Mobile Target Tracking Using a Radio Sensor Network

by

Nic Auth and Grant Hovey
Advisor: Dr. Suruz Miah

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

Abstract

A critical problem that must be solved in most applications of mobile robotics is the self localization of a mobile robot and map generation in an totally unknown environment using only readings from external sensors. This is important for many tasks that require the mobile robot to reference its own location and the location of landmarks in its environment such as navigation. There is a large breadth of research literature and experimental result related to this subject using many different methodologies of sensor data collection and algorithms to process said data. This is however still a highly active area of research as more resource efficient and accurate solutions are capable of being designed. One of the current challenges in simultaneous localization is the implementation of a cost efficient modular solution for operation in an indoor environment. This challenge becomes even more complicated if the scope is expanded to include a solution for the mobile target tracking problem. The mobile target tracking problem can be described as such; a mobile target and a follower robot are placed in an unknown environment, with neither the target or robots pose (position and orientation) being known. The follower is required to track the unknown trajectory of the mobile target using only measurements from external sensors. This project solves this problem using range only measurements of signals generated by a network of wireless sensors placed throughout the robot's operating environment. A custom built radio transceiver is mounted on the follower robot to communicate with the wireless sensors and gather the sensor measurements. The trajectory of the mobile target will be completely randomized within the operating environment. This mobile target will be represented by a designated sensor in wireless sensor network. A standard Extended Kalaman Filter Simultaneous Localization and Mapping (EKF-SLAM) algorithm will be used to turn the sensor measurements into usable position information. EKF-SLAM will provide us with an estimate of the follower robot's pose, the position of the mobile target, and it will generate a map of the environment by estimating the position of landmarks, in this case the wireless sensors in the operating area. A simple PI(proportional integral) controller will be used to determine the appropriate velocity to for the follower to track the mobile target and remain within a minimum safe distance. We firstly implemented this solution virtually in a commercial robotics simulator V-REP (Virtual Robot Experimentation Platform), and then using a mixture of pre-built and customized hardware using the Robotics Operating System (ROS) platform for system integration.

Acknowledgements

The authors would like to thank the Caterpillar College of Engineering and the Electrical and Computer Engineering Department as well as Mr. Christopher Mattus, and Mr. Nick Schmidt.

Table of Contents

List of Figures

Chapter 1

Introduction

Mobile robots are becoming an important facet of our modern economy and soon our daily lives. Applications for mobile robotics exist in many fields including; automated tractors and spraying equipment for agriculture, autonomous vehicles for transportation, automated forklifts for use in warehouses and industrial settings, and many more. In many of these applications a mobile robot may be placed into an environment that is not known to it a priori.

Sensory data must be collected by the mobile robot to build a map of the environment and to localize itself within the environment for it to be able to reliably navigate. This problem is known in the robotics community as Simultaneous Localization and Mapping or more succinctly as SLAM. There are currently many existing systems on the market to solve this problem using a variety of sensor types to gather data about the robot's environment and algorithms to process this data into a usable map and position information. Further complexities can arise when the SLAM problem is extended to include tracking and following the position of a mobile target.

This work approaches to solve the SLAM with mobile target tracking problem by extending an existing SLAM solution. [1] This SLAM system was implemented using range only measurements from a network of static radio sensors placed throughout the operating environment. These measurements are processed by an Extended Kalman Filter Simultaneous Localization and and Mapping algorithm (EKF-SLAM) into usable navigational data including position and orientation of the mobile robot and position of the radio sensors to serve as landmarks in the map of the environment. We modified this existing system both to improve its performance and adapt it to track and follow a moving target while avoiding obstacles in its path.

Chapter 2

Theory

This chapter discuses the theory used for our navigation model, for our measurement model, robot model, and the motion control model. First Section 2.1 analyzes the EKF-SLAM algorithm. Section 2.2 elaborates on the model using RSSI to determine the range and bearing of the radio sensors. Next, Section 2.3 shows the model of the robot used and its kinematics. Finally, Section 2.4 takes a look at the motion control strategy for the mobile robot. Figure 2.1 shows the high level system block diagram for our system. The following sections will elaborate on these blocks.

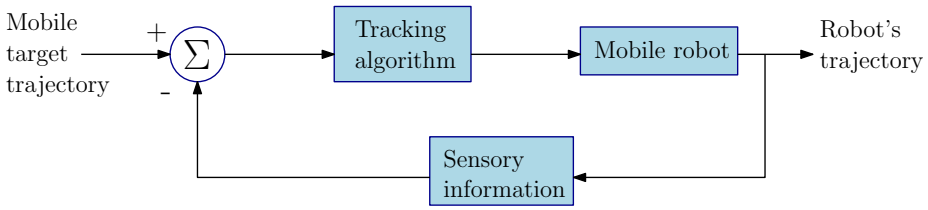


Figure 2.1: System block diagram

2.1 EKF-SLAM Algorithm

Simultaneous localization and mapping (SLAM) has been a common problem to solve for and an active area of research in the field of mobile robotics for many years. Using the extended Kalman filter (EKF) is widely considered the *de facto* standard model in the theory for navigation systems. The EKF-SLAM algorithm employed in this work recursively estimates the pose of the mobile robot, the static 2D positions of the radio sensors placed in the operating environment, as well as the mobile target modeled by an additional radio sensor. The robot model is subject to process noise ζ_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.*, $\zeta_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$. Taking into consideration the robot's process noise, the discrete-time model can be rewritten as

$$\mathbf{q}_{k+1,r} = \mathbf{f}_r(\mathbf{q}_{k,r}, \mathbf{u}_k, \zeta_k) \tag{2.1}$$

Assume the initial prediction of the robots pose is different from the reference. The initial state error covariance matrix is given by $\mathbf{P}_{0,rr} = diag((\mathbf{q}_{0,r} - \hat{q}_{0,r}) \cdot (\mathbf{q}_{0,r} - \hat{q}_{0,r}))$. The EKF-SLAM algorithm performs two standard steps for solving the the localization and mapping problem; predict and update. Suppose that s, $0 \leq s \leq s'$, radio sensors are mapped in the robot's workspace. These radio sensors are denoted in the set $\beta = \{b^{[1]}, \cdots, b^{[s]}\}$. 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.

$$\mathbf{q}_k = [\mathbf{q}_{k,r}, \mathbf{b}^{[1]}, \cdots, \mathbf{b}^{[s]}]^T$$

and

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{k,rr}\mathbf{P}_{k,r\beta} \\ \mathbf{P}_{k,\beta r}\mathbf{P}_{k,\beta\beta} \end{bmatrix}$$

where $\mathbf{P}_{k,\beta r} \in \mathbb{R}^{2s\times 3}$ is the covariance matrix that maps the state of the robot to the state of the radio sensors and $\mathbf{P}_{k,r\beta} \in \mathbb{R}^{3\times 2s}$ is the covariance matrix that maps the state of the radio sensors to the state of the robot. $\mathbf{P}_{k,\beta\beta} \in \mathbb{R}^{2s\times 2s}$ is the covariance matrix corresponding to the states of the radio sensors that are mapped in the robot's environment. Let α^- and α^+ denote the prediction and the update estimates of the quantity α. Prediction estimates of the robot's pose and the map before measurements are gathered are modeled with the following equations.

$$\hat{\mathbf{q}}^-_{k+1} = \hat{\mathbf{q}}^+_k + \mathbf{C}^T\mathbf{f}_r(\hat{\mathbf{q}}^+_{k,r}, \mathbf{u}_k, \mathbf{0}), \tag{2.2a}$$

$$\mathbf{P}^-_{k+1,rr} = \mathbf{F}_k\mathbf{P}^+_{k,rr}\mathbf{F}^T_k + \mathbf{L}_k\mathbf{Q}_k\mathbf{L}^T_k, \tag{2.2b}$$

$$\mathbf{P}^-_{k+1,r\mathcal{B}} = \mathbf{F}_k\mathbf{P}^+_{k,r\mathcal{B}}, \tag{2.2c}$$

$$\mathbf{P}^-_{k+1,\mathcal{B}r} = \left(\mathbf{P}^-_{k+1,r\mathcal{B}}\right)^T \tag{2.2d}$$

where

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \end{bmatrix}, \mathbf{F_k} = \frac{\partial \mathbf{f_r}}{\partial \mathbf{q_{k,r}}}|_{(\hat{\mathbf{q}}^+_{\mathbf{k,r}}, \mathbf{0})}, \mathbf{L_k} = \frac{\partial \mathbf{f_r}}{\partial \zeta_\mathbf{k}}|_{(\hat{\mathbf{q}}^+_{\mathbf{k,r}}, \mathbf{0})}$$

The measurement model at discrete time index $k+1$ is written as

$$\mathbf{y}^{[\mathbf{j}]}_{\mathbf{k+1}} = \mathbf{h}(\mathbf{q_{k+1,r}}, \mathbf{b}^{[\mathbf{j}]}, \zeta_{\mathbf{k+1}}), \tag{2.3}$$

with $\zeta_{\mathbf{k+1}} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, where $\mathbf{R} = \mathbf{diag}(\sigma^\mathbf{2}_\mathbf{d}, \sigma^\mathbf{2}_\mathbf{b})$ is the measurement noise covariance matrix where σ_d and σ_b are the standard deviations of the noise associated with the range and

bearing measurements respectively. The update estimate of the robot's pose and the map after taking measurements are modeled with the following equations.

$$\mathbf{S} = \begin{bmatrix}\mathbf{H}_r & \mathbf{H}_{\mathbf{b}^j}\end{bmatrix}\begin{bmatrix}\mathbf{P}^-_{k+1,rr} & \mathbf{P}^-_{k+1,r\mathbf{b}^j}\\ \mathbf{P}^-_{k+1,\mathbf{b}^j r} & \mathbf{P}^-_{k+1,\mathbf{b}^j\mathbf{b}^j}\end{bmatrix}\begin{bmatrix}\mathbf{H}_r^T\\ \mathbf{H}_{\mathbf{b}^j}^T\end{bmatrix} + \mathbf{R}, \tag{2.4a}$$

$$\mathbf{K}_{k+1} = \begin{bmatrix}\mathbf{P}^-_{k+1,rr} & \mathbf{P}^-_{k+1,r\mathbf{b}^j}\\ \mathbf{P}^-_{k+1,\mathcal{B}r} & \mathbf{P}^-_{k+1,\mathcal{B}\mathbf{b}^j}\end{bmatrix}\begin{bmatrix}\mathbf{H}_r^T\\ \mathbf{H}_{\mathbf{b}^j}^T\end{bmatrix}\mathbf{S}^{-1}, \tag{2.4b}$$

$$\hat{\mathbf{q}}^+_{k+1} = \hat{\mathbf{q}}^-_{k+1} + \mathbf{K}_{k+1}\mathbf{v}, \tag{2.4c}$$

$$\mathbf{P}^+_{k+1} = \mathbf{P}^-_{k+1} - \mathbf{K}_{k+1}\mathbf{S}\mathbf{K}^T_{k+1}, \tag{2.4d}$$

where the jacobians of the measurement model are

$$\mathbf{H_r} = \frac{\partial \mathbf{h}}{\partial \mathbf{q_{k+1,r}}}|_{(\hat{\mathbf{q}}^-_{\mathbf{k+1,r}},\mathbf{0})}, \mathbf{H_{b^j}} = \frac{\partial \mathbf{h}}{\partial \mathbf{b^{[j]}}}|_{(\hat{\mathbf{q}}^-_{\mathbf{k+1,r}},\mathbf{0})}$$

and

$$\mathbf{v} = \mathbf{y}^{\mathbf{[j]}}_{\mathbf{k+1}} - \mathbf{h}(\mathbf{q}^-_{\mathbf{k+1,r}}, \mathbf{b}^{\mathbf{[j]}}, \mathbf{0})$$

The EKF prediction model (2.2) and the update model (2.4) are employed recursively to estimate the robot's pose $\hat{\mathbf{q}}_{k,r}$ and the position of the radio sensors $\mathbf{b}^{\mathbf{[j]}}$ that are mapped in the robot's environment.
Suppose that a new measurement $\mathbf{y}^{\mathbf{[s+1]}}$ at discrete time index $k+1$ is received by the coordinator radio sensor on-board the mobile robot. The 2D position of the radio sensor that corresponds to this measurement must be updated on the map for the robot. An inverse measurement model is given by

$$\mathbf{b}^{\mathbf{[s+1]}} = \mathbf{g}(\mathbf{q}^+_{\mathbf{k+1,r}}, \mathbf{y}^{\mathbf{[s+1]}}) \tag{2.5}$$

$$= \begin{bmatrix}\hat{x}^+_{k+1} + (r^{[s+1]} + \xi_d)\cos(\hat{\theta}^+_{k+1} + \beta^{[s+1]})\\ \hat{y}^+_{k+1} + (r^{[s+1]} + \xi_b)\sin(\hat{\theta}^+_{k+1} + \beta^{[s+1]})\end{bmatrix} \tag{2.6}$$

The noise $\xi = [\xi_\mathbf{d}, \xi_\mathbf{b}]^\mathbf{T}$ 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 radio sensor associated with the new measurement $\mathbf{y}^{\mathbf{[s+1]}}$, let us define the jacobians $\mathbf{G_r} \in \mathbb{R}^{\mathbf{2\times 3}}$ and $\mathbf{G}_{\mathbf{y}^{[s+1]}} \in \mathbb{R}^{\mathbf{2\times 2}}$ based on this measurement with

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

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

$$\mathbf{P}_{\mathbf{b}^{(s+1)}\mathbf{b}^{(s+1)}} = \mathbf{G_r}\mathbf{P}^+_{\mathbf{k+1,rr}}\mathbf{G}^\mathbf{T}_\mathbf{r} + \mathbf{G_y}\mathbf{R}\mathbf{G}^\mathbf{T}_\mathbf{y}, \tag{2.7}$$

$$\mathbf{P}_{\mathbf{b}^{(s+1)}\mathbf{q}} = \mathbf{G_r}\mathbf{P}_{\mathbf{k+1,rq}} = \mathbf{G_r}\begin{bmatrix}\mathbf{P}^+_{\mathbf{k+1,rr}} & \mathbf{P}^+_{\mathbf{k+1,r}\beta}\end{bmatrix} \tag{2.8}$$

The augmented state and state covariance matricies are then updated yielding the final state for $\mathbf{q}$ and $\mathbf{P}$ before repeating the algorithm. These states are shown below

$$\mathbf{q_{k+1}} = \begin{bmatrix}\mathbf{q}^+_{\mathbf{k+1}}\\ \mathbf{b}^{\mathbf{s+1}}\end{bmatrix}$$

$$\mathbf{P_{k+1}} = \begin{bmatrix} \mathbf{P^{+}_{k+1}} & \mathbf{P^{T}_{b^{(s+1)}q}} \\ \mathbf{P_{b^{(s+1)}q}} & \mathbf{P_{b^{(s+1)}b^{(s+1)}}} \end{bmatrix}$$

2.2 Measurement Model

The radio sensors placed in the robot's operating environment are unknown to it and must be estimated by the robot in addition to its own pose. Each radio sensor has its own unique ID that it sends out to be received by the base radio sensor on the mobile robot. Along with the ID, the base sensor measures the received signal strength indication, RSSI, from each of the radio sensors in the workspace. Based on the model found in [[2]], the RSSI measurement $z_k^{[j]}$ of the j^{th} radio sensor is modeled by

$$z_k^{[j]} \approx P_{ref} - 10\eta \log_{10} r_k^{[j]}, \tag{2.9}$$

where $r_k^{[j]} = \sqrt{(x_k - x_k^{[j]})^2 + (y_k - y_k^{[j]})^2}$ is the ideal line of sight distance between the robot and the j^{th} radio sensor, P_{ref} is the power level at a reference distance of 1m, and η is the signal propagation constant. For our work, $\eta = 2$ and P_{ref} was found to be different for each radio sensor. The values we measured for our work are found in the table below.

P_{ref1}	P_{ref2}	P_{ref3}	P_{ref4}	P_{ref5}
-43	-34	-34	-39	-36

The bearing information for each radio sensor was determined from the RSSI measurements gathered. The bearing β for each respective radio sensor is considered to be the point where the RSSI is strongest. Figure 2.2 shows the model for the radio sensor network in the robot's environment.

2.3 Robot Model

The model for the mobile robot is an Ackermann steering vehicle. The discrete time kinematic equations are as follows

$$\begin{aligned} x_{k+1} &= x_k + T\nu_k \cos(\theta_k + \gamma_k), \\ y_{k+1} &= y_k + T\nu_k \sin(\theta_k + \gamma_k), \\ \theta_{k+1} &= \theta_k + T\nu_k \frac{\sin(\gamma_k)}{l} \end{aligned}$$

2.4 Motion Control Model

A PI controller was initially proposed to be used as the motion controller for our mobile robot, but during our actual implementation we found that using a constant velocity was easier to work with while dealing with errors.

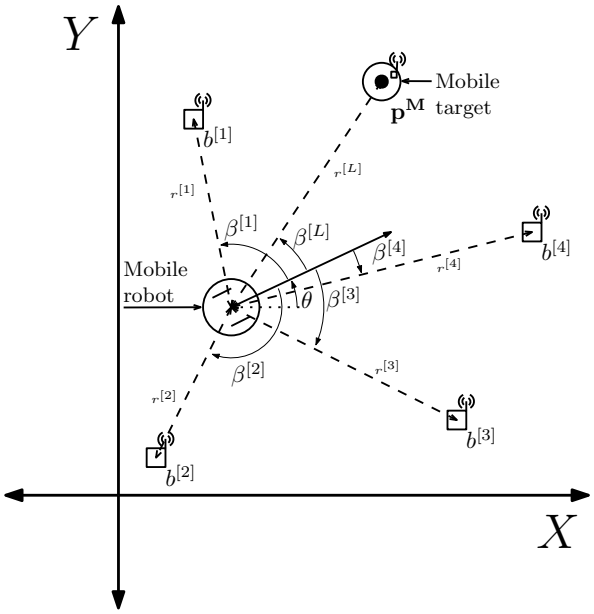


Figure 2.2: Radio sensor network for mobile target tracking localization and mapping.

2.5 Subsystem Diagram

A more detailed subsystem block diagram is illustrated in Figure 2.3.

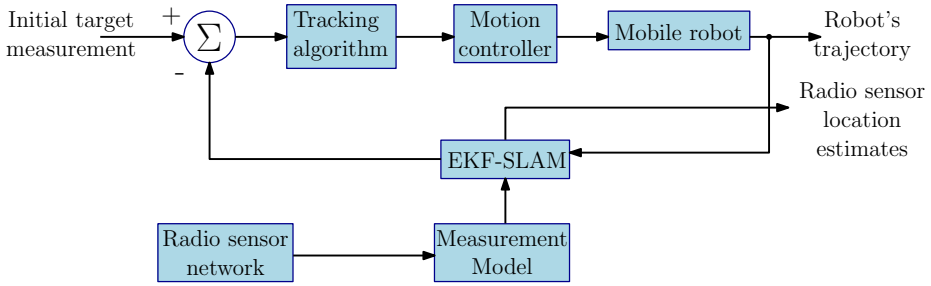
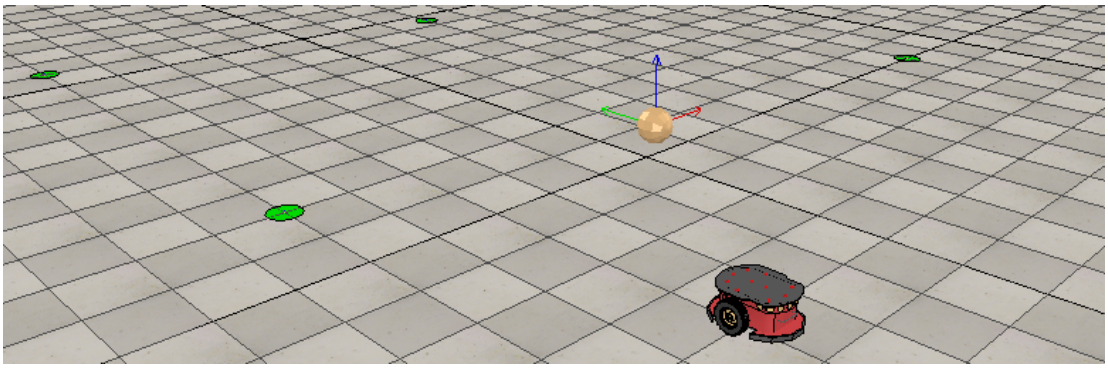


Figure 2.3: Subsystem block diagram

Chapter 3

Simulation



3.1 Virtual Robot Experimentation Platform

For the simulation of our algorithm, we chose to use the free, multi-platform robot simulator V-REP[1]. The V-REP robot simulator follows industry standard robotic platforms so the results are consistent in a way that is accurate to their real-world counterparts. This simulator is popular within the robotics community due its integrated development environment, large number of robot models available, the ability to write in a variety of programming languages including C/C++ and Matlab, its compatibility with ROS, and overall ease of use.

3.2 V-REP Setup

In order to simulate the EKF-SLAM and Mobile Target Tracking algorithms, a scene was first created in V-REP. Figure 3.1 illustrates the virtual workspace. The Pioneer 3-DX model was used to test our algorithm because that is the same mobile robot we had access to for testing.

[1] http://www.coppeliarobotics.com/

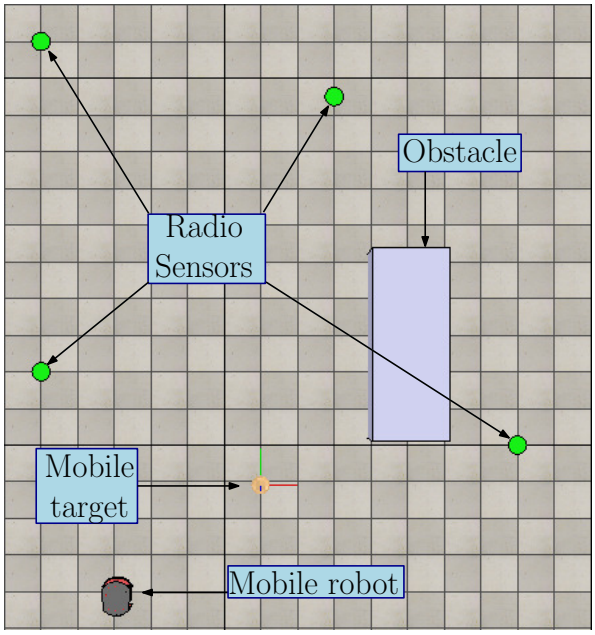


Figure 3.1: Virtual environment for experiments in V-REP

3.3 Ideal Mobile Target Case

For our first simulation case, the mobile robot was tasked with tracking the mobile target while also avoiding obstacles in its path. To ensure that our method for tracking the mobile target was working, observations for the mobile target were done continuously where as the static radio sensors observations were only gathered every 30 iterations like in our physical implementation. As seen in Figure 3.2, the radio sensor localiation error reduces very quickly. They are localized within 20-30cm of their true locations. Figure 3.3 shows the mobile target estimation error. As expected with observations being gathered continuously, the estimation error is very low. The mobile target is estimated within 20cm of its true location over time.

Running these tests were crucial in determining if our mobile target tracking was possible in theory, but in practice we were unable to get observations continuously because the robot must stop and scan each time it observes. Doing this would result in the robot almost never moving and the mobile target having moved very far away. Because we are unable to predict the movement of the mobile target, we are limited to gathering information about its location through these observations. We decided to run experiments where the mobile target was only estimated the same way the other radio sensors were updated. These results are discussed in the next section.

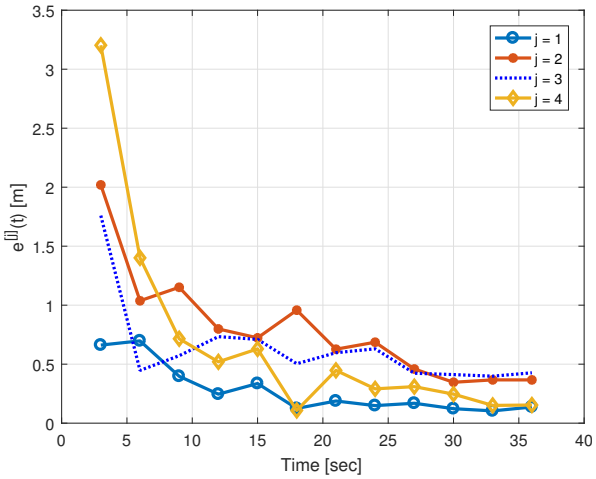


Figure 3.2: Radio sensor estimation error

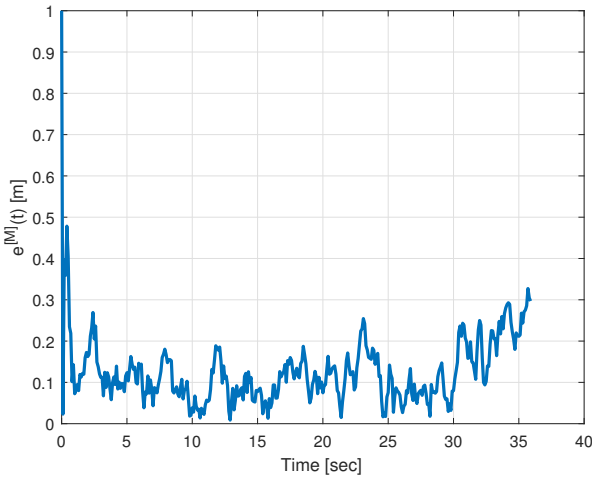


Figure 3.3: Mobile target estimation error with continuous update model

3.4 Restrictive Mobile Target Case

As described in the previous section, the position of the mobile target is difficult to estimate using our model. For our restrictive mobile target case, the mobile robot remains static to gather the best results for our mobile robot. This defeats the point of the mobile robot chasing the target, but using the model available to us, the estimates for the mobile target get worse as the robot also moves. Figure 3.4 shows the radio sensor estimation error. Same as the previous case, the estimation error for the radio sensors is within 20-30cm of their true location over 1000 iterations. Figure 3.5 shows the mobile target estimation error over 1000 iterations. The mobile target was localized around 35cm. The mobile target was moving on a random path at a very slow speed.

For our implementation, the mobile target is set a point and the mobile robot scans to localize on it, then moves to a safe distance away from the target, then we move the target to the new location and the process is repeated.

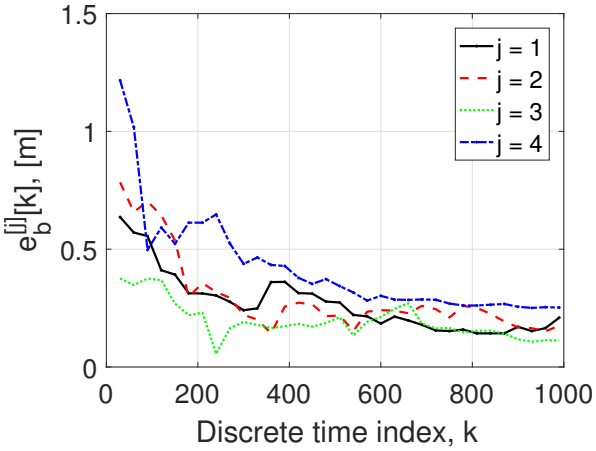


Figure 3.4: Radio sensor estimation error

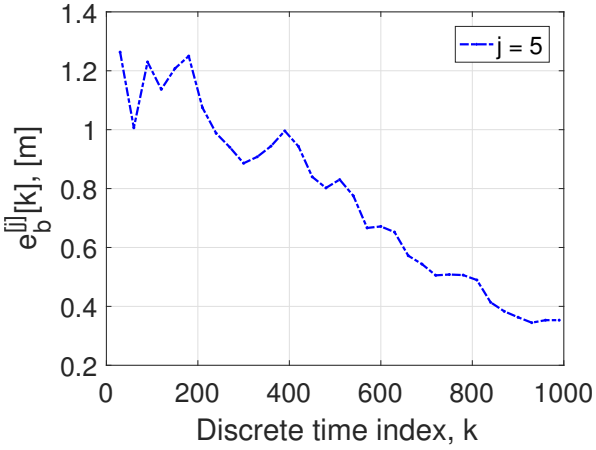


Figure 3.5: Mobile target estimation error in limited update model

Chapter 4

Implementation

4.1 Hardware

To implement the algorithm described and simulated in the previous sections, customized and off the shelf hardware was required. This includes a mobile robotics platform, a radio sensor network, and a customized radio transceiver. All of the hardware used in this work was already available to the Bradley ECE department.

4.1.1 Pioneer P3-DX

The mobile robotics platform chosen for this work was a Pioneer P3-DX show in fig 4.1. The P3-DX is purpose built for research so it is an excellent choice for conducting mobile robotics experimentation. The P3-DX is a differential drive robot equipped with 500 tick wheel encoders for a high degree of odometric accuracy. It is powered by three 12V lead-acid batteries each rated for 7.2 Ah allowing the P3-DX to power itself and the customized radio transceiver for many experiments between charging cycles. The P3-DX can also easily interface with the rest of our system using its serial communication port and ARIA programming framework.

4.1.2 Xbee Radio Sensor Network

For our radio sensor network and mobile target, we chose to use Xbee S2C radios manufactured by Digi show in fig 4.2. These radios have many advantages that make them excellent for research applications such as this. Each Xbee module is low cost, currently $17.50 a unit. Xbees are easily programmable using the XCTU application free for download off of the Digi website. Xbee radios can be powered off of 3V and low current between 31-45 mA. The radios communicate using the Zigbee protocol on a 2.4 Ghz ISM band. The Zigbee protocol allows the Xbees to be configured in a star network topology with one Xbee serving as a coordinator node that sends and receives packets from other the other radios in the network. A diagram of this is shown in fig 4.3 The coordinator node

Figure 4.1: The Pioneer P3-DX.

will be mounted on the customized radio transceiver. These radios can be interfaced with UART serial communication which was used by the customized radio transceiver to control and receive data from the coordinator of the Xbee network. Each Xbee in the network besides the coordinator will be programmed in AT mode sending a packet containing only its unique string ID. The coordinator will receive this ID packet along with RSSI for each radio.

4.1.3 Customized Radio Transceiver

A unique solution was needed to interface the mobile robot with the radio sensor network and a laptop running MATLAB. This took the form of the customized radio transceiver or more simply CRT show in fig 4.4 The main tasks of the CRT are gathering the RSSI readings from the radio sensor network, and controlling the P3-DX. The main components of the CRT include a Beaglebone Black Wireless microcomputer, a motor controller, a stepper motor, and an Xbee radio mounted on a parabolic dish. The Beaglebone serves as the main coordinator of communication between the other hardware using a variety of different protocols. This is shown in detail in fig 4.5. The motor is used to rotate the Xbee mounted on the parabolic dish while it is receiving RSSI readings. The bearing of the stepper motor is controlled by a motor controller that is in turn controlled by the GPIO pins of the Beaglebone Black. The bearing of the motor is used to determine the angle of the radio sensor later in the software. The mounted Xbee is the coordinator of the Xbee star network and will send pings to all of the other Xbee's in the network and then receive the responses in the form of the other radio's string IDs and RSSI. These responses are then sent to the Beaglebone Black over UART.

Figure 4.2: An Xbee Radio.

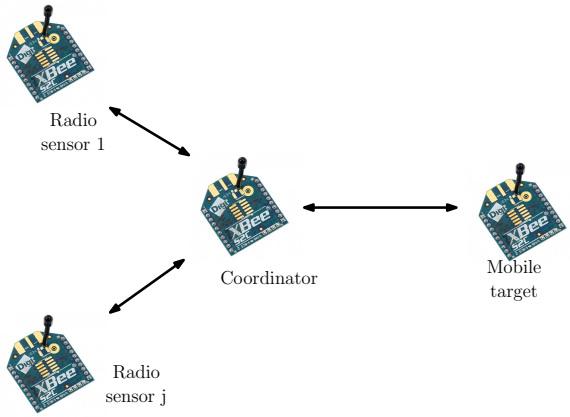


Figure 4.3: The Xbee Network Topology.

4.2 Software

The following section details the software packages and code used to implement the SLAM with mobile target tracking algorithm on the custom and off the shelf hardware.

4.2.1 Ubuntu

The Beaglebone Black micro-controller needed an operating system that was easily modifiable and compatible with the other pieces of software that were used. A special version of Ubuntu Linux compiled for the Beaglebone Black was acquired from https://elinux.org/BeagleBoardUbuntu#Ubuntu_.2816.04.4.29. Ubuntu was a perfect fit

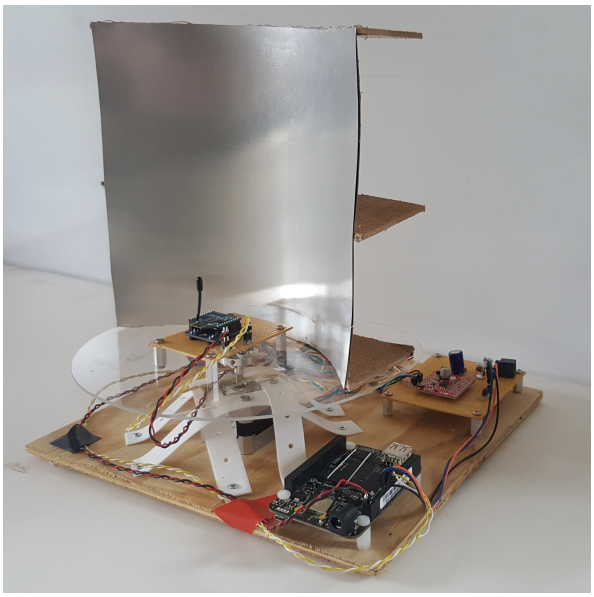


Figure 4.4: The Customized Radio Transceiver

for an operating system, it was free, allowed for access of low level GPIO on the Beaglebone, and is compatible with Robotics Operating System.

4.2.2 ROS

To facilitate communication between the different software processes and provide a layer of hardware abstraction Robotics Operating System (ROS) was leveraged. ROS is a free collection of robotics middle-ware. The version used in this work was Kinetic Kame obtained from http://www.ros.org/. We used many of ROS's features including the TCP/IP communication stack to send messages between our process nodes and the CMAKE tools to compile our C++ code. ROS also has a library for interfacing with the ARIA platform the Pioneer P3-DX uses. This implementation uses three ROS nodes. The master node on the Beaglebone Black is written in C++, a second node on the Beaglebone runs the ROSARIA framework to interface with the P3-DX, and a third node runs remotely on a laptop using MATLAB.

4.2.3 C++ Node

C++ was chosen to write the code for the ROS node that controls both the motor and coordinator Xbee to gather the radio measurements. This node also publishes the RSSI and bearing measurements to the remote node running MATLAB. ROS native supports C++ compilation using the included CMAKE build tools and can interface with the ROS API by using callback functions. This node also served as the ROS Master node and handled the core ROS services. A flowchart of the logic used in this node is shown in fig 4.6

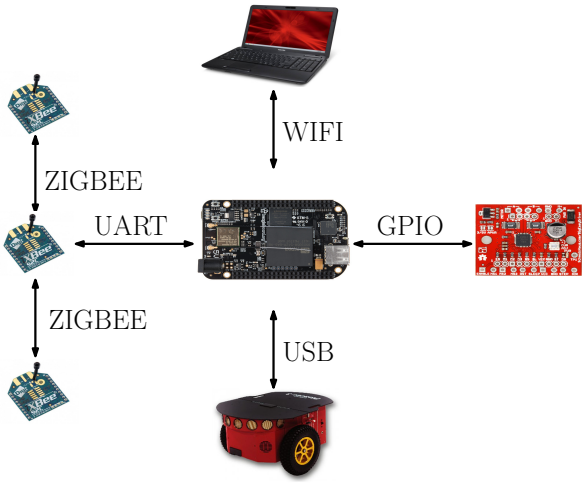


Figure 4.5: Communication Diagram

4.2.4 MATLAB Node

MATLAB running on a windows laptop was used to perform the EKF-SLAM algorithm on a remote node. This was done due to the ease of coding a complex algorithm like an extended Kalman in MATLAB compared to implementing it in pure C++ code. MATLAB is very easy to interface with ROS using the Robotics Systems Toolbox. This node also publishes the linear and angular velocity commands for the ROSARIA node to control the P3-DX. MATLAB was also used to record and plot data. A flowchart of this nodes logic can be see in fig 4.7

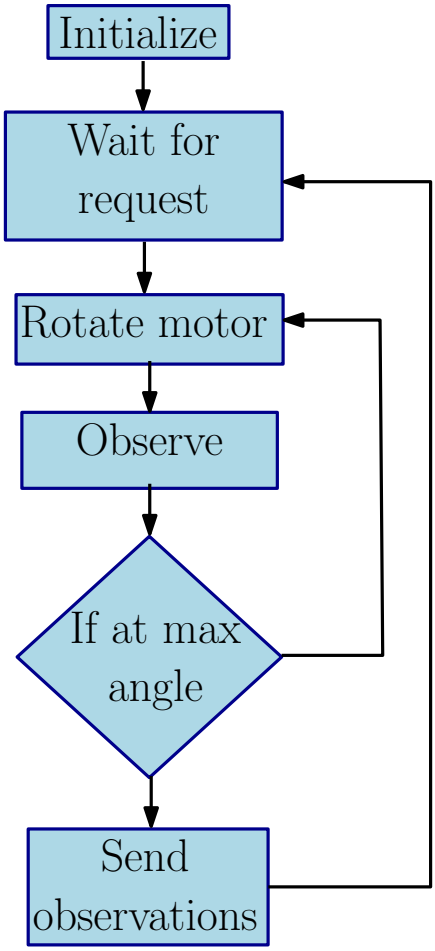


Figure 4.6: Flowchart of C++ Node.

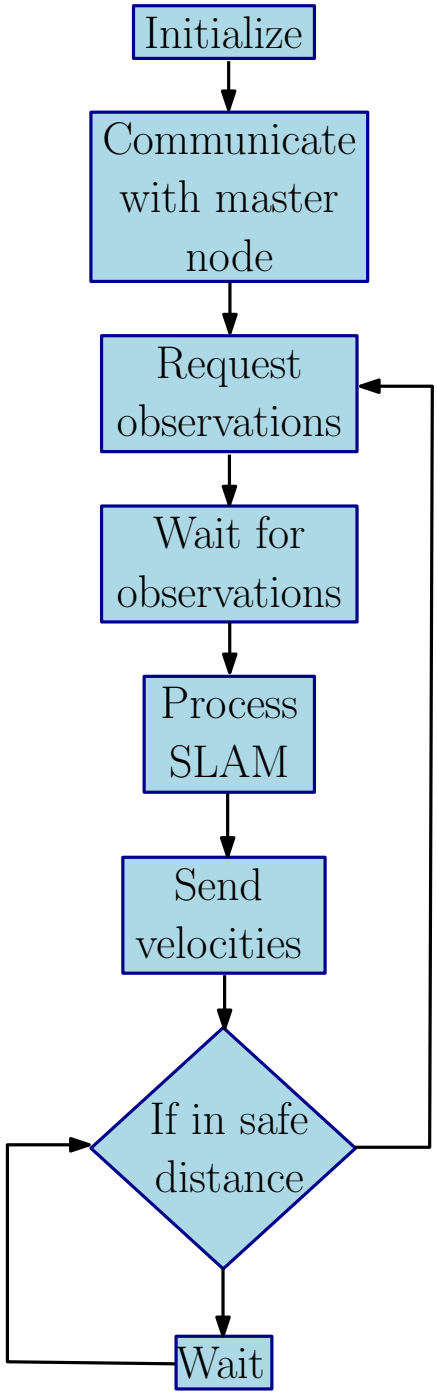


Figure 4.7: Flowchart of MATLAB node.

Chapter 5

Conclusion

This work of EKF-SLAM modified to track a mobile target produced satisfactory results. In simulation the mobile robot as able to follow the mobile target with a great degree of accuracy while simultaneously avoiding obstacles in its path. The physical implementation produced satisfactory results but encountered some issues with multipath noise. If an environment free of radio reflective surfaces was available the system should have had performance close to that of the simulation. This noise would cause some sensors measurements to be read with incorrect bearings and distances, and would often effect the position of the mobile target and the mobile robots estimate of its own pose. If this work is to be continued mitigating multipath noise should be a priority.

5.1 Experimental Results

The following are some results from one of the experiments run on this platform. The multipath issue is clearly visible as some beacons are localized closely and others have a more significant error. Shown in fig 5.1 is the estimated position error of the beacons over time and show in fig 5.2. A video of this experiment is uploaded on the website for this project.

5.2 Future Work

There are several ways that this project may be improved upon or adapted in the future.

- Perhaps there is some solution to the multi-path signal problem that can be implemented in the form of either a new antenna/beacon design or software algorithm.
- New sensor types such as LIDAR might be included to increase accuracy and speed of the mobile target tracking.
- Multiple Robots could used to cooperatively track the mobile target and improve accuracy.

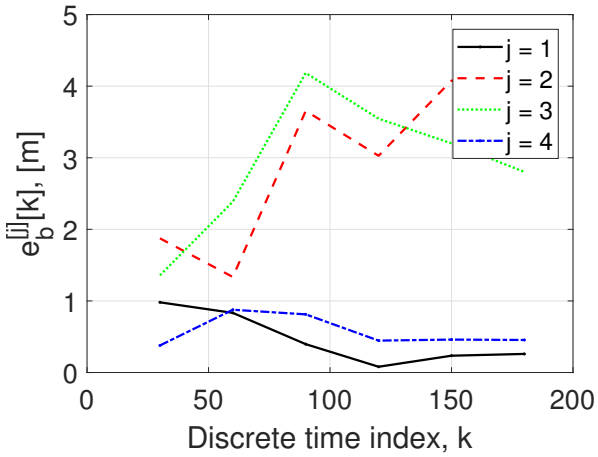


Figure 5.1: Beacon error over time.

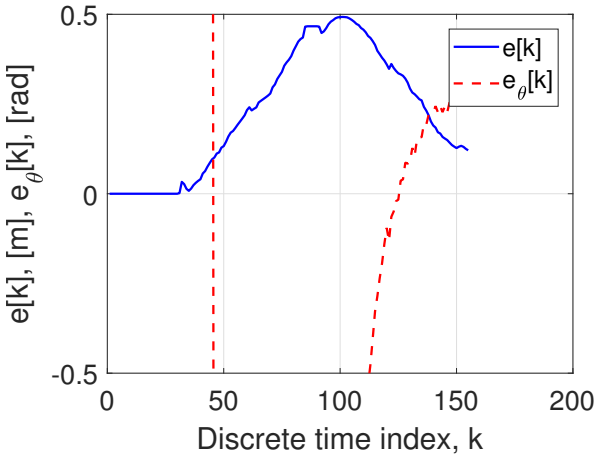


Figure 5.2: Robot estimation error over time.

APPENDICES

Appendix A

Setup for Running Simulations

This appendix gives the set of steps to simulate using V-REP.

A.1 V-REP Setup

The simulations completed were done using V-REP and the remote API for Matlab. Besides the mobile robot itself, the objects used in V-REP were all 'dummies' that were renamed as their respective radio sensor the mobile target. A solid cube was used during the obstacle avoidance testing during the ideal simulation. Besides the mobile target and the mobile robot, all the objects were static in location.

A.2 Running Simulation

Once the V-REP scene is ready, press the play button on the top toolbar to start V-REP. After this, press run on the Matlab script and the simulation will now run.

Appendix B

Setup for Running Experiments

The following section is for setting up the hardware implementation of this project

B.1 BeagleBone Black Wireless

Install ubuntu linux from http://rcn-ee.net/rootfs/2017-01-23/microsd/bone-ubuntu-16.04.1-console-armhf-2017-01-23-2gb.img.xz. to an SD card using a disc imaging program. Connect the BeagleBone Black to the internet. Install Kinetic Kame ROS from http://www.ros.org/ using the instructions provided. Copy the seniorProject directory from our github https://github.com/ghovey/SLAMDAR Install the ROSARIA packag using the provided instructions from https://github.com/amor-ros-pkg/rosaria Set the ROS environmental variables

`ROS_MASTER_URI`

and

`ROS_IP`

to match those of the wireless connection on the Beaglebone Black.

B.2 Xbee Radio Network

Configure your XBee radios to run in AT mode using the same PAN ID on every unit in the network including the coordinator. Give each radio a sequential node identifier such as beaconN where N is a number starting at 1, the last beacon in the sequence will be designated as the mobile target. Configure the XBee on the CRT to be the coordinator node.

B.3 Running Experiment

Download the MATLAB interface from our github https://github.com/ghovey/SLAMDAR Edit the Use your favorite console program to SSH into the Beaglebone Black at whatever its wireless IP address is. To start the master node type the following commands

```
sudo su
echo BB-UART2 >/sys/devices/platform/bone_capemgr/slots
cd seniorProject/catkin_ws
. devel/setup.bash
roslaunch ekf_slam node.launch
```

Then open another putty window to the Beaglebone Black. To start the ROSARIA node enter th following commands

```
cd seniorProject/catkin_ws
. devel/setup.bash
rosrun rosaria RosAria
```

Then hit run on the MATLAB script targettracking.m

Bibliography

[1] M. S. Miah, J. Knoll, and K. Hevrdejs, "Intelligent range-only mapping and navigation for mobile robots," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1164–1174, March 2018.

[2] Q. Dong and W. Dargie, "Evaluation of the reliability of rssi for indoor localization," in *2012 International Conference on Wireless Communications in Underground and Confined Areas*, Aug 2012, pp. 1–6.