reconfiguration procedure and the server determines to abandon the task for the recovered robot, the reconfiguration within T_B after the failure detection is required to execute the missed tasks and to reschedule tasks for robot A. If the failed robot B is recovered after reconfiguration due to the resource failure, the reconfiguration within T_B after recovery to add this resource to robot formation is required and also need to reschedule the task for the resource.

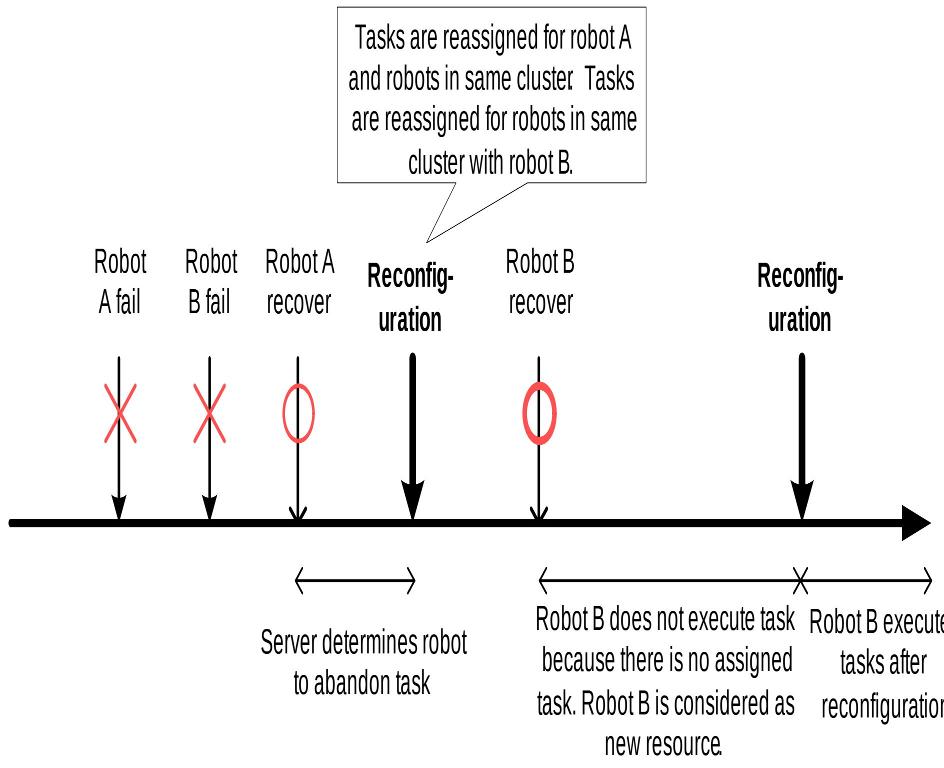

Tasks are reassigned for robot A and robots in same cluster. Tasks are reassigned for robots in same cluster with robot B.
Robot A fail
Robot B fail
Robot A recover
Reconfig-uration
Robot B recover
Reconfig-uration
Server determines robot to abandon task
Robot B does not execute task because there is no assigned task. Robot B is considered as new resource.
Robot B executes tasks after reconfiguration

Figure 3-19: Reconfiguration due to Robot Recovery: Two types of Recovered Robot

3.5.4 Resource utilization and redistribution

The resource redistribution is to gracefully degrade or upgrade overall system performance when detecting any failure and recovery of robots, and the new addition and removal of robots from teams. The following are the cases where the resource redistribution between the clusters can occur.

- There is more than one failed slave robot and no available slave robot in the cluster.
- There are more than one failed slave robot and available slave robots in the cluster, but resource redistribution between clusters are required to gracefully absorb overall system performance degradation.

The Resource redistribution to equally distribute tasks among the slave robots is performed according to the frequency of failures and recoveries because changing the robot formation halts the system and significantly increases the completion time due to the robot dispatch time between the regions. The optimum time to redistribute resource between robot clusters will be analyzed.

3.6 Summary

In this chapter, we presented how to control robots with robustness and to provide synchronization for the system with networked multi-robots. The robust robot control scheme included the failure detection mechanism, the recovery procedures, and the resource reconfigurations. The robust control of the mobile robots that are networked together presents many challenges as many performance degrading events can occur during the system operations. The scheme we presented covered some of the most probable events that may rise for the system and provided a novel approach to resolve the problems with the recovery procedures and the resource redistribution to optimize the system under various failure situations.

We also showed the synchronization method where a global time source was not available. Thus, the chapter has paved the way to implement the networked multi-robot system in places such as airports, hospitals, and government campuses where the robustness and the synchronization not only provides performance improvement but also key requirements for the deployment.

Chapter 4

Utilization of Multiple Mobile Robots for Map Based RF Access-Point Placement

4.1 Introduction

As mobile devices growingly have been replacing traditional landline telephones, adequate in-building wireless network coverage is becoming ever important. Predicting and designing in-building wireless network is much more complicated than the macro-area wireless network design [17]. This is because predicting propagation characteristics in indoor environment depends on parameters that are very hard to approximate if not impossible. These parameters include building materials for all structures within the building, window sizes, and so on [18]. Even with these parameters known, the propagation characteristics can not be acquired precisely since the most of the current design tools utilizes the empirical propagation model that may not provide accurate data for a particular indoor environment [19].

Another method that may be used to design in-building wireless network that have been used in the field is to measure actual propagation loss by walking around in the building with RF signal measuring devices [20]. This method can provide an accurate propagation map of the building. However, to be able to find optimal locations of the base stations, this practice requires many time-consuming iterations as the locations of the base stations changes.

With all the difficulties in predicting propagations in the building, the design based on the prediction needs to be verified as well once the installation of the base stations is finished, necessitating another round of a time-consuming process. In this chapter, we utilize the navigation and robust control mechanism for networked multiple mobile robot described in the previous chapters to overcome constraints such as lack of absolute location information, a global timing source, and data collection efficiency.

The remainder of this chapter has 4 sections. In Section 4.2, we present overview of the system model and background. Also, the problem is described. Section 4.3 discusses data acquisition strategy which includes data collection, synchronization, data representation and coverage analysis . In Section 4.4, an estimation strategy is presented and the effects of various system parameters are incorporated. Finally in Section 4.5, we summarize how the navigation algorithm and robust control mechanism can be applied to an real world application.

4.2 System Model and Problem Description

4.2.1 RFBOT

The model being proposed will use commercially available robots to navigate the building floor and collect RF information. We call these robots "RFBOTs". The RFBOTs will act as either base stations or mobile clients. The Base station RFBOTs (BTS-RFBOT) are the robots that act as BTSes transmitting beacon or continuous wave (CW) signals at a known transmit power level. The Mobile Station RFBOTs (MS-RFBOTS) are the robots that act as mobile stations. The MS-RFBOTs measure the signal strengths back to BTS-RFBOTs for the forward link RF conditions at the current locations. Each BTS-RFBOT is paired with a single or multiple MS-RFBOTs for RF data collection. There also is a central server that monitors and directs the BTS-RFBOTs to navigate in a synchronized way. The server also sends the navigation messages for the MS-RFBOTs via BTS-RFBOTs. The MS-RFBOTs only listens to their paired BTS-RFBOTs. Thus, the communication link between the server and the BTS-RFBOTs as well as between the BTS- and MS-RFBOTs are guaranteed to be reliable.

4.2.2 Grid Based Navigation

Since there are infinite number of points on the map that can be overlaid with signal strength values, the model needs to reduce the number of points. As a solution to this problem, we will use grid based navigation methodology. We partition the floor map with equal-sized grids within which the received RF characteristic from a signal source is assumed to be the same. The RFBOTs will navigate to each grid to measure RF signal strengths at the grid from another grid where the BTS-RFBOT is transmitting as shown in Figure 4-1 The key parameter in a grid based navigation methodology

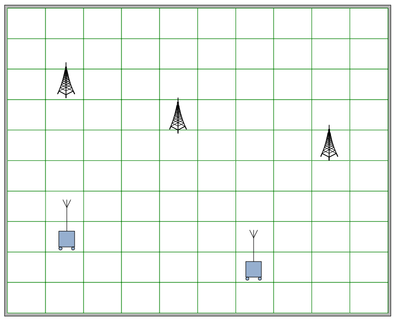


Figure 4-1: Grids with BTS and Client Mobile

is the size of grids. In this model, we will use equal sized square grids for simplicity. The size of the grid is very important in that it cannot be too large or too small. If the size of the grids are too large, the data collected will not properly represent the path loss values for the entire floor. If the sizes are too small, it will take too much time to collect the data for the practical purpose. The proper size of the grid should be based on frequency and resolution of the signal strengths map. The pedestrian speed also needs to be taken into account. For this chapter, we do not present the navigation algorithm again asthe topic has been already discussed in Chapter 2. We will assume RFBOT's operate according to the navigation algorithm.

4.2.3 Two-Tier Network Structure

As mentioned in Section 4.2.1, there are two communication networks required for the model. The first network ensures the controlling messages for the RFBOTs are delivered reliably. The second network is to actually measure the signals transmitted by the BTS-RFBOTS. The requirement of maintaining two networks within the same model presents some challenges in terms of interference and message collisions. The approach to mitigate the difficulty of maintaining two communication networks is two-tier network structure. Tier 1 of the double-tiered network is formed to establish the communication link between RFBOT's. Since the link itself is unreliable, we will construct a communication protocol that would ensure the reliable communication between RFBOTs. The second tier of the network is to send and receive actual RF signals that need to be measured. Tier 1 network is realized using 802.11 WiFi network along with a communication protocol. Tier 2 network will use the air interface technology or CW signal of a frequency that is different from Tier 1 network. In case when the signal measurement is done for the same WiFi frequency of the Tier 1 network, Tier 2 network will use the channel farthest from the channel Tier 1 uses to minimizes the interference.

4.2.4 Cluster of Grids

A BTS-RFBOT is always paried with either a single or multiple MS-RFBOTs. The combination of BTS- and MS-RFBOTs will cover a cluster of grids at a time. A cluster of grids simply is a collection of contiguous grids that a set of BTS- and MS-RFBOTs will navigate within to transmit and measure signal strengths. The size of the clusters depends on the coverage of Tier 1 network as we need to guarantee the network connectivity between the BTS-RFBOTs. If let d_{max} be the maximum distance the BTS-RFBOTs can be separated without losing connectivity, the length of diagonol of a cluster on the floor map should be $d/2$ to ensure the BTS-RFBOTs in the partitions are always connected. When the BTS-RFBOTs move around the map to survey the building, the BTS-RFBOTs maintain the separation distance such that the distance between adjacent BTS-RFBOTS are never separated by more than d_{max} as shown in Figure 4-2. The number of BTS- and MS-RFBOTs depends on

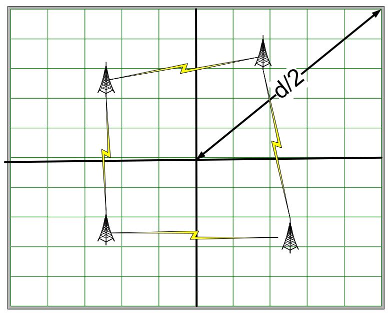


Figure 4-2: Network Connectivity with Partition

the size of the cluster given a time to finish the cluster. If we let B be the number of BTS-RFBOTs and N be the number of grids in a cluster, then the number of navigations that the BTS-RFBOTs need to make becomes the following:

$$No.of Navigations = floor(N/B) + ceiling(N/B) - 1 \tag{4.1}$$

For N by N grids, the number of trips becomes,

$$No.of Navigations = (floor(N/B) + ceiling(N/B) - 1)^N \tag{4.2}$$

If we assume the time to complete a trip to be T_trip and the number of MS-

RFBOTs to be M, the total time to complete the entire measurement becomes,

$$TotalTime = \frac{T_{trip}}{M} * (floor(N/B) + ceiling(N/B) - 1)^N \tag{4.3}$$

4.2.5 Problem Description and Motivation

Given signal strength constraints, the objective is to move the base stations and mobile stations to RF strength map of the interested area. Since there are practically infinite combination of the base station and mobile clients, the goal of this paper is to provide path plan of the base stations and mobile clients such that the time it takes to generate RF strength map is minimized.

The first problem is to create grids with the size within which the signal variation is ignorable. RF signal strength does not need be measured every centimeter, but if the size is too large, the signal strength variation within a grid becomes large enough so that optimal locations of the base stations that the model finds may not be optimal.

The second problem is to ensure that the RFBOT's are in the grid where it is instructed to be by the path planning algorithm. The RFBOT's can be in a wrong grid, yet the path planning model thinks the RFBOT's follow to the instructed location. The mechanical error tolerance of the RFBOT's need to be taken into consideration.

The third problem is that the base stations in a indoor environment are usually installed at ceiling levels. When the BTS-RFBOT's transmit and receive signals, the effect of the height of the antenna being used needs be considered and properly corrected.

The fourth problem is to ensure the data exchange between the MS-RFBOT,BTS-RFBOT, and the central management control. The RF data transfer should be robust and as error-free as possible. When the communication link fails, how to recognize the data link is lost. If the data is not received, how will the model treat the non-data situation.

4.3 Data Acquisition

4.3.1 Synchronization over data acquisition

When the MS-RFBOT conducts the data acquisition, it should follow the synchronization rules as follows:

- Each MS-RFBOT shall belong to a particular BTS-RFBOT
- BTS-RFBOTs shall be located on the clusters where the Tire-2 connectivity of the MS-RFBOT can be covered.
- Entire BTS-RFBOTs shall be interconnected over the mesh network so that the Tier 1 communication shall be reached to any BTS-RFBOTs within the Tire 2 communication coverage.

An ideal and simple topology for the data acquisition is to have every cluster have its own a pair of BTS-RFBOT and MS-RFBOT. This configuration will enable the entire cluster to conduct the data acquisition simultaneously. Under this configuration, we can estimate the data acquisition time in terms of a number of trips as follows:

$$NumberofTrips = N \tag{4.4}$$

where N is the number of grids in the largest cluster. The total data acquisition time is estimated as N * K, which are the number of grids in the largest cluster and the number of clusters.

4.3.2 Data Measurement

Let the base stations move to newer locations every T_b and the clients move to newer locations every T_c. Then T_b = M$\cdot T_c$ as in Figure 4-3 where M is number of locations visited by each client for any given base station configuration. At any given time T_c, the base station sends (broadcasts) signaling information with base station index, location, signal strength.

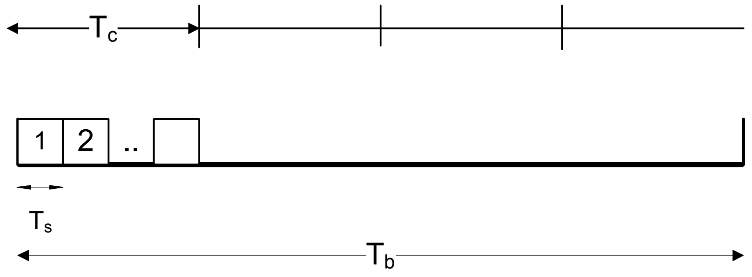


Figure 4-3: Timing

Since the RFBOTs need to exchange data with the server as well as among themselves, it is required for the model to acquire and share timing information. If base station configuration changes before the MS-RFBOTs finish navigating through all grids for signal information, the data collected by MS-RFBOTs will cause the model to use wrong information to select optimal configuration. Also, RFBOTs need to know when exactly to transmit or receive the RF signals to avoid the noise in path loss measurement. Even with multiple access systems such as CDMA, EVDO, or UMTS, the timing of transmission is important due to the navigation requirements. However, it is not easy to have global timing in indoor environment. We need to have handshaking mechanism in the model to make sure all data exchanges are acknowledged. This means before the base stations move to new locations, all clients must obtain all information. There needs to be a signalling timing framework to ensure synchronization among the all components of the model. We first assume that we have an ideal master timing source that distributes the timing information to RFBOTs. Then, we would describe how synchronization would be done without the master timing source.

System with a perfect timing source

Having a perfect timing source means that the RFBOTs have the information as to when they can send packets and when they can move to new grids. It also means that the both BTS and MS RFBOTs know exactly when to start transmitting and receiving RF signals to be measured. Let T_s be duration for which RFBOTs trnasmit signal, N be number of BTS, M be number of Mobile Stations then $T_c = N{\cdot}T_s$, where T_c is the time when the mobile station can move to the next grid in the path planning. The BTS's can start moving to the next configuration locations after $T_b = M{\cdot}T_c$ as in Figure 4-4. For one BTS configuration, the entire navigation time can be derived as $T_total = K{\cdot}(T_b + T_a)$ where K is number of BTS configurations and T_a is time it takes for BTS's to move to the new configuration.

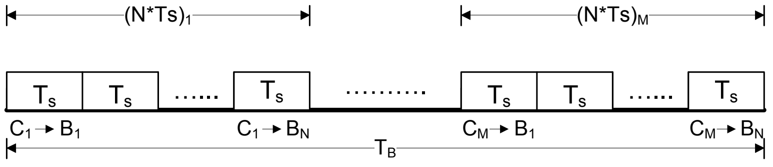


Figure 4-4: Synchronization

System without a master Timing Source

For the system without a master timing source to have synchronized movement, there needs to be localized timing information exchange between the elements within the system. Since there is no global timing information available, the task is performed by introducing handshaking between the server, BTS's, and clients. The only information required in this model is to know when the BTS's and clients finish their current tasks. When clients send acknowledgement and completion message to the BTS's, the BTS's take the next action in the planning. All BTS's relay the acknowledgement and completion messages from the clients to the server by sending their own acknowledgement and completion messages. The following state diagrams for Server, BTS, and Client show how the messages flow between them and how each state is triggered without knowing precise timing of each event.

4.3.3 Data Structure

The fundamental data of interest in optimally selecting BTS locations is the path loss values of the building for a particular frequency. The path loss is defined in Equation

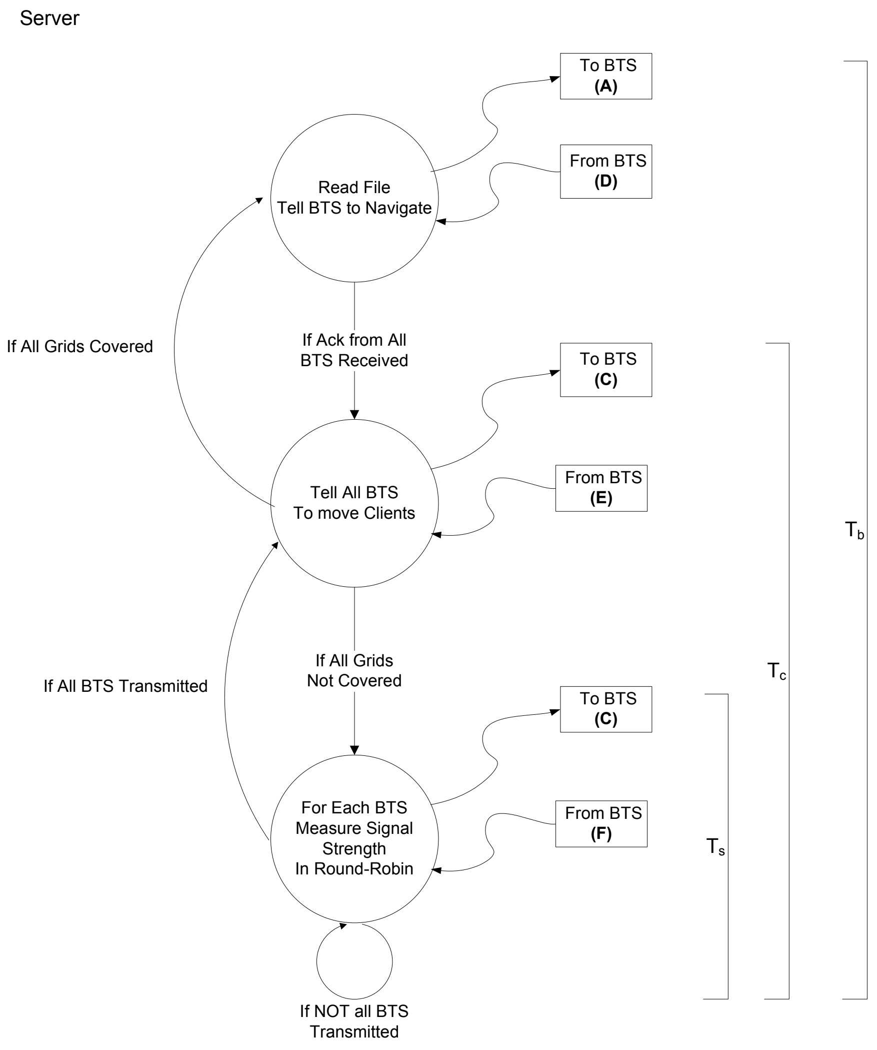

Server
To BTS (A)
From BTS (D)
Read File Tell BTS to Navigate
If All Grids Covered
If Ack from All BTS Received
To BTS (C)
From BTS (E)
Tell All BTS To move Clients
T_b
If All BTS Transmitted
If All Grids Not Covered
T_c
To BTS (C)
From BTS (F)
For Each BTS Measure Signal Strength In Round-Robin
T_s
If NOT all BTS Transmitted

Figure 4-5: State Diagram - Server

4.5.

$$PathLoss = TransmitPower - ReceivePower \tag{4.5}$$

With known transmit powers and measured signal strengths for all possible pairs of grids on the map, the path loss values can be easily obtained for all grid pairs. At every T_c, each MS-RFBOT records the signal strength from the BTS-RFBOT. At the end of T_b, each MS-RFBOT has a table of received signal strengths for the grid it covered in the cluster. By combining these tables, we will have a larger table of

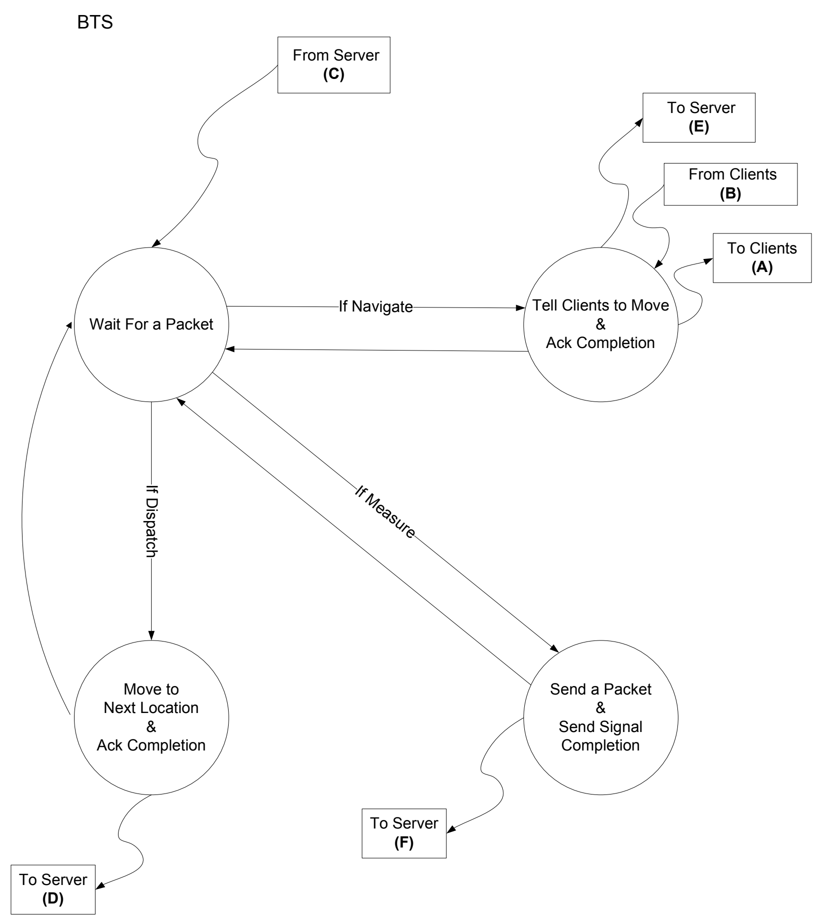


Figure 4-6: State Diagram - BTS

$N - 1$ entries assuming N grids on the map for a BTS-RFBOT location. Since the transmit power is already known for the BTS-RFBOT, we can create a table with the path loss values for each grids shown below:

$$\begin{bmatrix} (\underline{PL1}) & (\underline{PL2}) & \cdots & (\underline{PL_x}) \\ (\underline{PL_x}) & (\underline{PL_x}) & \cdots & (-) \\ \vdots & \vdots & \cdots & \vdots \\ (-) & (-) & (-) & (-) \end{bmatrix}$$

(-) denotes the case where no signal information was received.

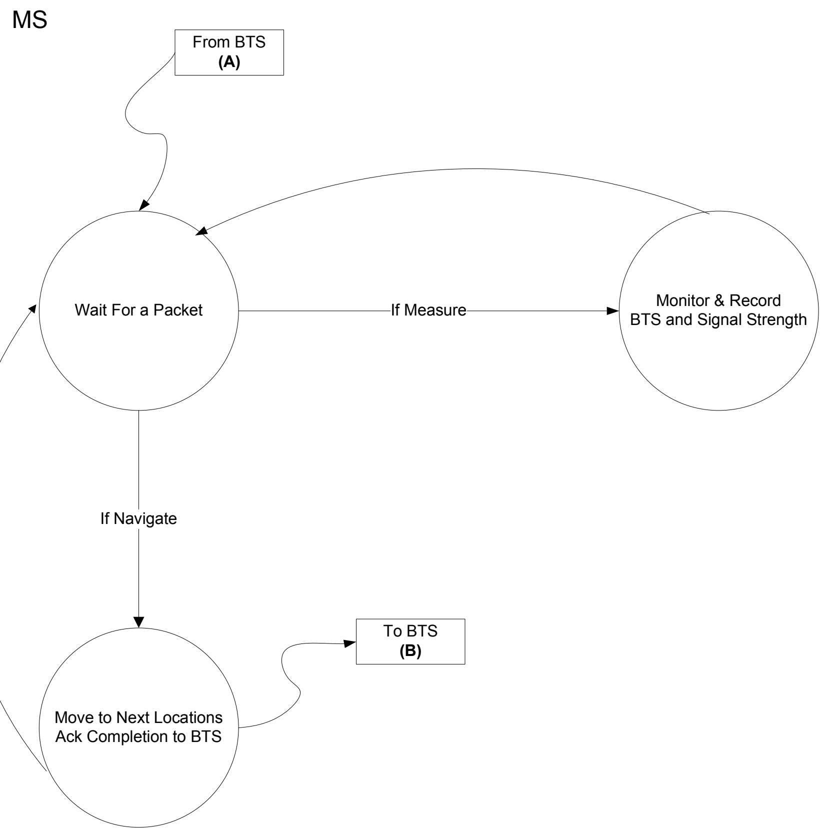

MS
From BTS (A)
Wait For a Packet
If Measure
Monitor & Record BTS and Signal Strength
If Navigate
To BTS (B)
Move to Next Locations Ack Completion to BTS

Figure 4-7: State Diagram - Client

After entire grids on the map are covered, we can generate an overall path loss matrix by combining the tables from all MS-RFBOTs. The final matrix will be N by N matrix where the rows corresponds to the transmit grids while the columns represents the receive locations. The entries will have path loss values for the grid ID same as column numbers.

$$\begin{bmatrix} \underline{PL_{11}}, \underline{PL_{12}}, \underline{PL_{13}}, \cdots & \cdots & \underline{PL_{1N}} \\ \vdots & \vdots & \vdots \\ \underline{PL_{N1}}, \underline{PL_{N2}}, \underline{PL_{N3}}, \cdots & \cdots & \underline{PL_{NN}} \end{bmatrix}$$

Once the database is built, finding optimal locations of the base stations becomes "Search and Optimize" with the design criteria. A simple SQL query can yield the base station configuration that meet the design criteria most optimally. With the optimal base station configuration, we can also pick optimal output power for each base stations without affecting the RF coverage of the network. The power optimization depends heavily on the technology because Signal to Noise Ratio, $SNR(dB)$ requirements and handoff capability features are different from one wireless technology to another.

The coverage of a grid by a BTS can fundamentally be represented by a binary variable. The value can take either "covered" or "not covered" by the BTS. The values of grid coverage are determined by whether the signal strength at the grid with given a transmit power is greater than the receiver sensitivity. In practice, however, the minimum signal strength is usually greater than the receiver sensitivity to ensure the demodulated information is above a certain quality level. Thus, with the required minimum signal strength, the coverage of grid i by BTS at grid j, C_{ij} can be determined by the inequality in Equation 4.7.

$$C_{ij} = \begin{cases} 1 & \text{if } P_r^i \geqq \ P_{min} \text{ or } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

where

$$P_r^i = P_t^j - PL_{ji} \tag{4.7}$$

Assuming N total grids, each grid has N number of C_{ij} values while the total number of C_{ij} values is N^2 for the entire map. Thus, the whole coverage map can be represented by a N by N matrix of either 0 or 1 for each entry with the rows representing grids with BTS locations while the columns representing grid locations of the receivers. Even though the BTS signal can reach the mobile station at a grid, there is no guarantee that the mobile station can also reach the BTS. To guarantee that the communication link for both directions are established, the link balance equation in Equation 4.9 needs to be satisfied.

$$C_{ij} = C_{ji} \tag{4.8}$$

We introduce a new matrix, $\underline{C'}$, where

$$\underline{C'} = \underline{C} + \underline{C^T} \tag{4.9}$$

The entries in $\underline{C'}$ with value of "2" signifies that there can be two way communication links between the grids corresponding to the rows and the columns of the entries.

4.3.4 Optimization Algorithm

Coverage Analysis without Handoff

The purpose of the coverage analysis is to come up with a set of optimal BTS locations that result in satisfying the desired coverage with the minimum number of BTS's. Even though there may be solutions with less number of BTS's to cover the desired area, the approach being proposed here is one of the simplest method to determine locations of the BTS's on the map. We assume that any grid on the map has at least one BTS location with two way communication links. The assumption will be removed in the later section. The process of finding locations of BTS now becomes an iterative process of finding the row with the most entries of value "2". When a location is identified after each iteration, the values at the grids covered by the location will be replaced by 0. The iteration continues until all grids are covered. When the all grids are covered and their coverage matrix values are replaced by 0's, the set of BTS locations are found.

Coverage Analysis with Handoffs

The algorithm needs a slight modification when handoff regions are required. The handoff regions can be defined as areas where more than two signals greater than P_{min} are present. This means that the algorithm no longer can exclude all grids where C_i is set to one. The handoff regions are required to provide non-interruptive transition for a mobile station from one base station to another base station. The handoffs get triggered when the signal strengths from the neighboring base stations become greater than a certain threshold, P_{hothr}. For example, if we set P_{hothr} = -80 dBm and when a mobile station is at an area where the signal level became to be -80 dBm, there should be a signal greater than $P_{min} + \delta_{ho}$ present to ensure the handoff is triggered. The algorithm needs to be modified to accommodate dual-signal presence in a grid. we also redefine the coverage indicator variable C_i to be trinary as in Equation 4.10

to indicate handoff region.

$$C_i = \begin{cases} 1 & \text{if } P_r^i \geqq \; P_{min} \\ 2 & \text{if } P_{hotr} \geqq \; P_r^i \geqq \; P_{min} \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

The first step in the algorithm with handoff support is the same as the algorithm without handoff support. The algorithm picks a grid when transmitted that covers the most grids where the signal strengths received are greater than P_{min}. Then, from the covered grids, the algorithm finds grids that meet the handoff trigger condition, $P_{min} < P_r^i < P_{hothr}$. With the grids identified for handoffs, find a grid that can cover the handoff region with the signal strengths greater than $P_{min} + \delta_{ho}$. If there is no single transmit grid that can cover all handoff grids, the algorithm can select a grid that covers the most handoff grids and find another grid that covers the most of the uncovered handoff grids. This process can be repeated until all handoff grids are covered. The algorithm then finds again the handoff regions for the new grids added to cover the initial handoff regions. This process repeats until all grids are covered. The last grid to be found should cover all grids that are not covered so far in the algorithm.

4.4 Analysis

4.4.1 Effects of Missing Data

When there are some grids that are missing information to determine the coverage, the coverage analysis needs to handle the case. There are two possible cases of missing pathloss information. The first case is when all pathloss data for a grid is missing. The second case is when some of the pathloss data is missing. The indoor RF propagation usually follows lognormal distribution. This characteristic provides a way to predict the coverage of a grid. The lognormal fading dictates that as the distance from the transmitter increases, the pathloss increases with a normally distributed fading that follows the lognormal distribution. To be able to predict the pathloss value at a grid with missing data, we use all other measured data for a given transmitter to estimate the path loss exponent, the mean and the standard deviation of the lognormal path loss model. The statistical parameters estimated by the actual measured data can be used to determine the coverage status of the grid. With the distribution parameters all known, we can predict the path loss values of the missing grids. When there are too many grids with missing path loss values, the parameters for the lognormal distribution cannot reliably be estimated. One has to decide on a desired confidence level of the estimated parameters to determine the minimum number of successfully measured path loss values. When the number of measured values are less than the minimum number of the values required to satisfy a certain preset confidence level and a confidence interval, the RFBOTs have to revisit those grids with missing path loss values to perform measurements. Figure 4-8 shows typical path loss values plotted against distances from a transmitter. The estimation for the parameters can be done with the least square error method [25].

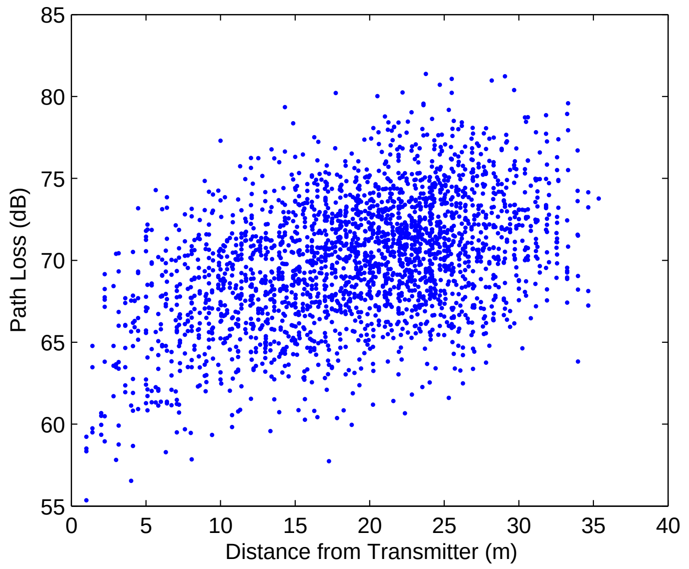


Figure 4-8: Path Loss vs. Distance

Figure 4-9 and Figure 4-10 show the estimated values of the path loss exponent, n and σ respectively for a map with 2500 (50x50) grids. The both charts display a significant degradation of estimated parameters when more than 80% of the grids are missing with the path loss values.

4.4.2 Effects of Navigation Error

In coverage analysis, a grid has only two values, covered or not covered. Thus by declaring all points within a grid are covered or not covered, we may introduce some errors when not all points within the grids agree with the value represented by the grid. We need to employ a statistical model to decide the grid size with a certain confidence level that the P_r represents all the points in the grid in terms of the

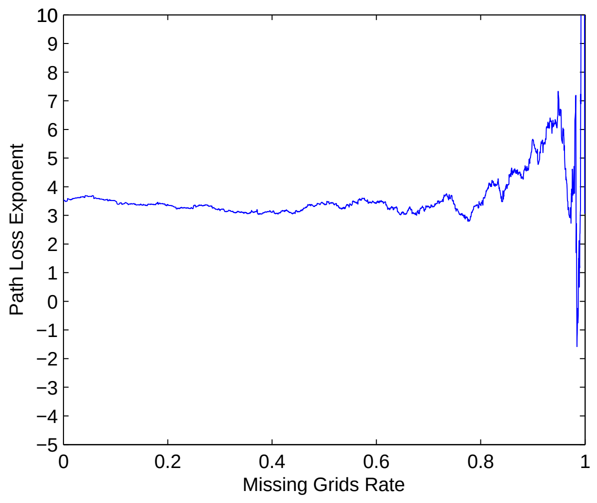


Figure 4-9: Path Loss Exponent Estimation, n=3.5

coverage indicator, C_{ij}. The indoor path loss has been shown to obey the distance power law in equation 4.11 [24].

$$PL = PL(d_0) + 10n\log(\frac{d_t}{d_0}) + X_\sigma \tag{4.11}$$

where n is the path loss exponent and X_σ is a normal random variable with a standard deviation of σ. With a transmit power, P_t given, the receive power, P_r becomes a Gaussian random variable with a mean, $\overline{P_r}$ and a standard deviation of σ.

$$\overline{P_r} = P_t - (PL(d_0) + 10n\log(\frac{d_t}{d_0})) \tag{4.12}$$

Since the path loss values follow inversely with the distance from the transmitter, the farthest point in the grid from the center of the grid would have the smallest $\overline{P_r}$. A grid that has the distance, d to a vertex from the center of a square grid has a

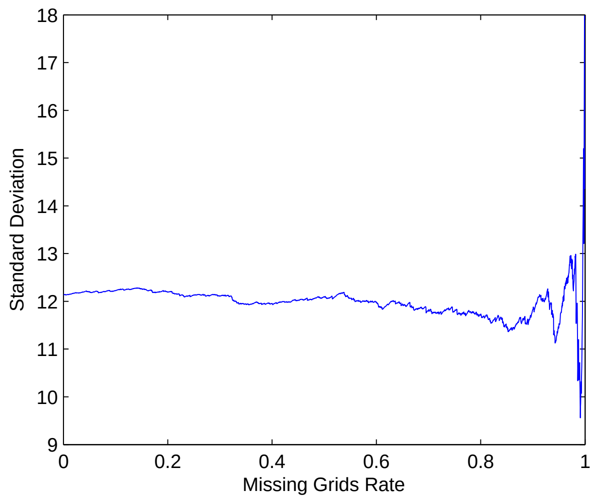


Figure 4-10: σ Estimation, $\sigma = 12$

length of $(d \cdot \sqrt{2})/2$ for each side.

If we let the distance from the transmitter to the center of the grid be d_t, then the $(P_r)_{farthest}$ at the farthest point from the center can be calculated in equation 4.13.

$$(P_r)_{farthest} = P_t - (PL(d_0) + 10n\log(\frac{d_t + d}{d_0}) + X_\sigma) \tag{4.13}$$

With the receive power at the center of the grid, $(P_r)_{center}$,

$$\begin{aligned} \overline{(P_r)}_\Delta &= \overline{(P_r)}_{center} - \overline{(P_r)}_{farthest} \\ &= 10n(\log(\frac{d_t + d}{d_0}) - \log(\frac{d_t}{d_0})) \\ &= 10n(\log(d_t + d) - \log(d_t)) \\ &= 10n\log(\frac{d_t + d}{d_t}) \end{aligned} \tag{4.14}$$

where X_σ is a normal random variable with a standard deviation σ. Thus, $(P_r)_\Delta$ is a

Gaussian variable with mean, $10n\log(d)$ and standard deviation, σ. For the coverage test, the worst case occurs when the grid is at the edge of the coverage area where two neighboring grids' coverage indicators are different. by moving from one grid to the other, the coverage is no longer there. To reduce the error at the coverage boundary, we need to introduce some margin, P_δ to the P_{min}. This means that at the coverage edge, if $(P_r)_\Delta < P_\delta$, the entire grid can be said to have coverage. If we let $Prob[(P_r)_\Delta < P_\delta]$ be a certain value x.

$$\begin{aligned} x &= 1 - Prob[(P_r)_\Delta > P_\delta] \\ &= 0.5 + erf(\frac{P_\delta - \overline{(P_r)_\Delta}}{\sigma}) \end{aligned} \tag{4.15}$$

From equation 4.15, with a known x, we can solve for d/d_t. For typical values of $n = 4, \sigma = 5dB, P_\delta = 5dB$, and $x = 0.95$, Figure 4-11 shows the maximum sizes of the grids for different coverage values.

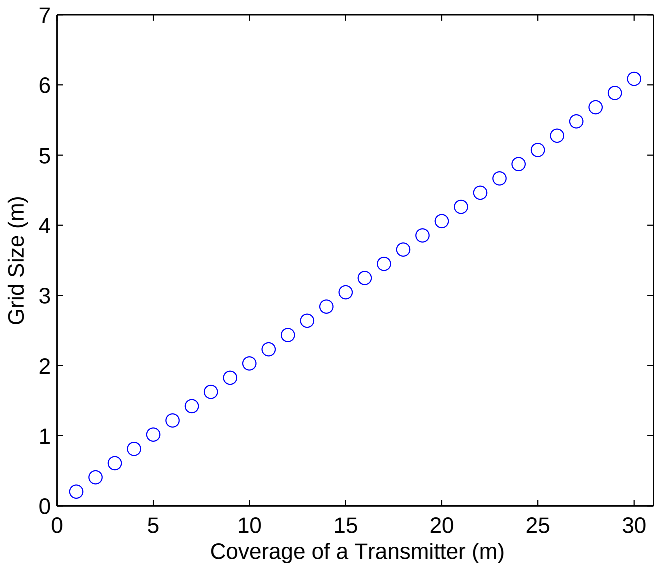


Figure 4-11: Grid Size vs. Coverage

4.5 Summary

This chapter has presented a novel approach for utilizing multiple networked robots to collect path loss data of an indoor environment. We have shown then grid based navigation can be used not only for accurate navigation but also for collecting just enough information out of infinite amount of possible data points. The possibility of utilizing multiple robots to aid in data collection opens door for many different application not just access point placement in a wireless network design. The strategies and methods presented in this paper can be expanded to accommodate many different dynamic environments where robots can perform with much higher efficiency while minimizing human intervention. The further study is warranted to insert more intelligence to the system model so that the whole model becomes more adaptive to dynamic changes unforseen by initial design. By making the model more adaptive, we can minimize the effect of possible errors in navigation and data acquisition. Also, the network topology can be expanded to form an ad-hoc network as necessary for more distributed and less prone to communication failures making the model even more robust.

Chapter 5

Contirbution and Future Research

5.1 Contribution

The following are the contributions of the dissertation:

- The self-localization using range finders without knowledge of absolute postioning information.
- The error compensation in mobile robot navigation using the range data where the navigational environment can dynamically change due static or moving obstacles as well as non-flat surface boundaries.
- The robust robot control schemes with failure detection, recovery, and reconfiguration inluding resource redistribution.
- The synchronization realization using two-tiered network topology without a source of global timing.

- Autonomous RF signal path loss data collection without human intervention utilizing self-localizing mobile robots.

5.2 Future research

The following are the future works that need to be done:

- Investigate self-localization without the distance information from the map.
- Investigate the model that employs an adaptive grid size scheme for more accuracy and efficiency.
- Investigate how to optimize the amount of boundary information for navigation.
- Enhance robust and navigation algorithm under interference from dynamic objects.
- Investigate collaborative map building of indoor environment.

Bibliography

[1] A. Sanfeliu, N. Hagita and A. Saffiotti, “Network Robot Systems,” *Robotics and Autonomous Systems*, Vol. 56, No. 10, pp. 793-797, 2008.

[2] E. Guizzo, “Three Engineers, Hundreds of Robots, One Warehouse,” *IEEE SEPCTRUM*, Jul. 2008.

[3] A. A. Zavala, “Autonomous Navigation and Positioning for Indoor Wireless Measurements using Mobile Robots,” *Proceedings of the 14th International Conference on Electronics, Communications, and Computers*, 2004.

[4] M. Mehmood, L. Kulik, and E. Tanin, “Autonomous navigation of mobile agents using RFID-enabled space partitions,“ *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information System*, 2008

[5] Gueaieb, W., Miah, M.S.. “An Intelligent Mobile Robot Navigation Technique Using RFID Technology,“ *IEEE Transactions on Instrumentation and Measurement*, Vol 57, Issue 9,Page(s):1908 - 1917, 2008

[6] M. Betke, L. Gurvits,“Mobile Robot Localization Using Landmarks“, *IEEE Transactions on Robotics and Automation*, vol. 13, No 2, Page(s): 251-263, 1997.

[7] R. Sim, G. Dudek,“Mobile robot localization from learned landmarks“, *Proc. of IEEE/RSJ IROS*, vol. 2, Page(s): 1060 .1065, 1998.

[8] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, “A Comparison of Line Extraction Algorithms using 2D Range Data for Indoor Mobile Robotics,” *Autonomous Robots*, Vol. 23, No. 2, pp. 97-111, 2007.

[9] http://www.drrobot.com/products_item.asp?itemNumber=X80

[10] http://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html

[11] M.B. Dias, M. Zinck, R. Zlot and A. Stentz, ”Robust Multirobot Coordination in Dynamic Environments,” *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3435-3442, Apr. 2004.

[12] H. Choset, ”Coverage for robotics. A survey of recent results,” Ann. Math. and AI, 31:113.126, 2001.

[13] S. V. Spires and S. Y. Goldsmith, ”Exhaustive geographic search with mobile robots along space-filling curves,” *Proceedings of the First International Workshop on Collective Robotics*, pages 1.12. Springer-Verlag, 1998.

[14] N. Hazon and G.A. Kaminka, ”Redundancy, Efficiency and Robustness in Multi-Robot Coverage,” *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 735-741, Apr. 2005.

[15] L.E. Parker, ”ALLIANCE: An architecture for fault tolerant Multirobot cooperation,” IEEE Trans. Robot. Automat., vol.14, pp220-240, Apr. 1998.

[16] T. Boggs and R. B. Gomez, “Fast Hyperspectral Data Processing Methods,” *SPIE AeroSense 2001 Conference Proceeding*, pp. 74-78, 16-20 April, Orlando, Florida.

[17] H. Hashemi, “The Indoor Radio Propagation Channel“, *Proceedings of the IEEE*, Vol 81, Issue 7, pp 943-968, Aug. 2002.

[18] T.S. Rappaport, *Wireless Communications - Principles and Practice*, Prentice Hall, 1991.

[19] B. Sujak, D.K. Ghodgaonkar, B.M. Ali and S. Khatun, “Indoor Propagation Channel Models for WLAN 802.11b at 2.4 GHz ISM Band,“ *Asia-Pacific Conference on Applied Electromagnetics, 2005. APACE 2005.* , pp. 5, Dec. 2005.

[20] A. Hills, J. Schlegel and B. Jenkins, “Estimating Signal Strengths in the Design of An Indoor Wireless Network,“ *IEEE Transactions on Wireless Communications*, vol 3, Issue 1, pp 17-19, Jan. 2004.

[21] F. Babich and G. Lombardi, “Statistical Analysis and Characterization of The Indoor Propagation Channel,“ *IEEE Transactions on Communications*, Vol 48, pp. 455-464, Mar. 2000.

[22] J. Yuan, “A General Photogrammetric Method for Determining Object Position and Orientation,” *IEEE Transactions on Robotics and Automation*, Vol. 5, Issue 2, pp. 129-142, 1989.

[23] A. Bandera, C. Urdiales and F. Sandoval, “An Hierarchical Approach to Grid-Based and Topological Maps Integration for Autonomous Indoor Navigation,“ *Proceedings. 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001.*, Vol 2, pp. 883-888, Oct. 2001.

[24] C. Perez-Vega and J.L. Garcia, “A Simple Approach to A Statistical Path Loss Model for Indoor Communications,“ *27th European Microwave Conference 1997.*, Vol 1, pp 617-623, Sept. 1997.

[25] N. Benvenuto and F. Santucci, “A Least Squares Path-Loss Estimation Approach to Handover Algorithms,“ *IEEE Transactions on Vehicular Technology*, Vol. 48,No, 2, pp 437-447, Mar. 1999.

[26] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering Transactions*, ASME, Ser. D 82, pp. 35-45, 1960.