Indoor Robot Localization and Mapping Using ZigBee Radio Technology

by

Kyle Hevrdejs, Jacob Knoll
Advisor: Dr. Suruz Miah

Electrical and Computer Engineering Department
Caterpillar College of Engineering and Technology
Bradley University

Abstract

Addressing the navigation (localization and motion control) problem of a mobile robot, coupled with its mapping problem, remains a significant challenge to date. The well–known simultaneous localization and mapping (SLAM) problem of mobile robots has been addressed in the literature without specifically taking into account the robot's motion control tasks. Moreover, its implementation can cost more than the robot itself. Robot's motion control strategies developed in the literature either (i) rely on sophisticated hardware platforms, (ii) assume noise-free environments, or (iii) are based on abstract theories which are validated using computer simulations only.

The work presented herein solves the navigation and mapping problems of mobile robots using open–source hardware and range–only measurements from a network of radio sources. The hardware platform used in this work is customized such that it is cost–effective and easy-to-implement, addressing the aforementioned issues in developing motion control strategies. Here, the robot estimates its position and orientation, builds a map of its operating environment using radio frequency (RF) signals received from radio sources. It then navigates through a path defined by a set of 2D points on the ground using a motion control strategy in cooperation with a tool of computational intelligence. The proposed robot navigation and mapping scheme is tested in an indoor laboratory environment and its performance is compared with the simulation counterpart using a commercial robot simulator.

Acknowledgements

The authors would like to thank everyone who helped us along the way including Mr. Christopher Mattus, Mr. Nick Schmidt, and the Electrical and Computer Engineering Department.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, mobile robot navigation has became a popular topic in research, military, and commercial applications. Successful navigation relies on addressing two major challenges in the field of robotics: (i) localization and mapping, and (ii) motion control. Previous works have solved the simultaneous localization and mapping problem (SLAM) alone without taking into account motion control, see the work in [6, 23, 12] and some references therein. In many of these instances, the hardware necessary to implement the SLAM algorithm can be more expensive than the robot itself. The robotics community has also been interested in the motion control problem and has addressed robot dynamics, kinematics, and its actuator constraints assuming the fact the map of the robot's operating environment is known. See the papers, [17, 18, 16, 19], for example. Proposed solutions to the SLAM problem have lately involved the use of RF signals from radio sources to localize a mobile robot [4, 14, 8, 9, 1]. Specifically, received signal strength indicator (RSSI) measurements have been common in these works, see [2, 13, 21, 22]. However, a major issue of using RSSI measurements for localizing a robot is that they are less sensitive to the robot's location. The signal-to-noise ratio associated with RSSI information is relatively low compared to other RF signals due to the fact that they are dependent on robot's operating environment. Also, RSSI measurements also rely on the radiation pattern of the antenna mounted on the selected radio. Despite these side effects of using RSSI measurements, they are a common choice for indoor/outdoor robot position and orientation (pose) estimation due to the fact that most radio sources are inexpensive, require little to no maintenance, and the RSSI data from them can be obtained relatively easy. In addition, radio sources have integrated circuits that store individual unit identifiers (ID) and the last RSSI in their memory. Therefore, radio sources can be uniquely identified by a receiver mounted on the robot, coupling RSSI measurements with IDs. Thus, bypassing the common data association problem encountered when performing measurements.

For this project, a mobile robot placed in an indoor environment must simultaneously navigate through a set of predefined waypoints on the ground while estimating its own pose and postions of radio sources in the operating space. Mobile robot navigation and SLAM has been done by many, but these works usually solve only one of the problems, use expensive equipment, or only show simulation results. The goal for this project was to simulate and implement a cost-effective, easy to use, and modular mobile robot navigation and mapping system. Due to the lack of reliable GPS signals indoors, RSSI measurements from radio sources in the robot's operating range are used to estimate the robot's pose in this work. Several drawbacks of using RSSI have been addressed by building an inexpensive custom radio transceiver which increases accuracy of measurements, aiding in the mapping of the radio source's positions. XBee radios are used as radio sources in this work. They are low-cost and communicate using the ZigBee protocol which allows for easy access to RSSI information. In order to simultaneous localize the mobile robot and map the XBee radios, or beacons, the well-known extended Kalman filter algorithm is utilized due to its resilience in noisy environments. To control the motion of the robot, a fuzzy logic controller and proportional controller have been implemented. A fuzzy logic controller uses human like reasoning to determine the robot's linear speed without requiring accurate approximate distances between the robot and it's current waypoint. To test the performance of the proposed solution, several simulations have been completed using a commercial robot simulator, and results from laboratory experiments using off-the-shelf hardware confirm the implementation of a fully-functional system.

The rest of the report is organized as follows:

- Chapter 2 discusses the simultaneous navigation and mapping algorithm.
- Chapter 3 shows a simulation of the navigation and mapping algorithm in V-REP.
- Chapter 4 describes the customized radio transceiver and its use for range and bearing approximation.
- Chapter 5 shows the experimental results of the implemented navigation and mapping algorithm in a laboratory environment.
- Chapter 6 draws conclusions from the results and presents future related work.
- Appendex A goes through the steps to simulate the algorithm in V-REP.
- Appendex B explains to the run an experiment using the customized radio transceiver and Pioneer 3-DX.

Chapter 2

Simultaneous Navigation and Mapping

This chapter discusses the EKF-SLAM algorithm, other mathematical models, and the robot's motion controllers. First, Section 2.1 the mobile robot's kinematics are modeled and illustrated. In Section 2.2, the EKF-SLAM algorithm is presented in detail. Section 2.3 covers the measurement model of using RSSI to determine range and bearing of the XBee radios. This is followed by a description of the fuzzy logic and proportional controller used for motion control in Section 2.4.

2.1 Robot Model

Even though a circular shaped differential-drive mobile robot was used to conduct the simulations and experiments for this project, the kinematic model of an Ackermann steering vehicle operating on a ground–fixed inertial reference frame X-Y was considered [3]. This is done so the proposed localization and mapping algorithm can be applied to a wide range of robotic vehicles. Let $\mathbf{q}_{k,r} \equiv [x_k, y_k, \theta_k]^T$ denote the pose (position and orientation) vector of the robot at time $t \geq 0$ with $t = kT$, $k \in \mathbb{N}_0$, the subscript r in $\mathbf{q}_{k,r}$ indicates robot's pose, and $T > 0$ being the sampling time. The robot's discrete-time model is approximated by the first-order Euler integration given as

$$x_{k+1} = x_k + T\nu_k \cos(\theta_k + \gamma_k), \tag{2.1a}$$

$$y_{k+1} = y_k + T\nu_k \sin(\theta_k + \gamma_k), \tag{2.1b}$$

$$\theta_{k+1} = \theta_k + T\nu_k \frac{\sin(\gamma_k)}{\ell}, \tag{2.1c}$$

where $\gamma_k \in (-\frac{\pi}{2}, \frac{\pi}{2})$ is the front wheel steering angle with respect to the robot's orientation $\theta_k \in [-\pi, \pi)$, ν_k is the linear speed, and ℓ is the distance between the drive wheels of the robot. A compact form of model (2.1) can be written as

$$\mathbf{q}_{k+1,r} = \mathbf{f}_r(\mathbf{q}_{k,r}, \mathbf{u}_k), \tag{2.2}$$

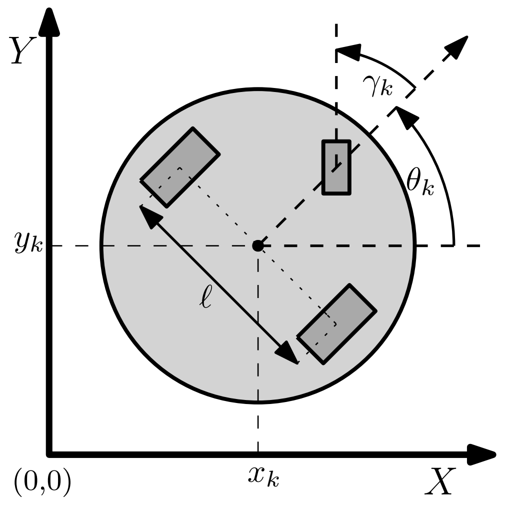


Figure 2.1: Illustration of the Ackermann steering vehicle model.

where the vector-valued vector function $\mathbf{f}_r : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}^3$ and the robot's motion control input $\mathbf{u}_k = [\nu_k, \gamma_k]^T \in \mathbb{R}^2$. It is assumed that the left and right wheels of the robot steer together under a no slip condition [3].

2.2 EKF-SLAM Algorithm

Simultaneous localization and mapping (SLAM) is a standard problem in the field of mobile robotics and has been an active research topic in recent years (see [21] and some references therein). A significant challenge in solving this problem is to overcome the effect of multipath on measurement signals when a robot operates in an indoor environment. Even though a large body of research has been conducted to solve this problem in an indoor environment, a well-balanced compromise between the accuracy of the SLAM algorithm and the cost to implement it has yet to be reached. The well–known Extended Kalman Filter (EKF)–SLAM method employed in the current work recursively estimates both the robot's pose and the 2D positions of the XBee radios in the environment despite noisy measurements. The EKF–SLAM technique is mostly exploited by the robotic community for its simplicity and capability to sustain exaggerated noise inherent in both robots and sensory measurements. The robot model is subject to process noise $\boldsymbol{\zeta}_k$ associated with the motion control input $\mathbf{u}_k$ at discrete time index k, for $k \in \mathbb{N}_0$. Without loss of generality, the process noise is assumed to be Gaussian with mean $\mathbf{0}$ and covariance $\mathbf{Q}_k$, *i.e.*, $\boldsymbol{\zeta}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$. Taking into consideration the robot's process noise, the discrete-time model (2.2) can be rewritten as

$$\mathbf{q}_{k+1,r} = \mathbf{f}_r(\mathbf{q}_{k,r}, \mathbf{u}_k, \boldsymbol{\zeta}_k). \tag{2.3}$$

Assume that the robot is initially perturbed from its reference pose. Hence, the initial state error covariance matrix is given by $\mathbf{P}_{0,rr} = \mathrm{diag}((\mathbf{q}_{0,r} - \hat{\mathbf{q}}_{0,r}) \odot (\mathbf{q}_{0,r} - \hat{\mathbf{q}}_{0,r}))$. Note that conventional EKF-SLAM performs two standard steps (prediction and update) for solving the localization and mapping problem of a mobile robot. Interested readers are encouraged to read papers [15, 20] and some references therein for more details on classical EKF and EKF-SLAM techniques that are commonly used in implementing mobile robot navigation techniques. Following [11], we briefly illustrate the main steps of the EKF-SLAM strategy for estimating the robot's pose (localization) and positions of radio sources (map). Suppose that s, $0 \leq s \leq s^{'}$, XBee radios are mapped in the robot's workspace and the corresponding map is denoted by the set $\mathcal{B} = \{\mathbf{b}^{[1]}, \ldots, \mathbf{b}^{[s]}\}$. Let $\mathbf{q}_k \in \mathbb{R}^{2s+3}$ denote the augmented state vector and $\mathbf{P}_k \in \mathbb{R}^{(2s+3)\times(2s+3)}$ be the corresponding augmented state covariance matrix, *i.e.*,

$$\mathbf{q}_k = \left[\mathbf{q}_{k,r}, \mathbf{b}^{[1]}, \ldots, \mathbf{b}^{[s]}\right]^T \text{ and } \mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{k,rr} & \mathbf{P}_{k,r\mathcal{B}} \\ \mathbf{P}_{k,\mathcal{B}r} & \mathbf{P}_{k,\mathcal{B}\mathcal{B}} \end{bmatrix},$$

where $\mathbf{P}_{k,r\mathcal{B}} \in \mathbb{R}^{3\times 2s}$ is the covariance matrix that maps the state of XBee radios to the state of the robot. Similarly, $\mathbf{P}_{k,\mathcal{B}r} \in \mathbb{R}^{2s\times 3}$ is the covariance matrix that maps the state of the robot to the state of XBee radios. The covariance matrix $\mathbf{P}_{k,\mathcal{B}\mathcal{B}} \in \mathbb{R}^{2s\times 2s}$ corresponds to the states of XBee radios that are mapped in the robot's environment. Let$(\cdot)^-$ and $(\cdot)^+$ denote the a priori (predication) and a posteriori (update) estimates of the quantity $(\cdot)$. A priori estimate of the robot's pose and its map (before taking into account measurements from XBee radios) is given by the classical EKF prediction model which is summarized as

$$\hat{\mathbf{q}}^-_{k+1} = \hat{\mathbf{q}}^+_k + \mathbf{C}^T \mathbf{f}_r(\hat{\mathbf{q}}^+_{k,r}, \mathbf{u}_k, \mathbf{0}), \tag{2.4a}$$

$$\mathbf{P}^-_{k+1,rr} = \mathbf{F}_k \mathbf{P}^+_{k,rr} \mathbf{F}^T_k + \mathbf{L}_k \mathbf{Q}_k \mathbf{L}^T_k, \tag{2.4b}$$

$$\mathbf{P}^-_{k+1,r\mathcal{B}} = \mathbf{F}_k \mathbf{P}^+_{k,r\mathcal{B}}, \tag{2.4c}$$

$$\mathbf{P}^-_{k+1,\mathcal{B}r} = \left(\mathbf{P}^-_{k+1,r\mathcal{B}}\right)^T, \quad \text{where} \tag{2.4d}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \end{bmatrix}, \mathbf{F}_k = \frac{\partial \mathbf{f}_r}{\partial \mathbf{q}_{k,r}}\bigg|_{(\hat{\mathbf{q}}^+_{k,r}, \mathbf{0})},$$

and $\mathbf{L}_k = \frac{\partial \mathbf{f}_r}{\partial \zeta_k}\Big|_{(\hat{\mathbf{q}}^+_{k,r}, \mathbf{0})}$. The measurement model (2.9) at discrete time index $k+1$ is written as

$$\mathbf{y}^{[j]}_{k+1} = \mathbf{h}(\mathbf{q}_{k+1,r}, \mathbf{b}^{[j]}, \boldsymbol{\xi}_{k+1}), \tag{2.5}$$

with $\boldsymbol{\xi}_{k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, where $\mathbf{R} = \text{diag}(\sigma^2_d, \sigma^2_b)$ is the measurement noise covariance matrix with σ_d and σ_b being the standard deviations of the noise associated with the range and bearing measurements, respectively, $\forall j \in \{1, 2, \ldots, s\}$. Let $\mathbf{v} = \mathbf{y}^{[j]}_{k+1} - \mathbf{h}(\mathbf{q}^-_{k+1,r}, \mathbf{b}^{[j]}, \mathbf{0})$ define the robot's measurement innovation vector. A posteriori estimate of the robot's pose and its map (after taking into account measurements from XBee radios) is given by the classical EKF update model which is summarized as

$$\mathbf{S} = \begin{bmatrix}\mathbf{H}_r & \mathbf{H}_{\mathbf{b}^j}\end{bmatrix} \begin{bmatrix}\mathbf{P}^-_{k+1,rr} & \mathbf{P}^-_{k+1,r\mathbf{b}^j} \\ \mathbf{P}^-_{k+1,\mathbf{b}^j r} & \mathbf{P}^-_{k+1,\mathbf{b}^j\mathbf{b}^j}\end{bmatrix} \begin{bmatrix}\mathbf{H}_r^T \\ \mathbf{H}_{\mathbf{b}^j}^T\end{bmatrix} + \mathbf{R}, \tag{2.6a}$$

$$\mathbf{K}_{k+1} = \begin{bmatrix}\mathbf{P}^-_{k+1,rr} & \mathbf{P}^-_{k+1,r\mathbf{b}^j} \\ \mathbf{P}^-_{k+1,\mathcal{B}r} & \mathbf{P}^-_{k+1,\mathcal{B}\mathbf{b}^j}\end{bmatrix} \begin{bmatrix}\mathbf{H}_r^T \\ \mathbf{H}_{\mathbf{b}^j}^T\end{bmatrix} \mathbf{S}^{-1}, \tag{2.6b}$$

$$\hat{\mathbf{q}}^+_{k+1} = \hat{\mathbf{q}}^-_{k+1} + \mathbf{K}_{k+1}\mathbf{v}, \tag{2.6c}$$

$$\mathbf{P}^+_{k+1} = \mathbf{P}^-_{k+1} - \mathbf{K}_{k+1}\mathbf{S}\mathbf{K}^T_{k+1}, \tag{2.6d}$$

where the jacobians of the measurement model (2.5) are

$$\mathbf{H}_r = \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{k+1,r}}\bigg|_{(\hat{\mathbf{q}}^-_{k+1,r},\mathbf{0})} \text{ and } \mathbf{H}_{\mathbf{b}^j} = \frac{\partial \mathbf{h}}{\partial \mathbf{b}^{[j]}}\bigg|_{(\hat{\mathbf{q}}^-_{k+1,r},\mathbf{0})}.$$

Therefore, the EKF prediction model (2.4) and the update rule (2.6) are employed recursively to estimate the robot's pose $\hat{\mathbf{q}}_{k,r}$ and the position of XBee radios, $\hat{\mathbf{b}}^{[j]},\ j = 1,\dots,s$, that are mapped in the robot's environment for $k \in \mathbb{N}_0$.
Suppose that a new measurement $\mathbf{y}^{[s+1]}$ at discrete time index $k+1$ is received by the base XBee on-board the mobile robot in its operating range. The 2D position of the XBee radio that corresponds to this measurement is not in the current map of the robot. Hence, an inverse measurement model is given by

$$\begin{aligned}\mathbf{b}^{[s+1]} &= \mathbf{g}\left(\hat{\mathbf{q}}^+_{k+1,r}, \mathbf{y}^{[s+1]}\right) \\ &= \begin{bmatrix}\hat{x}^+_{k+1} + \left(r^{[s+1]} + \xi_d\right)\cos\left(\hat{\theta}^+_{k+1} + \beta^{[s+1]}\right) \\ \hat{y}^+_{k+1} + \left(r^{[s+1]} + \xi_b\right)\sin\left(\hat{\theta}^+_{k+1} + \beta^{[s+1]}\right)\end{bmatrix}.\end{aligned}$$

Without loss of generality, the noise $\boldsymbol{\xi} = [\xi_d, \xi_b]^T$ is chosen such that $\xi_d \sim \mathcal{N}(0,\sigma_d^2)$ and $\xi_b \sim \mathcal{N}(0,\sigma_b^2)$. Before augmenting the XBee radio associated with the new measurement $\mathbf{y}^{[s+1]}$, let us define jacobians $\mathbf{G}_r \in \mathrm{R}^{2\times3}$ and $\mathbf{G}_{\mathbf{y}^{[s+1]}} \in \mathrm{R}^{2\times2}$ based on this measurement with

$$\mathbf{G}_r = \frac{\partial \mathbf{g}}{\partial \mathbf{q}^+_{k+1,r}}\bigg|_{(\hat{\mathbf{q}}^+_{k+1,r},\mathbf{0})} \text{ and } \mathbf{G}_{\mathbf{y}^{[s+1]}} = \frac{\partial \mathbf{g}}{\partial \mathbf{y}^{[s+1]}}\bigg|_{(\hat{\mathbf{q}}^+_{k+1,r},\mathbf{0})}.$$

The state covariance matrices associated with the new measurement are given by

$$\begin{aligned}\mathbf{P}_{\mathbf{b}^{[s+1]}\mathbf{b}^{[s+1]}} &= \mathbf{G}_r\mathbf{P}^+_{k+1,rr}\mathbf{G}_r^T + \mathbf{G}_{\mathbf{y}^{[n+1]}}\mathbf{R}\mathbf{G}^T_{\mathbf{y}^{[n+1]}}, \\ \mathbf{P}_{\mathbf{b}^{[s+1]}\mathbf{q}} &= \mathbf{G}_r\mathbf{P}_{k+1,r\mathbf{q}} = \mathbf{G}_r\begin{bmatrix}\mathbf{P}^+_{k+1,rr} & \mathbf{P}^+_{k+1,r\mathcal{B}}\end{bmatrix},\end{aligned}$$

where the covariance matrix $\mathbf{P}^+_{k+1,r\mathcal{B}} \in \mathbb{R}^{3\times2s}$. The augmented state and the state covariance matrix are then updated using the update rule

$$\mathbf{q}_{k+1} = \begin{bmatrix}\mathbf{q}^+_{k+1} \\ \mathbf{b}^{[s+1]}\end{bmatrix} \text{ and } \mathbf{P}_{k+1} = \begin{bmatrix}\mathbf{P}^+_{k+1} & \mathbf{P}^T_{\mathbf{b}^{[s+1]}\mathbf{q}} \\ \mathbf{P}_{\mathbf{b}^{[s+1]}\mathbf{q}} & \mathbf{P}_{\mathbf{b}^{[s+1]}\mathbf{b}^{[s+1]}}\end{bmatrix},$$

where the dimensions of the new state covariance matrices, $\mathbf{P}_{\mathbf{b}^{[s+1]}\mathbf{b}^{[s+1]}}$ and $\mathbf{P}_{\mathbf{b}^{[s+1]}\mathbf{q}}$, are 2×2 and $2 \times (2s+3)$, respectively. The new augmented state $\mathbf{q}_{k+1}$ and the augmented state covariance matrix $\mathbf{P}_{k+1}$ follows the classical EKF predication and update steps defined in (2.4) and (2.6), respectively, to recursively estimate the robot's pose and the position of the XBee radios in the updated map. While updating the state (robot's pose and map), the robot also tunes its motion controller in order to follow the path defined by a set of pre-defined 2D points, $\mathcal{P}$.

2.3 Measurement Model

Note that the XBee radios are placed at 2D positions of the robot's workspace are unknown to the robot. These positions are to be estimated by the robot in addition to its own pose. Each XBee radio has its own ID and is able to store the signal strength information, RSSI, of the signal received from the base XBee radio. Therefore, there is no data association problem which is typical in many RF-based sensor networks. The customized radio transceiver mounted on the robot broadcasts an RF signal in its operating region. It then receives the RSSI measurements back from the XBee radios in the operating range. Similar to [5], the line-of-sight (LoS) distance between the robot and the jth, $j = 1, \ldots, s^{'}$, XBee radio at discrete time index $k \in \mathbb{N}_0$ is approximated using the RSSI measurement given by

$$z_k^{[j]} \approx P_{\text{ref}} - 10\eta \log_{10} r_k^{[j]}, \tag{2.7}$$

where $r_k^{[j]} = \sqrt{(x_k - x_k^{[j]})^2 + (y_k - y_k^{[j]})^2}$ is the ideal LoS distance between the robot and the jth Xbee radio, P_{ref} is the power level (in this work, $P_{\text{ref}} = -29$ dBm) at a reference distance of 1 m, η is the signal propagation constant (here, $\eta = 2$), and $z_k^{[j]} < 0$ is the RSSI that the robot receives from the jth XBee radio. Furthermore, the bearing information $\beta_k^{[j]}$ (angle between the robot's current orientation and the jth, $j = 1, \ldots, s^{'}$, XBee radio) at discrete time index $k \in \mathbb{N}_0$ is also assumed to be maximum RSSI in a set of measurements made by the robot, stated by

$$\beta_k^{[j]} = \underset{[-\pi,\pi)}{\arg\max}\, \mathcal{R}^{[j]}, \tag{2.8}$$

where $\mathcal{R}^{[j]}$ denotes the set of RSSI measurements of the jth XBee radio. Therefore, the robot's measurement model considered in this work consists of noisy LoS distance (determined by the inverse model of (2.7)) and bearing, which are modeled by

$$\mathbf{y}_k^{[j]} = \left[r_k^{[j]}, \beta_k^{[j]} \right]^T + \boldsymbol{\xi}_k = \mathbf{h}(\mathbf{q}_{k,r}, \mathbf{b}^{[j]}, \boldsymbol{\xi}_k), \tag{2.9}$$

where $\boldsymbol{\xi}_k \in \mathbb{R}^2$ is the noise associated with the range (LoS distance) and bearing measurements at discrete time index $k \in \mathbb{N}_0$ and function $\mathbf{h} : \mathbb{R}^2 \times [-\pi, \pi) \times \mathbb{R}^2 \to \mathbb{R}^2$. Note that the model (2.9) will be employed in the localization and mapping algorithm which will be discussed in later chapters.

2.4 Motion Control Strategy

Suppose that the a posteriori estimate of the robot's pose at discrete time index k, $k \in \mathbb{N}_0$, is $[\hat{x}_k^+, \hat{y}_k^+, \hat{\theta}_k^+]^T$, which is determined by the EKF-SLAM algorithm illustrated in the previous section. The motion control strategy of the robot is implemented using a proportional (P) controller in cooperation with a fuzzy logic controller. Fig. 2.2 summarizes the robot's motion control scheme used in this work. The pre-processing block takes the robot's es-

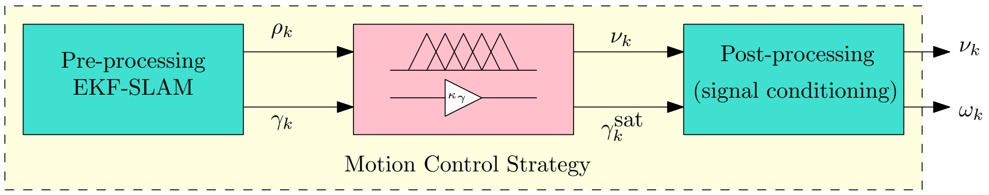


Figure 2.2: Robot's motion control mechanism.

timated pose and the ith, $i = 1, \ldots, n$, 2D point, $\mathbf{p}^{[i]}$, in the path. It then computes the Euclidean distance $\rho_k = \sqrt{\left(x^{[i]} - \hat{x}_k^+\right)^2 + \left(y^{[i]} - \hat{y}_k^+\right)^2}$ and the necessary steering angle γ_k, which are passed on to the motion controller block (see Fig. 2.2). For the robot to steer towards the ith, $i = 1, \ldots, n$, point in the path, its steering angle is determined by the P-controller using

$$\gamma_k^{\text{new}} = K_\gamma \left(\text{ATAN2}\left(y^{[i]} - \hat{y}_k^+, x^{[i]} - \hat{x}_k^+\right) - \hat{\theta}_k^+\right)$$

where $K_\gamma > 0$ is the proportional constant. Therefore, the robot's actual steering tune-up is then updated by

$$\Delta\gamma_k = \gamma_k^{\text{new}} - \gamma_k^{\text{old}},$$

where γ_k^{old} is the current steering angle of the robot's wheels. Due to limited steering rate, the steering tune-up $\Delta\gamma_k$ is saturated using the saturation function $\Delta\gamma^{\text{sat}} = \text{sign}(\Delta\gamma_k)\min(\Delta\gamma_{\max}, |\Delta\gamma_k|)$, where $\Delta\gamma_{\max} = T\Upsilon_{\max}$ with $\Upsilon_{\max}$ being the maximum allowed steering rate and the function $\text{sign}(\cdot)$ gives -1 for the argument $(\cdot) < 0$ and $+1$, otherwise. The robot's steering angle is updated using the update rule $\gamma_k^{\text{new}} = \gamma_k^{\text{old}} + \Delta\gamma^{\text{sat}}$, which is also saturated using $\gamma_k^{\text{sat}} = \text{sign}(\gamma_k^{\text{new}})\min(\gamma_{\max}, |\gamma_k^{\text{new}}|)$ with $\gamma_{\max}$ being the maximum steering angle of the robot allowed.
The robot moves towards the current point $\mathbf{p}^{[i]}$ in the path with the linear speed ν_k which is determined by a single-input single-output (SISO) fuzzy logic controller (FLC). The FLC implemented in this work is based on fuzzy logic theory [10], which is a tool of computational intelligence that has received thorough attention from the control community for its modeling capability of highly nonlinear systems. The current work relies on the robot's state estimation, therefore, finding the accurate distance between the robot and the current 2D point in the path is impossible. As such, a fuzzy logic controller is generally a choice due to its human like reasoning capability to determine the robot's speed without relying on the accurate mapping of the LoS distance between the robot and the current 2D point in

the path. In addition, an FLC is employed on-board the robot's microcontroller to mimic the behavior much like an indoor delivery robot (room service orders in hotel, for instance). The reasoning mechanisms of an FLC are represented by linguistic descriptions with a set of well-structured *if-then* rules. The *if–then* rules are based on heuristics, knowledge, and experience, and are often used to control the states of a given system. The information stored in the knowledge–base of the FLC is processed by an inference engine to determine appropriate control actions to be taken in any given operating condition. The SISO fuzzy logic control mechanism adopted in this work is similar to the one presented in [7] and its technical details are omitted here due for conciseness. Let $\rho_{\min}$ denote the minimum distance between the robot and the current 2D point, $\mathbf{p}^{[i]}$, in the path before changing its orientation towards the next point, $\mathbf{p}^{[i+1]}$. If the robot's position error with respect to the point $\mathbf{p}^{[i]}$ satisfies the condition $\rho_k \leq \rho_{\min}$, then it updates its steering angle towards the next point $\mathbf{p}^{[i+1]}$ until the point $\mathbf{p}^{[n]}$ in the path.

In this chapter, the simultaneous navigation and mapping was discussed in detail. While the robot used in this work is differential-drive, the kinematic model of an Ackermann steering vehicle is modeled to show the modularity of the algorithm. The popular EKF-SLAM algorithm was then outlined. This is followed by the measurement model to determine the range and bearing of the XBee radios in the the environment. The robot's motion is controlled by a fuzzy logic controller and a proportional controller. The next chapter shows a simulation of the simultaneous navigation and mapping algorithm.

Chapter 3

V-REP Simulation

In the previous chapter, the simultaneous localization and mapping strategy was presented. In the following chapter, this strategy is applied for use in a virtual robot simulator. The Virtual Robot Experimentation Platform (V-REP), is an industry standard robot simulator that is able to reproduce theoretical results in almost the same way as would be seen in an experimental environment. V-REP also simulates the physics of the objects and robots within the environment, leading to higher accuracy. The chapter is broken up as follows: Section 3.1 details the setup of the V-REP simulator, Section 3.2 gives some of the simulation parameters used in the algorithm, and Sections 3.3 and 3.4 present the simulation results.

3.1 V-REP Setup

In order to simulate the EKF-SLAM algorithm, a V-REP scene was constructed. An example scene containing the Pioneer 3-DX model is shown in Fig. 3.1. Based on the mobile robots available in the lab, the Pioneer 3-DX was chosen for use within in V-REP. There are multiple ways to interface with the simulator for both the control of objects and the gathering of data. For our purposes, MATLAB was chosen so data could be more

Figure 3.1: Pioneer 3-DX model used in V-REP simulations.

easily analyzed and plotted. The performance of the algorithm will be evaluated using the Root Mean Square Error. The general equation of this measure is defined as

$$\text{RMSE} = \sqrt{\sum_{k=0}^{k_f<\infty} (e[k])^2} \tag{3.1}$$

3.2 V-REP Simulation

The algorithm and control of the Pioneer 3-DX was written in MATLAB with the aim of being as modular as possible. Since V-REP currently does not support the simulation of RF signal propagation, the distance between the Pioneer model and beacons was calculated in m and converted to an RSSI value in $\text{dB} \cdot \text{m}$. Noise was then added to this value and converted back to a distance in m. Noise was also added to the bearing estimation in a similar manner.Two different simulations were run in V-REP. Other than the positions of waypoints and beacons, parameters remained the same for both simulation cases. These parameters are given in Table 3.1.

Table 3.1: Simulation parameters used within V-REP.

Name	Value	Unit
σ_ν	0.0119	$\text{m} \cdot \text{s}^{-1}$
σ_ω	0.0012	$\text{rad} \cdot \text{s}^{-1}$
σ_r	2	m
σ_β	18	$^\circ$
d	0.1950	m
ℓ	0.331	m

3.3 Simulation Case I

In the first case, the Pioneer navigated through a set of four waypoints placed on the ground (shown as red diamonds in Fig. 3.2). These waypoints were placed at $\mathbf{p}^{[1]} = [2,2]^T$ m, $\mathbf{p}^{[2]} = [2,7]^T$ m, $\mathbf{p}^{[3]} = [7,7]^T$ m, and $\mathbf{p}^{[4]} = [7,2]^T$ m. In a similar fashion, beacons are shown as green circles in Fig. 3.2. The true positions of these beacons are unknown to the algorithm in the sense that only the noisy range estimates are used to estimate their locations. However, these locations are read from V-REP by MATLAB for use in the real–time plotting of the simulation as well as error calculations. As the simulation was ran in V-REP, the data was retrieved and saved using MATLAB, allowing the data to be plotted (see Fig. 3.3).The data collected in this first simulation case shows promising results for the navigation and mapping by a mobile robot along a simple trajectory. As can be seen, the EKF-SLAM algorithm does a reasonable job of estimating the positions of the beacons as well as the pose of the mobile robot. For comparison, beacons are located at

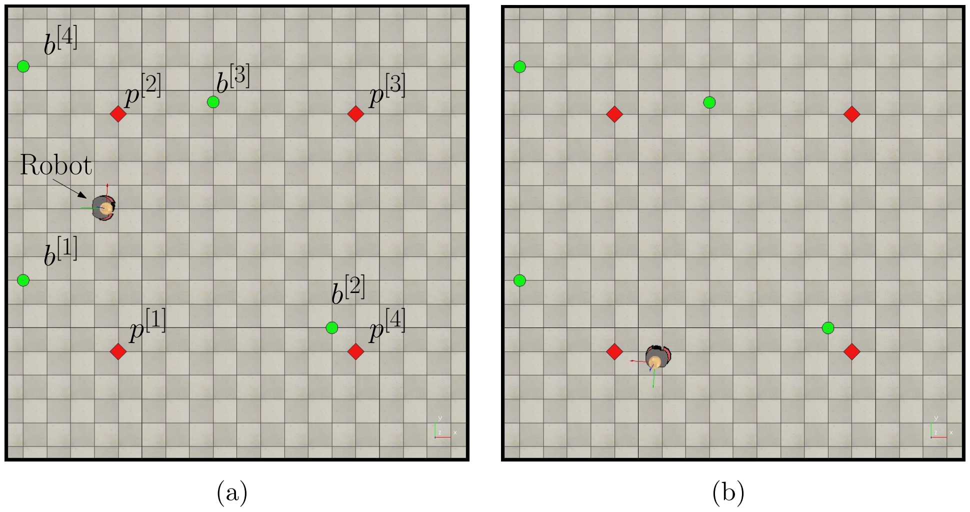


Figure 3.2: Case I: Robot configurations from V-REP while performing the simultaneous navigation and mapping algorithm at (a) $t = 25\,\mathrm{s}$ and (b) $t = 300\,\mathrm{s}$.

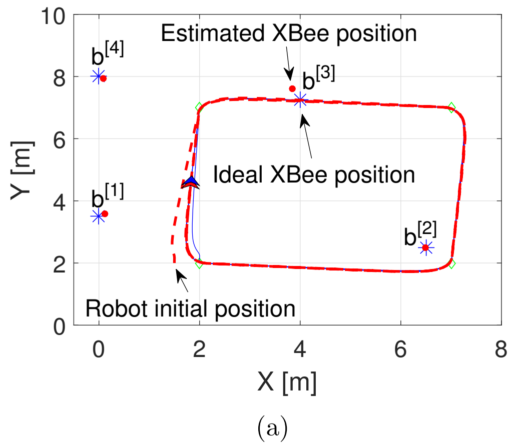
$b^{[1]} = [0, 3.5]^T$ m, $b^{[2]} = [6.5, 2.5]^T$ m, $b^{[3]} = [4, 7.25]^T$ m, and $b^{[4]} = [0, 8]^T$ m. Using Eq. 3.1, the RMSE for the robot's position error and the RMSE for the beacon position errors were calculated. These calculated values are shown in Table 3.2.

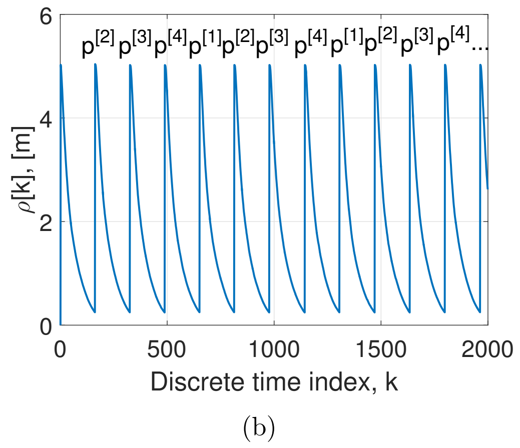
Table 3.2: Simulation Case I RMSE values.

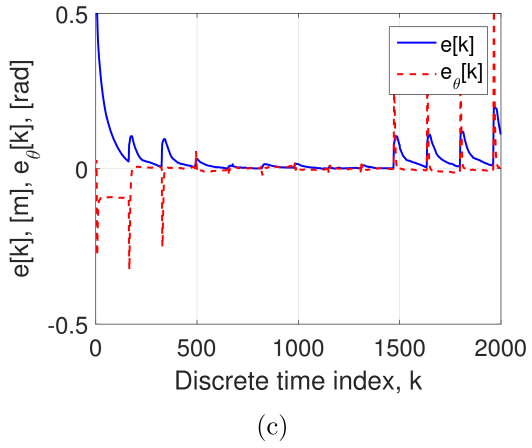
RMSE[m]	RMSE$_\theta$[rad]	RMSE$_b^{[1]}$[m]	RMSE$_b^{[2]}$[m]	RMSE$_b^{[3]}$[m]	RMSE$_b^{[4]}$[m]
0.69	0.04	0.36	0.32	0.36	0.25

3.4 Simulation Case II

To test the performance of the algorithm in a more complex environment (both trajectory and distribution of beacons), another test case was devised. This case has the robot navigating through six waypoints arranged so turns are tighter and the direction of travel alternates more. Similar to the simulation case presented in Section 2.4. Similar to the previous case, waypoints are represented by red diamonds. Shown in Fig. 3.4, the waypoints are located at $\mathbf{p}^{[1]} = [9, 3]^T$ m, $\mathbf{p}^{[2]} = [5, 5]^T$ m, $\mathbf{p}^{[3]} = [3, 7]^T$ m, $\mathbf{p}^{[4]} = [8, 8]^T$ m, $\mathbf{p}^{[5]} = [8, 2]^T$ m, and $\mathbf{p}^{[6]} = [2, 2]^T$ m. Following the same method as Case I, the data was collected from V-REP and plotted using MATLAB (shown in Fig. 3.5). Despite the more complex setup of this simulation, the algorithm still does a reasonable job of localizing the robot and mapping the environment. There is more error present in the estimation of the robot's pose. This is due to the instantaneous speed being applied that may be causing some wheel spin, leading to error in the estimation. As the speed decreases when approaching a waypoint, the error also decreases.In this simulation case, beacons are located at $b^{[1]} = [0, 0]^T$ m, $b^{[2]} = [9, 2]^T$ m, $b^{[3]} = [4, 4]^T$ m, and $b^{[4]} = [8, 7]^T$ m. The RMSEs







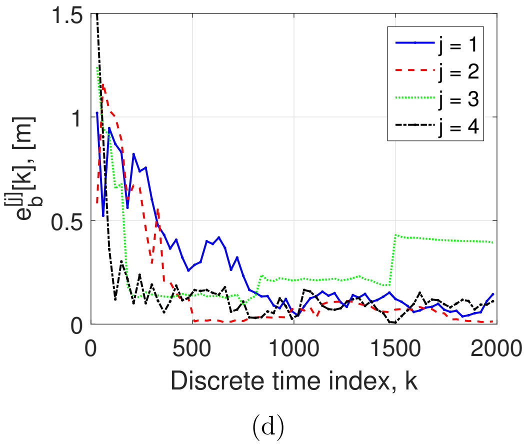
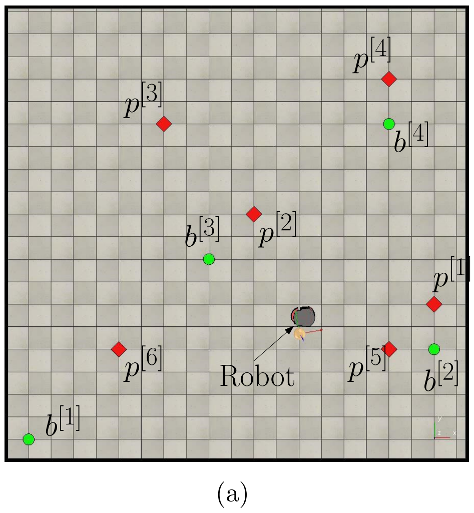


Figure 3.3: Data gathered from the first simulation case.



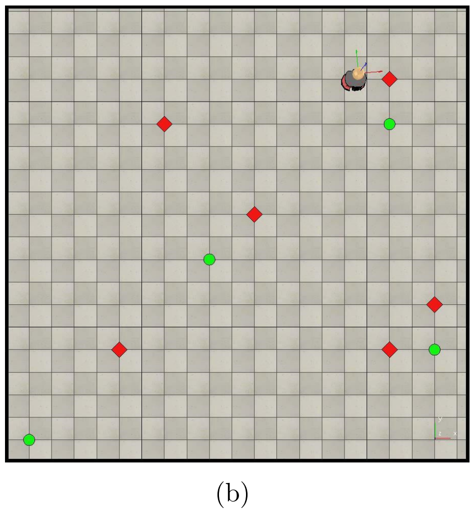
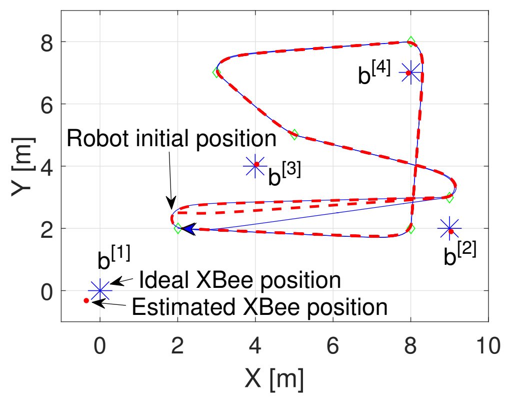
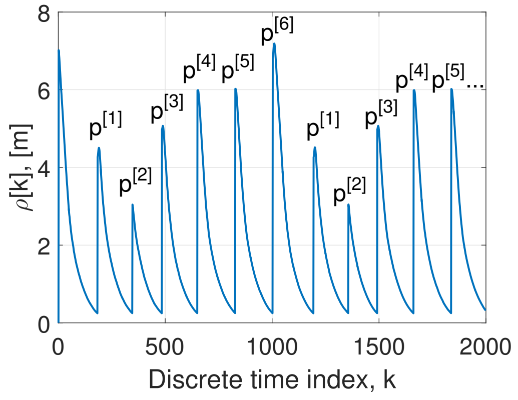


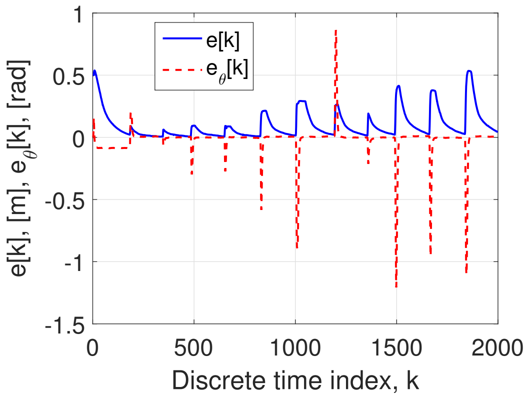
Figure 3.4: Case II: Robot configurations from V-REP while performing the simultaneous navigation and mapping algorithm at (a) $t = 25\,\mathrm{s}$ and (b) $t = 300\,\mathrm{s}$.



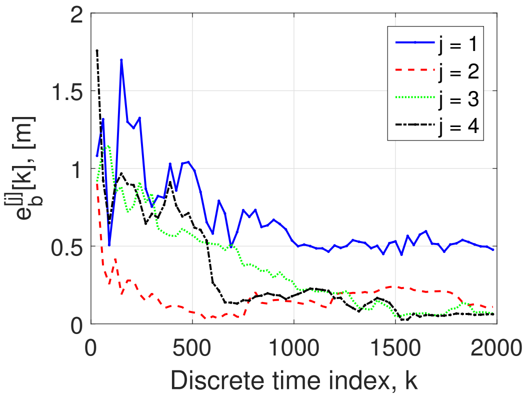
(a)



(b)



(c)



(d)

Figure 3.5: Data gathered from the second simulation case.

for this case were calculated in the same manner as in case I and are shown below in Table 3.3. Considering the noise present within each simulation case, the algorithm performs

Table 3.3: Simulation Case II RMSE values.

RMSE[m]	RMSE$_\theta$[rad]	RMSE$_b^{[1]}$[m]	RMSE$_b^{[2]}$[m]	RMSE$_b^{[3]}$[m]	RMSE$_b^{[4]}$[m]
0.15	0.14	0.73	0.21	0.46	0.47

well. In the more complex scenario of case II, more error was present. This error could be attributed to the length of the trajectory or the amount of time the simulation was run for.

In this chapter, two simulations in V-REP of the simultaneous navigation and mapping algorithm are presented. As it can be seen, the robot's pose and beacon positions are estimated reasonably. If simulations are run for longer times, the RMSE for the pose and beacon positions will be less, due to the recursive nature of the algorithm. In the next chapter, the customized radio transceiver which aids in the approximation of range and bearing of XBee radios is discussed. This hardware is necessary for the implementation of this algorithm.

Chapter 4

Customized Radio Transceiver

In this chapter, the methods for approximating range and bearing using the customized radio transceiver is outlined. The working principle behind this system is described in Section 4.1. Range and bearing approximation using the customized radio transceiver and XBee radios is covered in Section 4.2 and 4.3, respectively. The cost-effective hardware chosen to build the customized radio transceiver and its interconnections are illustrated in Section 4.4. Section 4.5 describes the open-source software running on a BeagleBone Black to acquire the RSSI measurements and communicate with other subsystems. Section 4.6 shows initial experimental results of using the customized radio receiver to approximate range and bearing.

4.1 Working Principle of the Transceiver

In order to implement the algorithm simulated previously, a customized radio transceiver was designed to measure received signal strength indicators (RSSI) from XBee S2C modules, herein beacons, dispersed in the environment. This range-only data is utilized to determine the range and bearing of each XBee with respect to the mobile robot. The design is centered around another XBee S2C module acting as the coordinator of the wireless personal area network (WPAN). The coordinator XBee is mounted within a parabolic reflector, which is mounted on a stepper motor. This design allows the reflector and coordinator XBee to rotate on the Z-axis. Due to the nature of the parabolic reflector, RSSI measurements for beacons are greatest when the reflector is pointing toward that beacon. The reflector also improves the sensitivity of the measurements, allowing for greater accuracy. The coordinate system of the customized radio transceiver is assumed to be the same as that of the mobile robot upon which it is mounted (see Fig. 4.1). This assumption allows for easier calculations and data processing within the EKF-SLAM algorithm.

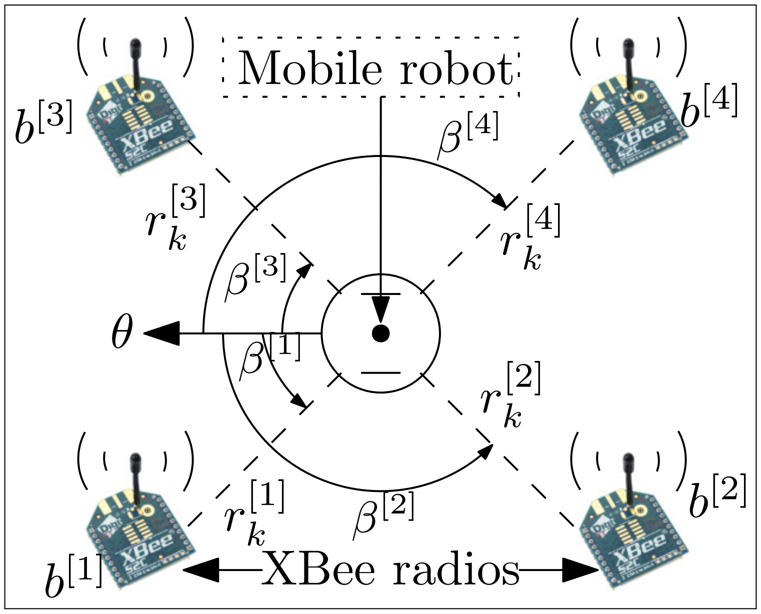
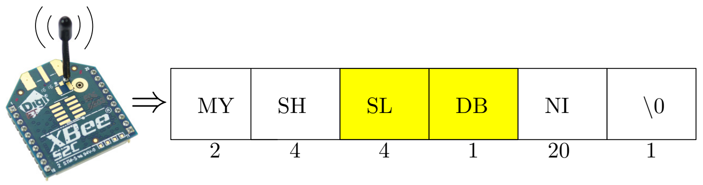


Figure 4.1: Experimental setup including four beacons $b^{[j]}$ where $r_k^{[j]}$ and $\beta^{[j]}$ denote range and bearing respectively



MY	SH	SL	DB	NI	\0
2	4	4	1	20	1

Figure 4.2: An example ZigBee data packet obtained using the *ND* command.

4.2 Range Approximation

Due to the simplicity of the ZigBee protocol utilized by the XBee S2C, the command set was all that was necessary to estimate the range between the customized radio transceiver and the beacons distributed throughout the environment. The BeagleBone Black micro-computer sends the node discover *ND* command serially to the coordinator XBee. This initiates the coordinator XBee to request that each XBee in the WPAN to respond with a data packet much like the one shown in Fig. 4.2. The coordinator XBee then fowards the receiverd data packets for each discovered XBee serially to BeagleBone Black for processing. Using the serial number *SL* field of the data packet, each XBee module is uniquely identified. This allows the RSSI stored in *DB* to be associated with its corresponding source. Upon gathering the RSSI measurements, the measurement model discussed in Section 2.3 was used to convert the RSSI measurements to a distance in meters.

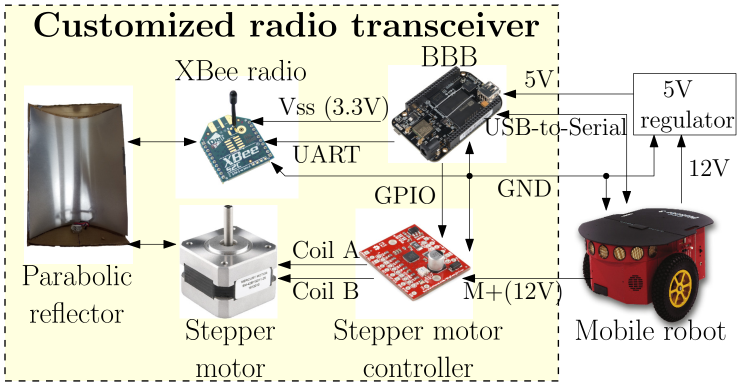


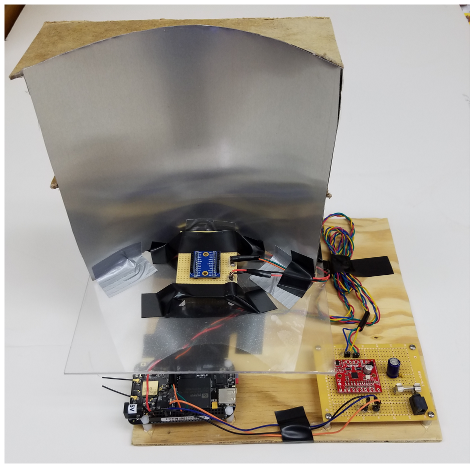
Figure 4.3: The connection diagram of the customized radio transceiver.

4.3 Bearing Approximation

The range estimation technique can be extended to the estimation of bearing as well. By rotating the parabolic reflector through a set of known positions, data can be gathered in order to estimate the bearing of each beacon. It is assumed that the strongest RSSI measurement for the jth beacon is present when the reflector is oriented in the LoS of the jth beacon. This operation to determine bearing is described mathematically in measurement model in Section 2.3. Depending on the size of the rotational step used, the accuracy can be changed (more/less measurements will cause higher/lower accuracy).

4.4 Hardware

The customized radio receiver, shown in Fig. 4.4, was built to gather RSSI measurements from all angles while minimizing cost. The main component of the customized radio transceiver is the BeagleBone Black. It has a 1 GHz processor, 512 MB of primary memory with a variety of configurable input/output ports while costing approximately \$70. The particular model of BeagleBone Black chosen was the BeagleBone Black Wireless. This new BeagleBone Black has built–in WiFi and Bluetooth capabilities, simplifying the setup and implementation. The BeagleBone Black interfaces with the coordinator XBee and stepper motor driver through its GPIO. The XBee S2C is relatively low cost, easily configurable with the XCTU software, and an API mode for structured communication. Therefore, it is utilized in both the customized radio transceiver and the beacons. The beacons are powered by a 9 V battery which is stepped down to 3.3 V by a voltage regulator and any noise is filtered by a 10 µF capacitor. To increase RSSI measurement sensitivity, an aluminum sheet was curved into a parabolic shape with hardboard. A stepper motor was chosen to rotate the reflector and XBee as the exact amount of rotation can be determined from the number of pulses inputted to the stepper motor driver. A simple power supply circuit was also designed to provide 12 V from the Pioneer 3-DX to the stepper motor driver. The interconnections of the custom radio transceiver are illustrated in Fig. 4.3.


(a)

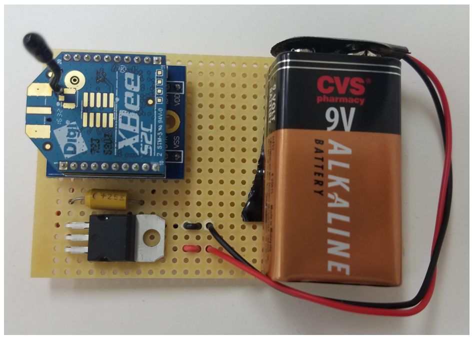

(b)

Figure 4.4: (a) The completed customized radio transceiver and (b) an assembled beacon used to implement the EKF-SLAM algorithm.

4.5 Software

The BeagleBone Black runs the Ubuntu 16.04 as to support the most recent version of the Robot Operating System. The Robot Operating System (ROS) is a collection of software frameworks written for robot development. It allows for a high–level abstraction from operations happening within the system, leading to easier programming and control. In this project, ROS is utilized for the control of the Pioneer 3-DX and the communication between subsystems. All software on the BeagleBone Black was written in C++ for compatibility with the open-source libraries used for interfacing with XBees and for easier use of GPIO on the BeagleBone Black. Since the BeagleBone Black is running Ubuntu, it was easy to write code on a laptop for easier checking of syntax and then transferring to the BeagleBone Black for implementation. Using the two open–source libraries previously described, the software was written to promote customization. By changing parameters within the code, the angle increment between observations of the environment can be tuned for the best performance for the current situation. For the experimental cases performed in the lab, the angle increment was chosen to be 9°. This was done to balance the speed of operation with the number of data points to determine the bearing to beacons.

4.6 Initial Experimental Results

Before using the customized radio transceiver to implement the EKF-SLAM algorithm, a simple range and bearing estimation test was done to gauge the performance of the constructed system. The environment for this preliminary experiment consisted of the customized radio transceiver and four beacons. These beacons were placed at known positions in order to quantify the performance. To test the customized radio transceiver's ability to measure RSSI for use in range estimations, four beacons were placed at ranges

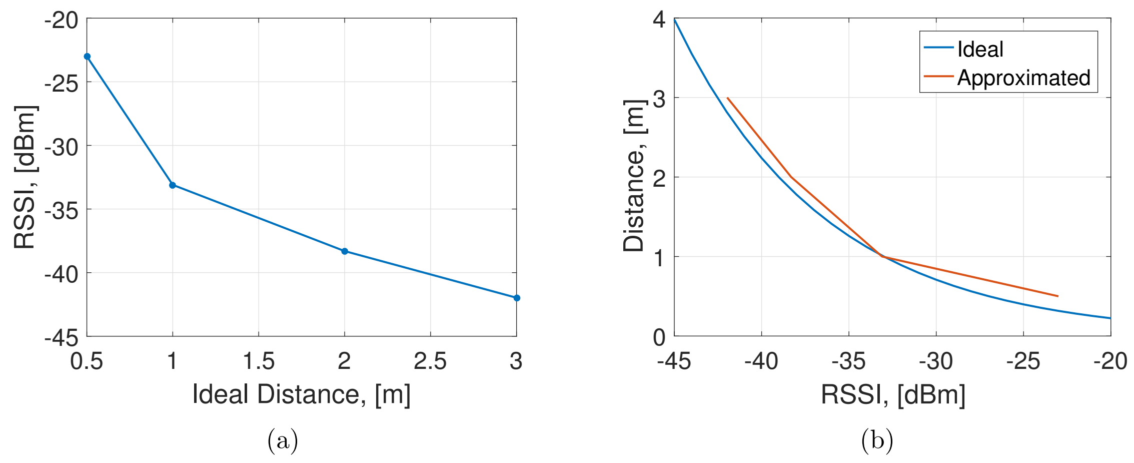


Figure 4.5: (a) Measured RSSI vs. ideal distance and (b) distance vs RSSI comparing the ideal range from equation (2.7) and the approximate range from measurements.

of 0.5 m, 1 m, 2 m, and 3 m in a straight line from the reflector, which was placed on top of the Pioneer 3-DX much like it would be while running the actual EKF-SLAM algorithm. To get the best possible results, the XBee radios were placed on the same 2D plane as the reflector and the coordinator XBee. Without rotating the reflector (no power was supplied to the motor driver), 40 RSSI measurements were taken and the data was collected. The average of these measurements for each beacon can be seen in Table 4.1 as well as a comparison with the ideal range curve using equation (2.7) in Fig. 4.5. For these preliminary experiments, $P_{\text{ref}} = -33$ dB · m and $\eta = 2$. Notice how the RSSI measurements, when plot-

Table 4.1: Average of RSSI measurements obtained with customized radio transceiver

Ideal Range	0.5 m	1 m	2 m	3 m
Measured RSSI [dB · m]	-23	-33.103	-38.308	-41.974
Approximated Range[m]	0.3162	1.0119	1.8425	2.81
Error [m]	0.1838	-0.0119	0.1575	0.19

ted, decrease in value logarithmically as would be expected. The bearing estimation was tested in a similar fashion. With the robot placed at $\mathbf{q} = [1\ \text{m}, 1\ \text{m}, 0\ \text{rad}]^T$, and beacons placed in the corners of the workspace at $\mathbf{b}^{[1]} = [0, 0]^T$ m, $\mathbf{b}^{[2]} = [0, 3]^T$ m, $\mathbf{b}^{[3]} = [3, 0]^T$ m, and $\mathbf{b}^{[4]} = [3, 3]^T$ m. Measurements were taken while rotating the reflector a full 360° and the data was plotted. This plot can be seen in Fig. 4.6. The actual estimations of bearing can be seen in Table 4.2. Notice how in Fig. 4.6 the RSSI value decreases when the reflector is not pointing toward the beacon. This is the underlying principle used to determine the bearing. Examining the error in Table 4.2, all estimates are within about 21°. While this may seem high, it is actually quite reasonable considering a rotational step size of 9° is being used. If this rotational step size was decreased, the accuracy of bearing estimates would increase.

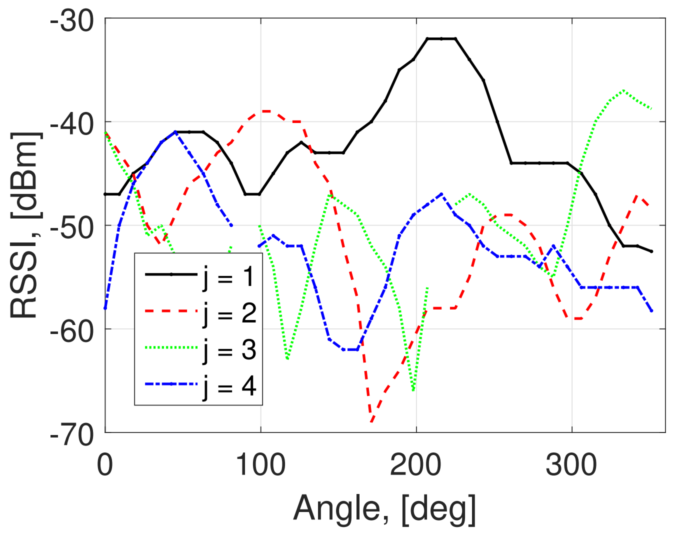


Figure 4.6: Data collected during the bearing estimation test.

Table 4.2: Bearing approximation using customized radio transceiver.

	$\mathbf{b}^{[1]}$	$\mathbf{b}^{[2]}$	$\mathbf{b}^{[3]}$	$\mathbf{b}^{[4]}$
Actual β	225°	120°	333°	45°
Measured β	207°	99°	330°	45°
Error	18°	21°	3°	0°

This chapter discusses the customized radio transceiver and initial performance tests of range and bearing approximation. The transceiver and XBee radio beacons were built to use off-the-shelf and be cost-effective. Range and bearing are approximated adequately and error is expected due to the noisy environment. The next chapter combines the simultaneous navigation and mapping technique from Chapter 2 with the customized radio transceiver into a fully-implemented system where results of multiple experiments are reported.

Chapter 5

Implementation

Using the customized radio transceiver from the previous chapter to approximate range and bearing of the XBee radios, and the EKF-SLAM algorithm presented in Chapter 2, an experimental setup was designed and is described in Section 5.1. The following sections represent four experiments with varying trajectories and XBee radio positions. A simple square trajectory is tested in Section 5.2, a more complex trajectory in Section 5.3, and a figure-eight trajectory in Section 5.4. Section 5.5 is an experiment with a real-world trajectory, but an unexpected error has corrupted some results.

5.1 Experimental Setup

Since the Pioneer 3-DX was chosen as the mobile robot for the simulation cases, it was also chosen for use in the implementation. The Pioneer 3-DX is a widely used mobile robot in research due to its accuracy and the available accessories. In this setup, it is not equipped with any accessories except the customized radio transceiver. All noise parameters were also kept the same as those given in Table 3.1 to keep results consistent. XBee radios are placed in the robot's operating range and their positions are unknown to the robot. Multiple experimental cases were run to validate the algorithm and will each be discussed in the following sections.

5.2 Experimental Case I

A simple trajectory was chosen for the first experimental case in order to verify that everything was working correctly. With the Pioneer 3-DX having an initial pose of $\mathbf{q} = [1.5, 0, \frac{\pi}{2}]^T$ m, four waypoints were placed at $\mathbf{p}^{[1]} = [0.6, 0.6]^T$ m, $\mathbf{p}^{[2]} = [0.6, 2.4]^T$ m, $\mathbf{p}^{[3]} = [2.4, 2.4]^T$ m, and $\mathbf{p}^{[4]} = [2.4, 0.6]^T$ m to form a square (shown in Fig. 5.1). For this case, the EKF-SLAM algorithm was run for 1000 iterations (not including the time it takes to conduct an observation). Upon completion, the data was processed and the plots seen in Fig. 5.2. As it can be seen in Fig. 5.2, the algorithm does a satisfactory job of

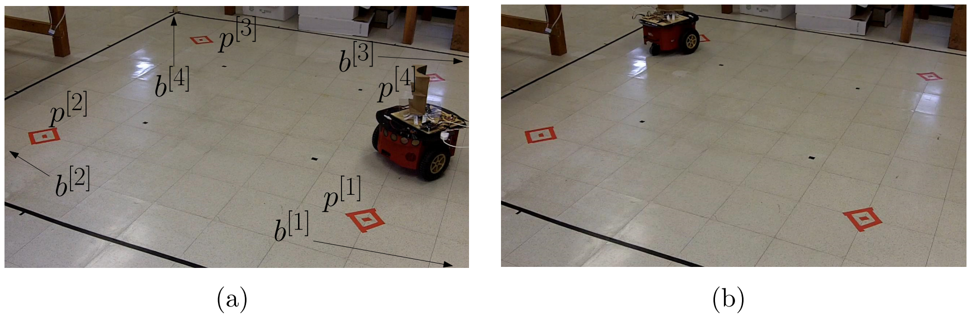


Figure 5.1: Case I: Robot's configuration while performing the simultaneous navigation and mapping algorithm at time (a) $t = 25$ s and (b) $t = 300$ s.

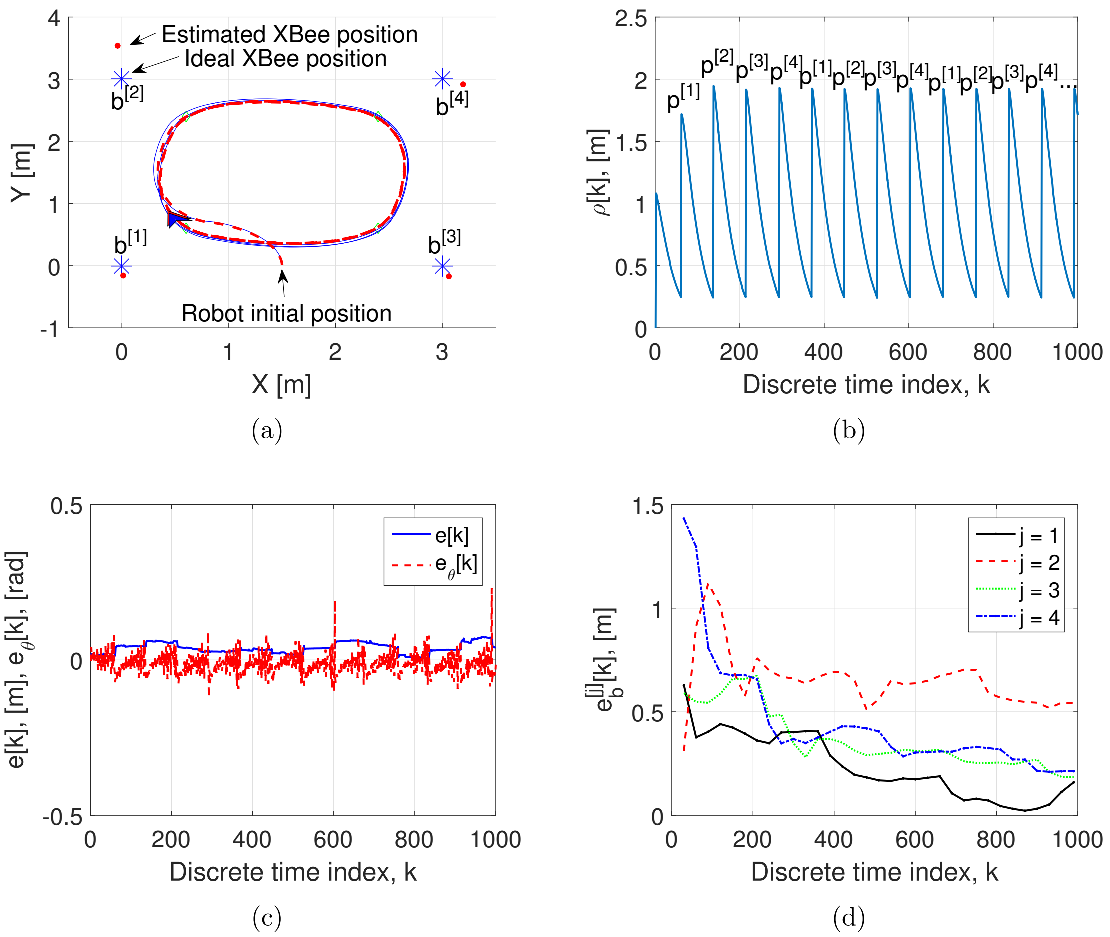
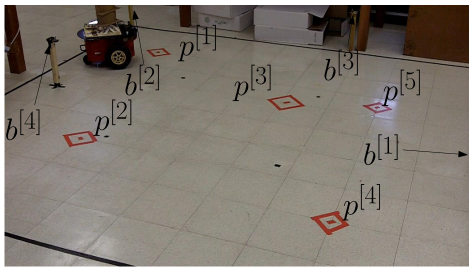
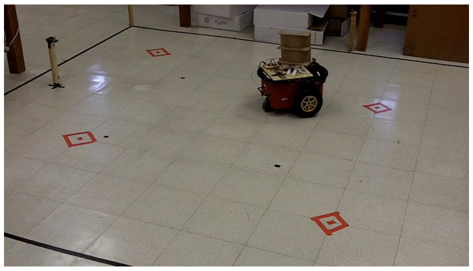


Figure 5.2: Case I: MATLAB plots showing (a) the ideal (solid line) and estimated (dashed line) robot trajectories and estimated XBee radio positions, (b) the distance estimate used by the fuzzy logic controller, (c) the robt's position estimation error, and (d) the beacon position estimation error.



(a)



(b)

Figure 5.3: Case II: Robot's configuration while performing the simultaneous navigation and mapping algorithm at time (a) $t = 25$ s and t= (b) $t = 300$ s.

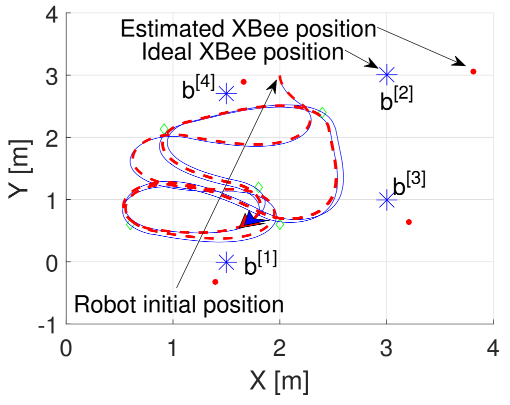
both localizing the robot and mapping the beacons placed in the corners of operating area (located at $\mathbf{b}^{[1]} = [0, 0]^T$ m, $\mathbf{b}^{[2]} = [0, 3]^T$ m, $\mathbf{b}^{[3]} = [3, 0]^T$ m, and $\mathbf{b}^{[4]} = [3, 3]^T$ m). Note the estimated position of beacon 2. While the estimate is roughly 0.5 m off from the true position, the accuracy of the estimate was still improving as the experiment ended. If the experiment continued, this error would have decreased to the amounts seen in the other beacons. Considering the physical size of the Pioneer 3-DX and the operating environment, the mapping performance is satisfactory. The localization accuracy seen in this experiment is also satisfactory in that the error was almost nil with respect to the size of mobile robot. For performance comparison between experimental cases, the RMSE for the robot position estimation error and beacon position estimation error were calculated using (3.1). These values are shown in Table 5.1.

Table 5.1: Experimental Case I RMSE values.

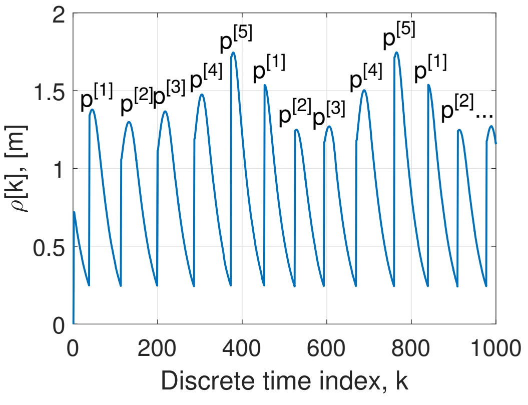
RMSE[m]	RMSE$_\theta$[rad]	RMSE$_b^{[1]}$[m]	RMSE$_b^{[2]}$[m]	RMSE$_b^{[3]}$[m]	RMSE$_b^{[4]}$[m]
0.04	0.03	0.28	0.67	0.39	0.53

5.3 Experimental Case II

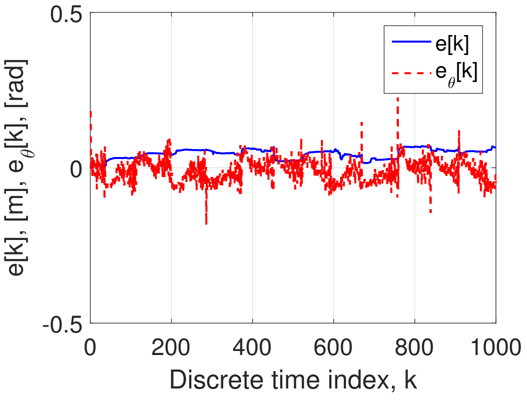
Upon satisfactory completion of the simple trajectory presented in Section 4.2, a more complex trajectory was designed with the robot's initial pose being $\mathbf{q} = [2, 3, \frac{-\pi}{2}]^T$ m. This path consists of five waypoints arranged to form a concave shape. As seen in Fig. 5.3, waypoints are located at $\mathbf{p}^{[1]} = [2.4, 2.4]^T$ m, $\mathbf{p}^{[2]} = [0.92, 2.13]^T$ m, $\mathbf{p}^{[3]} = [1.8, 1.2]^T$ m, $\mathbf{p}^{[4]} = [0.6, 0.6]^T$ m, and $\mathbf{p}^{[5]} = [2, 0.6]^T$ m. Much like Case I, the algorithm was run for 1000 iterations and data was plotted using MATLAB. These plots can be seen in Fig. 5.4. As seen in Fig. 5.4, the algorithm is again able to perform satisfactorily with respect to the size of the trajectory, complexity, and the operating area. However, there is more error present in this case when estimating the positions of the beacons (located at $\mathbf{b}^{[1]} = [0, 0]^T$ m, $\mathbf{b}^{[2]} = [9, 2]^T$ m, $\mathbf{b}^{[3]} = [4, 4]^T$ m, and $\mathbf{b}^{[4]} = [8, 7]^T$ m). This can be seen most clearly in the beacon position estimation error plot for $\mathbf{b}^{[2]}$, which, at the time the experiment ended,



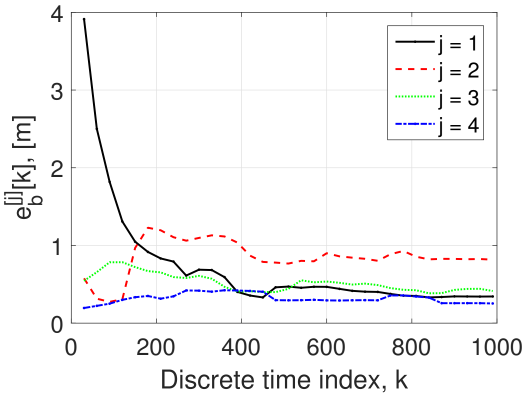
(a)



(b)



(c)



(d)

Figure 5.4: Case II: MATLAB plots showing (a) the ideal (solid line) and estimated (dashed line) robot trajectories and estimated XBee radio positions, (b) the distance estimate used by the fuzzy logic controller, (c) the robt's position estimation error, and (d) the beacon position estimation error.

had an error close to 1 m. This can be attributed to the length of the trajectory being followed leading to less measurements being done in the vicinity of $\mathbf{b}^{[2]}$. If the number of max iterations were to be increased, this error would decrease. The RMSE values for this experiment can be seen in Table 5.2. While the performance in this case was not as good

Table 5.2: Experimental Case II RMSE values.

RMSE[m]	RMSE$_\theta$[rad]	RMSE$_b^{[1]}$[m]	RMSE$_b^{[2]}$[m]	RMSE$_b^{[3]}$[m]	RMSE$_b^{[4]}$[m]
0.05	0.04	1.03	0.88	0.53	0.33

as in the first case, they are still satisfactory since the beacon position estimation error is decreasing as the number of iterations increase and the error for the robot is negligible with respect to its own size.

5.4 Experimental Case III

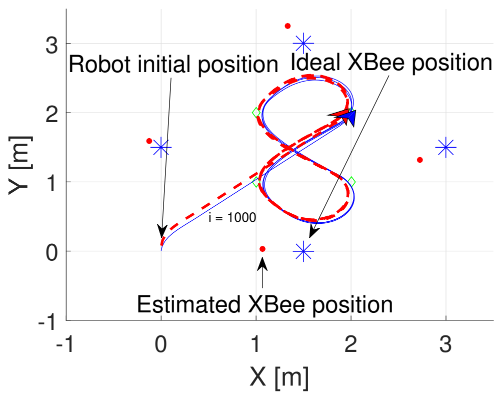
For the following two experimental cases, the images of the mobile robot driving in the lab environment are omitted as they are not necessary for the analysis of the results. In this experimental case, the Pioneer 3-DX's initial pose is $\mathbf{q} = [0, 0, \frac{\pi}{2}]^T$ m and follows a small figure-eight with waypoints at $\mathbf{p}^{[1]} = [2, 2]^T$ m, $\mathbf{p}^{[2]} = [1, 2]^T$ m, $\mathbf{p}^{[3]} = [2, 1]^T$ m, and $\mathbf{p}^{[4]} = [1, 1]^T$ m. After running for 1000 iterations, the data was plotted (see Fig. 5.5). As shown in Fig. 5.5, the algorithm had no issues following the defined trajectory. Again, due to the nature of RSSI measurements, there is noticeable error in the beacon position estimations. The error for $\mathbf{b}^{[1]}$ is actually increasing after about 400 iterations. While this is not desireable, the accuracy of the other three beacons is still satisfactory. Over time, the error for $\mathbf{b}^{[1]}$ would begin to decrease again, but over just 1000 iterations of the EKF-SLAM algorithm, its performance is still reasonably acceptable. The RMSE values for this experiment are shown in Table 5.3.

Table 5.3: Experimental Case III RMSE values.

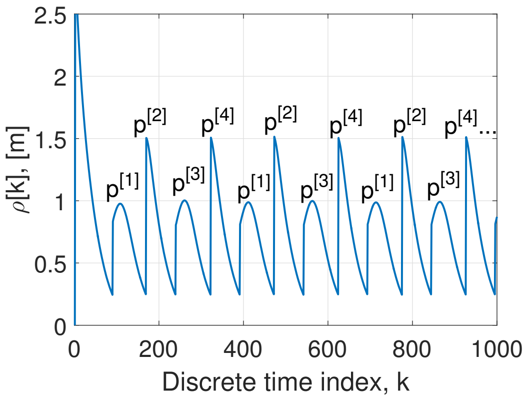
RMSE[m]	RMSE$_\theta$[rad]	RMSE$_b^{[1]}$[m]	RMSE$_b^{[2]}$[m]	RMSE$_b^{[3]}$[m]	RMSE$_b^{[4]}$[m]
0.06	0.04	0.41	0.36	0.34	0.66

5.5 Experimental Case IV

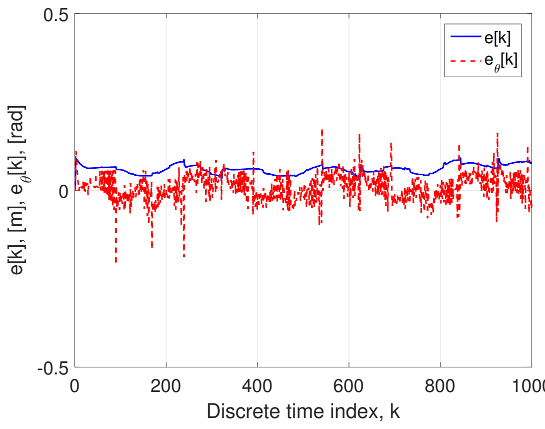
We also ran an experiment that would be similar to a real–world scenario, with a longer trajectory, more obstacles in the environment affecting measurements, and a longer run time. In this test, the robot's initial pose is $\mathbf{q} = [0, 0, \frac{\pi}{2}]^T$ m and waypoints were placed at $\mathbf{p}^{[1]} = [2, 1.5]^T$ m, $\mathbf{p}^{[2]} = [2.4, 8.4]^T$ m, $\mathbf{p}^{[3]} = [-0.6, 8.5]^T$ m, and $\mathbf{p}^{[4]} = [-0.9, 1.2]^T$ m. Unfortunately, one of the beacons lost power and a bug in the code caused the program



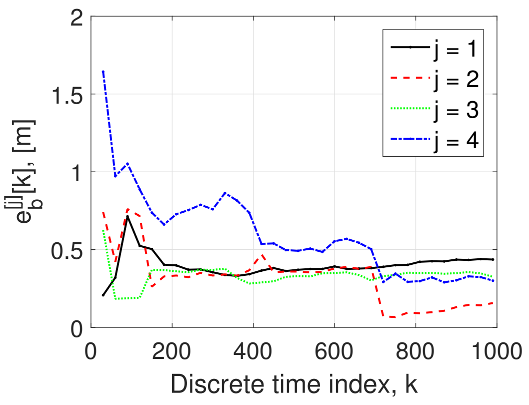
(a)



(b)



(c)



(d)

Figure 5.5: Case III: MATLAB plots showing (a) the ideal (solid line) and estimated (dashed line) robot trajectories and estimated XBee radio positions, (b) the distance estimate used by the fuzzy logic controller, (c) the robt's position estimation error, and (d) the beacon position estimation error.

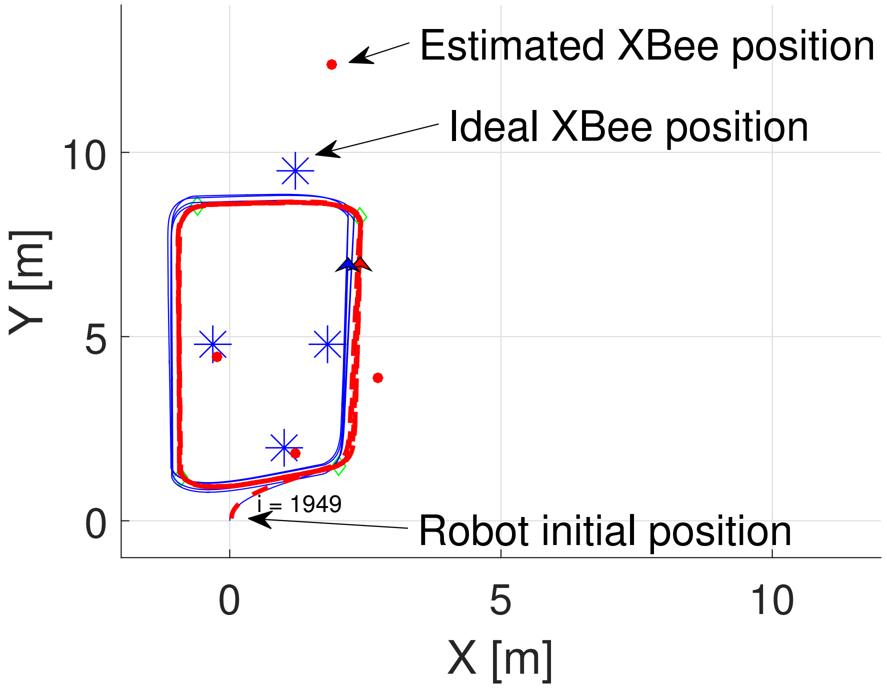


Figure 5.6: Case IV: The ideal (solid line) and estimated (dashed line) robot trajectories and estimated XBee radio positions.

to fail and most of the data was lost. The only data that was able to be recovered was the robot's current estimation of its position, the estimates of where the beacons were placed, and the MATLAB figure showing the final trajectory. This figure can be seen in Fig. 5.6. Using the last known estimates of the positions of the four beacons (located at $\mathbf{b}^{[1]} = [1, 2]^T$ m, $\mathbf{b}^{[2]} = [1.8, 4.8]^T$ m, $\mathbf{b}^{[3]} = [1.2, 9.5]^T$ m, and $\mathbf{b}^{[4]} = [-0.3, 4.8]^T$ m.) the error was able to be calculated. This error can be seen in Table 5.4. Despite the program

Table 5.4: Experimental Case IV final estimation error values.

$e_b^{[1]}$[m]	$e_b^{[2]}$[m]	$e_b^{[3]}$[m]	$e_b^{[4]}$[m]
0.28	1.32	2.97	0.36

crashing about 50 iterations before it would have been completed, it can still be seen that there is a reasonable amount of accuracy with respect to the size of the trajectory being followed. If the data had been able to be recovered, the plots would have looked similar to the other experimental cases with decreasing estimation error. Due to the data loss, the RMSE was not able to be calculated.

As it can be seen from the first three experimental cases, the mobile robot navigates through the waypoints very well and the positions of the XBee radios are mapped reasonably. The last experiment shows that this system can be applied to a real-world scenario and it is unfortunate that the data was lost. Like the simulations from Chapter 2, if the experiments are run for a longer period of time, the RSME of the robot's pose and XBee radio positions will decrease. The following chapter will draw conclusions from this work and propose future related work.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

As can be seen in the work presented in this reported, the designed system used in conjunction with the widely used EKF-SLAM algorithm offers a modest amount of accuracy in short to medium range (around 3 m) and when used in noisy environments. Over larger trajectories the error is more apparent but over time the error is still decreasing due to the nature of the EKF-SLAM algorithm. The original specifications planned for this project were (1) a cost less than \$500, (2) have the robot localized within 30 cm of its true position, and (3) estimate the positions of beacons within 20 cm of their true positions. The first specifications was easily met as the total cost of the customized radio transceiver is about \$140 and each beacon was about \$20. The second two specifications are a little more complex, as the RMSE was chosen instead to measure algorithm performance. However, looking at the error plots for each experimental case, the error for both the robot's position estimate and the beacon position estimates are around the desired specification. Of course, by running the algorithm for a longer period of time the specifications will eventually be met.

6.2 Future Work

While the presented system provides a reasonable amount of accuracy for the cost, there are still some improvements that could be made to improve the overall performance. First, the addition of a slip ring (such as the one found here https://www.adafruit.com/product/736) and a new mounting solution would let the reflector assembly rotate without wires becoming tangled around the stepper motor's axle. This could lead to faster performance since the reflector would not need to rotate back to the home position after each observation. To that extent, the software could be modified so the direction of the reflector's rotation would alternate for every observation scan, eliminating the problem of tangled wires. Another possible improvement would be further tuning of the observation scan functions themselves. It would not be too difficult to minimize the amount pauses or to find the optimal speed for the stepper motor rotation. If responses from beacons could be minimized, the time it takes to make a complete observation would decrease and it would probably appear as if the reflector was not stopping at all. One other major aspect of the algorithm that could be improved is the method of prediction for $\mathbf{q}$ since there are still some inaccuracies in certain situations such as fast turns or when the Pioneer 3-DX is coming to a stop for an observation.

System performance could be improved by using a store bought reflector and antenna. This was not pursued in the current work due to our limited RF knowledge and desire for a less expensive, more attainable system. The hardware for range and bearing estimation subsystem could be implemented using a microcontroller instead of the BeagleBone Black. This may increase efficiency and reduce the amount of computations done by the BeagleBone Black.

With the completion of this work, several new avenues of research have been opened using our work as the base. Some possible topics include an EKF-SLAM algorithm that only relies on bearing estimations for use in the localization and mapping subsystem. This is still a relatively new technique for use in localization and mapping algorithms and would be a novel extension of the completed work. In future extensions of this project, it may be advantageous to find a way to accurately measure the true position of the mobile robot instead of comparing to encoder values as those are not always correct.

APPENDICES

Appendix A

Running V-REP Simulations

This appendix gives a simple set of steps to simulate the algorithm using V-REP. In Section A.1 the V-REP setup is explained and in Section A.2 the steps necessary for running the simulation are given.

A.1 Setup

The simulations presented in this report were completed using V-REP and the remote API for MATLAB. There are no objects included in the scene other than the markers used to show where waypoints and beacons are located. The code is written in such a way that dummy objects placed in V-REP act as beacons. These dummies are stationary so at the beginning of the MATLAB code, the locations are just retrieved and stored. This allows the visual aspect in V-REP to be used directly by MATLAB.It is not necessary to create a custom V-REP scene to simulate the algorithm as a sample scene is included with the simulation code. The simulation code can be found within the file named *simulationV10.m*. The V-REP scene associated with this MATLAB script is called *simulationV10.ttt*. If desired, the dummy objects labelled as beacons can be repositioned and the change can be seen in MATLAB when the simulation is started, as detailed in the next section.

A.2 Runtime

If the provided V-REP scene is being used for the simulation, no settings need to be changed. Press the play button on the top toolbar to start the simulation. This only starts the V-REP simulation. Now in MATLAB, open the provided MATLAB code. There should be no changes necessary for it to run so just press *Run* again. It will take a couple seconds for it to connect to the simulator but once everything is set up the Pioneer 3-DX model should begin to move in V-REP. This will run until the maximum iterations (customizeable within the MATLAB script) is met. The program will then plot all the data collected during the simulation and save the recorded video that was created by MATLAB.

Some examples of the plots created after the V-REP simulation completes are shown in this report (see Fig. 3.3 and Fig. 3.5).

Appendix B

Running Experiments

This appendix gives the steps necessary for running the algorithm on a physical robot, namely the Pioneer 3-DX. Sectionsec:append-bbb-setup gives a detailed set of instructions to set up the BeagleBone Black Wireless. Section B.2 gives some notes on things to consider when setting up XBees. Section B.3, explains the setup needed to run the algorithm and then Section B.4 details how to start the actual experiment.

B.1 BeagleBone Black Wireless Setup

In order to run the experiments in the lab, quite a bit of setup is needed. This setup is done with the assumption that the BeagleBone Black Wireless is being used.

Before setting up anything on the BeagleBone Black, the disk image for Ubuntu 16.04 LTS must be downloaded, put on a 16 GB microSD card. The disk image used for this project can be found at `http://rcn-ee.net/rootfs/2017-01-23/microsd/bone-ubuntu-16.04.1-console-armhf-2017-01-23-2gb.img.xz`. This can be placed on a microSD using Win32 Disk Imager (available at `https://sourceforge.net/projects/win32diskimager/`).

Once the disk image is on the microSD card, put it in the BeagleBone. If you are using a new BeagleBone, you will need to hold down the boot select button, *S2*, while connecting the power for the first time so the BeagleBone will automatically boot from the microSD card in the future. At the time of install, the image is taking up only 2 GB of the microSD. This must be expanded to use all of the available space. The Linux program *Parted* will be used to accomplish this. Unfortunately, this program is not included in the image and must be installed first, which requires an Internet connection. For conciseness, it is assumed that the network used for ROS communication is also connected to the Internet. In the case that it is not, the process of connecting to a new network will need to be repeated each time it is desired to change networks. The command below will connect the BeagleBone Black to the network with the name *ECE-Robots-1* using interface *wlan* without a password. It also sets a static IP of *192.168.1.3* that is necessary for the ROS node to connect to MATLAB in later sections.

```
sudo nmcli con add con-name wireless ifname wlan type wifi ssid ECE-Robots-1 \ip4 192.
```

These values can be changed to whatever is needed for your specific router. You should now be connected to the Internet. Before installing *Parted*, you should update Linux using the below command.

```
sudo apt-get update
```

Now install *Parted* using the following command and once it is installed, find the name of the boot device.

```
sudo apt-get install parted
df -h /
```

Most likely the name of the boot partition will be */dev/mmcblk0*. If it is something else then the command will need to be changed accordingly.

```
sudo fdisk /dev/mmcblk0
p
```

Take note of the boot partition (marked with an asterisk) after using the *p* command within fdisk. Also write down the starting sector.

```
d
n
```

Use the defaults until you are asked for the starting sector. For the starting sector, use the exact same value that you wrote down earlier or nothing will work when you reboot. Continue accepting the defaults until it says a new partition has been created. Use *p* to view the list of partitions again. If the boot partition is not marked as such, use the following command.

```
a
```

To finish setting up the partitions, use these commands.

```
w
sudo partprobe
sudo resize2fs /dev/mmcblk0p1
```

The filesystem now fills the entire microSD card. It is a good idea to run updates again now just in case.

```
sudo apt-get update
sudo apt-get upgrade
```

Now it is time to setup ROS. All of the commands for this are given below.

```
cd
mkdir seniorProject
cd seniorProject
git clone https://www.github.com/attie/libxbee3
cd
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /e
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9
sudo apt-get update
sudo apt-get install ros-kinetic-ros-base
sudo rosdep init
rosdep update
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt-get install python-rosinstall
```

The base ROS install will take a good amount of time since there is a lot to download, unpack, and install. There is still more to install, as detailed below.

```
. /opt/ros/kinetic/setup.bash
cd seniorProject
mkdir -p /catkin_ws/src
cd /catkin_ws/src
catkin_init_workspace
cd ../
catkin_make
. /devel/setup.bash
cd /src
git clone https://github.com/amor-ros-pkg/rosaria.git
sudo apt-get install ros-kinetic-tf
```

Now that the ROS package for the Pioneer is downloaded, ARIA must be downloaded and installed before continuing. These instructions assume you are still */seniorProject/catkin_ws*.

```
cd ../
wget http://robots.mobilerobots.com/ARIA/download/current/ARIA-src-2.9.1.tar.gz
tar -xvzf ARIA-src-2.9.1.tar.gz
cd Aria-src-2.9.1/src
make
cd ../ArNetworking
make
cd ../
sudo make install
```

With ARIA compiled, the library can be added to the library cache.

```
sudo nano /etc/ld.so.conf
```

In this file, add the following text in a new line at the end of the file.

```
/usr/local/Aria/lib
```

Save and exit the file. Continuing with the installation of ROS components,

```
sudo ldconfig
rosdep install rosaria
```

Everything needed for rosaria is now downloaded and the package can be built. Make sure you have successfully compiled and installed ARIA before doing any of the commands below or it will not work (build errors).

```
cd ../catkin_ws
catkin_make
```

Note: There is a alternate version of ROSARIA used in this project that allows the pose in the robot odometry to be changed using ROS, this lets multiple experiments be run without restarting the robot. This version can be downloaded at https://github.com/TheSmallHill/rosaria.

Now that everything is setup for the Pioneer, download the EKF-SLAM package from https://github.com/TheSmallHill/ekf_slam/tree/ros-matlab-1. Make sure to download the branch called *ros-matlab-1*. In the case that this branch no longer exists, download the master branch as *ros-matlab-1* has been merged to the master. Use the following command from the base folder of the catkin workspace to build just the *ekf_slam* package.

```
catkin_make --pkg ekf_slam
```

This is the end of the setup for ROS. The following section gives some high–level comments on the setup of XBees.

B.2 XBee Network

Instructions to set up an XBee network will not be given in this appendix as there is a multitude of tutorials online. Some things to consider will be given instead. The XBee mounted within the rotating reflector needs to be setup as the coordinator of the network and beacons will be setup as endpoints. In order for the algorithm to function, the XBees must be given node identifiers in the form *beacon#* where # is the beacon number. All beacons being used must be numbered in order starting at one as well. All beacons have the same pan ID as well so any message the coordinator sends will go to all beacons in the network. If desired, more information on the XBee setup can be sought in the lab notebook for this project.

If these considerations are kept in mind when setting up the XBee network there should be no issues. The next section details how to start the ROS nodes for EKF-SLAM and the Pioneer.

B.3 ROS Node

The setup for running experiments is a little more involved than the setup for the simulation cases. First plug in the customized radio transceiver into the two DC power jacks coming out of the Pioneer 3-DX. Be sure to use the correct jack for each connection, the DC power connector with an inline voltage regulator must be plugged into the BeagleBone Black Wireless. If the BeagleBone Black has not already been configured for connecting to a wireless network, that must be done over a direct serial connection. The wireless connection is required for MATLAB to control the customized radio transceiver and the Pioneer 3-DX.Once you are able to connect to the BeagleBone Black using an SSH connection, a couple commands must be run before starting the ROS node. These commands are detailed below.

```
sudo su
echo BB-UART2 > /sys/devices/platform/bone_capemgr/slots
exit
cd seniorProject/catkin_ws
source devel/setup.bash
```

At this point the ROS node is ready to be started on the BeagleBone Black. To do this, use the following command,

```
roslaunch ekf_slam node.launch
```

The Pioneer 3-DX should make an activation sound and the startup information will be displayed in your SSH session window.

B.4 Runtime

Now that the ROS node is running on the BeagleBone Black, all that needs to be done is to change the robot's initial pose, waypoint locations, and the beacon locations. At this time these values must be measured by hand and entered into the program manually. Open the MATLAB script titled *ekfSlam.m* and do this. Make sure you place the Pioneer near its intended starting position and then press *Run*. Once the program starts successfully, no further action is required.If desired, the values for the proportional controller and maximum velocities can be adjusted before pressing *Run*.

Bibliography

[1] T. Alhmiedat, F. Omar, and A. Abu Taleb. A hybrid tracking system for zigbee wsns. In *2014 6th Int. Conf. on Comput. Sci. and Inf. Technol. (CSIT)*, pages 71–74, March 2014.

[2] L. Cheng, C. d. Wu, and Y. z. Zhang. Indoor robot localization based on wireless sensor networks. *IEEE Trans. Consum. Electron.*, 57(3):1099–1104, August 2011.

[3] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB.* Springer–Verlag, Berlin Heidelberg, 1 edition, 2011.

[4] E. DiGiampaolo and F. Martinelli. Mobile robot localization using the phase of passive uhf rfid signals. *IEEE Trans. Ind. Electron.*, 61(1):365–376, Jan 2014.

[5] Qian Dong and W. Dargie. Evaluation of the reliability of rssi for indoor localization. In *Int. Conf. on Wirel. Commun. in Underground and Confined Areas*, pages 1–6, Aug 2012.

[6] C. Gentner, T. Jost, W. Wang, S. Zhang, A. Dammann, and U. C. Fiebig. Multipath assisted positioning with simultaneous localization and mapping. *IEEE Trans. Wireless Commun.*, 15(9):6104–6117, Sept 2016.

[7] Wail Gueaieb and Suruz Miah. An intelligent mobile robot navigation technique using RFID technology. *IEEE Trans. Instrum. Meas.*, 57(9):1908–1917, September 2008.

[8] Dirk Hahnel, Wolfram Burgard, Dieter Fox, Ken Fishkin, and Matthai Philipose. Mapping and localization with RFID technology. In *IEEE Int. Conf. on Rob. and Autom.*, pages 1015–1020, New Orleans, LA, United States, April 2004.

[9] Donghwa Jeong and Kiju Lee. Directional rss-based localization for multi-robot applications. In *Proceedings of WSEAS conf. on signal processing, robotics, and automation*, 2013.

[10] Fakhreddine Karray and Clarence W. de Silva. *Soft Comput. and Intell. Syst. Design, Theory, Tools and Appl.* Addison-Wesley, Pearson Education Limited,, Essex, England, 4th edition, 2004.

[11] Jacob Knoll, Kyle Hevrdejs, and Suruz Miah. Virtual robot experiments for navigation in structured environments. In *The 26th Int. Symp. on Ind. Electron.*, Edinburgh, Scotland, June 2017.

[12] E. Leitinger, P. Meissner, M. Lafer, and K. Witrisal. Simultaneous localization and mapping using multipath channel information. In *IEEE Int. Conf. on Commun. Workshop*, pages 754–760, June 2015.

[13] V. Malyavej, W. Kumkeaw, and M. Aorpimai. Indoor robot localization by rssi/imu sensor fusion. In *Int. Conf. on Electr. Eng./Electron., Comput., Telecommun. and Inf. Technol.*, pages 1–6, May 2013.

[14] F. Martinelli. A robot localization system combining rssi and phase shift in uhf-rfid signals. *IEEE Trans. Control Syst. Technol.*, 23(5):1782–1796, Sept 2015.

[15] Suruz Miah and Wail Gueaieb. A stochastic approach of mobile robot navigation using customized rfid systems. In *Int. Conf. on Signals, Circuits and Syst.*, Jerba, Tunisia, November 2009.

[16] Suruz Miah and Wail Gueaieb. Mobile robot trajectory tracking using noisy rss measurements: An rfid approach. *ISA Trans.: The Journal of Automation, Elsevier*, 53(2):433–443, March 2014.

[17] Suruz Miah and Wail Gueaieb. Rfid-based mobile robot trajectory tracking and point stabilization through on-line neighboring optimal control. *J. Intell. Rob. Syst.*, 78(3–4):377–399, June 2015.

[18] Suruz Miah, Wail Gueaieb, and Davide Spinello. Linear time-invariant feedback operator for mobile robot trajectory tracking. In *IEEE Int. Instrum. and Meas. Technol. Conf.*, Pisa, Italy, May 2015.

[19] G. Oriolo, A. De Luca, and M. Vendittelli. Wmr control via dynamic feedback linearization: design, implementation, and experimental validation. *IEEE Trans. Control Syst. Technol.*, 10(6):835–852, Nov 2002.

[20] Joan Sola, , Teresa Vidal-Calleja, , Javier Civera, and Jose Maria Martinez Montiel. Impact of landmark parametrization on monocular ekf-slam with points and lines. *Int. J. Comput. Vision*, 97(3):339–368, 2012.

[21] D. Song, C. Y. Kim, and J. Yi. Simultaneous localization of multiple unknown and transient radio sources using a mobile robot. *IEEE Trans. Rob.*, 28(3):668–680, June 2012.

[22] S. Tomic, M. Beko, and R. Dinis. Rss-based localization in wireless sensor networks using convex relaxation: Noncooperative and cooperative schemes. *IEEE Trans. on Veh. Technol.*, 64(5):2037–2050, May 2015.

[23] H. Zhou, K. Ni, Q. Zhou, and T. Zhang. An sfm algorithm with good convergence that addresses outliers for realizing mono-slam. *IEEE Trans. on Ind. Inf.*, 12(2):515–523, April 2016.