Multirobot Localization Using Heuristically Tuned Extended Kalman Filter

by

Ruslan Masinjila

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

A mobile robot needs to know its pose (position and orientation) in order to navigate and perform useful tasks. The problem of determining this pose with respect to a global or local frame is called localization, and is a key component in providing autonomy to mobile robots. Thus, localization answers the question *Where am I?* from the robot's perspective.

Localization involving a single robot is a widely explored and documented problem in mobile robotics. The basic idea behind most documented localization techniques involves the optimum combination of noisy and uncertain information that comes from various robot's sensors. However, many complex robotic applications require multiple robots to work together and share information among themselves in order to successfully and efficiently accomplish certain tasks. This leads to research in collaborative localization involving multiple robots. Several studies have shown that when multiple robots collaborativelly localize themselves, the resulting accuracy in their estimated positions and orientations outperforms that of a single robot, especially in scenarios where robots do not have access to information about their surrounding environment.

This thesis presents the main theme of most of the existing collaborative, multi-robot localization solutions, and proposes an alternative or complementary solution to some of the existing challenges in multirobot localization. Specifically, in this thesis, a heuristically tuned Extended Kalman Filter is proposed to localize a group of mobile robots. Simulations show that when certain conditions are met, the proposed tuning method significantly improves the accuracy and reliability of poses estimated by the Extended Kalman Filter. Real world experiments performed on custom-made robotic platforms validate the simulation results.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my thesis supervisor, Pierre Payeur, Ph.D., for his patience, useful remarks and continuous guidance throughout my research and writing of this thesis. I would also like to thank my family and friends, for their invaluable support, advice and encouragement.

Dedication

To my family and friends.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of the field

Mobile Robotics is one of the fastest growing fields of engineering [1]. Mobile robots are commonly used to perform tasks in remote or areas hazardeous to humans [2, 3]. Other applications for mobile robots include military, exploration, and entertainment [4]. At any time, a mobile robot must know where it is in order to be able to safely move to another location. The movement can be achieved with the help of a human operator, who guides the robot remotely via a set of commands [5] or autonomously, in which case the robot guides itself using available information about the environment [6].

The process by which a mobile robot determines its pose is known as localization [7]. The most widely used method to determine position and orientation of the mobile robot is dead-reckoning [8], which is a form of *relative localization*. In this method, a mobile robot determines its current pose from its previous pose with the help of on-board motion sensors (proprioceptive sensors) and the robot's kinematics model. Examples of commonly used proprioceptive sensors include wheel encoders, gyroscopes and accelerometers. The disadvantage of dead-reckoning is the accumulation of errors over long distances, which can lead to poor localization performance. In another method, called *absolute localization*, a mobile robot extracts information about the environment using its exteroceptive sensors, compares the extracted information with known information about the environment (*a-priori*), and deduces its pose relative to some global frame of reference [9]. Examples of exteroceptive sensors include sonars, laser rangefinders, and radars.

Multirobot systems are increasingly being used to perform complex tasks. For example, Amazon uses thousands of autonomous Kiva robots in some of its warehouses which speed up the process of retrieving and shipping goods from the large warehouse to customers [10]. Without knowing where the robots are, and where they should go, the robots would collide with each other, resulting in damages to the robots themselves and the goods they carry, as well as delays. Therefore, accurate localization in each robot is extremely important in order to avoid such problems. In the last two decades, research in multi-robot

systems has gained a lot of attention [11]. The key reason is the increase in efficiency when multiple robots collaborate in order to accomplish a given task [11, 12]. As a result, a number of algorithms that support multi-robot localization have been developed. These algorithms estimate the pose of every robot in a team by combining motion measurements from proprioceptive sensors of individual robots, with robot-to-robot measurements from their exteroceptive sensors [13].

Multi-robot localization schemes are either centralized, decentralized, or multi-centralized [14]. In a centralized scheme, all computations are done in one processing center, which can be one of the robots (the leader). Although this scheme provides optimal pose estimation for all robots in a group, it is poorly scalable, computationally expensive, and is vulnerable to single-point failure (for example, if the leader malfunctions). In a multi-centralized scheme, each robot maintains information about its own pose, and that of every other robot in the group. In other words, every robot maintains a copy of the entire group's state. This scheme is robust to single-point failure but results in higher communication and computation costs. In a decentralized scheme, each robot maintains only the information about its own pose, and only updates this information locally when relative robot-to-robot measurements become available. This scheme can provide approximate solution to a multi-robot localization problem at reduced communication and computational costs but often leads to inconsistent (overly optimistic or pessimistic) estimates if the interdependencies between robots are neglected [15].

1.2 Problem Statement

In effective localization, whether in single or multi-robot systems, information from the proprioceptive sensors is generally combined (fused) with the information from exteroceptive sensors in such a way that more weight is given to more accurate sensors, and vice versa. Probabilistic algorithms have widely been used in robot localization with great success [7, 9]. In distributed systems, however, probabilistic algorithms such as the Extended Kalman Filter (EKF) have a tendency of becoming inconsistent if the interdependencies in the system are ignored. This thesis presents a consistent, decentralized, multirobot localization solution using a heuristically tuned Extended Kalman Filter which combines information from proprioceptive and exteroceptive sensors.

1.3 Objectives

The main objectives of this thesis are:

- Develop a method based on Extended Kalman Filter for consistent localization of a group of robots in GPS-denied, featureless environments.
- Evaluate the performance of the developed method through simulations.
- Test and validate the localization method experimentally in the real world using custom-made robotic platforms.

1.4 Thesis Organization

The previous sections of this chapter introduced the localization problem involving multiple robots, and outlined the objectives of this thesis. The rest of the chapters are organized as follows:

- Chapter 2 presents the relevant work in the field and previous approaches to multi-robot localization, their effectiveness, and limitations.
- In Chapter 3 a decentralized, multirobot localization solution based on a heuristically tuned Extended Kalman Filter is developed, and its performance evaluated through simulations. The results of the tuned filter are compared with that of the original filter.
- Chapter 4 describes the real-world experiments that validate the performance of the solution using custom-made robotic platforms.
- Chapter 5 concludes the thesis with a summary of the work, the contributions made, and possible directions of future research and improvements.

Chapter 2

Literature Review

It is critical that a mobile robot knows its pose (position and orientation) at all times relative to some chosen reference. Without knowledge of its pose, a robot may be involved in collisions, move to wrong places or perform tasks at wrong locations. Localization is one of the most fundamental components in providing a robot with autonomous capabilities [7, 16]. The localization problem can be classified into two main categories, namely *relative* and *absolute* [17]. Effective localization methods usually combine one or more methods from each of these categories. This chapter introduces the localization problem in mobile robots and describes the two main categories of localization in Sections 2.1 and 2.2. Section 2.3 presents the probabilistic framework which forms the basis for most existing state-of-the-art solutions to the localization problem. Section 2.4 describes the notion of multirobot localization and provides a summary, comparison and limitations of some of the existing methods used to localize teams of robots.

2.1 Relative Localization

In relative localization, often referred to as dead-reckoning, a mobile robot uses its on-board motion sensors to determine its new pose from its previous pose [18]. These sensors are known as *proprioceptive* sensors, because they measure the motion by monitoring changes happening within the robot, for example, rotations of the wheels. Since the next robot's pose is always derived from its current pose, the position and orientation errors in relative localization grow unbounded [9], and therefore this category of localization is unsuitable for long distances. Relative localization is further divided into two subgroups, namely *odometry* and *inertial navigation*.

2.1.1 Odometry

In this method, wheel encoders are mounted onto the drive motors (or wheels) in order to count the number of revolutions of each wheel. Using the shape and size of the robot (i.e., the robot's geometry), and the number of revolutions each wheel makes, it is simple to estimate the new pose of a mobile robot provided that its initial pose is known [9, 8].

Odometry is widely used in mobile robotics because the method is inexpensive, self-contained, and can be used in combination with other methods for more effective localization [19]. However, when used on its own, odometry performs very poorly over long distances due to various sources of errors. In particular, errors in orientation result in large deviations in the position [20]. These sources of errors are either systematic (deterministic), or non-systematic (random). Systematic errors are caused by imperfections in the mechanical design of the robot. Examples of these imperfections include unequal wheel diameters, misalignment of the wheels, or the resolution of the wheel encoders. On the other hand, non-systematic errors are caused by interaction of the robot with unpredicted features of the environment, for example, when the robot moves on uneven surface or when the wheels of the robot slip due to slippery surface, over acceleration or collisions with obstacles [8].

2.1.2 Inertial Navigation

This method uses gyroscopes to measure the rate of change of rotation, and accelerometers to measure the rate of change of velocity [21]. The orientation of a mobile robot can be determined by integrating measurements from the gyroscope once, whereas the position of the robot is found by integrating the measurements from the accelerometer twice. Like odometry, inertial navigation is self-contained and is independent of external information sources and is also susceptible to drift errors over long distances. However, gyroscopes in particular are more accurate in estimating orientation of the robot than odometers, and thus can be used to significantly reduce positional errors that arise due to errors in orientation. On the other hand, accelerometers are very sensitive to bumpy surfaces since they pick up components of the gravitational field, and may result in large positional errors due to the fact that the position of the robots is calculated by integrating acceleration twice [19]. The effect of gravitational interference in accelerometers can be reduced using a tilt sensor [22].

2.2 Absolute Localization

In absolute localization, a mobile robot uses its *exteroceptive* sensors (e.g., laser rangefinders, and sonars) to extract information from the environment through measurements. The information usually contains the locations of features of interest with respect to the position of the robot [21]. This information is then matched with known information about

the environment, called *priori*, in order to estimate the robot's pose. Unlike relative localization, absolute localization is independent of previous measurements and therefore the errors do not grow unbounded [9]. Absolute localization is further divided into two sub-groups, namely *landmark localization* and *map-based localization*.

2.2.1 Landmark Localization

Landmarks are distinct features in an environment that the robot's extreroceptive sensors can detect. Landmarks can be of geometric shapes, such as rectangles, cylinders, and circles, and may include some additional information in the form of color, or bar-codes [21]. In general, landmarks should have fixed and known positions in the environment. However, in dynamic environments, the location of landmarks may change with time, resulting in a more complicated localization problem for mobile robots [23]. Landmarks are either *artificial* or *natural*.

2.2.1.1 Artificial Landmarks

These are man-made landmarks that are placed at specific locations which are known in advance in order to facilitate absolute localization of a robot [24]. Three or more artificial landmarks can be used to reliably and accurately determine the absolute pose of a robot through triangulation[1], or trilateration[2]. The main disadvantage of artificial landmarks is that it requires careful engineering of the environment which may not always be possible in hazardous or hard to reach places [17]. Artificial landmarks are further classified as *active* (energy emitters), or *passive* (non-energy emitters).

Active Artificial Landmarks. These landmarks, also known as *beacons*, use energy to actively transmit their locations, which can be picked up by the receivers of a mobile robot. When these receivers detect three or more beacons, the robot's absolute pose can be determined through triangulation or trilateration. Examples of active artificial landmarks include cellular networks, television networks, and Global Navigation Satellite Systems such as GPS (Global Positioning System) [26]. The main disadvantages of active artificial landmarks are:

- Expensive to install and maintain.
- Consume a lot of energy in order to transmit signals over large distances.
- Signals from these landmarks can be distorted through refraction or reflection, thus resulting in localization errors [26, 25]

[1] Triangulation uses angles and distances between known, fixed, landmarks and the robot to determine the robot's pose [25]

[2] Trilateration uses only distances [19]

Passive Artificial Landmarks. These landmarks differ from active artificial landmarks in that they do not transmit their locations, and therefore they do not need to be powered. Instead, a mobile robot must locate and identify three or more of these landmarks in order to localize itself through triangulation or triliteration. The identification of landmarks usually involves extraction of certain features from the landmarks (e.g., shape, color, edges, corners) which are then matched to features from a known database. Feature extraction can be based on image processing [27], or proximity and range sensors [28].

2.2.1.2 Natural Landmarks

These landmarks are not specifically engineered for the purpose of localization, and because of this, a robot might need to perform several, time-consuming observations in order to determine its unique position. Therefore, when compared to artificial landmarks, the computational complexity for identification of natural landmarks is higher while the reliability is lower [29, 24], especially for outdoor natural landmarks which usually have arbitrary shapes. The low reliability of natural landmarks can be improved by increasing the number of natural landmarks, however, the computational complexity also increases. To overcome this problem, Thrun [24] proposed a neural network based solution which keeps a balance between computational complexity and the number of natural landmarks used for localization. The proposed solution also improves the autonomy of the robot, optimality and flexibility within the environment.

2.2.2 Map-based Localization

Map-based localization is a type of absolute localization in which a mobile robot creates a local map of its surroundings using data from its exteroceptive sensors, and compares it to a global map stored in memory [21]. A successful match allows the robot to determine its global position and orientation. This technique is also often used to update a global map in a dynamic environment, or generate a new global map from multiple, locally generated maps [25]. The most common types of maps are metric, and topological [30]. Metric maps contain the geometric properties of the environment, while topological maps represent the connectivity of different places in the environment.

A more complicated case of map-based localization happens when the robot does not have access to the global map. In this case, the robot must incrementally build the map of the environment and at the same time determine its position and orientation. In the literature, the resulting problem is known as SLAM (*Simultaneous Localization and Mapping*) [31, 32] or CML (*Concurrent Mapping and Localization*) [33, 34]. SLAM presents the most practical, yet most difficult problem in mobile robotics since it is unlikely that a robot exploring an uncharted environment has access to the map of the environment prior to exploration. For this reason, over the past two decades there has been extensive research into the problem of SLAM. At a theoretical and conceptual level, the problem of SLAM can now be considered solved [32], and has been successfully implemented indoors [35, 36,

37, 38], outdoors [39, 40], as well as in aerial [41, 42], and underwater [43] environments. Many of the existing mapping solutions are robust to noise and cope with a wide range of environments.

Despite the progress made over the years in map-based localization, there still exist a large number of open, challenging problems that could take decades to be solved [30]. In particular, most existing mapping methods assume that the world is static, and therefore do not perform well in dynamic environments. Furthermore, most existing mapping techniques assume that the environment contains enough geometric features, such as corners, edges, and straight lines in a hallway, in order to reliably determine the pose of a mobile robot [21]. For this reason, existing map-based localization techniques perform well in structured environments, such as those found indoors, but poorly in unstructured, or large, open areas commonly found outdoors, where useful physical features are either insufficient or lacking [30, 44] . Therefore, extending existing mapping techniques to such environments, or developing completely new ones, is an important research goal, and has prompted a number of researchers to explore multirobot systems as a plausible solution to the above problems.

2.3 Probabilistic Framework

The pose of a mobile robot can be estimated from its previous pose using proprioceptive sensors (Section 2.1) or through perception of its surrounding environment using exteroceptive sensors (Section 2.2), or both. However, due to noise associated with sensors, and other factors such as wheel slippage or imperfections in the mechanical design of the robot, it is practically impossible to determine the actual or "true" pose of the robot in the real world through measurements [23]. For this reason, the pose of a robot is often expressed probabilistically, in which case the pose of a robot is expressed in terms of a Probability Density Function (PDF), which considers not one, but all possible poses of the robot based on information available from both proprioceptive and exteroceptive sensors. The probability density function represents the robot's internal belief (*bel*) about its own state and that of the environment. The state includes variables such as position and orientation of the robot, as well as the locations of the landmarks in the environment [23]. In general, the belief (*bel*) is calculated by incorporating information, in the form of probabilities, from models which characterize robot's localization. These models are described below.

Action or Motion Model. This model represents the change in internal state of the robot (position, orientation, or velocity) as a result of actuation caused by control input, u_t. This change can be expressed probabilistically as follows [45, 46]:

$$P(x_t|u_t, x_{t-1}) \tag{2.1}$$

Equation (2.1) defines the probability of the robot's position to be at x_t at time step t given that a control input u_t was applied to the robot when its position at time step $t-1$ was x_{t-1}. In many cases, the motion model can be derived from the robot's kinematic model [45] as described in Section 3.1.1.

Measurement or Perception Model. This model represents the relation between observation measurements and the robot's pose. Suppose a robot makes an observation, s_t, when its pose is x_t, the perception model relates the observation and the robot's pose probabilistically as follows [45]:

$$P(s_t|x_t) \tag{2.2}$$

Unlike the motion model, the probability density of the perception model is often difficult to compute due to high dimensionality associated with raw sensor data. For example, if a camera with a resolution of 640 by 480 is one of the robot's sensors, then every image contains 307,200 pixels (in grayscale). Thus, it becomes computationally expensive to determine the probability of every image s_t at each possible location, x_t. In order to overcome the problem of high dimensionality, raw sensor data is usually projected to a lower dimensional *feature vector*, z_t. For example, in landmark localization, a feature vector usually consists of bearing, (ϕ), and relative distance, (ρ), of a landmark to a robot, instead of a stream of numbers that a sensor outputs. Formally, the feature vector in landmark localization is expressed as follows [23]:

$$z_t = [\rho_t, \phi_t] \tag{2.3}$$

The belief, *bel*, mentioned earlier is generally calculated using the *Bayes filter* algorithm under Markov assumptions [47, 23, 45, 46]. The assumption states that the knowledge about the future (x_{t+1}) is independent of the knowledge about the past (x_{t-1}), and vice-versa, provided that the knowledge about the present (x_t) is known. Under this assumption, the Bayes filter is a recursive algorithm which computes the robot's most recent belief, $bel(x_t)$, from its previous belief, $bel(x_{t-1})$, the most recent control input, u_t, which takes the robot from x_{t-1} to x_t, and the measurement vector, z_t, at x_t. Since robot localization can be considered as a two step process (action and perception) alternating between each other, it is customary to express the belief for each step separately [45]. The belief computed right after the action but before incorporating measurements is called *a-priori belief*, denoted by $\overline{bel}(x_t)$ or $bel^{-1}(x_t)$. It combines the previous belief, $bel(x_{t-1})$, with the probability of the control input, u_t, that changes the location of the robot from x_{t-1} to x_t. The process of calculating *a-priori* is called *prediction*. On the other hand, the belief computed after measurements are taken is called *a-posteriori belief*, denoted by $bel(x_t)$, and is found by incorporating the most recent measurements, z_t, to $\overline{bel}(x_t)$. This process is known as *correction* or *measurement update*. Thrun. *et al.* provide a detailed derivation of the general Bayesian probabilistic framework for mobile robot localization in [23], the shorthand of which is shown in Table 2.1

Table 2.1: The general algorithm for the Bayes Filter

Input: $bel(x_{t-1}), u_t, z_t$
Prediction: $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx$
Measurement Update: $bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t)$

Where:
u_t is the control input.
z_t is the measurement vector.
x_{t-1} is the previous pose of robot.
x_t is the current pose of the robot.
$bel(x_t)$ is the current belief.
$bel(x_{t-1})$ is the previous belief.
η is the normalization constant.

It is important to note that the general Bayesian filter in Table 2.1 is an abstract concept that provides a probabilistic framework for recursive state estimation [48]. In order to implement this filter, the belief, $bel(x_t)$, the motion model, $P(x_t|u_t, x_{t-1})$, and the perception model, $P(z_t|x_t)$, have to be specified. As a result, a number of algorithms based on the Bayesian filter have been developed and successfully applied in mobile robotics for localization and/or mapping [24, 34, 30, 48, 31, 32, 23]. The different types of implementations of the Bayesian filter form two main families, namely Gaussian and Nonparametric filters [23].

2.3.1 Gaussian Filters

Gaussian filters are among the earliest family of Bayesian filters. The basic idea behind Gaussian filters is the representation of the belief, $bel(x_t)$, by multivariate normal distribution characterized by two parameters, the *mean* (mu), and the *covariance* ($\sum$). Gaussian filters are unimodal, meaning that their belief has a single peak (or maximum). This makes Gaussian filters suitable for tracking problems because the belief is focused around the true state. On the other hand, Gaussian filters perform poorly in situations where multiple hypotheses exist, for example, in *global localization* when the initial location of a robot is unknown. Furthermore, Gaussian filters, as their name suggests, assume that the noise that affects the action and perception models is Gaussian [23]. Despite these shortcomings, Gaussian filters are still very popular and widely used in robotics. The most common Gaussian filters are the *Kalman Filters* (KF) and their extensions such as the *Extended Kalman Filter* (EKF) and the *Unscented Kalman Filter* (UKF) [23, 48].

2.3.1.1 Kalman Filters

The Kalman Filter, invented by Rudolph Emil Kalman in the 1950s [23], is an optimal state estimator which assumes that the system state, the action and perception (measurement) models are linearly dependent in the presence of Gaussian noise [48] as shown in equations 2.4 and 2.5.

$$x_t = Ax_{t-1} + Bu_t + \epsilon_a \tag{2.4}$$

$$z_t = Hx_t + \epsilon_m \tag{2.5}$$

where:
$\mathbf{x_t}$ is the system state at time t.
$\mathbf{A}$ is the state transition model.
$\mathbf{x_{(t-1)}}$ is the system state at time $(t-1)$.
$\mathbf{B}$ is the control-input model.
$\mathbf{H}$ is the observation model.
$\mathbf{u_t}$ is the control input.
$\epsilon_\mathbf{a}$ and $\epsilon_\mathbf{m}$ represent zero-mean, independent, white Gaussian noise.

In practice, most systems are not linear. For this reason, the nonlinear versions of Kalman Filter, such as Extended Kalman Filters (EKF), and Unscented Kalman Filters (UKF) are used instead [49, 50]. EKF is a unimodal, Gaussian filter which approximates a nonlinear system through linearization using first order Taylor expansion. This approximation can introduce large errors in the estimated state of the system if the system is highly nonlinear, which in turn results in sub-optimal performance or even large deviations [49]. The limitation of the EKF can be overcome using the Unscented Kalman Filter (UKF) which is capable of estimating the state of system of any nonlinearity to the third order Taylor expansion.

The Kalman filter and its extensions have been widely and successfully applied to various localization and tracking problems due to their simplicity, robustness and tractability despite their restrictive assumptions [51, 52, 23, 53, 54, 30]. For these reasons, this thesis presents the Extended Kalman Filter as the main algorithm for distributed multirobot localization. Its implementation is discussed in detail in Chapter 3.

2.3.2 Nonparametric Filters

Unlike Gaussian filters, which assume that the system's state and dynamics are unimodal, Gaussian distributions, Nonparametric filters impose no such assumptions [23, 30]. Instead, they represent the belief *bel* arbitrarily (non-Gaussian) by a finite number of weighted samples. This property makes Nonparametric filters capable of dealing with situations when the initial location of a robot is unknown (i.e., global localization) [23, 48]. The most common Nonparametric filter is the Particle Filter (PF), which has received a lot of attention recently due to its ease of implementation and the ability to efficiently deal with non-Gaussian, multimodal distributions.

While Particle Filters (Nonparametric) are more versatile than the Kalman Filter and its extensions (Gaussian), the versatility comes at an increased computational cost. In general, the computational cost of Particle Filters increases exponentially with the dimension of the system state (x_t), whereas the cost of Kalman filters is polynomial with the dimension of the system state [48]. Thus, the choice between Particle and Kalman filters is usually governed by the desired versatility and computational cost.

2.4 Related work in Multirobot Localization

Research in multirobot systems began in the late 1980's [55, 56]. Prior to that, research in robotics mostly focused on single robot systems. Formally, a multirobot system is defined as a collection of two or more robots working together to achieve a common goal [11]. Studies have shown that multirobot systems are more effective and efficient at accomplishing complex tasks, such as localization, when compared to single robot systems [11, 34]. Furthermore, multirobot systems can be more fault tolerant, thus improving the systems' reliability and robustness. The main idea in multirobot localization is the use of robots themselves as *portable landmarks* to others in order to improve the localization accuracy, especially in unchartered areas with insufficient landmarks [57, 58]. Multirobot localization (and mapping) methods published in the literature are mostly adaptations of the existing single-robot localization techniques to multiple robots [11, 59]. These adaptations have been successfully implemented in centralized, multi-centralized, and decentralized multirobot systems.

2.4.1 Centralized Multirobot Systems

Centralized multirobot systems [60, 61, 58, 62, 63] consist of multiple, networked robots, whereby one of the robots, the leader, serves as the processing center, also known as the Fusion center (FC) [64] as shown in Figure 2.1. During localization, the processing center stores the states (poses along with their uncertainties) of all the robots in one place, usually in the form of an augmented matrix, and jointly updates all the states whenever new information from interacting[3] robots is received over a network. The information normally includes odometric and inter-robot measurements, such as robot-robot distances and relative angles. Centralized multirobot systems can provide optimal pose estimates for all robots in a system. However, since all the data and computations are managed by one robot (processing center), the system is vulnerable to single-point-failures in case this robot fails. Furthermore, algorithms for centralized, multirobot systems are poorly scalable, because as the number of robots in a team increases, the computation time increases exponentially [65] while the communication cost increases linearly. Therefore, localization in centralized, multirobot systems is not feasible for a large number of robots.

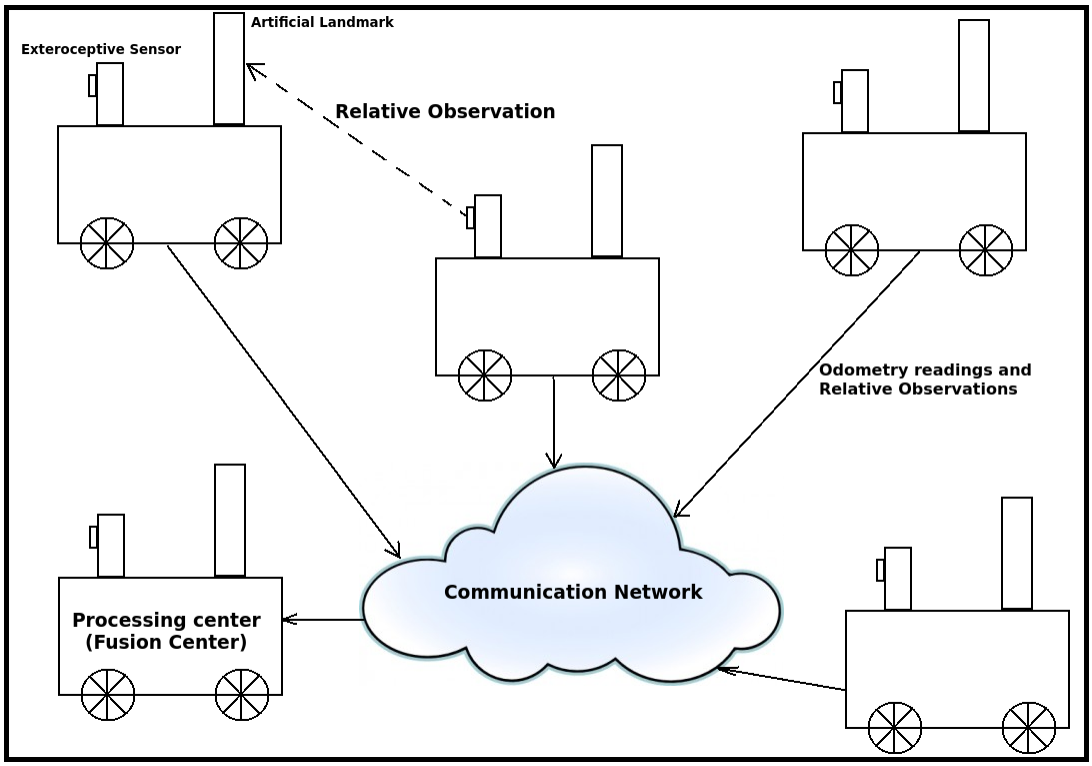


Figure 2.1: Centralized Multirobot System

[3]Interaction occurs whenever two or more robots detect each other through their exteroceptive sensors

2.4.2 Multi-centralized Multirobot Systems

Multi-centralized multirobot systems such as [66] and [64] are similar to centralized multirobot systems with the exception that each robot in the system serves as a processing center as shown in Figure 2.2, and updates the states of all the robots whenever two or more robots observe each other. In other words, each robot maintains a copy of the entire group's state. This property makes multi-centralized, multirobot systems robust to single-point-failures, because if one robot fails, others would still have access to a copy of the entire state of the system. However, the robustness due to redundancy comes at an even higher computational and communication cost when compared to centralized multirobot systems [14].

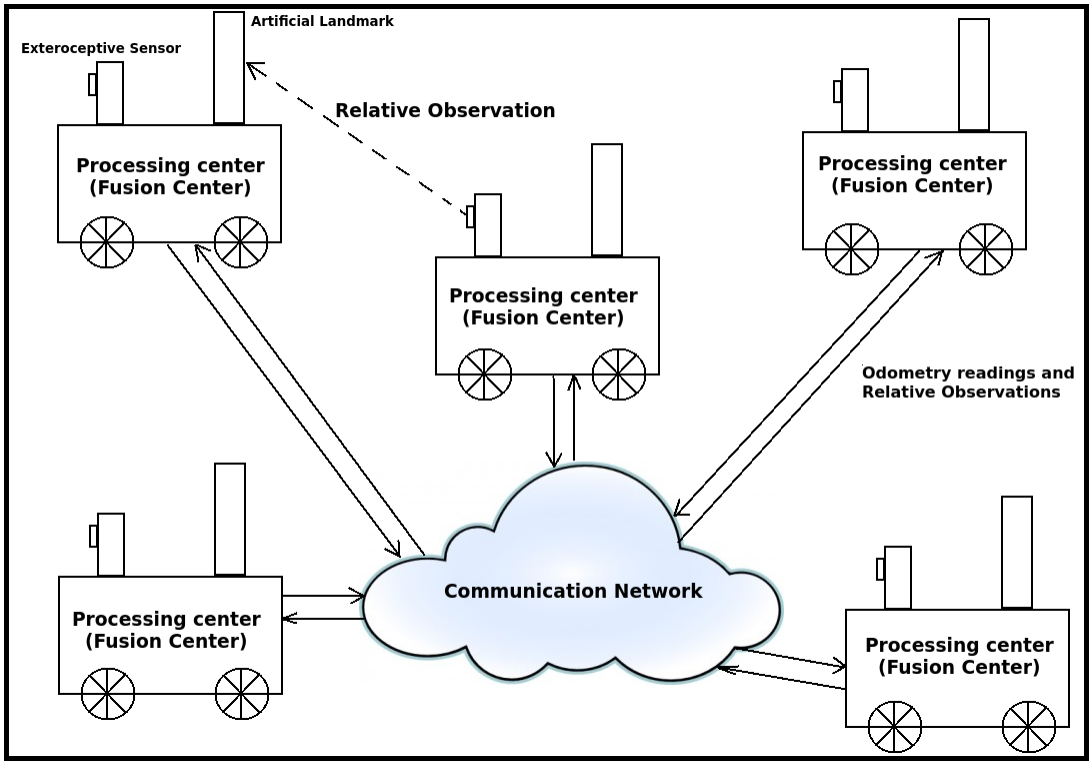


Figure 2.2: Multicentralized Multirobot System

2.4.3 Decentralized Multirobot Systems

In decentralized multirobot systems, each robot maintains only the information about its own pose, and updates this information locally when interacting with other robots, allowing the system to become scalable while significantly reducing computational and communication costs. Furthermore, decentralized multirobot systems are not susceptible to single-point failures and can span over a larger area since interacting robots don't have to be within communication range of the rest of the robots. For these reasons, decentralized, multirobot systems have been the main focus of research in multirobot localization, leading to the development of various optimal [67, 68], and suboptimal (approximate) [13, 69, 70, 71, 65, 14] decentralized localization algorithms. The biggest challenge in decentralized systems is the problem of *double-counting*, or *data incest*, which arises when interacting robots exchange correlated or interdependent data with each other, resulting in inconsistent estimates which are either overly-optimistic, or overly-pessimistic [72]. For example, in Figure 2.3, two robots, *robot0* and *robot1*, initially are at locations A and B with covariances $\sum_A$ and $\sum_B$ respectively.

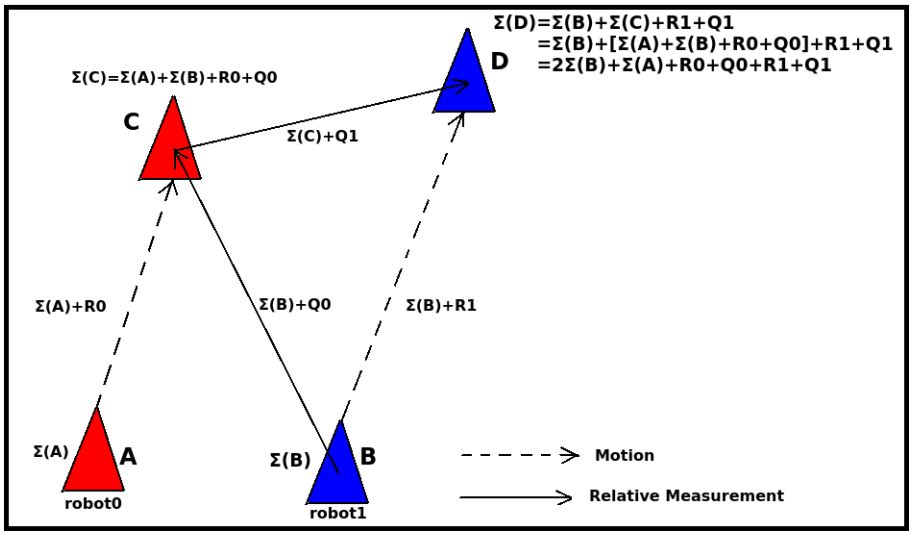


Figure 2.3: Double counting or data incest which occurs in interdependent, distributed systems.

As *robot0* moves from A to C, it propagates its initial covariance, $\sum_A$, from A to C, and accumulates additional error, R_0, due to noise in its proprioceptive sensors. At C, *robot0* refines its estimated pose through relative observation of *robot1* using its exteroceptive sensors. These measurements will propagate the covariance of *robot1*, $\sum_B$, from point B to C, as well as uncertainty, Q_0, due to noise from exteroceptive sensors. Similarly, as *robot1* moves from B to D, it propagates its initial covariance, $\sum_B$, from B to D, and accumulates more error, R_1, from its proprioceptive sensors. However, if at D, *robot1* uses *robot0* to improve its estimated location via relative measurements, it will propagate part of its own covariance, $\sum_B$, from when it was at B, in addition to the uncertainty, Q_1 , due to noise in its exteroceptive sensors. This means *robot1* will have its original covariance

included twice as it moves from B to D, which results in inconsistent estimates, often overly optimistic in which case the uncertainty associated with estimated poses is much smaller than the actual uncertainty. This type of distributed approach is called *Naïve* , because it ignores the interdependencies or correlations between robots.

Optimal, decentralized localization algorithms, such as [67] by Roumeliotis and Bekey, and [68] by Nerurkar *et al.*, are capable of providing consistent pose estimates of all robots in a team by keeping track of interdependencies between the robots. The optimal performance however, comes at very high computational and communication costs. In order to minimize these costs, several researchers have resorted to approximate or sub-optimal decentralized localization algorithms. Some of the approximate algorithms such as [69, 71] and [57] ignore the possible interdependencies between the robots at the risk of providing inconsistent estimates, while others, such as [13, 70] and [14] are capable of providing consistent estimates through the effective handling of interdependencies between robots. Table 2.2 provides a summary of some of the optimal, and sub-optimal (approximate) decentralized, multirobot localization systems available in the literature.

Table 2.2: Summary of some of the decentralized multirobot localization systems presented in the literature.

	Authors				
	Roumeliotis [67]	Nerurkar [68]	Panzieri [57]	Carrillo-Arce [13]	Wanasinghe [14]
Year	2000	2009	2006	2013	2014
Algorithm	Classic Extended Kalman Filter	Maximum A Posteriori	Interlaced Extended Kalman Filter	Covariance Intersection Filter	Split-Covariance Intersection Filter
Performance	Optimal	Optimal	Approximate	Approximate	Approximate
Consistency	Consistent	Consistent	Inconsistent	Consistent*	Consistent
Computational Complexity**	$O(N^4)$	$O(N^2)$	$O(N)$	$O(N)$	$O(N)$
Communication Complexity	$O(N^2)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
Simulations		✓	✓	✓	✓
Real world Experiments	✓		✓	✓	
Scalability	Poor	Poor	Good	Good	Good

*While Carrillo-Arce et al. claim that their method provides consistent results, Wanasinghe et al. of [14] allege that this method could still lead to inconsistent results because it treats all estimates as interdependent, and ignores possible independencies.
**N is the number of robots in a system.

The results provided in Table 2.2 clearly show the advantages and disadvantages of optimal and sub-optimal (approximate) decentralized systems in terms of their computational and communicational complexities as well as their consistencies. Section 2.5 will describe how the accuracy and consistency of such systems is usually evaluated.

2.5 Evaluation of Accuracy and Consistency in Robot Localization

The belief $bel_(x_t)$ represents the estimated robot position (also called the mean), μ_t, as well as the estimated uncertainty, $\sum_t$.

$$bel(x_t) = (\mu_t, \sum_t)$$

The accuracy of the multirobot systems is usually evaluated by taking the *Root Mean Squared Error (RMSE)* for each robot in a system [73]. RMSE indicates how close (or far) the estimated position of the robot is to the true (actual) position. Suppose a mobile robot moves along a predefined path (ground truth) several times, then at each known location of the robot, the RMSE in x, y, and θ is respectively given by:

$$RMSE_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x}_i)^2}{n}} \tag{2.6}$$

$$RMSE_y = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2}{n}} \tag{2.7}$$

$$RMSE_\theta = \sqrt{\frac{\sum_{i=1}^{n}(\theta_i - \overline{\theta}_i)^2}{n}} \tag{2.8}$$

Where:

n is the number of times the robot passes through the same known location.

$RMSE_x$ is the Root Mean Squared Error in x-axis.

$RMSE_y$ is the Root Mean Squared Error in y-axis.

$RMSE_\theta$ is the Root Mean Squared Error in orientation.

$[x_i, y_i, \theta_i]$ is the actual pose of the robot at the ith run.

$[\overline{x}_i, \overline{y}_i, \overline{\theta}_i]$ is the corresponding estimated pose of the robot at ith run.

On the other hand, the consistency of the estimator is evaluated graphically by determining whether or not the actual pose of the robot falls within a reasonable area (in case of 2D), or volume (in case of 3D) of its estimated pose as shown in Figure 2.4 (Top). The area, or volume, around the estimated pose represents the uncertainty, or all possible locations where the robot is likely to be found. In case of 2D localization, the area is an ellipse, whereas in 3D localization, the volume is an ellipsoid. The estimated covariance matrix, $\sum_t$, determines the shapes and sizes of the ellipses and ellipsoids. If the actual pose of the robot falls outside this area (or volume) at a specified probability (usually 95% or 99%), then the estimator is said to be *overconfident or optimistic* as shown in Figure 2.4 (middle). However, the area or volume around the estimated pose should not be too large relative to the length of the vector connecting the estimated and actual positions of the robot. If that happens, the estimator becomes *pessimistic* as shown in Figure 2.4 (bottom). Both cases, optimism and pessimism, result in inconsistent, or non-credible[4] estimator [74] .

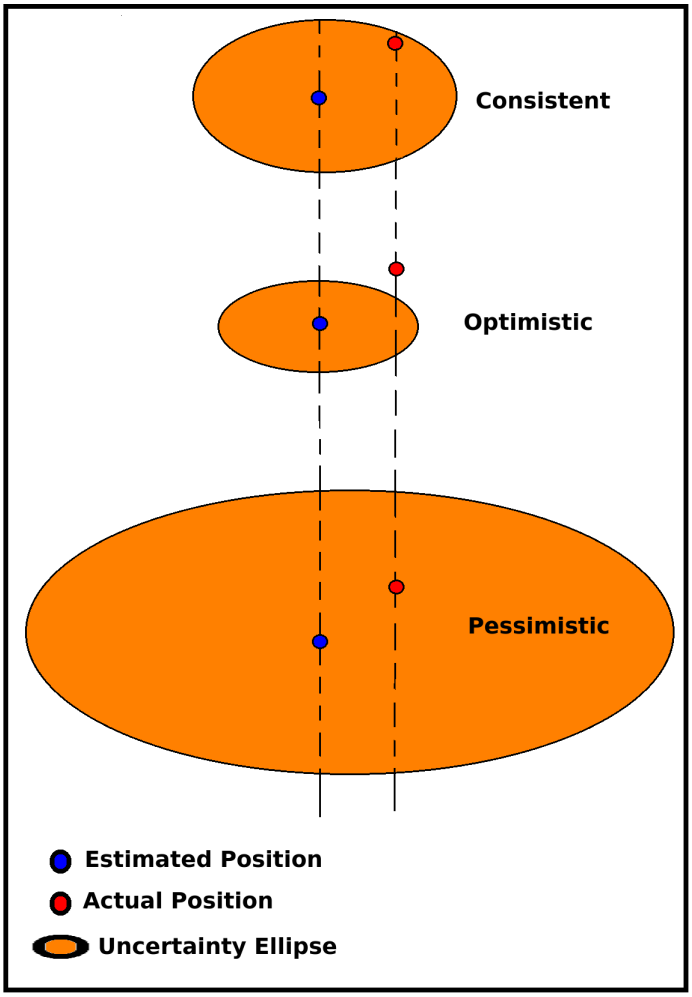


Figure 2.4: Examples of consistent (top), optimistic (middle), and pessimistic (bottom) cases with same RMSE.

[4]The terms credibility and consistency are often used interchangeably in the literature

Mathematically, the consistency of the estimator is evaluated using the *Normalized Estimation Error Squared (NEES)* which relates the absolute error of the pose, $[\mu - \overline{\mu}]$, to the estimated uncertainty, $\overline{\sum}$, by the following formula:

$$NEES = \frac{[\mu - \overline{\mu}]^T \overline{\sum}^{-1} [\mu - \overline{\mu}]}{\|\mu\|} \tag{2.9}$$

Where:

μ is the actual pose of the robot.

$\overline{\mu}$ is the corresponding estimated pose of the robot.

$\overline{\sum}$ is the estimated Covariance Matrix (i.e., estimated uncertainty).

$\|\mu\|$ is the dimension of the state vector, μ.

The NEES of an ideal estimator is 1. On the other hand, the NEES of an optimistic estimator is greater than 1 while that of a pessimistic estimator is smaller than 1. Often times the *Average Normalized Estimation Error Squared (ANEES)* is used to determine the credibility of an estimator at a specified probability (usually 95%). This is done by calculating the mean NEES at each known location of the robot as it moves along a predefined path several times (Equation 2.10).

$$ANEES = \frac{1}{n\|\mu\|} \sum_{i=1}^{n} [\mu_i - \overline{\mu}_i]^T \sum_i^{-1} [\mu_i - \mu_i] \tag{2.10}$$

Where:

n, μ, $\overline{\mu}$, $\overline{\sum}$ and $\|\mu\|$ have the same meaning as in equations 2.6, 2.7, 2.8 and 2.9.

The estimator is deemed credible/consistent if the ANEES value falls within the bounds of a Chi-squared test at a specified probability.

Some of the most recent successful, decentralized multirobot localization methods, such as [14] and [13] provide consistent estimates through the use of a technique called Covariance intersection, which fuses information under unknown interdependencies through convex combination of mean and covariances. The main drawback of this technique is that it requires additional computational time needed to iteratively determine the most optimal way of combining mean and covariances in order to achieve consistent estimated results [75].

One of the approaches used to overcome this problem involves heuristic tuning of the Extended Kalman Filter by deliberately adding artificial noise into the system which offsets the effects of double-counting and linearization and consequently results in consistent estimations. This process is referred to as *injection of stabilization noise* and was introduced by Maybeck [76]. Since then, it has been shown that under certain conditions, such as when the degree of nonlinearity of a system is not too high, the consistency of EKF estimates can be adequately maintained through deliberate addition of stabilization noise [77, 78]. In the context of robot localization, consistency in EKF poses can be achieved through the inflation of landmark covariances after each update [77], or through the inflation of sensor covariances [79]. However, formal ways for determining the adequate inflation of the covariances have not yet been defined. Thus, in this thesis a formal methodology for improving the consistency of EKF estimates through artificial inflation of landmark covariances in single robot localization is proposed, and further extended to multirobot systems.

2.6 Chapter Summary

This chapter introduced the localization problem in mobile robots and presented the probabilistic framework which forms the basis for most existing state-of-the-art solutions to the localization problem. Section 2.4 introduced the concept of multirobot localization and outlined the strengths and limitations some of the latest state-of-art methods used to localize teams of robots and proposed a heuristic approach to overcome these limitations.

Chapter 3

Proposed Approach to Multirobot Localization

This chapter presents an investigation of a localization algorithm based on the Extended Kalman Filter before and after heuristic tuning of the filter. Section 3.1 introduces the general EKF algorithm for single robot localization in an environment where the locations of landmarks are known *a-priori* (beforehand) and presents the simulation results. Section 3.2 proposes a formal heuristic methodology of tuning the original Extended Kalman Filter through artificial inflation of landmark covariance matrices and demonstrates the effect of tuning on the consistency of EKF estimates in single robot localization. Section 3.3 extends the proposed methodology to decentralized multirobot localization whereby the robots carry their own *portable landmarks* in environments where fixed (static) artificial or natural landmarks are lacking or insufficient.

3.1 Single Robot Localization using Extended Kalman Filter

Many real world systems are not linear. For example, a robot moving with a constant velocity seldom moves in a straight line. For such systems, the Extended Kalman Filter (EKF) is used instead of the Linear Kalman Filter. The EKF approximates the true belief, *bel*, by a multivariate Gaussian using mean and covariance through first order Taylor expansion of a nonlinear state function. In the case of robot localization, the mean represents the robot's pose (location and orientation), while the covariance represents the uncertainty associated with this pose. EKF recursively estimates the belief through weighted average, commonly referred to *fusion*, of beliefs from various measurement sources in such a way that more weight is placed on measurements from more accurate sources. For example, suppose a mobile robot moves in a straight line along the x-axis, from an arbitrary location on the axis, and stops at exactly x=4. Using its less accurate proprioceptive sensors, such as wheel encoders, as well as its motion model, the robot predicts its new location to be

somewhere between x=2.5 and x=6, with a peak at x=4.25 as shown in Figure 3.1. However, upon using its more accurate exteroceptive sensor, for example, a laser rangefinder, to observe nearby landmarks, the robot finds its new location to be between x=3.1 and x=4.8, with a peak at x=3.95 (Figure 3.2). The EKF then fuses the two beliefs by putting more weight on the more accurate exteroceptive sensor. The refined robot's location now lies between x=2.8 and x=5.2, with a peak at x=4 (Figure 3.3). Thus, through weighted combination of measurements from various sources, the EKF is able to provide more accurate estimates.

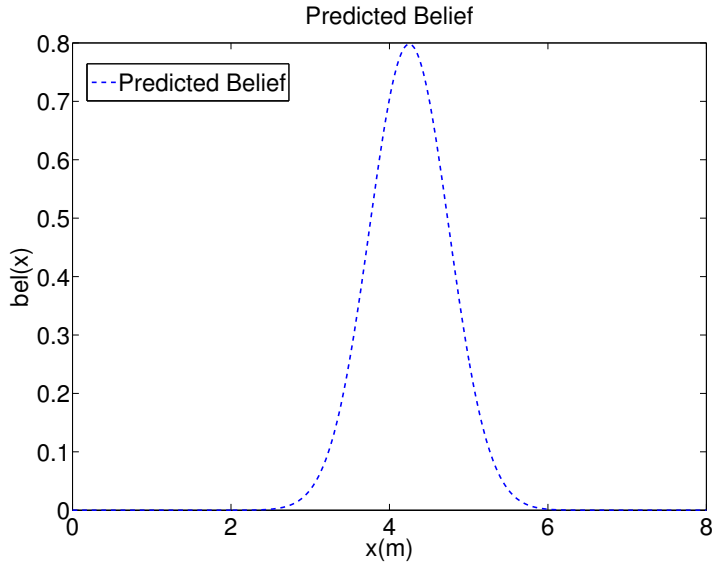


Figure 3.1: Predicted belief

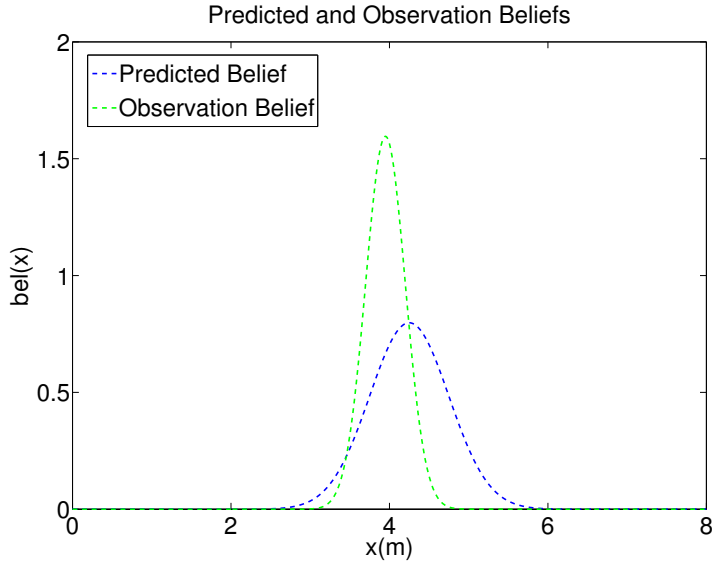


Figure 3.2: Predicted and observed beliefs

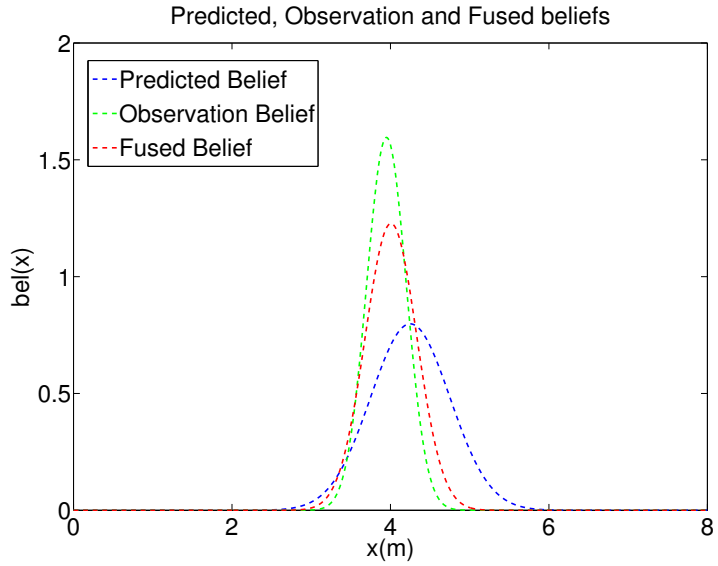


Figure 3.3: Weighted average of predicted and observed beliefs.

Extended Kalman Filters have extensively and successfully been used in absolute landmark localization (and mapping) [80, 77, 30]. The algorithm keeps the uncertainty associated with the pose of the moving robot bounded, by periodically correcting the position estimates through relative observations of fixed landmarks. For example, as the robot moves in a straight line in Figure 3.4, its uncertainty, characterized by ellipses[1], gradually grows. However, when the robot observes a landmark whose location is known *a-priori*, the uncertainty is reduced.

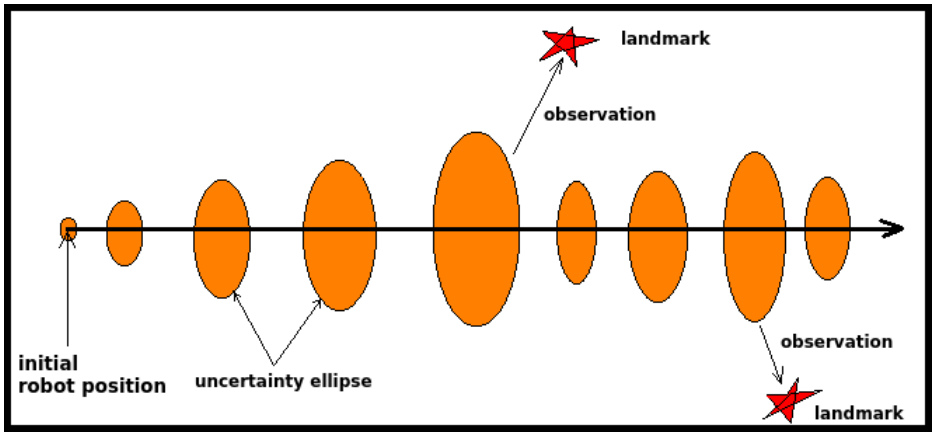


Figure 3.4: Landmark localization using Extended Kalman Filter.

[1]In 2D localization, the ellipses represent the area where the robot is most likely to be located. The smaller the ellipse, the higher the confidence

The Extended Kalman Filter described in this chapter uses two models introduced in Section 2.3 in order to estimate the robot location. The two models are *Action Model* and the *Perception Model*.

3.1.1 Action (Motion) Model

The action model estimates the robot's new belief, $\overline{bel}(x_t)$, from its previous belief, $bel(x_{t-1})$, its kinematic (geometric) model, and control input u_t. The action model considered here is that of a 2-wheeled mobile robot with wheel encoders in any of the wheel configurations shown in Figure 3.5.

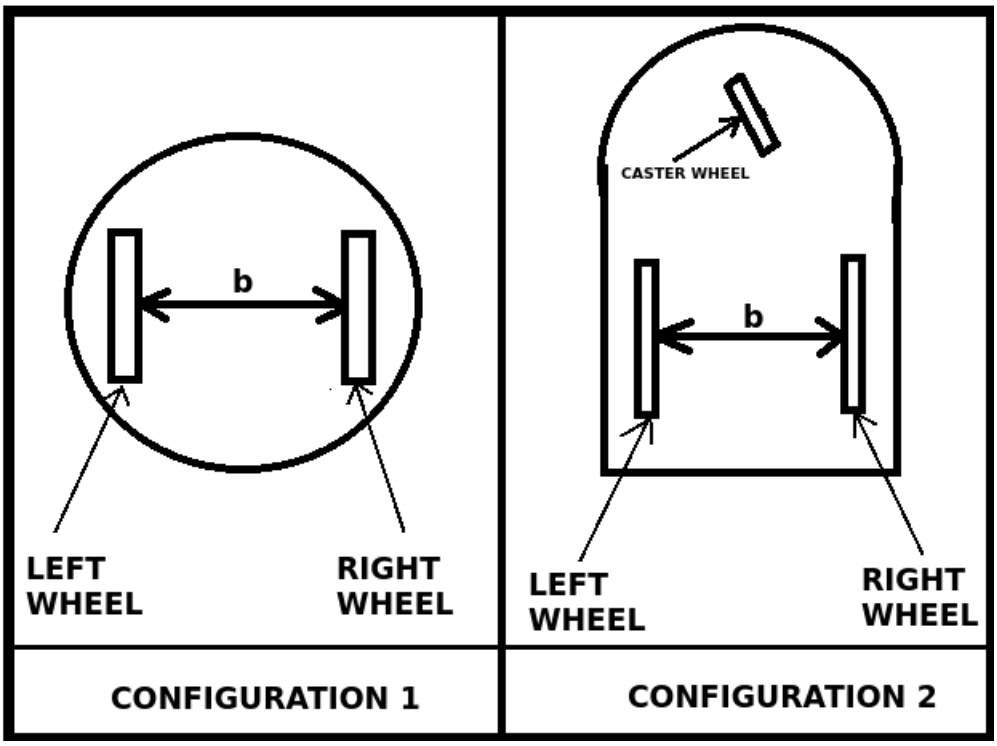


Figure 3.5:
CONFIGURATION 1: Two fixed wheels.
CONFIGURATION 2: Two fixed wheels and a caster.

For both configurations in Figure 3.5, the control input is given by:

$$u_t = \begin{bmatrix} \Delta L \\ \Delta R \end{bmatrix}$$

Where:
ΔL is the distance moved by the left wheel
ΔR is the distance moved by the right wheel

Figure 3.6 illustrates how the pose of a 2-wheeled mobile robot changes as it moves from A to B when a control input $u_t = [\Delta L, \Delta R]^T$ is applied.

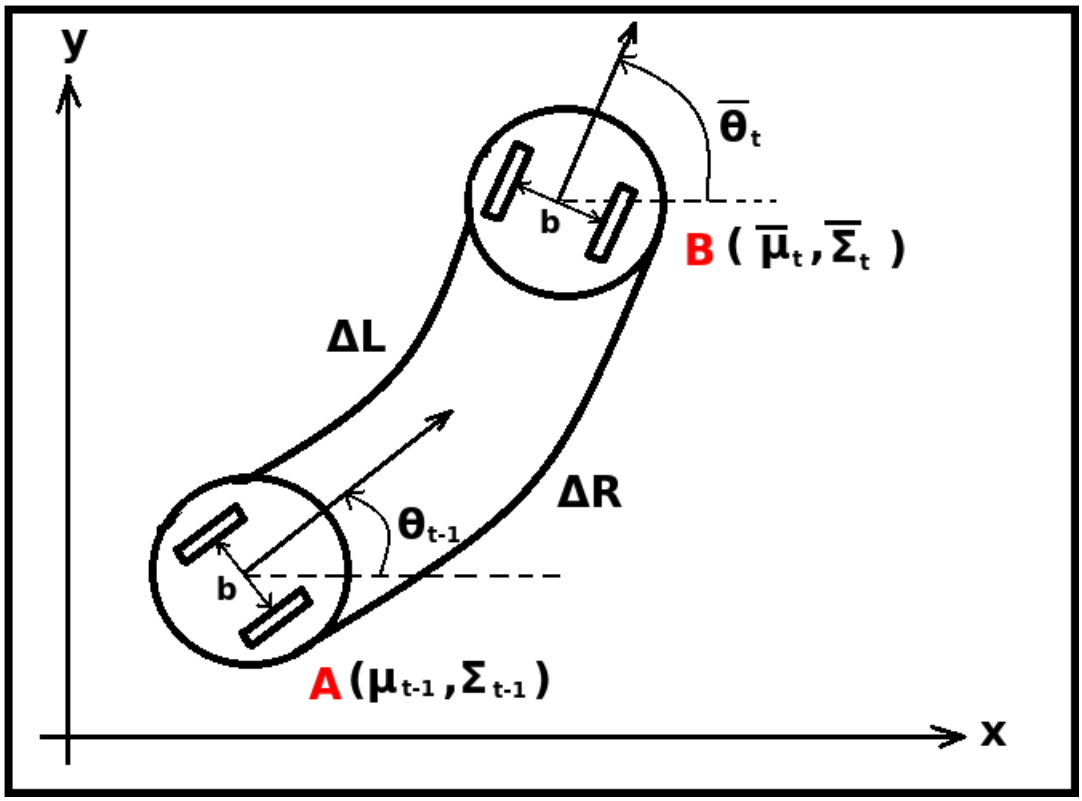


Figure 3.6: Motion of a 2-wheeled robot when a control input $[\Delta L, \Delta R]^T$ is applied

The new pose of the robot ($\overline{\mu}_t$), estimated from its previous pose (μ_{t-1}), and control input (u_t) is given by:

$$\overline{\mu}_t = \mu_{t-1} + \begin{bmatrix} (\Delta S)cos(\theta_{t-1} + \Delta\theta) \\ (\Delta S)sin(\theta_{t-1} + \Delta\theta) \\ 2\Delta\theta \end{bmatrix} \tag{3.1}$$

Where:

$\overline{\mu}_t = [\overline{x}_t, \overline{y}_t, \overline{\theta}_t]^T$

$\mu_{t-1} = [x_{t-1}, y_{t-1}, \theta_{t-1}]^T$

$\Delta S = \frac{\Delta R + \Delta L}{2}$

$\Delta\theta = \frac{\Delta R - \Delta L}{2b}$

b is the distance between the wheels as shown in Figure 3.5

The robot's uncertainty associated with its estimated new pose is determined as follows:

$$\overline{\sum}_t = G_{\mu_t} \sum_{t-1} G_{\mu_t}^T + G_{u_t} U_t G_{u_t}^T \tag{3.2}$$

$\sum_{t-1}$ is the covariance matrix of the robot at its previous pose, μ_{t-1}.

G_{μ_t} and G_{u_t} are the Jacobians of the estimated pose, $\overline{\mu}_t$, at the previous pose, μ_{t-1}, and control input, u_t, respectively, given by:

$$G_{\mu_t} = \frac{\partial \overline{\mu}_t}{\partial \mu_{t-1}} = \begin{bmatrix} 1 & 0 & -(\Delta S) sin(\theta_{t-1} + \Delta\theta) \\ 0 & 1 & (\Delta S) cos(\theta_{t-1} + \Delta\theta) \\ 0 & 0 & 1 \end{bmatrix}$$

$$G_{u_t} = \frac{\partial \overline{\mu}_t}{\partial u_t} \tag{3.3}$$

$$= \frac{1}{2b} \begin{bmatrix} (\Delta S) sin(\theta_{t-1} + \Delta\theta) + b cos(\theta_{t-1} + \Delta\theta) & -(\Delta S) sin(\theta_{t-1} + \Delta\theta) + b cos(\theta_{t-1} + \Delta\theta) \\ -(\Delta S) cos(\theta_{t-1} + \Delta\theta) + b sin(\theta_{t-1} + \Delta\theta) & (\Delta S) cos(\theta_{t-1} + \Delta\theta) + b sin(\theta_{t-1} + \Delta\theta) \\ -2 & 2 \end{bmatrix} \tag{3.4}$$

U_t is the covariance matrix of the control input. It specifies the uncertainties associated with the left and right wheels, assuming that the noise associated with each wheel is white Gaussian proportional to the distance that each wheel travels.

$$U_t = \begin{bmatrix} (K_L|\Delta L|)^2 & 0 \\ 0 & (K_R|\Delta R|)^2 \end{bmatrix}$$

Where:
K_L is the percentage error in the left wheel.
K_R is the percentage error in the right wheel.

$\overline{\mu}_t$ and $\overline{\sum}_t$ determined in Equations 3.1 and 3.2 through the action model collectively form *a-priori* belief, $\overline{bel}(x_t)$ which can be compactly expressed as follows:

$$\overline{bel}(x_t) = (\overline{\mu}_t, \overline{\sum}_t) \tag{3.5}$$

The process of calculating the *a-priori* belief is known as *prediction*. The next Section describes the *measurement model*, and how this model is used to refine the $\overline{bel}(x_t)$ obtained in the *Prediction* step.

3.1.2 Perception or Measurement model

This model provides the relationship between observation measurements, such as range and bearing, of a landmark relative to a robot. In the Extended Kalman Filter, this model is used to refine the pose estimated in the *Prediction* step. The refinement process is often referred to *correction* or *measurement update*. The measurement model considered here is of the form:

$$Z_t = [\rho, \phi]$$

Where:
ρ is the relative distance between a mobile robot and an observed landmark.
ϕ is the bearing of the landmark relative to the robot as shown in Figure 3.7.

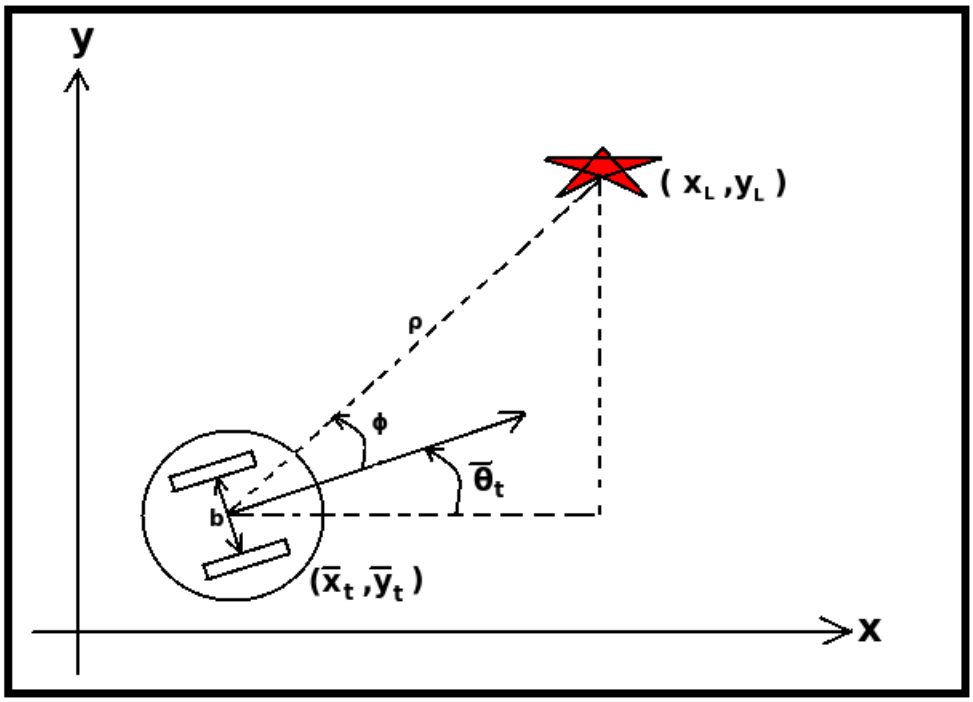


Figure 3.7: Distance (ρ) and bearing (ϕ) of an observed landmark (red star) with respect to a mobile robot.

From Figure 3.7, it can be shown with the help of trigonometric identities that the *estimated* distance ($\overline{\rho}$) and bearing ($\overline{\phi}$), of the landmark relative to the estimated pose of the robot, $\overline{\mu}_t = [\overline{x}_t, \overline{y}_t, \overline{\theta}_t]$, is given by:

$$\overline{Z}_t = \begin{bmatrix} \overline{\rho} \\ \overline{\phi} \end{bmatrix} = \begin{bmatrix} \sqrt[2]{(x_L - \overline{x}_t)^2 + (y_L - \overline{y}_t)^2} \\ atan2(y_L - \overline{y}_t, x_L - \overline{x}_t) - \overline{\theta}_t \end{bmatrix} \tag{3.6}$$

The uncertainty associated with the estimated relative measurements in Equation 3.6 is given by the following formula:

$$S = H_R \overline{\sum}_t H_R^T + H_L \sum_L H_L^T + Q_t \tag{3.7}$$

$\sum_L$ is the uncertainty associated with the location of the landmark.
H_R and H_L are the Jacobians of the estimated measurements, $\overline{Z}_t$, at the estimated pose of the robot, $\overline{\mu}_t = [\overline{x}_t, \overline{y}_t, \overline{\theta}_t]$, and the known location of the landmark, $\mu_L = [x_L, y_L, \theta_L]$, respectively.

$$H_R = \frac{\partial \overline{Z}_t}{\partial \overline{\mu}_t} = \frac{1}{q} \begin{bmatrix} -(x_L - \overline{x}_t) & -(y_L - \overline{y}_t) & 0 \\ \frac{(y_L - \overline{y}_t)}{q} & -\frac{(x_L - \overline{x}_t)}{q} & -q \end{bmatrix} \tag{3.8}$$

$$H_L = \frac{\partial \overline{Z}_t}{\partial \mu_L} = \frac{1}{q} \begin{bmatrix} (x_L - \overline{x}_t) & (y_L - \overline{y}_t) & 0 \\ -\frac{(y_L - \overline{y}_t)}{q} & \frac{(x_L - \overline{x}_t)}{q} & 0 \end{bmatrix} \tag{3.9}$$

Where:

$q = \sqrt[2]{(x_L - \overline{x}_t)^2 + (y_L - \overline{y}_t)^2}$

Q_t is the covariance matrix of the relative measurements. It specifies the uncertainties associated with relative distance and angle, assuming white Gaussian noise.

$$Q_t = \begin{bmatrix} (\Delta\rho)^2 & 0 \\ 0 & (\Delta\theta)^2 \end{bmatrix}$$

Where:
$\Delta\rho$ is the error in observed relative distance.
$\Delta\phi$ is the error in the observed relative bearing.

The uncertainties associated with estimated pose, $\overline{\sum}_t$, and estimated measurements, S, are used to calculate the gain matrix, K. This matrix determines by how much the pose estimated through motion of the robot in Equation 3.1 should be adjusted by the estimated relative observations obtained in Equation 3.6.

$$K_t = \overline{\sum}_t H_R^T (S)^{-1} \tag{3.10}$$

The gain is then used to update the predicted pose, $\overline{\mu}_t$, and covariance, $\overline{\sum}_t$, as follows:

$$\mu_t = \overline{\mu}_t + K_t (Z_t - \overline{Z}_t) \tag{3.11}$$

Where:
$\overline{Z}_t$ is the *estimated* relative measurements from Equation 3.6.
Z_t is the *actual* relative measurements that a mobile robot takes using its sensors.

$$\sum_t = (\mathbf{I} - K_t H_R)\overline{\sum}_t \tag{3.12}$$

Where $\mathbf{I}$ is the identity matrix.

μ_t and $\sum_t$ collectively form the *a-posteriori* belief, $bel(x_t)$, and can be compactly expressed in the following form:

$$bel(x_t) = (\mu_t, \sum_t) \tag{3.13}$$

Table 3.1: Summary of EKF algorithm for a 2-wheeled mobile robot

Input: $[\Delta L, \Delta R]^T, [K_L, K_R]^T, [x_{t-1}, y_{t-1}, \theta_{t-1}]^T, \sum_{t-1}), [\rho, \phi]^T, \sum_L$

Prediction Step

1. $\Delta S = \frac{\Delta R + \Delta L}{2}$
2. $\Delta\theta = \frac{\Delta R - \Delta L}{2b}$
3. $[\overline{x}_t, \overline{y}_t, \overline{\theta}_t]^T = [x_{t-1}, y_{t-1}, \theta_{t-1}]^T + \begin{bmatrix} (\Delta S)cos(\theta_{t-1} + \Delta\theta) \\ (\Delta S)sin(\theta_{t-1} + \Delta\theta) \\ 2\Delta\theta \end{bmatrix}$
4. $G_{\mu_t} = \frac{\partial \overline{\mu}_t}{\partial \mu_{t-1}}$
5. $G_{u_t} = \frac{\partial \overline{\mu}_t}{\partial u_t}$
6. $U_t = \begin{bmatrix} (K_L|\Delta L|)^2 & 0 \\ 0 & (K_R|\Delta R|)^2 \end{bmatrix}$
7. $\overline{\sum}_t = G_{\mu_t} \sum_{t-1} G_{\mu_t}^T + G_{u_t} U_t G_{u_t}^T$

Correction Step

8. $[\overline{\rho}_t, \overline{\phi}_t]^T = \begin{bmatrix} \sqrt[2]{(x_L - \overline{x}_t)^2 + (y_L - \overline{y}_t)^2} \\ atan2(y_L - \overline{y}_t, x_L - \overline{x}_t) \end{bmatrix}$

9. $q = \sqrt[2]{(x_L - \overline{x}_t)^2 + (y_L - \overline{y}_t)^2}$

10. $H_R = \frac{1}{q} \begin{bmatrix} -(x_L - \overline{x}_t) & -(y_L - \overline{y}_t) & 0 \\ \frac{(y_L - \overline{y}_t)}{q} & -\frac{(x_L - \overline{x}_t)}{q} & -q \end{bmatrix}$

11. $H_L = \frac{1}{q} \begin{bmatrix} (x_L - \overline{x}_t) & (y_L - \overline{y}_t) & 0 \\ -\frac{(y_L - \overline{y}_t)}{q} & \frac{(x_L - \overline{x}_t)}{q} & 0 \end{bmatrix}$

12. $Q_t = \begin{bmatrix} (\Delta\rho)^2 & 0 \\ 0 & (\Delta\theta)^2 \end{bmatrix}$

13. $S = H_R \overline{\sum}_t H_R^T + H_L \sum_L H_L^T + Q_t$

14. $K_t = \overline{\sum}_t H_R^T (S)^{-1}$

15. $[x_t, y_t, \theta_t]^T = [\overline{x}_t, \overline{y}_t, \overline{\theta}_t]^T + K_t([\rho_t, \phi_t]^T - [\overline{\rho}_t, \overline{\phi}_t]^T)$

16. $\sum_t = (\mathbf{I} - K_t H_R) \overline{\sum}_t$

Return: $([x_t, y_t, \theta_t]^T, \sum_t)$

Section 3.1.3 implements the Extended Kalman Filter of Table 3.1 in a simulated environment for a general case involving a single robot surrounded by N static observable landmarks.

3.1.3 Simulation of Landmark Localization of a Single Robot

Localization of a single robot using static landmarks whose locations are known *a-priori* was simulated in MATLAB [81] in order to verify that the Extended Kalman Filter works properly on a single robot, before adopting the algorithm to localize multiple robots in a distributed manner.

The simulator keeps track of the following:

1. Ground truth (μ_t_actual): The actual pose of a mobile robot.
2. Odometry pose ($\mu_t\_odometer$): The pose of a robot determined using wheel encoders.
3. Filtered pose (μ_t): The pose of a robot determined by EKF algorithm.
4. Covariance matrix ($\sum_t$): Covariance matrix associated with the filtered pose, μ.

The following assumptions were made when implementing the simulator:

1. The robot moves in obstacle-free environment. This assumption becomes relevant when considering a group of mobile robots traversing a large, open field.
2. Landmarks are stationary, and placed within the sensing range of the exteroceptive sensors.
3. The robot's relative bearing sensor is assumed to be located at the center of the axis joining the two wheels and has 360° field of view as shown in Figure 3.9.
4. The correspondence between sensor observations and the landmarks is known (*i.e* the mobile robot is able to positively identify a landmark through relative observations).
5. Additive white Gaussian noise is added to both control input, u_t, and relative measurements, Z_t in order to reflect real-world uncertainties associated with the wheel encoders and range sensors. In particular, both encoder and range sensor errors were intentionally set high to reflect the low resolution and accuracy in the sensors used in real-world experiments (Chapter 4).

The following simulation parameters correspond to the geometry of the actual platforms used in real-world experimentation (Chapter 4), the accuracy of the sensors, and the resolution of the ground truth

1. Encoder error as a percentage of distance traveled: $K_L = K_R = 0.05$
2. Initial covariance matrix associated with robot pose: $\sum_0 = diag([0.15m^2, 0.15m^2, 0.15rad^2])$
3. Covariance matrix associated with landmark location: $\sum_L = diag([0.15m^2, 0.15m^2, 0.15rad^2])$

4. Covariance matrix for relative measurements: $Q_t = diag([0.1m^2, 0.1rad^2])$
5. Distance between wheels: $b = 0.4m$
6. Control input: $u_t = [\Delta L, \Delta R] = [0.25m, 0.25m]$
7. Covariance matrix for control input: $R_t = diag([(0.05\Delta L)^2, (0.05\Delta R)^2])$

The flowchart in Figure 3.8 presents the high level implementation of the single robot landmark localization simulator for the general case with N observable landmarks around a robot.

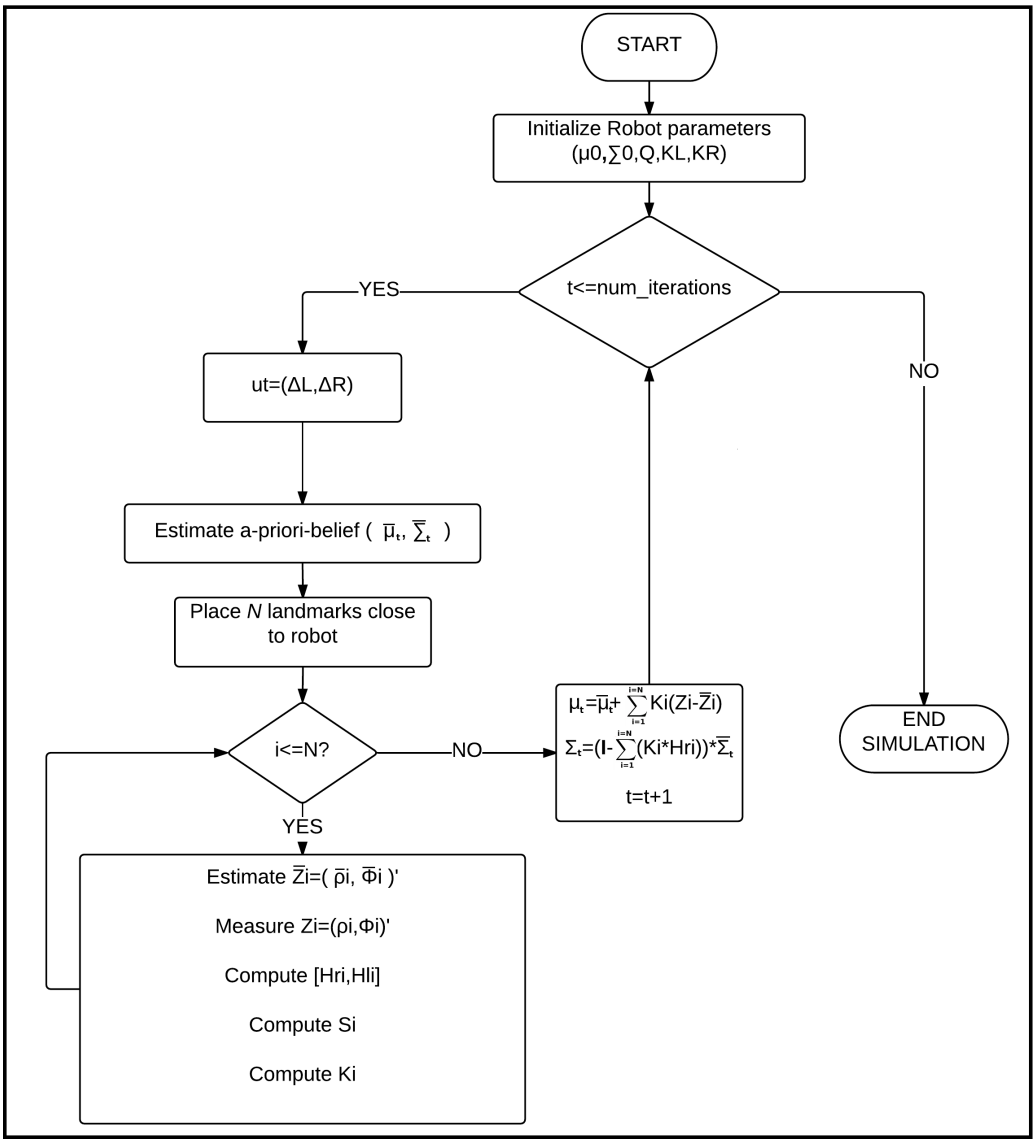


Figure 3.8: High Level implementation of the EKF localization algorithm

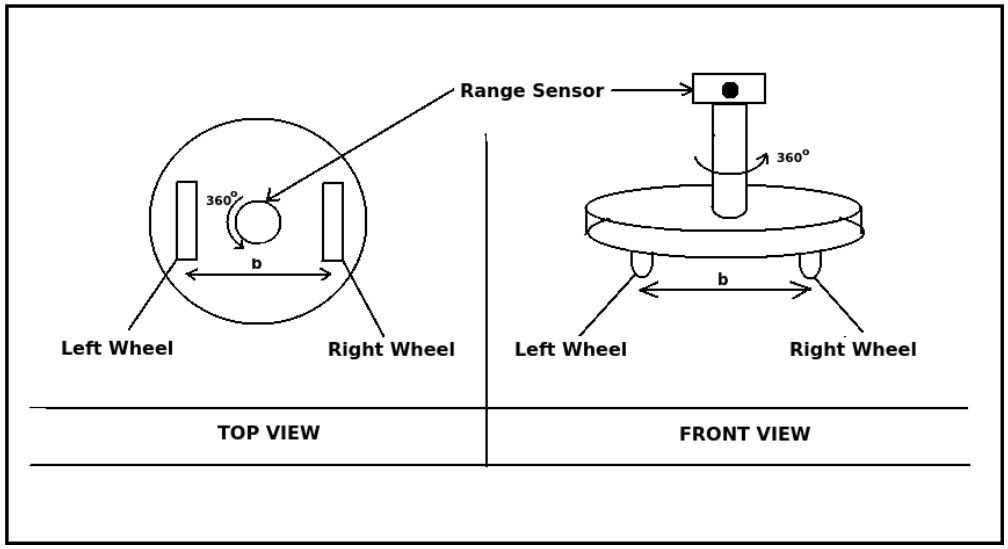


Figure 3.9: Assumed location of a range sensor in a simulated robot.

50 simulation runs were made for each N=1, 2, 3, 4, and 5 where N is the number of static observable landmarks within the field of view of the moving robot. In all simulation runs, the robot follows the same ground truth of about 40m. However, as a result of adding random noises into the system, the EKF and odometry trajectories differ from one run to the next, resulting in a sample of possible EKF and odometry trajectories corresponding to the uncertainties associated with proprioceptive and exteroceptive sensors. From this sample, the consistency of the estimator can be computed using Equation 3.18.

Figures 3.10 through 3.17 depict an example of how the EKF and odometry trajectories of a robot change over time in relation to ground truth in a single simulation run with N=5. The ground truth, EKF, and pure odometry trajectories are shown in blue, red and black respectively. The ground truth shows the actual trajectory of the robot. It is calculated using the motion model of the robot *before* adding white Gaussian noise into the system. On the other hand, pure odometry trajectory is calculated from the motion model of the robot *after* adding random white Gaussian noise into the control input given by the covariance matrix $R_t = diag([(0.05\Delta L)^2, (0.05\Delta R)^2])$. EKF trajectory is determined using both the motion and measurement models according to the algorithm presented in Table 3.1. In each simulation run, the initial EKF location (red circle) is randomly initialized around the actual location (blue circle) with the uncertainty given by the initial covariance matrix $\sum_0 = diag([0.15m^2, 0.15m^2, 0.15rad^2])$ in order to reflect the discrepancies[2] between EKF and actual locations of the robot. It can be seen that the odometry estimates start to diverge from the ground truth in the early stages of the robot's journey as seen in Figure 3.10 while the EKF estimates begin to converge towards the ground truth. EKF estimates are significantly better than pure odometry because the Extended Kalman Filter refines odometry estimates through relative landmark observations.

[2]These discrepancies may arise if the robot estimates its initial pose using its own sensors

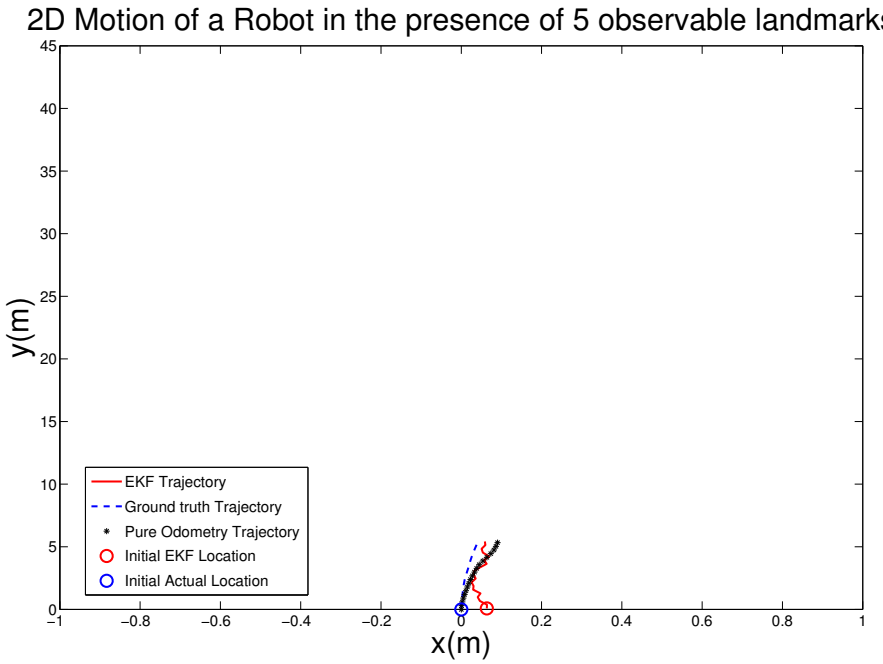


Figure 3.10: Trajectories in one of the simulation runs.

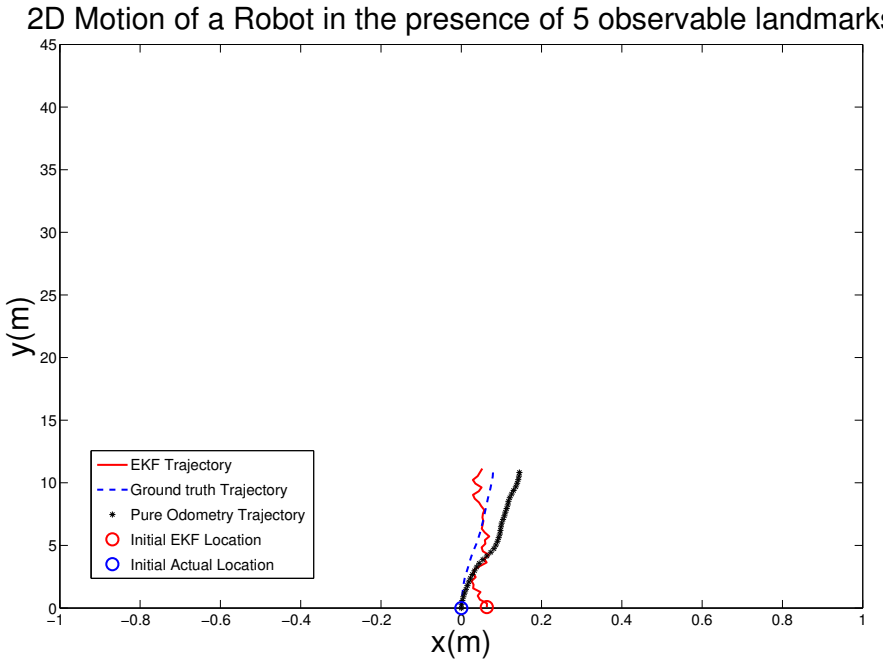


Figure 3.11: Continuation of the trajectories in the same simulation run.

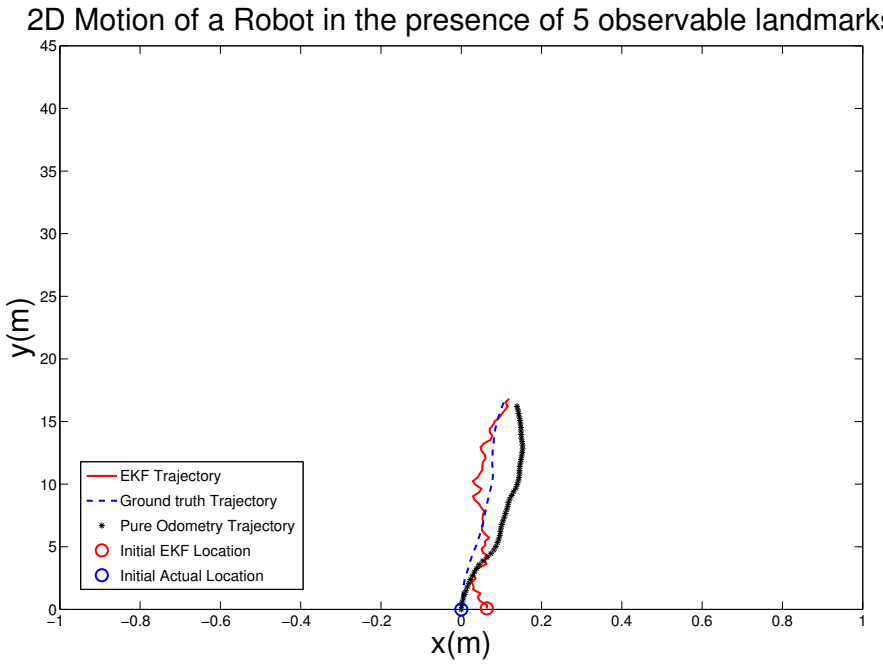


Figure 3.12: Continuation of the trajectories in the same simulation run.

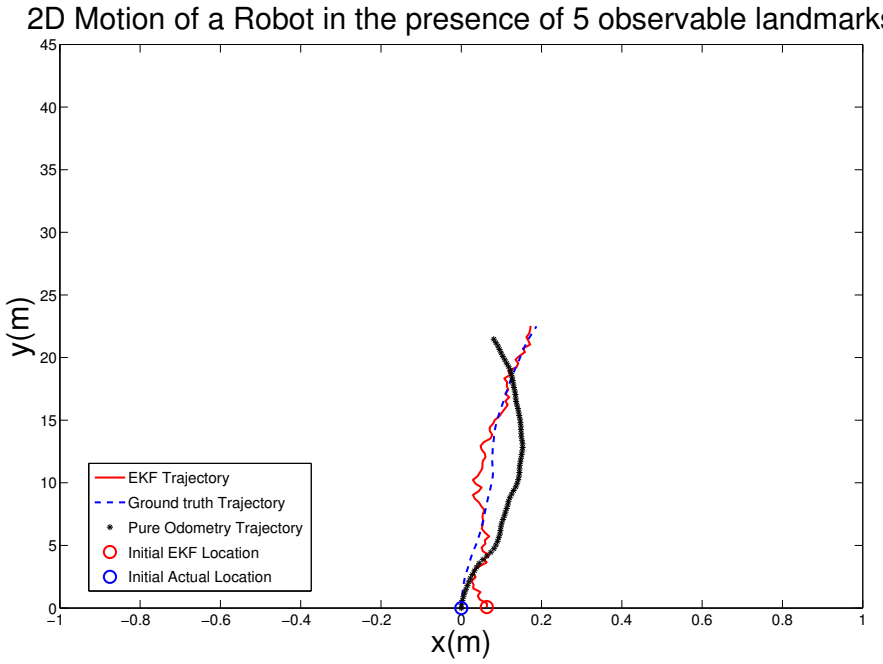


Figure 3.13: Continuation of the trajectories in the same simulation run.

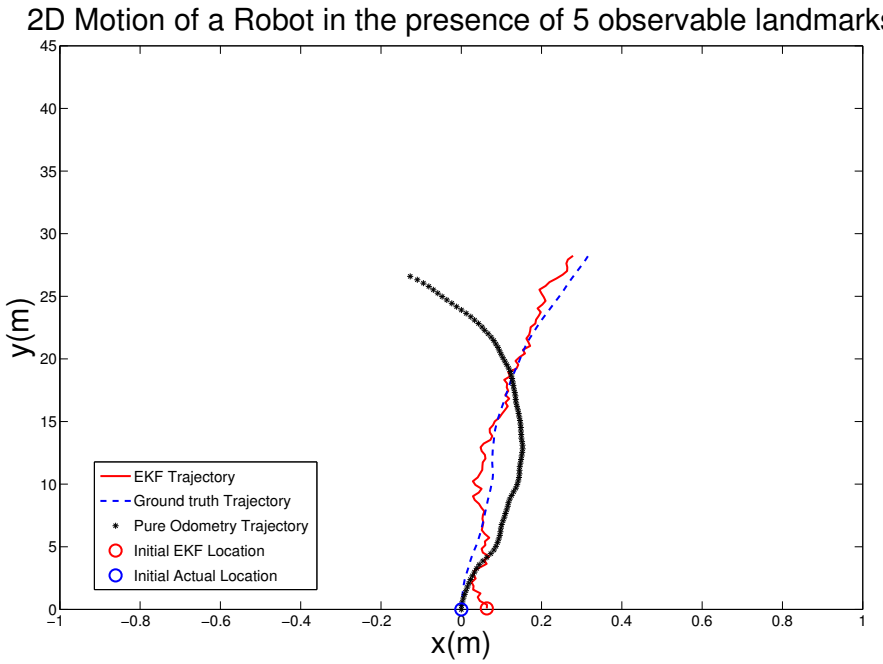


Figure 3.14: Continuation of the trajectories in the same simulation run.

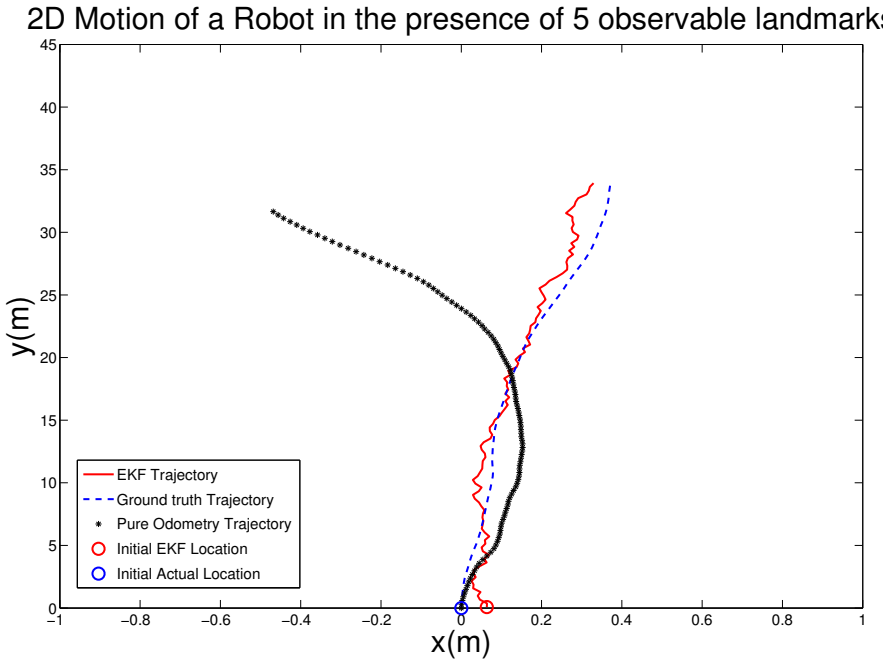


Figure 3.15: Continuation of the trajectories in the same simulation run.

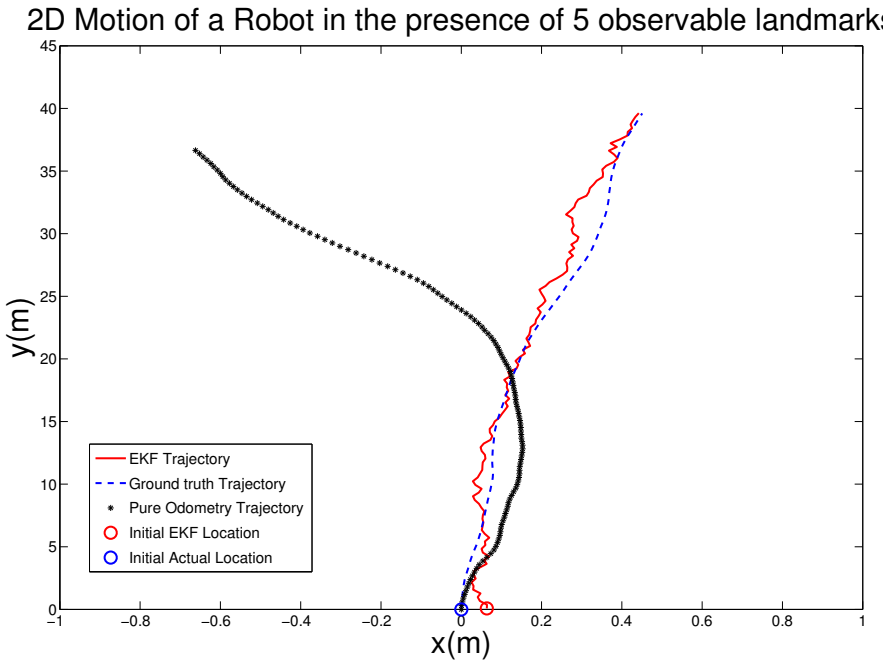


Figure 3.16: Continuation of the trajectories in the same simulation run.

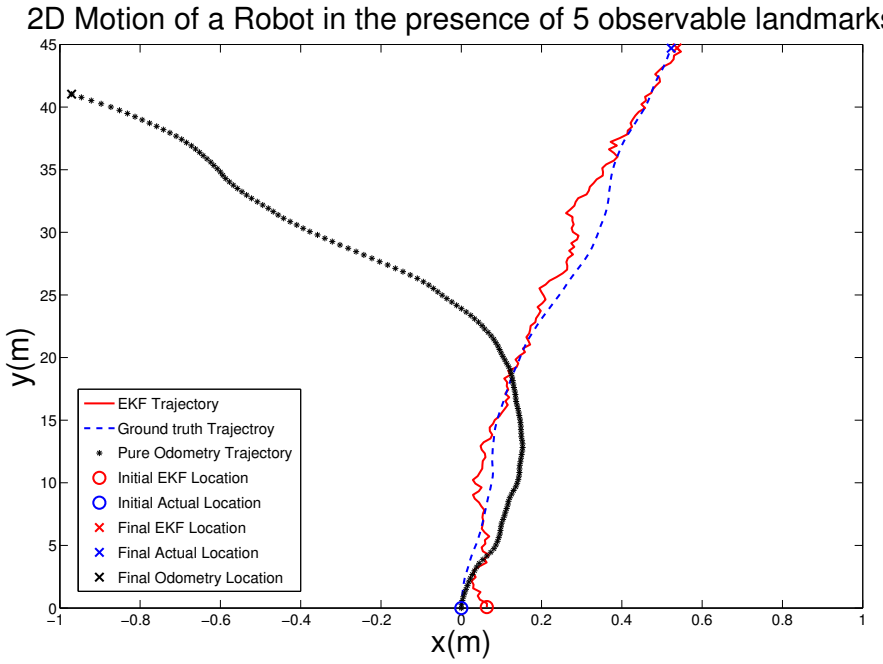


Figure 3.17: Termination of the trajectories in the same simulation run.

The average results for 50 simulation runs are shown in Tables 3.2 and 3.3 for N=1 through N=5. For each run, *Mean Absolute Error (MAE)* at every EKF and Odometry position and orientation were calculated as follows:

$$MAEP_t = \frac{1}{n}\sum_{k=1}^{n} \|\mu_t\_actual(x_k, y_k) - \mu_t(x_k, y_k)\| \tag{3.14}$$

$$MAEO_t = \frac{1}{n}\sum_{k=1}^{n} \|\mu_t\_actual(\theta_k) - \mu_t(\theta_k)\| \tag{3.15}$$

Where:
n is the total number of runs.
$MAEP_t$ is the Mean Absolute Error in Position at time t.
$MAEO_t$ is the Mean Absolute error in Orientation at time t.

The average of $MEAP_t$ and $MEAO_t$ for the entire journey of the robot was calculated as follows:

$$Average(MAEP) = \frac{1}{T}\sum_{t=1}^{T} \|MAEP_t\| \tag{3.16}$$

$$Average(MAEO) = \frac{1}{T}\sum_{t=1}^{T} \|MAEO_t\| \tag{3.17}$$

Where:
T is the total number of iterations (steps) per run.

Additionally, *Average Normalized Estimation Error Squared* ($ANEES_t$) at each position of the robot was calculated using the following equation:

$$ANEES_t = \frac{1}{3n}\sum_{k=1}^{n} (\mu_t\_actual(x_k, y_k, \theta_k) - \mu_t(x_k, y_k, \theta_k))^T \overline{\sum_k}^{-1} (\mu_t\_actual(x_k, y_k, \theta_k) - \mu_t(x_k, y_k, \theta_k)) \tag{3.18}$$

Where n is the total number of simulation runs.

At any time t, an estimator is said to be consistent if its $ANEES_t$ value falls within the Chi Squared bounds at a given probability. The most common probability is 95%. At this probability, the bounds are as follows:

$$\left[\left(1 - \frac{2}{27n} - 1.96\sqrt{\frac{2}{27n}}\right)^3, \left(1 - \frac{2}{27n} + 1.96\sqrt{\frac{2}{27n}}\right)^3\right] \tag{3.19}$$

For $n = 50$, the $ANEES_t$ bounds for consistency are $[0.7865, 1.2387]$

Table 3.2: Simulation results for single robot localization using Pure Odometry

Number of Landmarks	Average Simulation Time (s)	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
1	0.0452	3.9053	0.3021	N/A*
2	0.0671	3.5545	0.2808	N/A
3	0.0887	2.9401	0.3057	N/A
4	0.1093	3.3349	0.2674	N/A
5	0.1339	3.7495	0.2920	N/A

* Odometry does not keep track of Covariances, hence ANEES is not applicable.

Table 3.3: Simulation results for single robot localization using Extended Kalman Filter

Number of Landmarks	Average Simulation Time (s)	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
1	0.0452	0.0636	0.0353	0
2	0.0671	0.0489	0.0264	0
3	0.0887	0.0435	0.0227	0
4	0.1093	0.0382	0.0199	0
5	0.1339	0.0356	0.0178	0

From Tables 3.2 and 3.3 it can be seen that the Mean Absolute Error in both EKF positions and orientations rapidly converges to the smallest possible MAE as the number of observable landmarks around the robot increases (Figure 3.18 and 3.19).

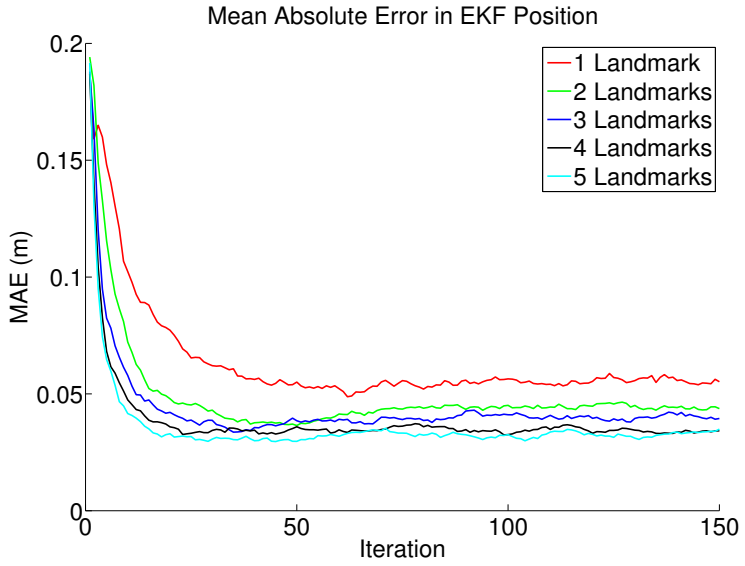


Figure 3.18: Mean Absolute Error in EKF positions for 1, 2, 3, 4 and 5 landmarks around the robot.

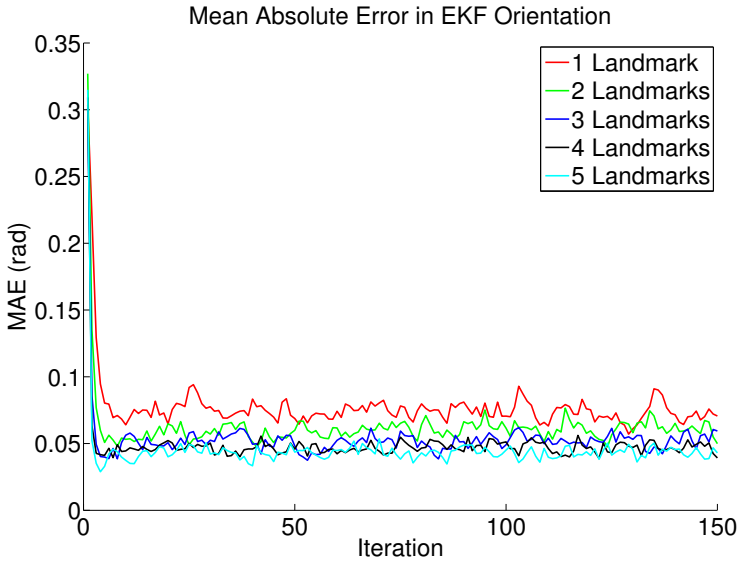


Figure 3.19: Mean Absolute Error in EKF orientations for 1, 2, 3, 4 and 5 landmarks around the robot.

On the other hand, the computation time of the EKF algorithm increases linearly as the number of landmarks, N, increases (Figure 3.20). This is due to the fact that the inner loop of the flowchart in Figure 3.8 is executed N times in each iteration step. Therefore, past a certain number of landmarks (3 landmarks in this example), the improvement in localization accuracy becomes insignificant relative to the added computational cost. This phenomenon is commonly referred to as the rate of diminishing return.

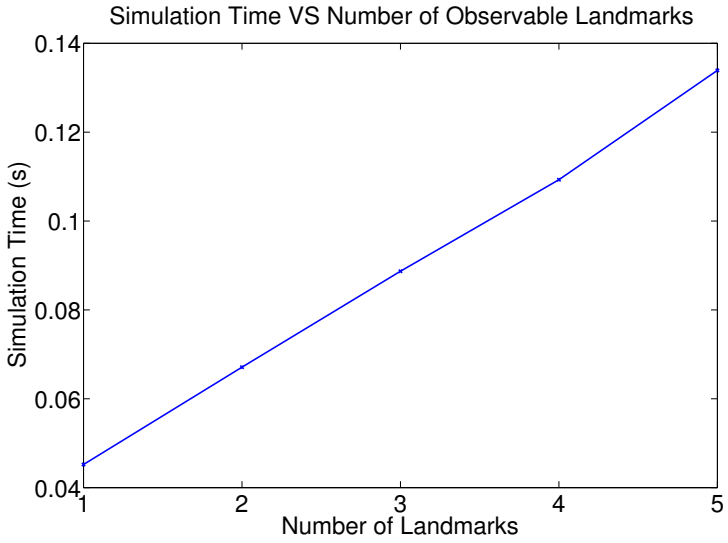


Figure 3.20: Average simulation time versus number of landmarks

Another very important observation that can be made is that the estimator is inconsistent (Figure 3.21), despite using static landmarks for correcting measurements leading to bounded errors. Specifically, in this case the estimator is pessimistic (ANEES smaller than lower bound of the Chi Squared test) because its covariance matrix converges to values larger than the actual covariance matrix. This implies that the estimated standard deviations in estimated poses are larger than the actual standard deviations as shown in Figure 3.22.

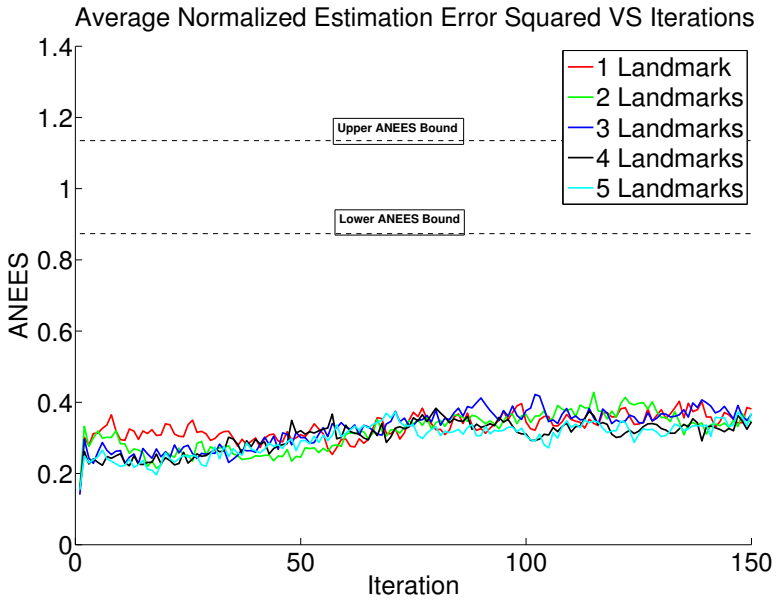


Figure 3.21: Pessimistic Average Normalized Estimation Error Squared.

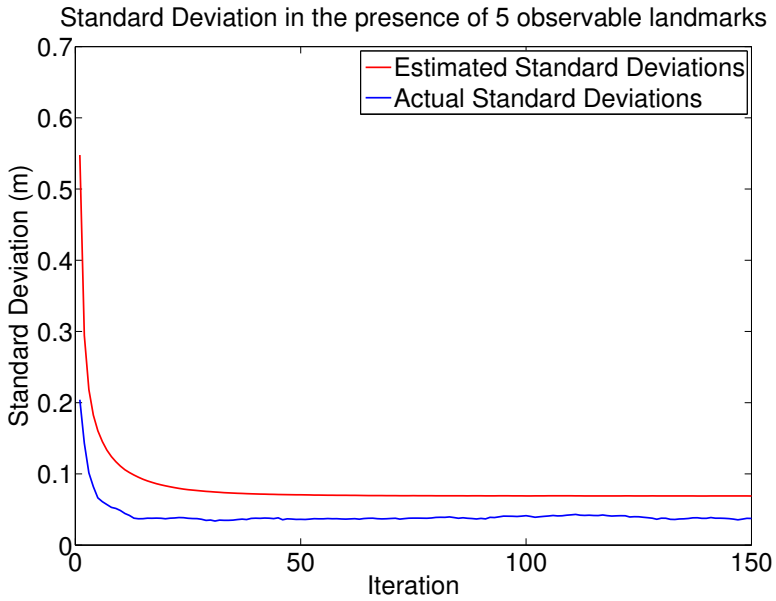


Figure 3.22: Estimated and actual standard deviations in positions of a robot during 5-landmark localization.

In other cases, the estimator could be optimistic (ANEES higher than upper bound of the Chi squared test). These inconsistencies stem forward from the very nature of the Extended Kalman Filter (i.e., independence assumption and approximation of nonlinear systems through linearization) and is a well known fact in the tracking and controls community. Section 3.2 describes how the inconsistent estimator is heuristically tuned in order to obtain consistent estimates.

3.2 Heuristic Tuning of Extended Kalman Filter for Consistent Localization

One of the techniques for tuning an inconsistent Extended Kalman Filter involves careful injection of artificial noise, called the *stabilising noise* [77, 51] into the system in order to offset the effects of linearization of nonlinear systems. However, this method requires substantial understanding of the system behaviour, and does not guarantee that the system will remain consistent over extended periods of time. In the case of localization, Bailey *et.* al. suggest that inconsistencies in Extended Kalman Filter could be compensated by heuristically inflating landmark covariances after each update [77], or through the inflation of sensor covariances [79]. However, it is unclear how to quantify adequate inflation, nor have formal methodologies for determining the correct inflation of the covariances been defined.

Fortunately, in single robot landmark localization, as long as the robot executes non-erratic movements between consecutive relative observations (for example, by avoiding tight/sharp turns), the behaviour of the system can be fairly understood or anticipated. For instance, the graphs for Mean Absolute Error in position and orientation (Figures 3.18 and 3.19), and Average Normalized Estimation Error (3.21) exhibit fairly simple response of the system and therefore the system can be heuristically tuned to achieve the desired response. Thus, this section of the thesis proposes a formal methodology for deducing the magnitude by which the landmark covariance matrix can be artificially inflated in order to achieve consistent EKF pose estimates in single robot localization, and extends the idea to multirobot localization in featureless environments in Section 3.3. The magnitude by which the covariance matrix is inflated will be referred to as the *Covariance Inflation Index*, C.

3.2.1 Inflation of Landmark Innovation Covariance

Recall that during the Correction step of the EKF algorithm, the uncertainty matrix associated with relative measurements (also known as innovation) matrix, S, is given by the following formula:

$$S = H_R \overline{\sum}_t H_R^T + H_L \sum_L H_L^T + Q_t$$

Where:

$$H_R = \frac{1}{q}\begin{bmatrix} -(x_L - \overline{x}_t) & -(y_L - \overline{y}_t) & 0 \\ \frac{(y_L - \overline{y}_t)}{q} & -\frac{(x_L - \overline{x}_t)}{q} & -q \end{bmatrix} ; \; H_L = \frac{1}{q}\begin{bmatrix} (x_L - \overline{x}_t) & (y_L - \overline{y}_t) & 0 \\ -\frac{(y_L - \overline{y}_t)}{q} & \frac{(x_L - \overline{x}_t)}{q} & 0 \end{bmatrix} ; \; Q_t = \begin{bmatrix} (\Delta\rho)^2 & 0 \\ 0 & (\Delta\theta)^2 \end{bmatrix}$$

Suppose the innovation matrix is represented by the following shorthand formula:

$S = S_R + S_L + Q_t$

Where:

$S_R = H_R \overline{\sum}_t H_R^T.$

$S_L = H_L \sum_L H_L^T.$

A closer look at the term S_L reveals that only the first two elements of the leading diagonal of the landmark covariance matrix, $\sum_L$, and their corresponding correlations appear in the term. This is because the third column of the matrix H_L consists of zeroes. In a physical world, this means that the orientation of the landmark does not contribute at all in the propagation of error from the landmark to the robot through relative observation. The following example illustrates this:

Let $H_L = \begin{bmatrix} L11 & L12 & 0 \\ L21 & L22 & 0 \end{bmatrix}$ and $\sum_L = \begin{bmatrix} M11 & M12 & M13 \\ M21 & M22 & M23 \\ M31 & M32 & M33 \end{bmatrix}$

$S_L = H_L \sum_L H_L^T$

$= \begin{bmatrix} L11(L11M11 + L12M21) + L12(L11M12 + L12M22) & L21(L11M11 + L12M21) + L22(L11M12 + L12M22) \\ L11(L21M11 + L22M21) + L12(L21M12 + L22M22) & L21(L21M11 + L22M21) + L22(L21M12 + L22M22) \end{bmatrix}$

In fact, if the third row and column of the landmark covariance matrix, $\sum_L$, were replaced by zeroes (i.e., M31=M32=M33=M13=M23=M33=0), the result would still be the same. Therefore, inflation of the entire landmark covariance matrix prior to propagation would only affect linear components (x and y) of the landmark pose (i.e., M11, M12, M21 and M22) while completely avoiding the angular component (θ) which is the greatest source of error in nonlinear systems. The inflation can be done as follows:

$$\sum_L^* = C\sum_L = \begin{bmatrix} CM11 & CM12 & CM13 \\ CM21 & CM22 & CM23 \\ CM31 & CM32 & CM33 \end{bmatrix} = \begin{bmatrix} CM11 & CM12 & 0 \\ CM21 & CM22 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.20}$$

The new matrix $\sum_L^*$ in Equation 3.20 has a really nice property in that it preserves any correlations between the x and y components of the original landmark covariance matrix $\sum_L$. This can be proven using Pearson Correlation formula [82] as shown below:

Pearson Correlation Coefficient between M11 and M22 before covariance inflation:

$$\rho_{M11,M22} = \frac{M12}{\sqrt{M11}\sqrt{M22}} \tag{3.21}$$

Where $\rho_{M11,M22}$ is the correlation coefficient between M11 and M22 before inflation

Pearson Correlation Coefficient between M11 and M22 After covariance inflation:

$$\rho*_{M11,M22} = \frac{CM12}{\sqrt{CM11}\sqrt{CM22}} = \frac{CM12}{C\sqrt{M11}\sqrt{M22}} = \frac{M12}{\sqrt{M11}\sqrt{M22}} \tag{3.22}$$

Where $\rho*_{M11,M22}$ is the correlation coefficient between M11 and M22 after inflation

From equations 3.21 and 3.22 it can be seen that the correlation coefficients between M11 and M22 (i.e., between x and y) does not change after multiplying the original landmark covariance matrix by the Covariance Inflation Index C. Geometrically, C affects the size of the uncertainty ellipses around the landmark positions, but not their shapes. The next section revisits the simulations for single robot localization in the presence of 5 landmarks and demonstrates how to achieve consistency in EKF estimates by adjusting C.

3.2.2 Improving the consistency in Single Robot Localization

Figure 3.23 shows the ANEES for 50 simulation runs before inflating the landmark covariance matrices. In all simulations, the robot executes non-erratic motions along a predefined path similar to Figures 3.10 through 3.11 and uses 5 landmarks to correct its estimates. The figure shows that the original EKF is pessimistically inconsistent since the ANEES is always lower than the lower bound of the double sided Chi Squared test at 95% Probability.

Figure 3.24 shows the ANEES for 50 simulation runs with inflated landmark covariance matrix in the correction step of the EKF algorithm according to Equation 3.20. For every set of 50 simulation runs, a different value of C was used until the best results were obtained. The figure shows that the system becomes fairly consistent when the Covariance Inflation Index, C, is about 20. The same methodology can be applied when the robot uses any other number of landmarks to correct its position.

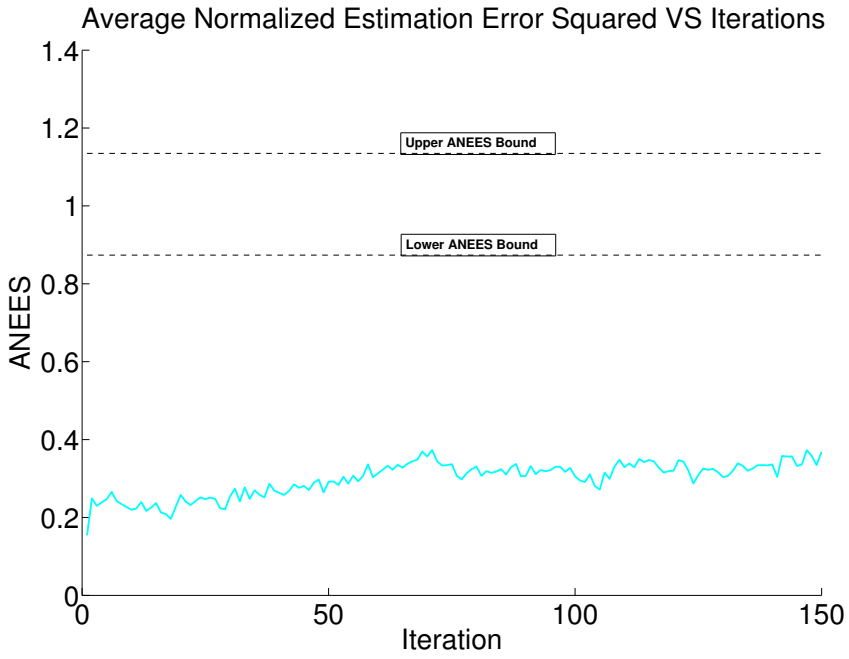


Figure 3.23: Average Normalized Estimation Error for a 5-landmark localization before inflation of landmark covariance matrix.

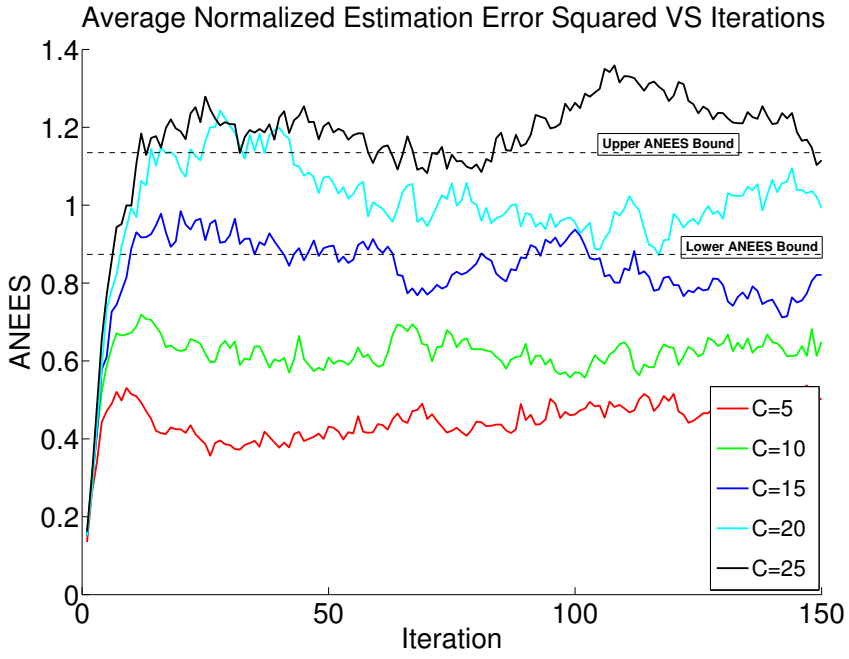


Figure 3.24: Average Normalized Estimation Error for a 5-landmark localization after inflation of landmark covariance matrix.

Table 3.4: Simulation results for single robot localization before inflation of landmark covariance matrix.

Number of Landmarks	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
5	0.0345	0.0179	0

Table 3.5: Simulation results for single robot localization after inflation of landmark covariance matrix with C=25.

Number of Landmarks	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
5	0.1965	0.0296	81.3

The results in Tables 3.4 and 3.5 show that the proposed heuristic method of inflating the landmark covariance matrix helps to improve the consistency of the system. However, due to the very nature of the Extended Kalman Filter and its underlying assumption that non-linear systems can be linearized, the method proposed here would work best in cases when the change in orientation of the robot, θ, is small between consecutive observations in order to minimize the degree of nonlinearity. In the physical world, this means that the robot would have to avoid making sharp turns between consecutive observations by keeping the velocities (or displacements) of its driving wheels close to each other between observations. Although this constraint seems impractical for small, indoor robots, which have to make sharp turns in order to avoid obstacles, the proposed tuning method could potentially be applied to consistently localize a small group of larger, outdoor robots traversing across an open field in a more or less straight line motions, thereby minimizing the risk of collisions between nearby robots. Section 3.3 describes how a group of robots can be consistently localized using a heuristically tuned Extended Kalman Filter.

3.3 Multirobot Localization using Extended Kalman Filter

The localization process described in Section 3.1 is only possible if the mobile robot has prior information, such as locations of fixed landmarks, about the environment. The prior knowledge about the environment is usually in the form of a map. Upon successful identification of a landmark in an environment, the robot uses the information about the landmark, stored in a map, in order to correctly localize itself. However, in many practical cases, for example, in uncharted areas, the map of the environment is not available to the robot beforehand. In this situation, the robot must incrementally build its own map, and at the same time use this map to localize itself. This problem is known as SLAM. Unfortunately, a map of an environment can only be built if there are sufficient observable and distinguishable features in the environment. In large, open areas, or in highly symmetrical indoor environments (such as hallways), mapping becomes difficult. Kurazume *et al.* [58] first introduced the concept of mobile landmarks in 1994, whereby a team of robots is divided into two groups, say A and B. As group A moves, group B acts as landmarks to group A, and vice versa. Since then, the concept of mobile landmarks has been studied several times in centralized [60, 61, 58, 62, 63] and decentralized [67, 68, 13, 69, 70, 71, 65, 14] multirobot systems. As mentioned in Chapter 2, centralized multirobot systems tend to be consistent at a higher computational and communication cost, whereas decentralized multirobot systems are generally inconsistent but with lower computational and communication cost. Studies have shown that in multirobot systems at least one robot should possess absolute positioning capabilities (such as GPS or static landmark) in order to have bounded error in the estimated pose of each robot in a group [67]. Therefore, in GPS-denied and featureless environments, the main goal of such systems is to reduce the rate at which the error of estimated pose of each robot in the system grows, while trying to keep the estimates consistent.

This section describes a distributed approach to localization of multiple, networked mobile robots using a heuristically tuned version of the Extended Kalman Filter introduced in Section 3.1.3. In this case however, fixed landmarks are unavailable, and instead, artificial geometric landmarks, such as cylinders, are mounted on the robots as shown in Figure 3.25. Therefore, the position, $[x_L, y_L]^T$, and the associated uncertainty. $\sum_L$, of each landmark is given by the estimated position, $[x_t, y_t, \theta_t]^T$, and uncertainty, $\sum_t$, of the carrier robot respectively. In a typical scenario, as one of the robots moves from one location to another, the rest remain stationary. At its new location, the robot that had just moved to a new location uses its exteroceptive sensors to improve its location through relative observations of stationary robots. The position and associated uncertainty of each observed stationary robot (i.e., position and uncertainty of *portable landmarks*) are transmitted to the moving robot over a network as depicted in Figure 3.26. Alternatively, the stationary robots can estimate their own poses relative to the moving robot using their own exteroceptive sensors as will be described in Section 4.1.7.

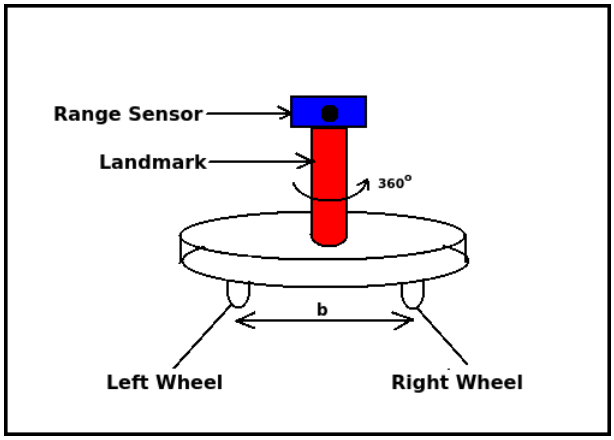


Figure 3.25: Geometric landmark mounted on a mobile robot

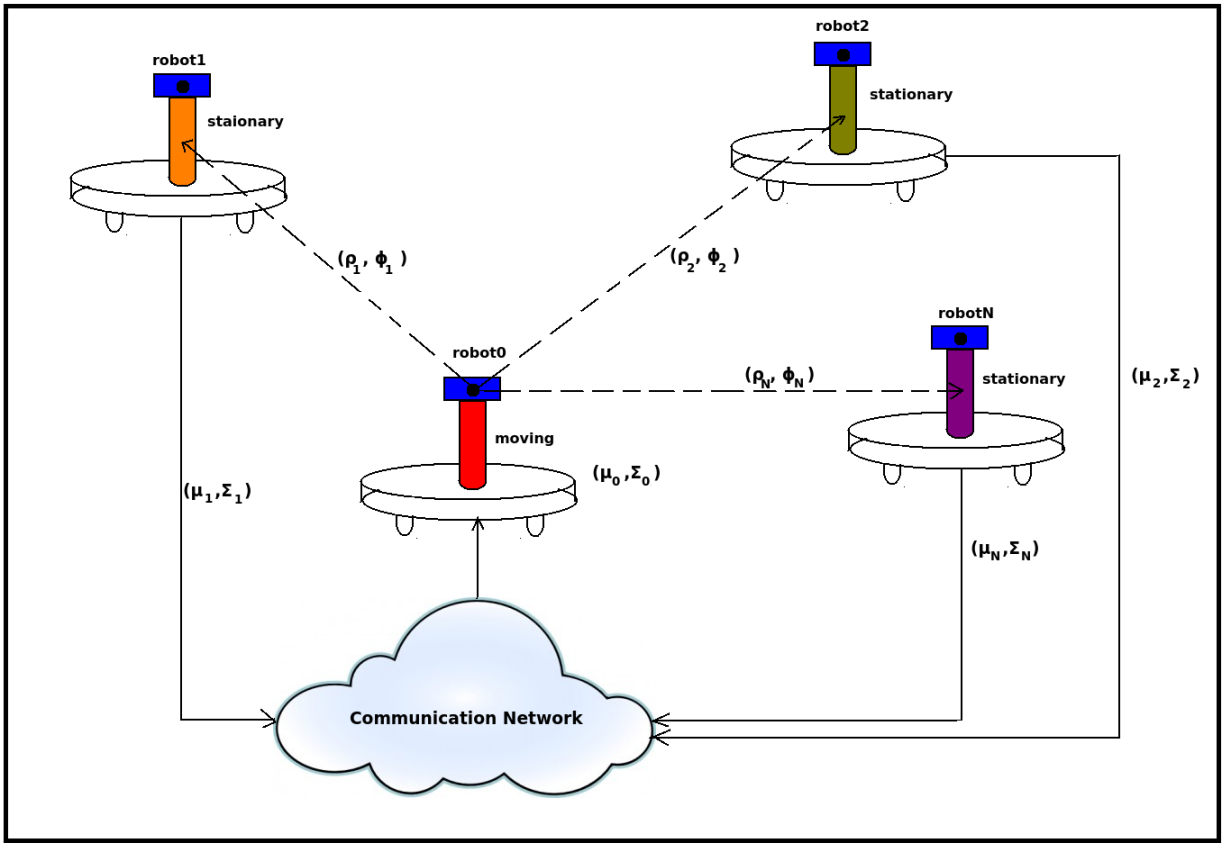


Figure 3.26: Relative observations and exchange of poses (μ) and uncertainties ($\sum$) over a network.

3.3.1 Simulation of Multirobot Localization using Standard EKF Algorithm

The simulator described in Section 3.1.3 was used to track the actual (μ_t_actual), odometer ($\mu_t\_odometry$), and EKF poses (μ_t) along with their associated covariances ($\sum_t$) for each robot in the group. In this case however, and unlike the situation described in Section 3.1.3 where static landmarks were assumed available, the simulator now uses the latest EKF poses and associated uncertainties of stationary robots, which act as landmarks, to calculate the innovation covariance matrix, S, in the Correction Step of Table 3.1. The rest of the parameters, and assumptions remain unchanged. Additional assumptions include:

1. The geometric landmark mounted on each robot is located at the center of the axis joining the two wheels.
2. All robots in a team are identical with the same motion and measurement models.
3. The initial belief (EKF pose and associated uncertainty) of each robot is randomly initialized around the actual ground truth of the robot using the initial covariance matrix. This is done in order to reflect the discrepancies, which may arise if the robots have to use their own sensors to estimate their initial poses, between ground truth and EKF estimates.
4. Robots move within sensing and communication range from each other.

50 Simulation runs were performed for 2, 3, 4 and 5 robots before and after artificial inflation of covariance matrices. In each simulation run, the robots execute non-erratic motions, one at a time, along their predefined ground truths in an open area, while more or less maintaining the distance of separation between them in order to be within sensing distance from each other at all times and avoid collisions (Figure 3.27) until each robot travels for about 45m. The order in which the robots move is random, but each robot must move only once before every other robot in the team gets a chance to move. Randomization of the order in which the robots move helps to counter the problem of overfitting which could happen if the robots moved in a specific order. For each robot, the ANEES is calculated using Equation 3.18 whereas MAE in position and orientation was calculated using the equations 3.14 and 3.15 respectively.

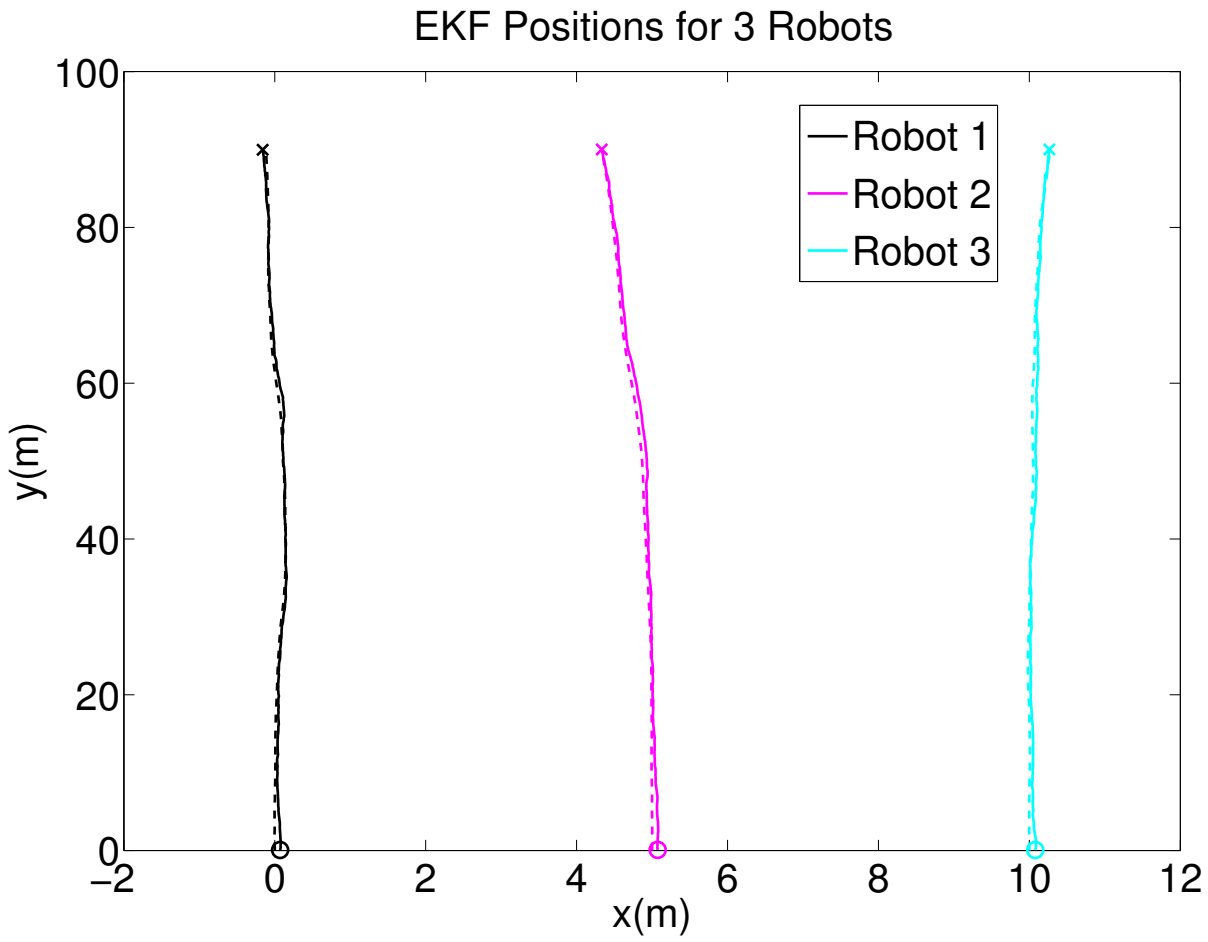


Figure 3.27: Example of a non-erratic 2D Localization of 3 communicating robots before inflation of the covariance matrices. Solid lines are the EKF trajectories whereas broken lines are the ground truths. The circles and crosses represent the robots' start and end locations respectively.

Tables 3.6 through 3.9 show the average results for 50 simulation runs before inflation of covariance matrices.

Table 3.6: Simulation results for 2-robot localization before inflation of covariance matrices.

Robot ID	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
1	0.8744	0.0523	0
2	0.8784	0.0524	0

Average Simulation time (s): 0.3909

Table 3.7: Simulation results for 3-robot localization before inflation of covariance matrices.

Robot ID	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
1	0.4700	0.0405	0
2	0.4305	0.0403	0
3	0.4647	0.0408	0

Average Simulation time (s): 0.8114

Table 3.8: Simulation results for 4-robot localization before inflation of covariance matrices.

Robot ID	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
1	0.3484	0.0350	0
2	0.3149	0.0345	0
3	0.3315	0.0352	0
4	0.3527	0.0346	0

Average Simulation time (s): 1.3047

Table 3.9: Simulation results for 5-robot localization before inflation of covariance matrices.

Robot ID	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
1	0.3412	0.0316	0
2	0.3131	0.0318	0
3	0.3102	0.0311	0
4	0.3243	0.0318	0
5	0.3645	0.0314	0

Average Simulation time (s): 1.8919

Figures 3.28 through 3.31 show the MAE and ANEES for 2, 3, 4 and 5 robot localization respectively.

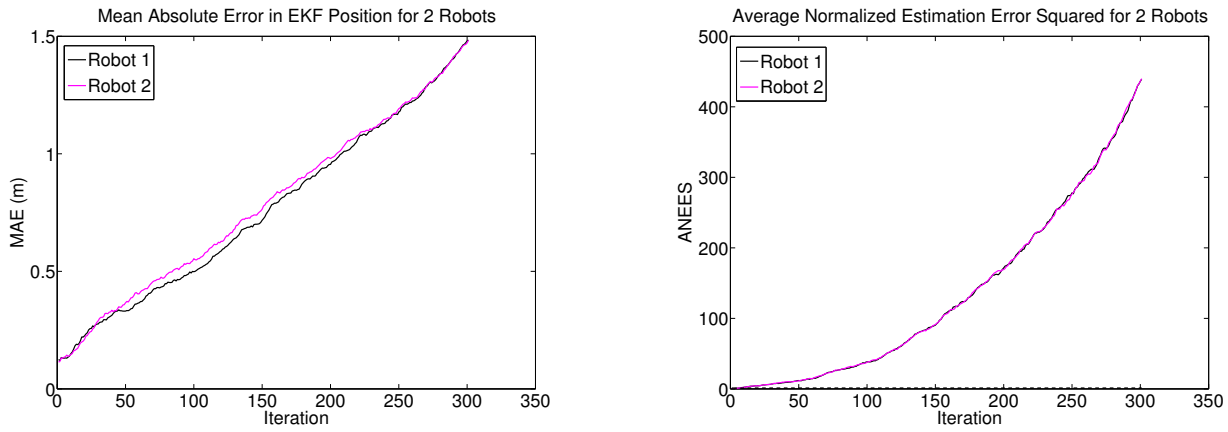


Figure 3.28: MAE (left) and ANEES (right) for 2-robot localization.

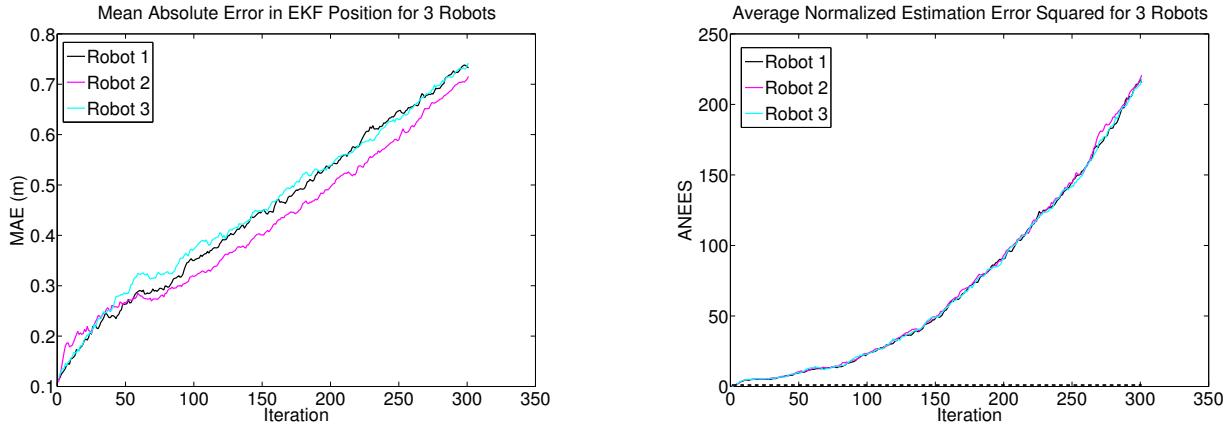
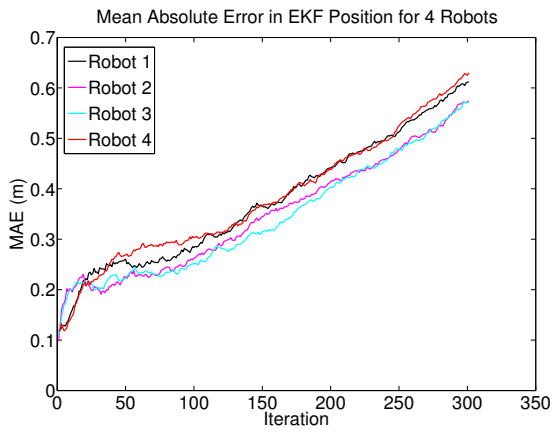


Figure 3.29: MAE (left) and ANEES (right) for 3-robot localization.



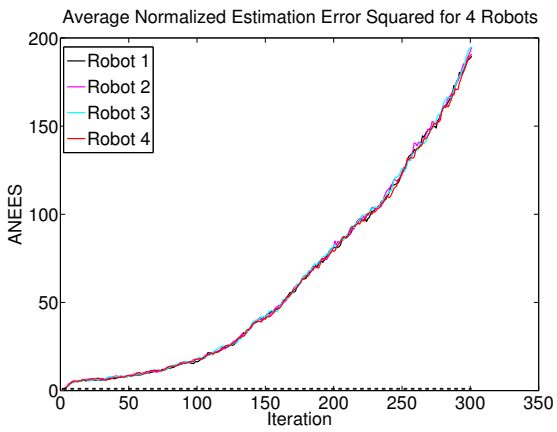
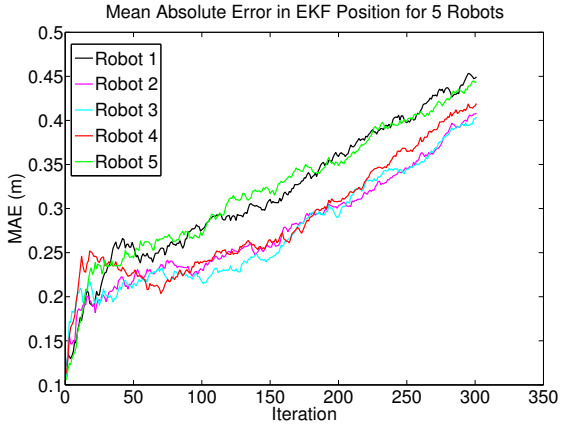


Figure 3.30: MAE (left) and ANEES (right) for 4-robot localization.



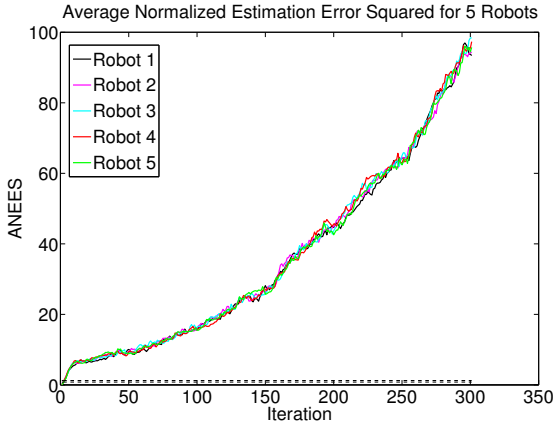


Figure 3.31: MAE (left) and ANEES (right) for 5-robot localization.

The following conclusions can be drawn from the results in Tables 3.6 through 3.9 and Figures 3.28 through 3.31 suggest the following:

1. Both MAE and ANEES for all robots in all group sizes are unbounded. This is because none of the communicating robots has absolute localization capabilities via static landmarks and/or GPS.

2. The rate at which MAE and ANEES grow decreases as the number of robots increases (evident from the y-axis scale).

3. Estimated poses of each robot in all group sizes are inconsistent. In this case, the estimates are overly optimistic because the actual standard deviations of the estimated poses grow without bounds, while the corresponding estimated standard deviations converge to wrong values (Figure 3.32).

4. Computation time increases linearly as the number of robots, N, increases. This is because the same algorithm used to localize a single robot in the presence of static landmarks was used to localize multiple robots.

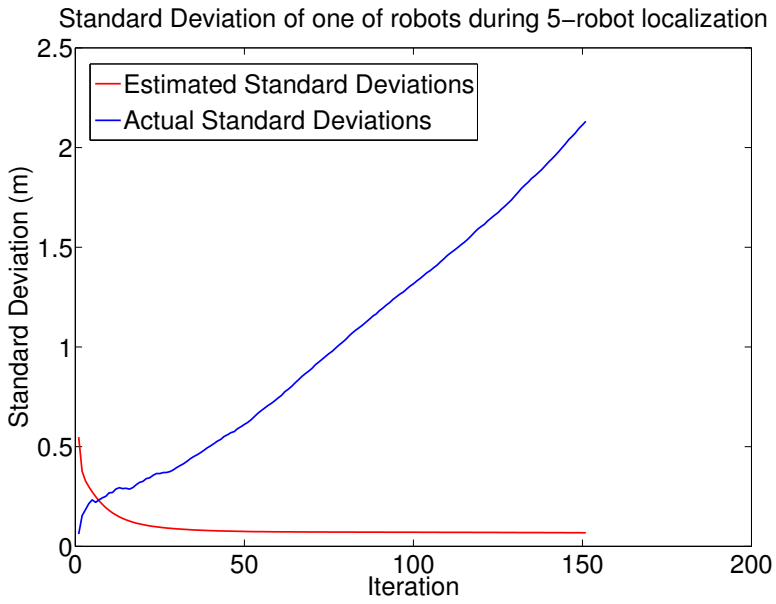


Figure 3.32: Estimated and actual standard deviations in positions of one of the robots during 5-robot localization.

In order to make the estimates consistent, the covariance matrices of the stationary robots will be inflated as described for single robot localization in Section 3.2. This time however, the Covariance Inflation Index C has to steadily increase since the errors associated with the estimated positions of the robots are unbounded (unlike the case of a single robot localization using static landmarks). Through various simulations, the Covariance Inflation Index for stationary robots that act as landmarks was heuristically determined to be of the form:

$$C = AD_t \tag{3.23}$$

Where:
D_t is the distance, determined from the wheel encoders, covered by the robot up to time t.
A is experimentally determined constant.

The next section describes the procedure for improving consistency in multirobot localization through inflation of covariance matrices of the stationary robots.

3.3.2 Improving Consistency in Multirobot Localization through Heuristic Tuning of EKF Algorithm

.

Case I: Moving One Robot at a Time

The methodology described here is for 5 robots moving in such a way that only one robot is allowed to move at a time, while the remaining robots remain stationary. For each set of 50 simulation runs, a different value of A is used until the desired results are achieved. Simulations show that by increasing or decreasing the value of A, the ANEES curves for all robots shift down or up accordingly. Therefore, the ANEES curves can be shifted into the consistent (black broken lines) region simply by adjusting the value of A as shown in Figures 3.33 through 3.36.

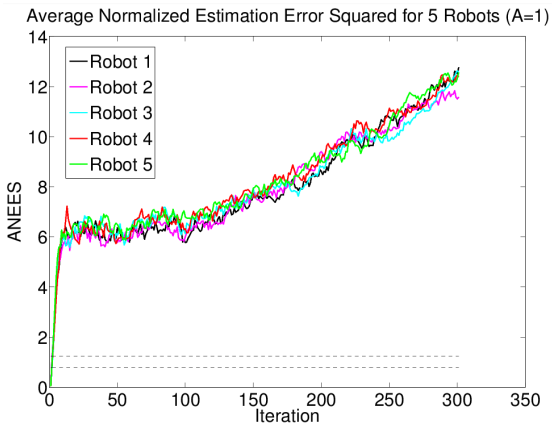


Figure 3.33: ANEES of 5 robots, one moving at a time, when A=1.

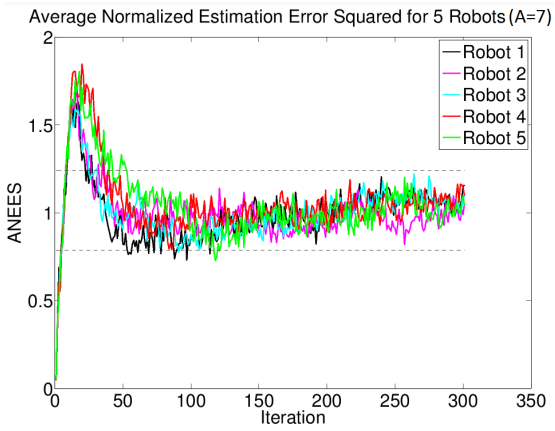


Figure 3.34: ANEES of 5 robots, one moving at a time, when A=7.

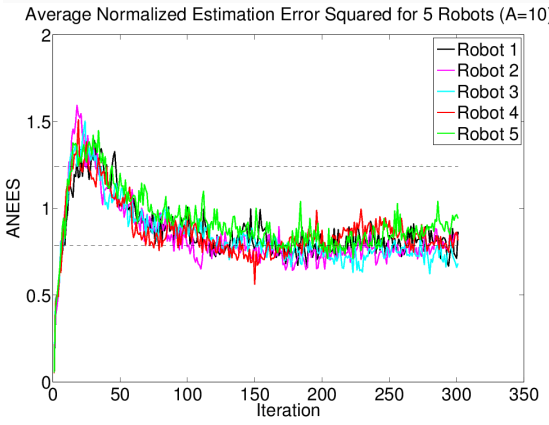


Figure 3.35: ANEES of 5 robots, one moving at a time, when A=10.

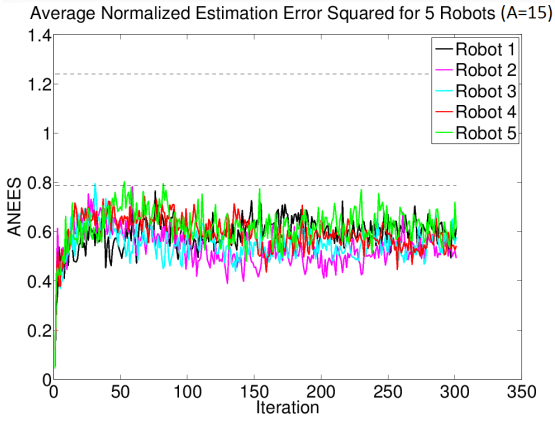


Figure 3.36: ANEES of 5 robots, one moving at a time, when A=15.

Figures 3.33 through 3.36 show that the best ANEES results were obtained when A=7. With this value, the corresponding MAEP, MAEO and ANEES for each robot are shown in Table 3.10.

Table 3.10: Case I: Simulation results for 5-robot localization after inflation of covariance matrices with A=7.

Robot ID	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
1	0.2365	0.0339	89.7010
2	0.2415	0.0345	91.0299
3	0.2433	0.0343	92.3588
4	0.2435	0.0348	89.3688
5	0.2421	0.0339	84.0532

Case II: Moving More than One Robot at a Time

The results in Figures 3.33 through 3.36 and Table 3.10 were obtained when only one robot was allowed to move at each step, while the rest of the robots remained stationary. In a more general case, one or more randomly chosen robots are allowed to move at each step while at least one other robot remains stationary. In other words, the number of robots that move, and the number of robots that remain stationary is random. Figures 3.37 through 3.40 depict the ANEES for 5-robot localization when, at each step, a random number of robots between 1 and 4 inclusive are allowed to move while the rest of the robots remain stationary.

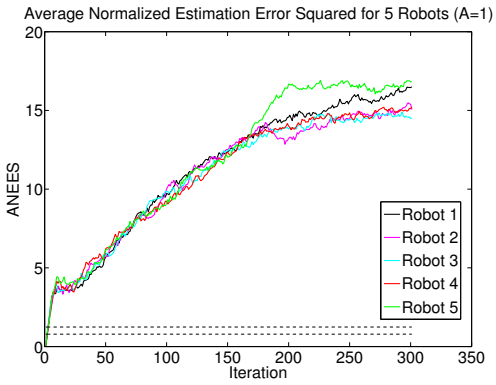


Figure 3.37: ANEES of 5 robots, at least one stationary, when A=1.

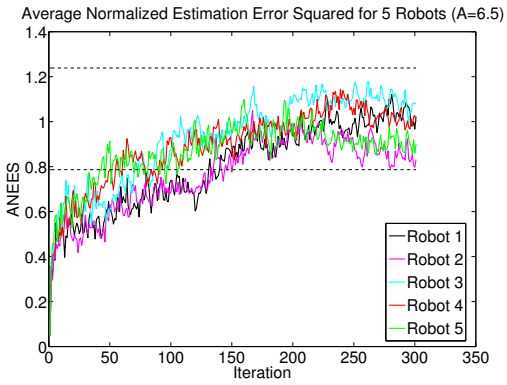


Figure 3.38: ANEES of 5 robots, at least one stationary, when A=6.5.

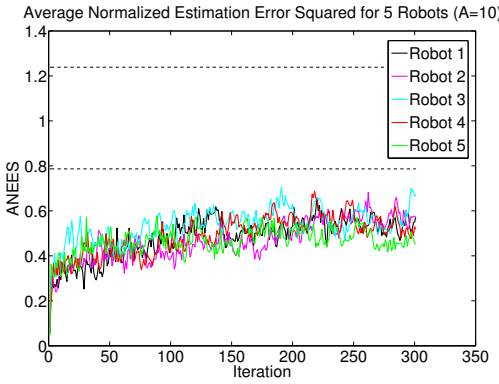


Figure 3.39: ANEES of 5 robots, at least one stationary, when A=10.

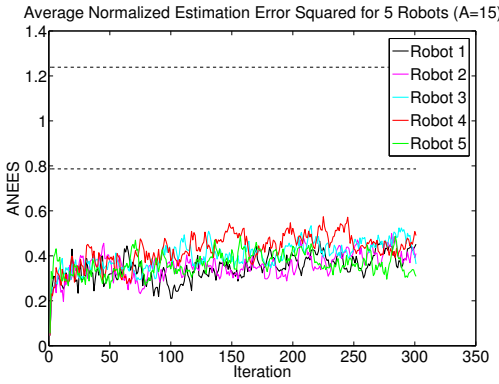


Figure 3.40: ANEES of 5 robots, at least one stationary, when A=15.

Table 3.11 shows the MAEP, and MAEO at the best ANEES (when A=6.5 as shown in Figure 3.38) for each of the 5 robots when more than one of the robots was allowed to move at each step. When compared with Table 3.10, it can be seen that the error associated with estimated positions and orientations for each robot, given by MAEP and MAEO respectively, is higher. Furthermore, there is significant reduction in the percentage of consistent ANEES for each robot when more than one robot is allowed to move. These results suggest that the best outcome is achieved when only one robot is allowed to move, while the rest remain stationary (i.e., Case I).

Table 3.11: Case II: Simulation results for 5-robot localization after inflation of covariance matrices with A=6.5.

Robot ID	Average(MEAP) (m)	Average(MEAO) (rad)	Consistent ANEES (%)
1	0.3315	0.0427	73.0897
2	0.3451	0.0411	43.8538
3	0.3233	0.0459	51.8272
4	0.3435	0.0482	77.0764
5	0.3512	0.0441	69.1030

The following is a summary of the advantages and disadvantages of the proposed consistent, multirobot localization technique using artificially inflated covariances.

Advantages

1. Computation time increases linearly as the number of robots grows.
2. Easy to tune, by trial and error, the amount by which the magnitude of the covariance matrices of the stationary robots can be inflated.
3. The Mean Absolute Errors of each robot in a distributed system after tuning (Table 3.10) are lower than the Mean Absolute Errors before tuning (Table 3.9).

Disadvantages

1. Cannot cope with distributed multirobot systems with high degree of nonlinearity. This stems from the fact that the Extended Kalman Filter performs poorly in such systems due large errors incurred through linearization of such systems. Therefore, the tuning technique proposed in this thesis would work best in scenarios whereby the robots in the system are not subjected to erratic motions.
2. The Covariance Inflation Index is specific to a system, and therefore the EKF has to be retuned every time the parameters of a system, such as the number of robots or types of sensors, change.

3.4 Chapter Summary

This chapter presented a formal heuristic tuning methodology of the classic Extended Kalman Filter through artificial inflation of landmark covariance matrices and demonstrated the effectiveness of tuning methodology, through simulations, in improving the consistency of EKF estimates in single robot as well as multirobot localization. The simulation results show that the proposed tuning methodology significantly improves the consistency in EKF at no additional computation or communication costs.

Chapter 4

Experimental Implementation of the Proposed Multirobot Localization Algorithm

This chapter presents the real-world implementation and evaluation of the proposed distributed, consistent, EKF algorithm using two custom-built mobile robots. Section 4.1 describes the platforms, sensors, and various electronic and mechanical components of the robots. Section 4.2 describes different parts of the software that controls the robots. Section 4.3 covers the real world experiments on localization using the two robots and presents the results obtained in the experiments.

4.1 Hardware Design

The two robotic platforms used in the experiments were built by the author of this thesis using some of the latest electronic components available on the market. The physical design of the platforms, as well as connection of various components was inspired by the earlier generation of the platforms built by 4th year undergraduate students for their final year project in 2007. The following sections describe each of the main hardware components in more detail.

4.1.1 Base Chassis

The chassis of the platform comes from an autonomous lawn mower called Robomow RL500 [83]. All of the original components from the first generation of the platform were removed, leaving only the chassis and driving motors as shown in Figure 4.1. A different electric circuit was installed as seen in Figure 4.2

Figure 4.1: Empty chassis of RL500.

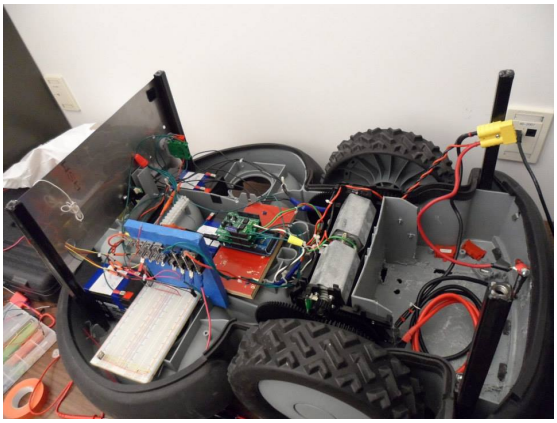


Figure 4.2: RL500 chassis with a new electric circuit.

The chassis has the wheel configuration for which the motion model of the Extended Kalman Filter was derived (i.e., two wheels and a front caster in Figure 3.5). Therefore, the EKF algorithm derived in Chapter 3 can be implemented on the two real robots without any modifications to the motion model. Each wheel is driven by a 24V, 150W DC motor. The two motors control the speed and direction of the robot through differential steering.

4.1.2 Microcontroller

The microcontroller used is the Arduino Mega 2560 [84]. The main functions of the microcontroller are:

1. Steer the robot differentially. This means that the speed and direction of the platform is controlled by differences in angular velocities of the driving motors. For example, if the right wheel rotates faster than the left wheel when the robot moves forward, the robot will turn left, and vice versa. The microcontroller controls the speeds of the motors through Pulse Width Modulation (PWM), and the direction of rotation through Megamoto H-bridges [85].
2. Estimate the distance traveled by robot by counting the number of revolutions each wheel makes with the help of optical wheel encoders described in Section 4.1.7.
3. Rotate the turret motor.
4. Send estimated distances to, and receive commands, such as PWM values for the motors, from a computer via usb cable.

4.1.3 Aluminium Roof

The aluminium roof is supported by four square steel posts as shown in Figure 4.3. The roof provides the surface for mounting sensors, portable landmarks, and a laptop.

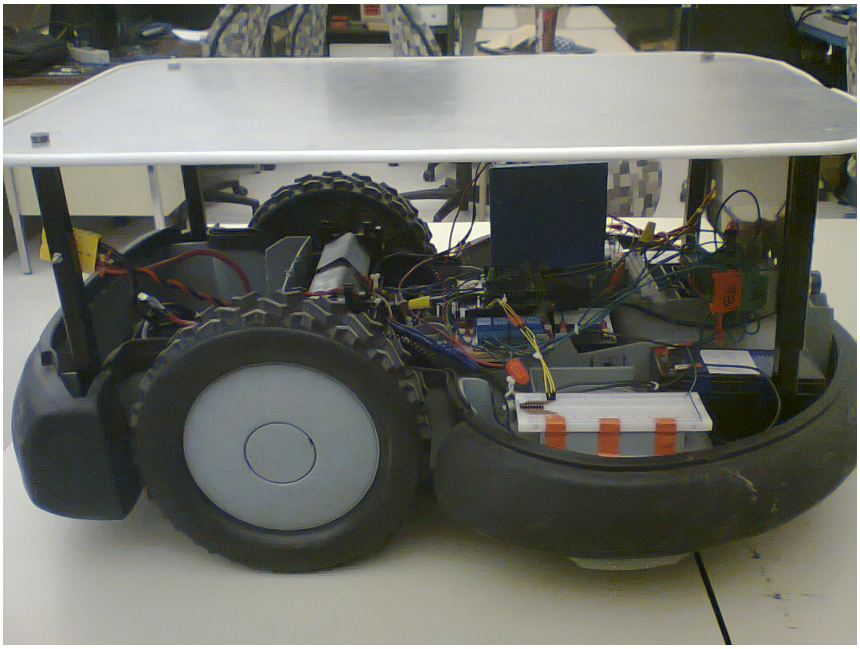


Figure 4.3: Platform with Aluminium roof.

4.1.4 Power Supply

The robot's power supply, shown in Figure 4.4, consists of three small 12V 7Ah DC batteries (B1, B2 and B3), and one large 12V 70Ah DC battery (B4). Two of the small batteries (B1 and B2) are connected in series to provide 24V to each of the driving motors, while the third small battery (B3) provides regulated 12V to a microcontroller, and regulated 5V to turret motor, optical sensors for the wheel encoders, and H-bridges. The large battery powers an onboard laptop and a Microsoft Kinect sensor through via a DC-AC converter. All batteries are connected in such a way that when the robot is completely powered off, the batteries can easily be recharged via a system of relay switches that select the batteries to be charged.

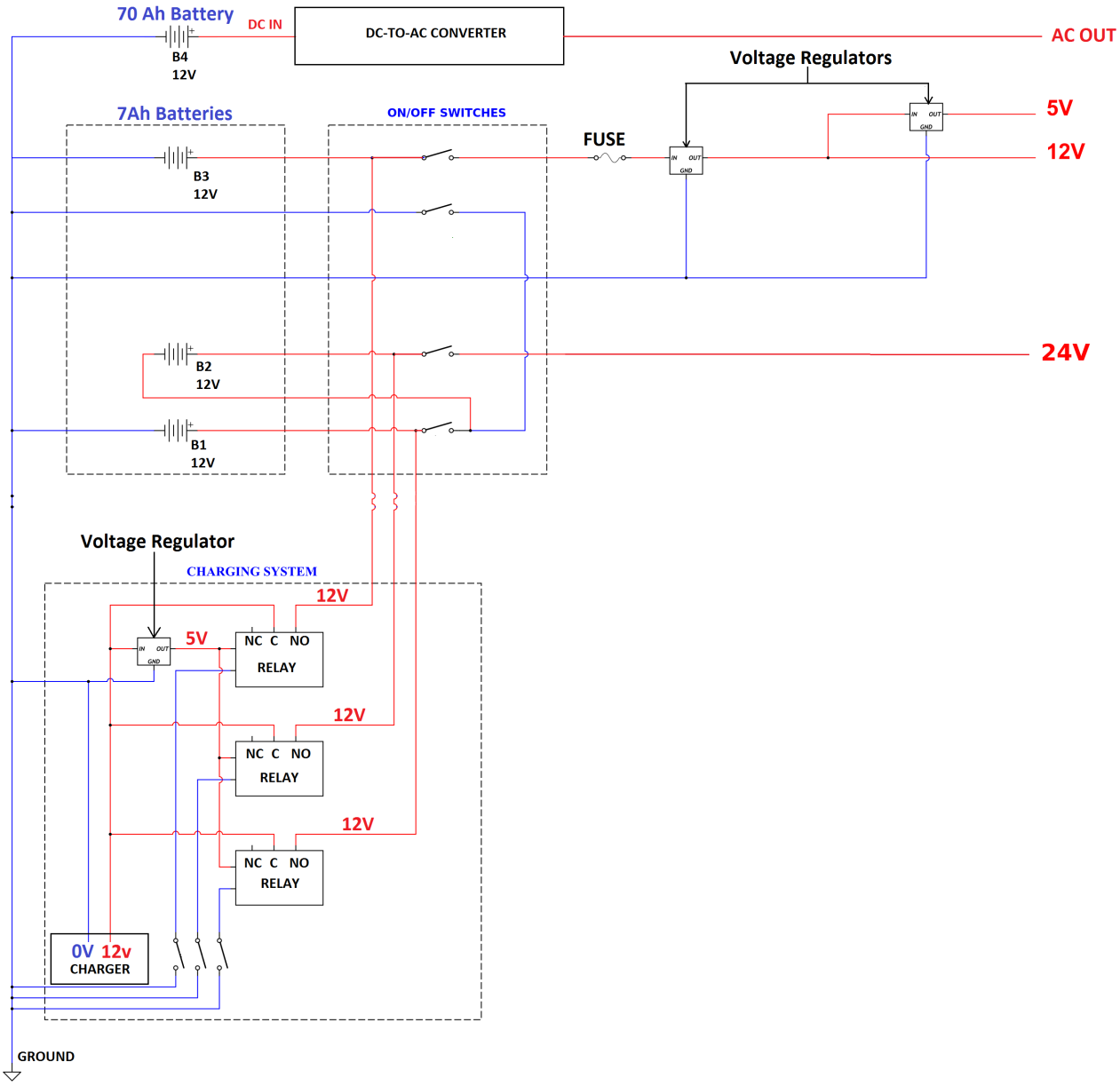


Figure 4.4: Power Supply.

4.1.5 Turret

The turret consists of a base and a rotating plate (Figure 4.5). The base of the turret houses a servo motor (Figure 4.6) and is fastened onto the aluminium roof of the robot by means of screws (Figure 4.7). A plexiglass system used to hold different sensors is screwed onto the rotating plate as shown in Figure 4.8. The whole turret assembly is shown in Figure 4.9. The servo motor provides sufficient torque to turn the assembly by 180 degrees. This helps the mounted sensors scan a wider field of view in order to locate landmarks, and in turn, provides the robot the flexibility to scan for multiple landmarks from any direction the robot faces without having to reorient itself.

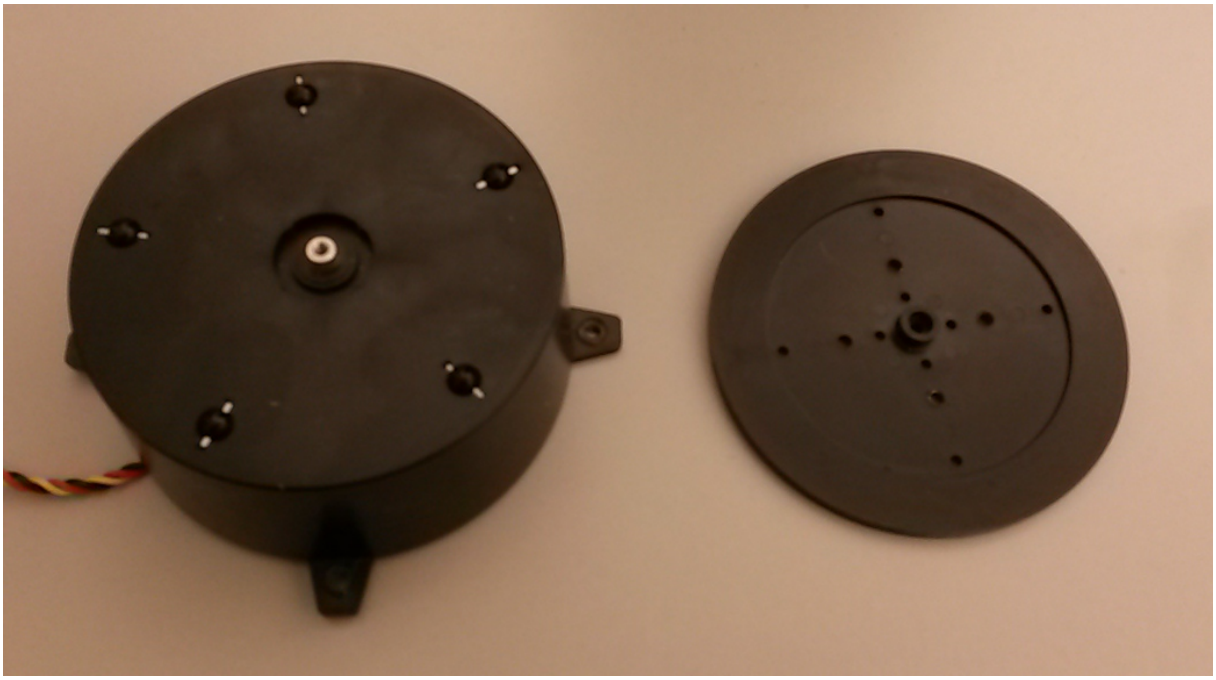


Figure 4.5: Disassembled turret consisting of a base (left) and a rotating plate (right).

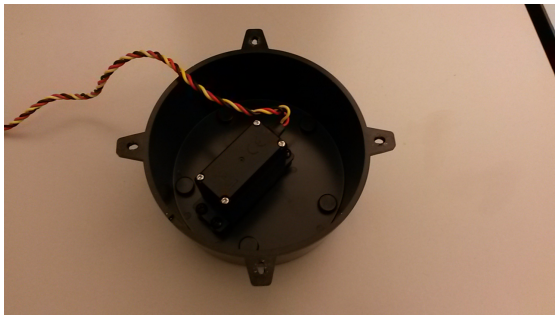


Figure 4.6: Servo motor inside the turret base.

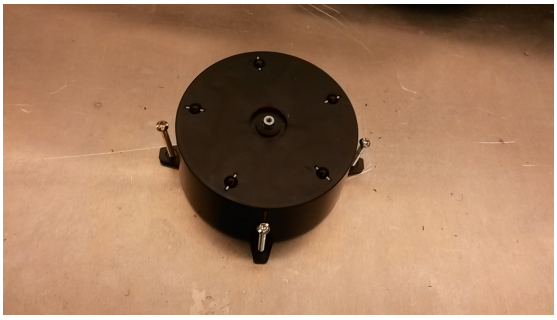


Figure 4.7: Base of turret fastened onto aluminium roof.

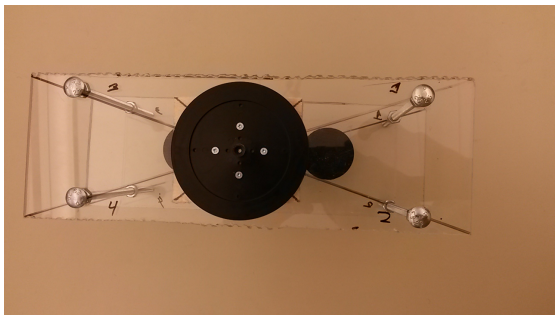


Figure 4.8: Multilevel plexiglass system attached onto rotating plate.

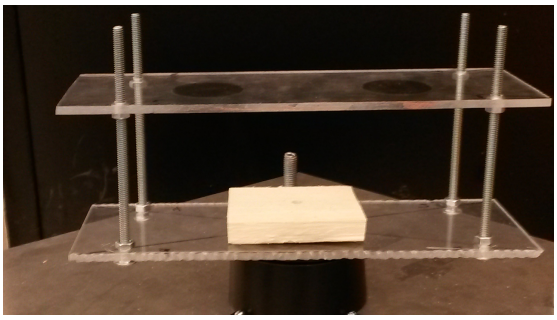


Figure 4.9: Complete turret assembly with multilevel plexiglass system.

4.1.6 Portable landmark

The portable landmark consists of a cylindrical can mounted on top of the aluminium roof right above the midpoint of the axis joining the two driving wheels (Figures 4.10 and 4.11). This midpoint is the origin of the robot's *base frame*. The arrangement ensures that the location of the landmark corresponds to the location of the robot. Cylindrical shape was chosen over other shapes because its curved surface is symmetrical about its vertical axis. This property makes the landmark appear the same when viewed from any direction perpendicular to its vertical axis, thus providing a uniform profile which helps in identification of the landmark in open areas. Additionally, the landmarks can be color-coded to discriminate in between robots.

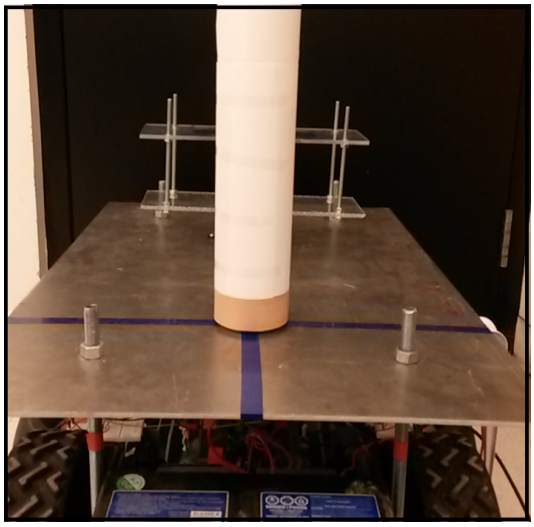


Figure 4.10: Rear View of the cylindrical portable landmark.

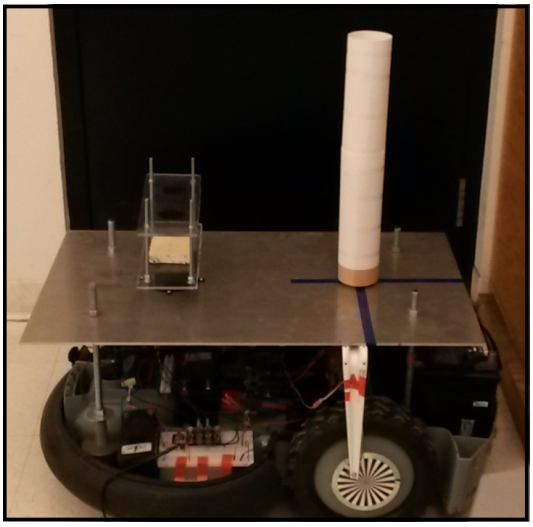


Figure 4.11: Side view of the cylindrical portable landmark.

4.1.7 Sensors

Each mobile robot is equipped with a Microsoft Kinect for Xbox 360, which acts as exteroceptive sensor, and optical wheel encoders, which act as proprioceptive sensors. During localization, the wheel encoders were used to roughly estimate the pose of a moving robot in the *prediction* step of the EKF algorithm, whereas Microsoft Kinect for Xbox 360 was used to refine the estimates in the *measurement correction* step of the algorithm.

4.1.7.1 Microsoft Kinect for Xbox 360

Microsoft Kinect for Xbox 360 [86] was used for the following reasons:

1. It comes with RGB and depth sensors which can be used for visual identification and depth perception of objects.
2. It is fully supported and compatible with Robot Operating System (ROS) [87], the software framework which controls the two robots.
3. It is relatively low-cost when compared to dedicated laser rangefinders such as those manufactured by HOKUYO AUTOMATIC CO., LTD. [88]

The sensor is mounted on the turret with the help of the plexiglass system as shown in Figure 4.12.

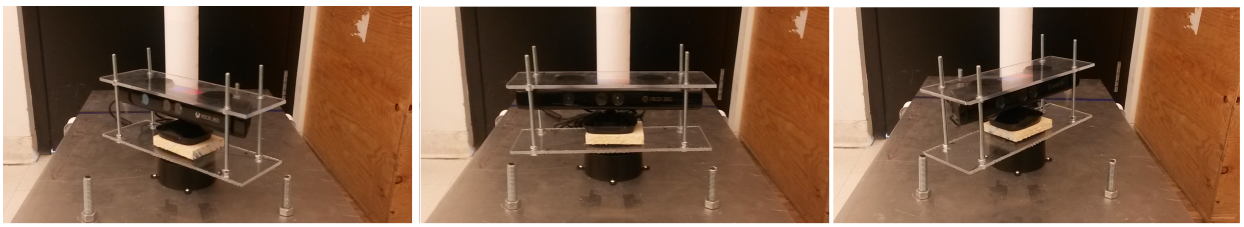


Figure 4.12: Turret with Microsoft Kinect for Xbox 360.

Because the sensor is not mounted right above the base frame (Figure 4.13), the coordinates of any point relative to the sensor have to be transformed relative to the base frame. If the Kinect's depth sensor provides $(x_k, y_k)^T$ coordinates relative to itself, as depicted in Figure 4.13, the transformation of these coordinates to the base frame of the robot becomes:

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix} + \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} \tag{4.1}$$

Where:
d is the distance of Kinect from the base frame.
x_k and y_k are the coordinates of an object (such as landmark) relative to Kinect.
θ is the angle Kinect makes with the x-axis of the base frame.
x_b and y_b are the coordinates of an object (such as landmark) relative to base frame.

The angle that Kinect makes with the x-axis of the base frame, θ, is controlled by sending PWM values to the servo motor inside the base of the turret. The motor is calibrated in such a way that it allows the sensor to face three different directions perpendicular to each other such that the possible values of θ are -90, 0 and 90 degrees (Figure 4.14). Intermediate values of θ were not used because the exact relation between the PWM values from the motor and the orientation of the turret is unknown.

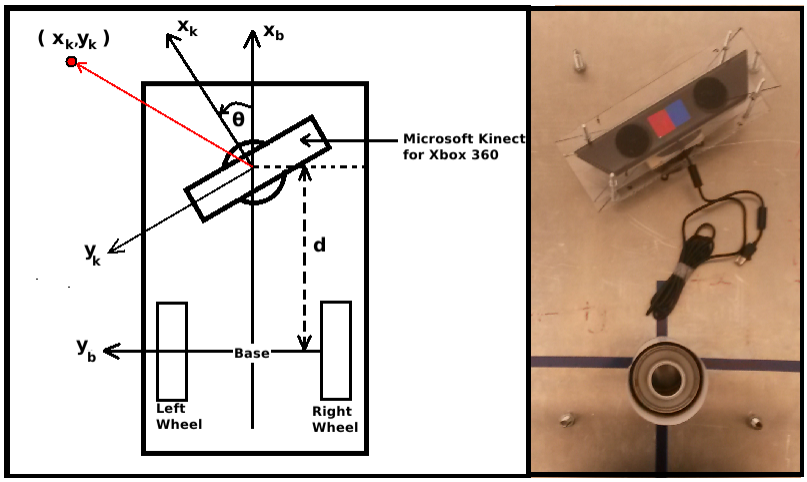


Figure 4.13: Location of the Kinect sensor relative to the base frame.
Left: With coordinate axes. Right: On actual robot.

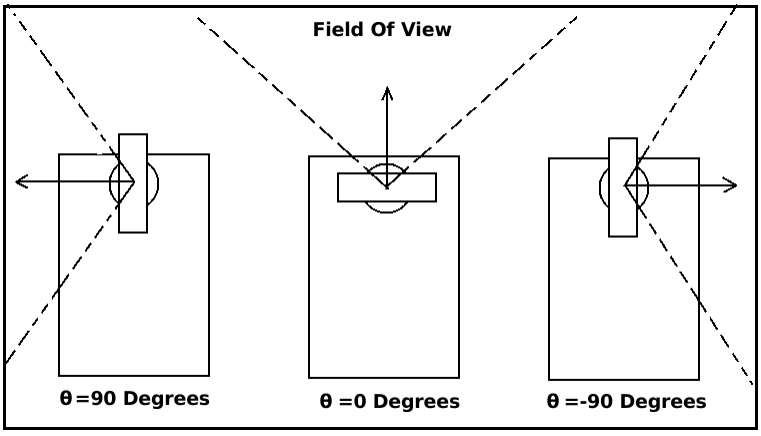


Figure 4.14: Three orientations of Kinect sensor relative to base frame.

Measurement vector and Covariance matrix for Kinect

Once the coordinates of the landmark relative to the base frame of the robot, $[x_b, y_b]^T$, are calculated using Equation 4.1, the measurement vector, z_t, can be determined as follows:

$$z_t = \begin{bmatrix} \rho \\ \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x_b^2 + y_b^2} \\ \\ atan2(y_b, x_b) \end{bmatrix} \tag{4.2}$$

The measurement covariance matrix, Q_t, associated with the measurement vector, z_t, for Kinect's depth sensor is given by the standard deviations in x_k, and y_k coordinates of an observed point relative to the sensor. Typically, an uncalibrated Kinect for Xbox 360 has a standard deviation of up to $\pm 2cm$ in x_k, and y_k when the observed point is 2m from Kinect's origin along the corresponding axes [89, 90, 91]. It follows that for a cylindrical portable landmark with radius r, at location $[x_k, y_k]$ relative to Kinect of an observing robot, the measurement covariance matrix can be approximated by:

$$Q_t = \begin{bmatrix} (\Delta\rho)^2 & 0 \\ \\ 0 & (\Delta\phi)^2 \end{bmatrix}$$

Where $\Delta\rho$ and $\Delta\phi$ are determined using the rules of error propagation as follows:

$$\Delta\rho = \left(\frac{\Delta x_k + r}{x_k} + \frac{\Delta y_k + r}{y_k} \right) \rho$$

$$\Delta\phi = \left| atan\left(\frac{y_k}{x_k} - \left(\frac{\Delta x_k + r}{x_k} + \frac{\Delta y_k + r}{y_k} \right) \frac{y_k}{x_k} \right) - atan\left(\frac{y_k}{x_k} + \left(\frac{\Delta x_k + r}{x_k} + \frac{\Delta y_k + r}{y_k} \right) \frac{y_k}{x_k} \right) \right|$$

Thus, for a cylinder with radius $r = 0.035m$, and $\Delta x_k = \Delta y_k = 0.02m$, the corresponding maximum error in distance, ρ, and orientation, ϕ, of the landmark within 2m radius of the sensor can be approximated by:

$$\Delta\rho = 0.11m$$
$$\Delta\phi = 0.06rad$$

Equation 4.2 is for the case when the moving robot uses its range sensor, Kinect, to determine the relative distance and orientation of the cylindrical landmark mounted on the stationary robot. Alternatively, the pose and orientation of the stationary landmark relative to the moving robot can be determined by observing the moving robot using the range sensors mounted on the stationary robot. This can be done with the help of an additional landmark, which will be referred to as auxiliary landmark, mounted along the x or y axis of the robots as shown in Figures 4.15 and 4.16.

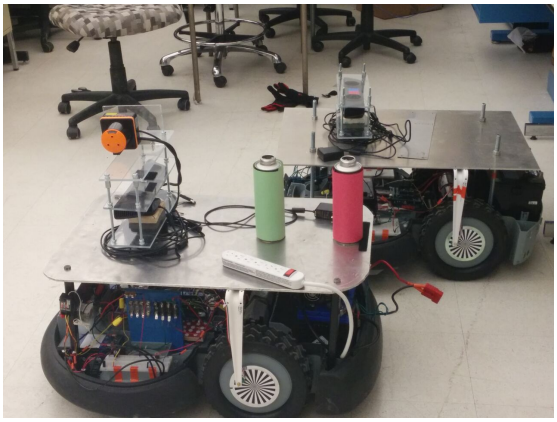


Figure 4.15: Auxiliary landmark (red) mounted along the x-axis next to the base landmark (green).

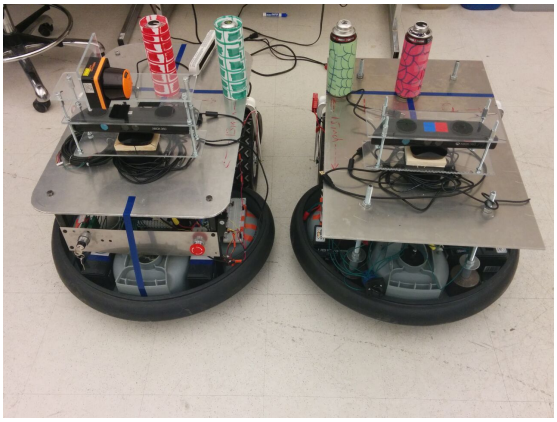


Figure 4.16: Auxiliary landmark (green) mounted along the y-axis next to the base landmark (red).

Referring to figure 4.17, let's suppose that the coordinates of the base landmark, B, and that of the auxiliary landmark, A, of a moving robot are respectively $[x_b, y_b]$ and $[x_a, y_a]$ relative to the base frame of a stationary robot, S. The distance, ρ, of the stationary robot, relative to the moving robot, is given by:

$$\rho = \sqrt{x_b^2 + y_b^2} \tag{4.3}$$

The orientation of the stationary robot relative to the moving robot can be computed using the dot product between the vectors $\overline{AB}$ and $\overline{SB}$. It can be shown that the angle of the stationary robot, ϕ, relative to the moving robot, is given by:

If both landmarks are visible:

$$\phi = \frac{y_b - y_a}{\|y_b - y_a\|} \left(\pi - \frac{x_b(x_b - x_a) + y_b(y_b - y_a)}{(\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2})(\sqrt{x_b^2 + y_b^2})} \right) \tag{4.4}$$

If the base landmark is obstructed by auxiliary landmark:

$$\phi = \pi \tag{4.5}$$

If the auxiliary landmark is obsrtucted by base landmark:

$$\phi = 0 \tag{4.6}$$

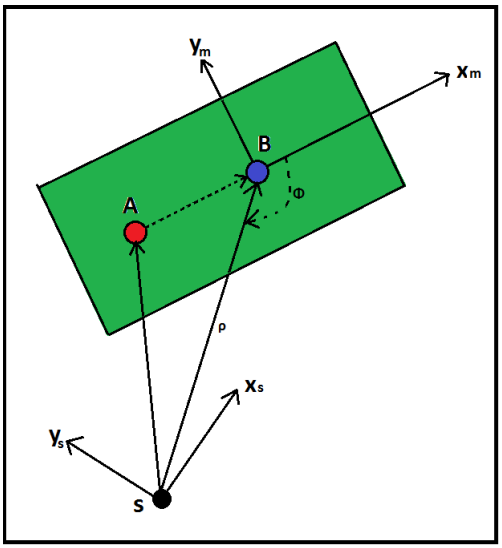


Figure 4.17: Using auxiliary landmark along the x-axis of a moving robot to determine distance and orientation of a stationary robot.

This method allows all stationary robots to simultaneously track the moving robot and determine their own distances, as well as orientations, relative to the moving robot, using their own range sensors. Thus, this method distributes the process of acquiring relative measurements among the stationary robots and therefore provides data for the correction step of the algorithm faster than when a single moving robot takes all the measurements itself, one landmark at a time. The method also ensures that multiple sensors are used to provide data needed for the measurement update and therefore improves data quality[1] as well as robustness. The downside of this method is that each stationary robot has to make measurements twice (once for each base and auxiliary landmark). The error per stationary robot associated with such measurements is at least twice the error incurred when a single moving robot takes all its measurements, one stationary robot at a time. For this reason, and because only two platforms were used in real world experiments, it was decided that the moving robot should take the relative measurements using its Kinect sensor and determine the distance and orientation of the stationary robot using Equation 4.2.

[1]If a single robot with an inaccurate sensor takes all the measurements, the results will be less accurate all the time, but when multiple robots take the measurements, the results will be a mix of various accuracies all the time.

4.1.7.2 Wheel encoders

The wheel encoders consist of paper disks with radially alternating black and white stripes. The paper disks are glued to the wheels as shown in Figure 4.18. Infrared sensors suspended from the aluminium roof by shelf-brackets detect white-to-black and black-to-white transitions on the encoders. By counting the number of transitions made, it is possible to estimate the distance traveled by each wheel, $[\Delta L, \Delta R]^T$, as follows:

$$\begin{bmatrix} \Delta L \\ \\ \Delta R \end{bmatrix} = \begin{bmatrix} \frac{2\pi r T_L}{N} \\ \\ \frac{2\pi r T_R}{N} \end{bmatrix}$$

Where:

r is the radius of the wheel.
T_L is the number of counts (transitions) in the left wheel encoder.
T_R is the number of counts (transitions) in the right wheel encoder.
N is the total number of white and black stripes on each encoder.

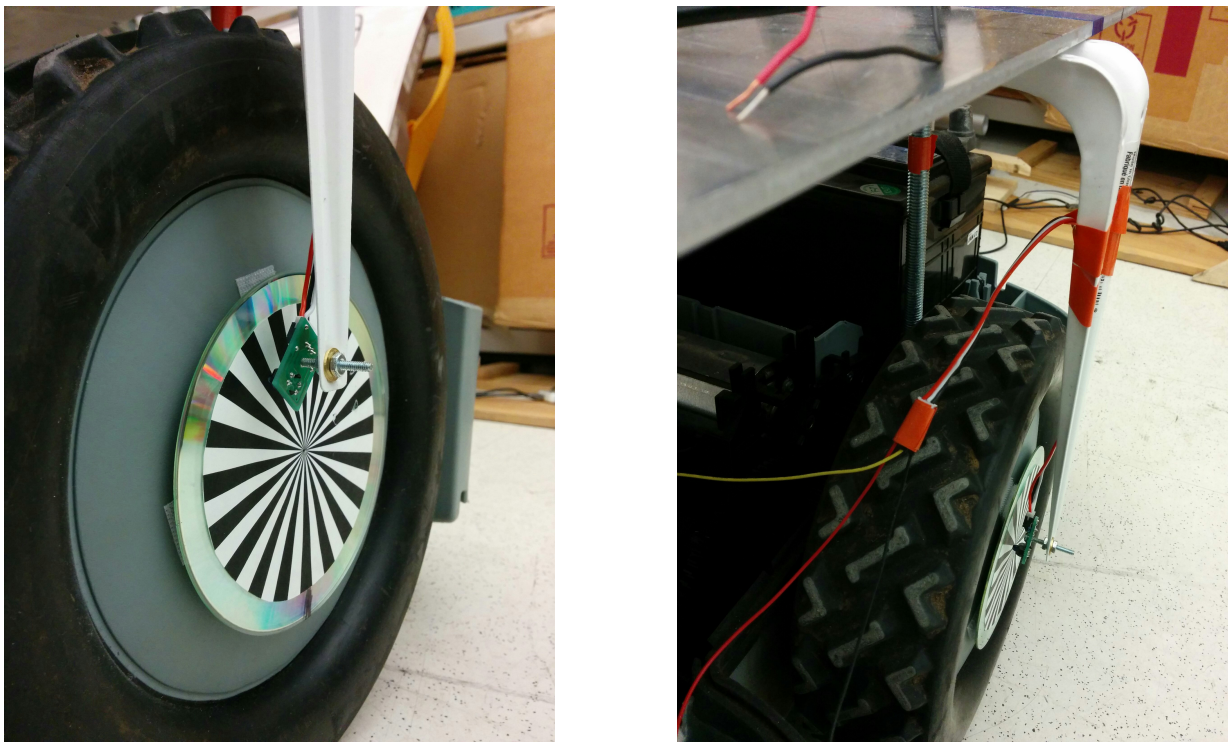


Figure 4.18: Wheel encoders.

Using the distance traveled by each wheel, the position and orientation of the robot can be roughly estimated. These estimates are then refined by fusing them (using EKF algorithm) with relative measurements from Kinect sensor.

Covariance matrix for encoders
The error associated with the distance estimated by wheel encoders is proportional to the distance traveled by the wheel, usually expressed as a percentage of the distance traveled, K.

$E_L = K_L \Delta L$
$E_R = K_R \Delta R$

Where:

K_L is the percentage error in left the encoder.
K_R is the percentage error in the right encoder.
ΔL is the distance moved by the left wheel.
ΔR is the distance moved by the right wheel.

Therefore, the covariance matrix for the wheel encoders is given by:

$$R_t = \begin{bmatrix} (K_L \Delta L)^2 & 0 \\ 0 & (K_R \Delta R)^2 \end{bmatrix}$$

The percentage errors, K_L and K_R, were determined empirically by moving the robot over a known distance and finding the difference between encoder estimates and the corresponding ground truths. The percentage errors for the encoders was found to be around 0.05. Therefore, R_t becomes:

$$R_t = \begin{bmatrix} (0.05 \Delta L)^2 & 0 \\ 0 & (0.05 \Delta R)^2 \end{bmatrix}$$

4.1.8 Laptop

Each robot carries a laptop that runs a 12.04 version of 64-bit Ubuntu, an open source Linux distribution [92]. The laptop serves as the main processing center in each robot where all computations and inter-robot communications take place. The software which runs on these laptops is described in Section 4.2.

4.1.9 Wireless Joystick

The wireless joystick shown in Figure 4.19 is the main interface through which a human operator interacts with all robots and their sensors. The joystick allows an operator to perform the following tasks:

1. Initialize robot parameters (initial poses and covariances of all robots).
2. Select a robot to move.
3. Steer the selected robot from one location to a new location.
4. Request data from wheel encoders, $[\Delta L, \Delta R]^T$, and Microsoft Kinect, $[\rho, \phi]^T$, of the selected robot.
5. Send the data to an Extended Kalman Filter for fusion.
6. Repeat 2-5

Figure 4.19: Wireless joystick for teleoperation of robotic platforms.

4.2 Software Design

Each robot runs a modular piece of software built using Robot Operating System (ROS) framework [93]. ROS provides powerful tools, libraries and conventions that simplify the process of creating complex robotic software for a wide variety of commercial and custom-made robotic platforms. The software consists of loosely coupled modules, called nodes, that communicate with each other. Figure 4.20 depicts the the top level architecture of the software, designed and implemented by the author of this thesis. It shows the interaction between two robots, i and j, over a network, and the type of information exchanged between them. Each robot runs four nodes, namely, master_node, kinect_node, arduino_node and ekf_node. The following sections will briefly describe each of these nodes.

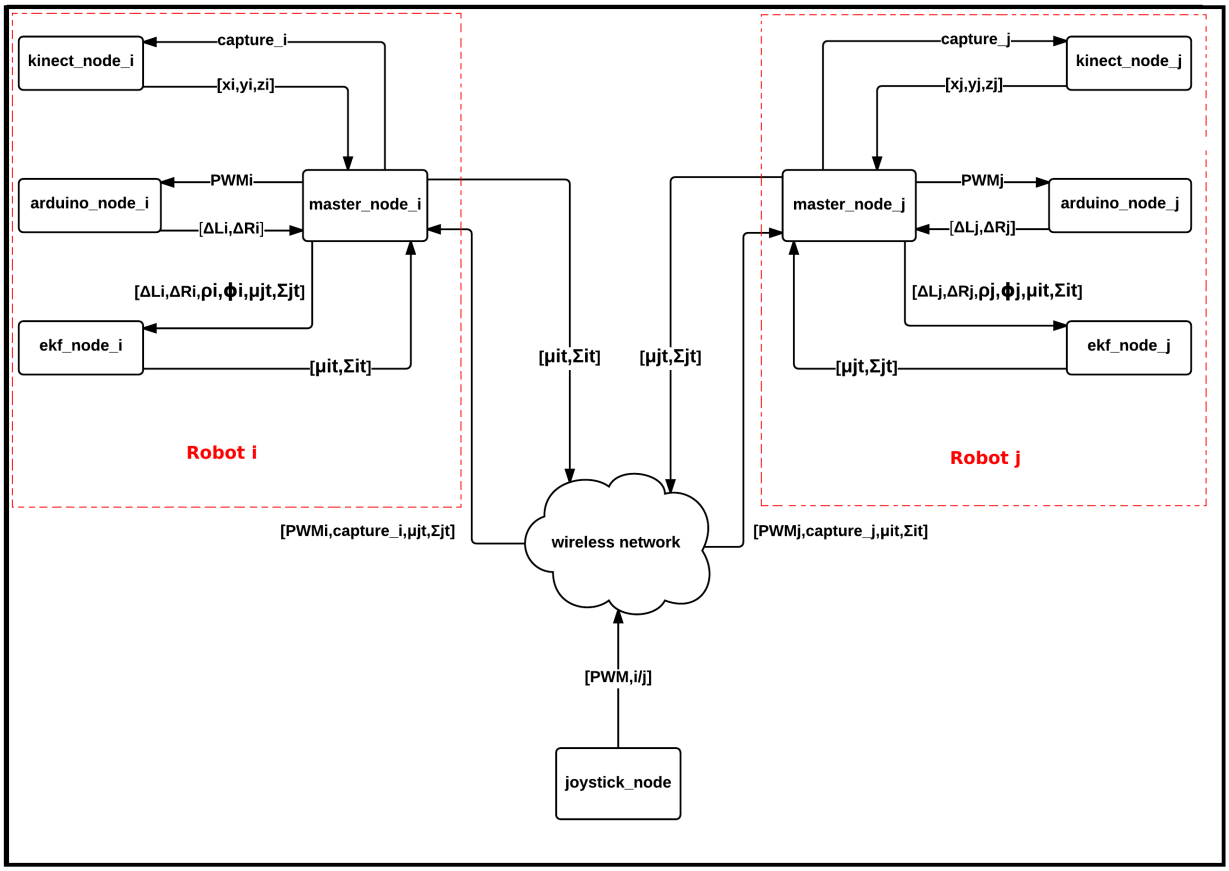


Figure 4.20: Top-level Software architecture for the custom-made robots.

4.2.1 Master Node

The master_node is the central node in each robot. It is responsible for facilitating interaction between other nodes. Specifically, the master node of each robot communicates with master nodes of every other robot over a network, and serves as a hub where data from nodes within the same robot is pre-processed, and exchanged. Communication between the robots happens over a Wi-Fi hotspot established on a smartphone. Alternatively, any robot's laptop could also be used as a Wi-Fi hotspot for inter-robot communication. This mode of communication enables the robots to communicate with one another without the need of a pre-existing network infrastructure.

4.2.2 Arduino Node

The arduino_node executes steering commands transmitted by the joystick_node via the master_node. These commands bring about the motion of the platform. The arduino_node also tracks the approximate distance traveled by each wheel, $[\Delta L, \Delta R]$, with the help of optical encoders. These distances are then sent to the ekf_node via the master_node for pose estimation in the prediction step of the Extended Kalman Filter. This node is also responsible for turning the turret which holds the Microsoft Kinect sensor.

4.2.3 EKF Node

The ekf_node executes the Extended Kalman Filter algorithm whenever new sensor information arrives. The algorithm is exactly the same as the one used in the simulations presented in Chapter 3. The only difference is that the simulator uses randomly generated inputs with zero mean, white Gaussian noise, whereas the ekf_node in ROS uses input from wheel encoders and Kinect.

4.2.4 Kinect Node

This node handles the RGB and depth images coming from Kinect sensor. The RGB channel is used for visual identification of the portable landmarks (by the teleoperator) mounted on nearby stationary robots, whereas the depth channel is used to get the x, and y coordinates of the landmarks relative to the moving robot. Because the depth (IR) and RGB cameras on Kinect are physically separated, there is an offset between depth and color images. Before taking any measurements, the two images have to be registered. This means that every pixel in the depth image is aligned with its corresponding pixel in the RGB image [90]. The process goes through three main steps.

Step 1
Suppose $[x_d, y_d]$ is a pixel on the depth image plane whose depth value is $depth(x_d, y_d)$. The corresponding 3D coordinates, $[X_D, Y_D, Z_D]$, of the pixel relative to the depth camera can be computed using the intrinsic parameters of the depth camera as follows:

$$X_D = \frac{(x_d - c\_x_d) \times (depth(x_d, y_d))}{f\_x_d} \tag{4.7}$$

$$Y_D = \frac{(y_d - c\_y_d) \times (depth(x_d, y_d))}{f\_y_d} \tag{4.8}$$

$$Z_D = depth(x_d, y_d) \tag{4.9}$$

Where the intrinsic parameters of the depth camera are defined as follows:

(c_x_d, c_y_d) is the optical center of the depth camera in pixels.

f_x_d, f_y_d are the focal lengths of the depth camera in pixels.

Step 2
The next step is to transform the 3D coordinates, $[X_D, Y_D, Z_D]$, relative to the depth camera, into the frame of reference of the RGB camera using the extrinsic parameters as follows:

$$[X_{RGB}, Y_{RGB}, Z_{RGB}]^T = R \times [X_D, Y_D, Z_D]^T + T \tag{4.10}$$

Where the extrinsic parameters are defined as follows:

R is a 3x3 rotation matrix between the depth and RGB cameras.

T is a 3x1 translation vector between the depth and RGB cameras.

Step 3
The last step is to reproject the 3D coordinates relative to the RGB camera, $[X_{RGB}, Y_{RGB}, Z_{RGB}]$, into the RGB image plane using the intrinsic properties of the RGB camera. The resulting coordinates of the pixel in the RGB image plane are calculated as follows::

$$x_{rgb} = \frac{X_{RGB} f\_x_{rgb}}{Z_{RGB}} + c\_x_{rgb} \tag{4.11}$$

$$y_{rgb} = \frac{Y_{RGB} f\_y_{rgb}}{Z_{RGB}} + c\_y_{rgb} \tag{4.12}$$

Where the intrinsic parameters of the RGB camera are defined as follows:

(c_x_{rgb}, c_y_{rgb}) is the optical center of the color camera in pixels.

f_x_{rgb}, f_y_{rgb} are the focal lengths of the color camera in pixels.

For this thesis, an Openni library, installed in ROS, was used to handle both depth and RGB images. The library has a builtin registration capability that automatically aligns depth images with their RGB counterparts using Kinect's factory intrinsic and extrinsic parameters [94].

Figure 4.21 shows the depth image (left) next to the RGB image (right). The green horizontal line seen in the RGB image acts as a cursor, and can be moved around using a joystick. When a *capture* command is sent from the joystick, the x, y coordinates of the nearest point on the cursor are sent to the ekf_node for measurement update. Therefore, when the cursor is placed over the cylindrical portable landmark (Figure 4.22) and the *capture* command is executed, the coordinates of the nearest point on the cylinder (plus radius of the cylinder) are sent to the ekf_node as $[\rho, \phi]^T$ (after transformation of coordinates as described in Section 4.1.7). In the current version of the system, the process of identifying landmarks, and acquiring relative landmark coordinates is under complete control of the remote operator. In the future, the process will be automated.

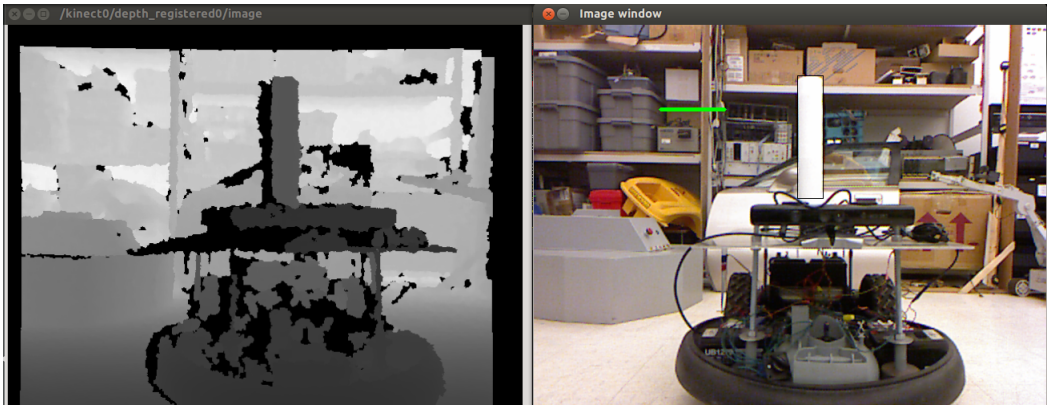


Figure 4.21: Depth Image (Left) and RGB Image (Right) from Kinect with a joystick-controlled line cursor (Green).

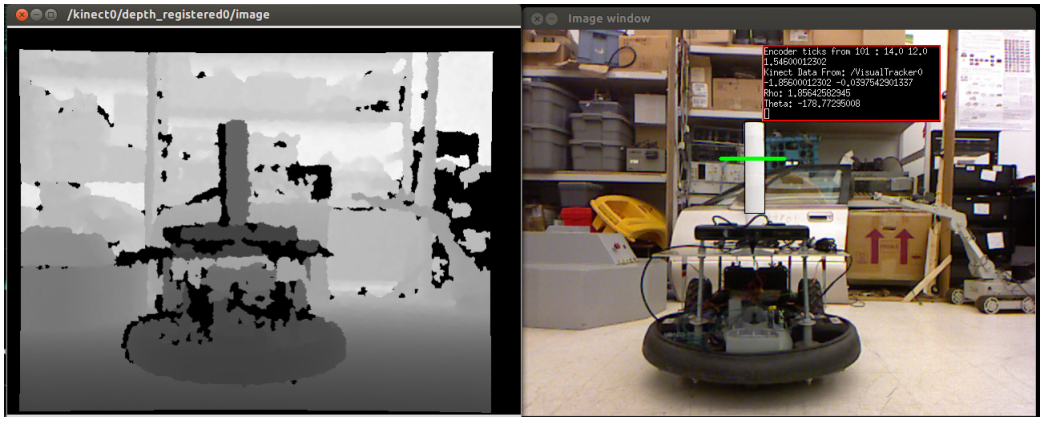


Figure 4.22: Depth Image (Left) and RGB Image (Right) from Kinect with a line cursor over the cylindrical landmark.

4.2.5 Joystick Node

This node transmits all joystick commands to the master_node, where they are preprocessed and sent to the target node. For example, when a remote user moves the Left/Right or Forward/Reverse analog sticks of the joystick, the resulting analog signals from the joystick are sent to the master_node, where they are mapped to PWM values. The PWM values are then sent to the arduino_node where they bring about differential motion of the robot.

4.3 Real World Evaluation of the Proposed Localization Method

4.3.1 Experimental Setup and Environment

The experimental setup used to validate the proposed localization method consists of two robotic platforms, introduced in Sections 4.1 and 4.2. The platforms move in tandem formation inside a narrow corridor over alternating black and reddish-brown, one-foot (0.30m approx.) square tiles on the floor as shown in Figure 4.23. The square tiles define the grid system with which the ground truths of the two robots can be manually determined to within $\pm 0.15m$ (i.e., half the grid resolution) of their actual positions as illustrated in Figure 4.24.

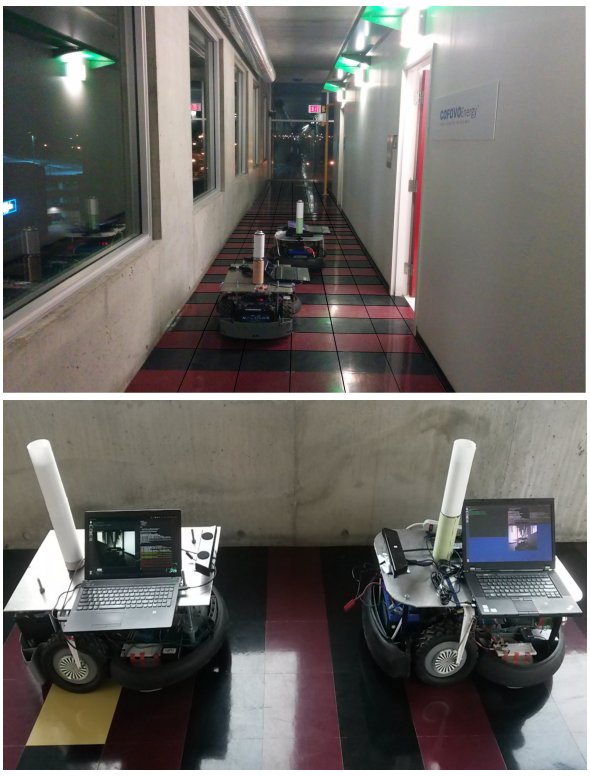


Figure 4.23: Experimental setup for evaluation of multirobot localization.

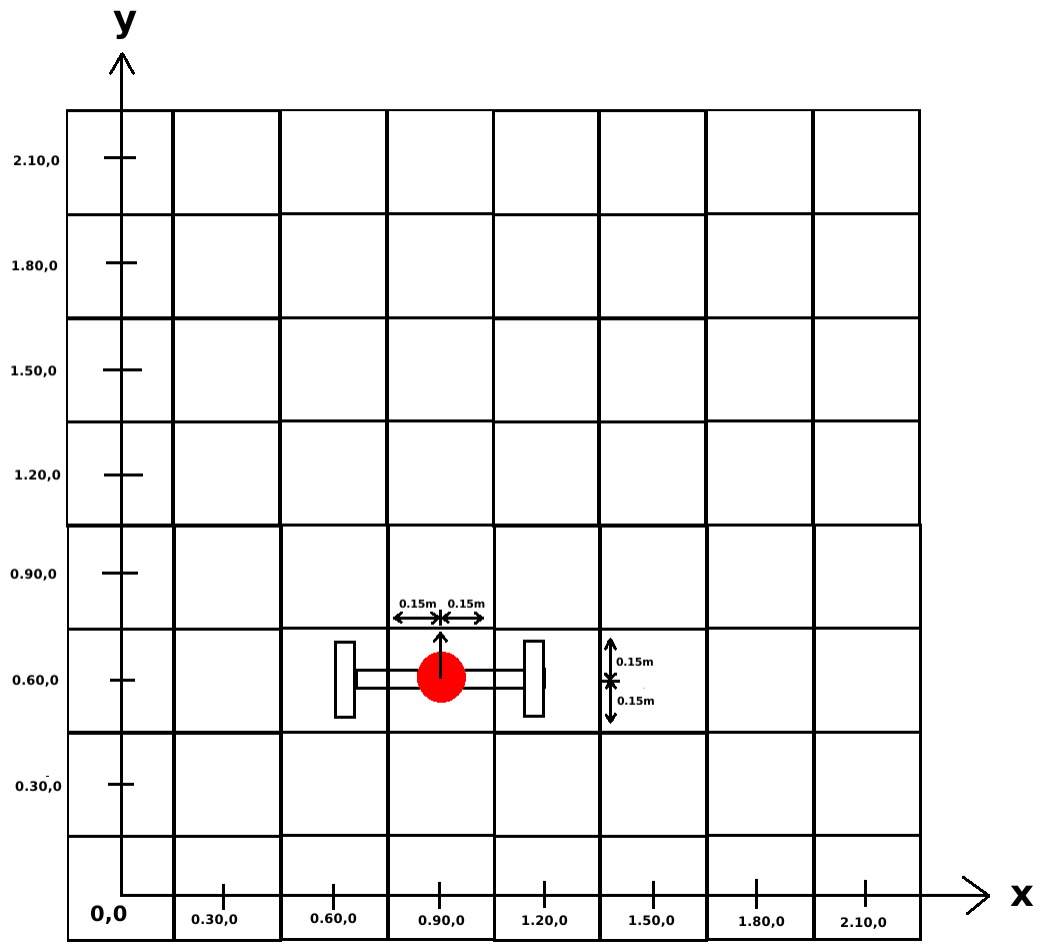


Figure 4.24: Using square tiles for estimating ground truths of the robots.

The robots are manually steered by a joystick in such a way that when the robots come to a stop, the portable landmark is within a tile. The position of the robot is then given by $[x_t \pm 0.15, y_t \pm 0.15]m$ where $[x_t, y_t]$ are the coordinates of the center of a tile in the global frame. For example, in Figure 4.24, $[x_t, y_t] = [0.90, 0.60]$, therefore the ground truth of the robot is $[0.90 \pm 0.15, 0.60 \pm 0.15]$

Since the real-world tests were performed inside narrow corridors using only two platforms, Kinect sensors had to be remounted differently so that each robot would always be within the field of view of another robot's Kinect as shown in Figure 4.25. The new configuration uses fixed Kinect sensors, as opposed to turret-mounted sensors described in Section 4.1.7. This new configuration eliminates the uncertainties induced by the turret motor, resulting in more accurate relative angle and distance measurements. The disadvantage of this configuration is that it requires the two robots to always be in tandem formation. In larger, open, areas, turret mounted sensors would be more appropriate.

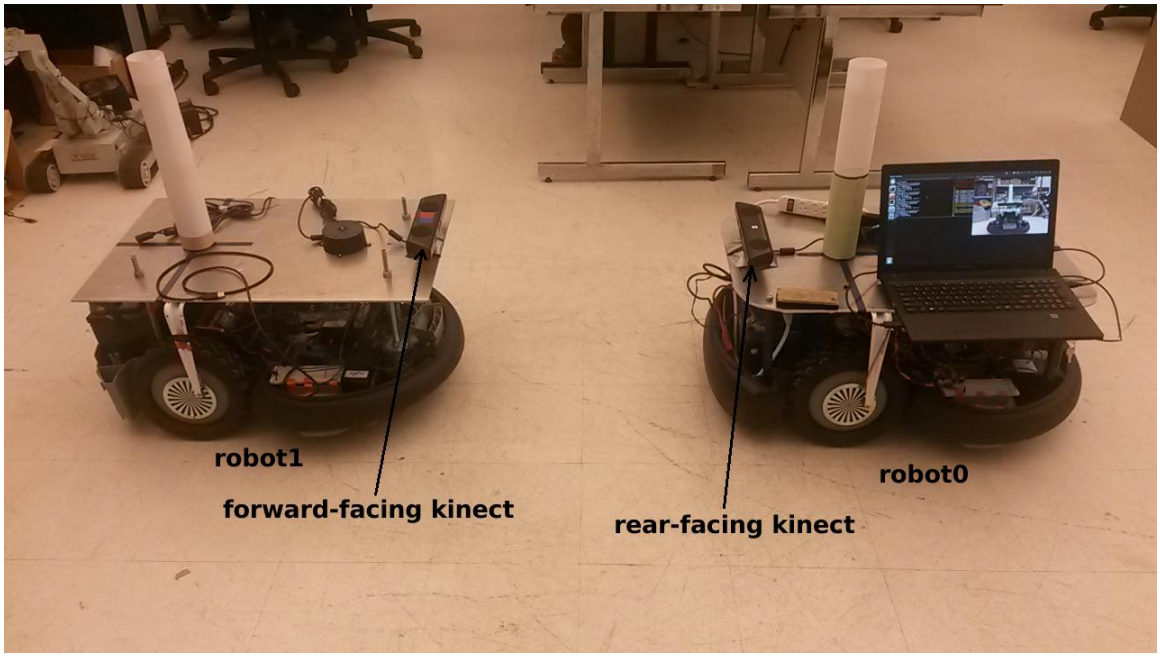


Figure 4.25: Fixed rear-facing and forward-facing Kinect Sensors on robot0 and robot1 respectively.

4.3.2 Evaluation Procedure

The starting pose of robot1 (platform with forward-facing Kinect sensor) was initialized to $[x, y, \theta] = [0, 0, \pi/2]$ (i.e., at origin and parallel to global y-axis). The robot with the rear-facing camera (robot0), was placed directly in front, and 5 square tiles away from the first robot as shown in Figure 4.26. The coordinates of this robot was therefore $[x, y, \theta] = [0, 1.5, \pi/2]$. The initial covariance matrix for each robot was initialized to uncertainty corresponding to the size of a square tile.

Localization starts by manually moving robot0 to the next square in a straight line, using a wireless joystick. From this new location (Figure 4.27), the relative coordinates of robot1, which acts as a landmark to robot0, are manually acquired using Kinect on robot0 as explained in Section 4.2.4. The two robots then switch roles such that robot1 becomes the moving robot while robot0 acts as a landmark (Figure 4.28). They continue to move this way, each time alternating between their roles, until each robot covers about 50 square tiles (about 15 meters approx). At each step during their journey, the ground truth (determined by the square tiles in Figure 4.24), the corresponding control input, $u_t = [\Delta L, \Delta R]$, coming from the wheel encoders, as well as relative measurement, $Z_t = [\rho, \phi]$, coming from Kinect sensors is logged in a text file for offline tuning of the filter. This experiment was repeated 10 times in order to get a sufficient amount of data samples for ANEES calculations.

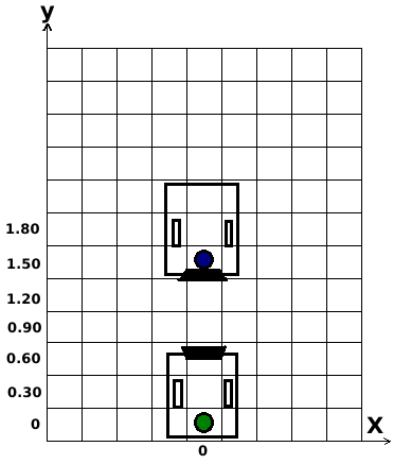


Figure 4.26: Initial poses for robot0 and robot1.

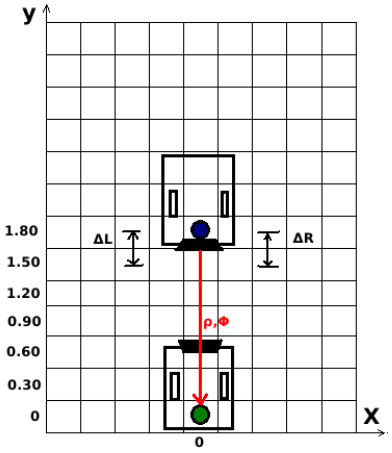


Figure 4.27: robot0 moves forward 1 square, and captures coordinates of robot1.

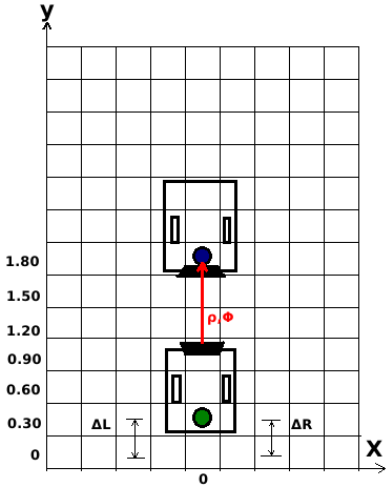


Figure 4.28: robot1 moves forward 1 square, and captures coordinates of robot0.

The Extended Kalman Filter was then tuned as described in Section 3.2 using the experimental data collected from the first 10 real world until the results that provided the best consistent ANEES values were obtained. The experiment was then repeated 10 more times with artificially inflated covariance matrices using the best Covariance Inflation Index obtained through tuning.

4.3.3 Experiment Results

Figures 4.29 and 4.30 show the trajectories of robot0 and robot1 respectively, averaged over 10 runs before tuning the filter through artificial inflation of the landmark covariances. The red and green error ellipses around EKF poses of each robot represent the estimated and actual uncertainties associated with EKF poses respectively. It can be seen that, for each robot, the estimated error ellipses (red) converge around the EKF poses. On the other hand, the actual error ellipses (green) diverge due to the fact that the Mean Absolute Error in each robot gradually increases as shown in Figure 4.31. As a result, the ANEES[2] for each robot quickly diverges away from the consistent region (broken lines) as shown in Figure 4.32, leading to 0% of consistent estimates in both robots. Each iteration on the x-axis of Figures 4.31 and 4.32 corresponds to a distance covered by a robot as it moves from the center of one tile to the center of the next(0.30m approx).

[2]Graphically, ANEES at an estimated position can be thought of as a ratio of the size of the actual error ellipse (green) to the size of the estimated error ellipse (red). This explains why ANEES is greater than 1 (optimistic case) when the estimated ellipse is smaller than actual ellipse

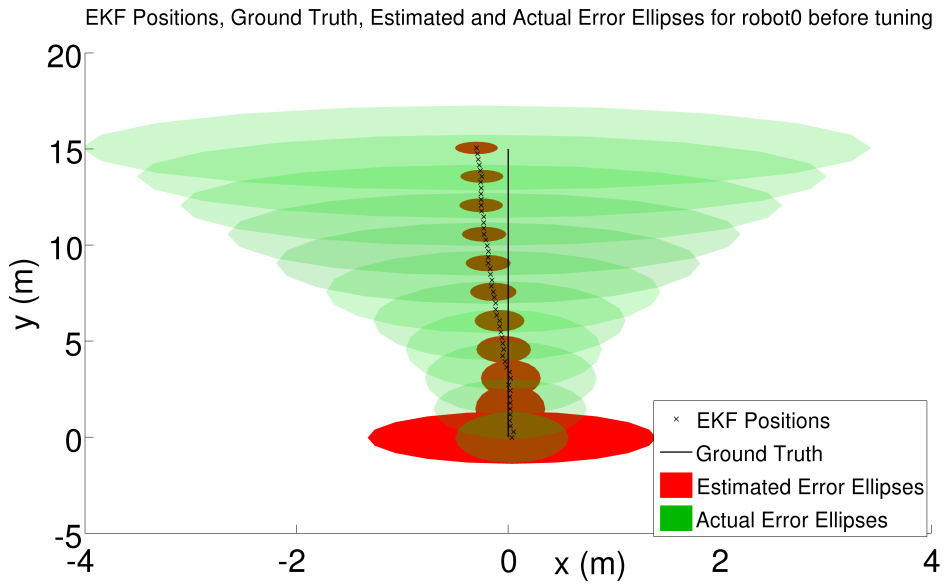


Figure 4.29: EKF positions, ground truth, estimated and actual error ellipses for robot0 before tuning.

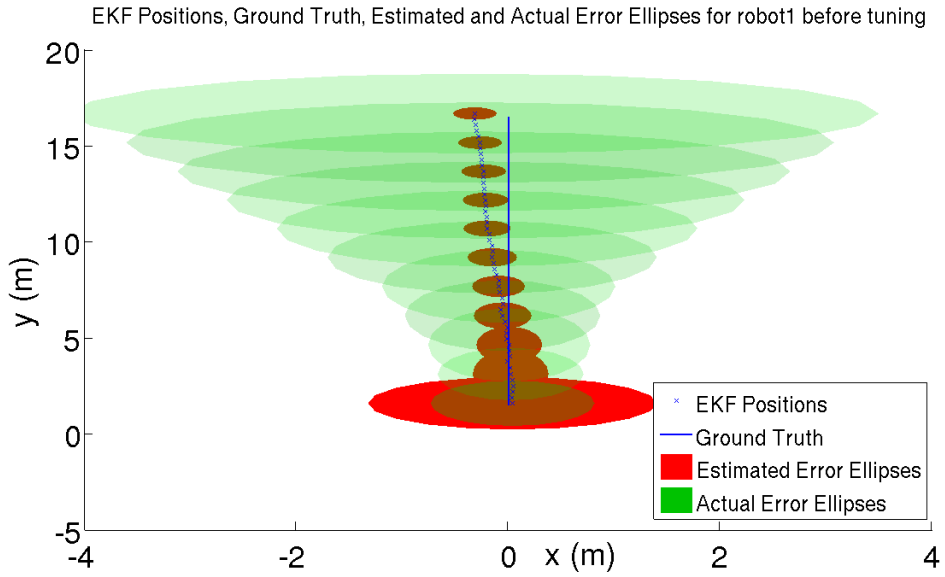


Figure 4.30: EKF positions, ground truth, estimated and actual error ellipses for robot1 before tuning.

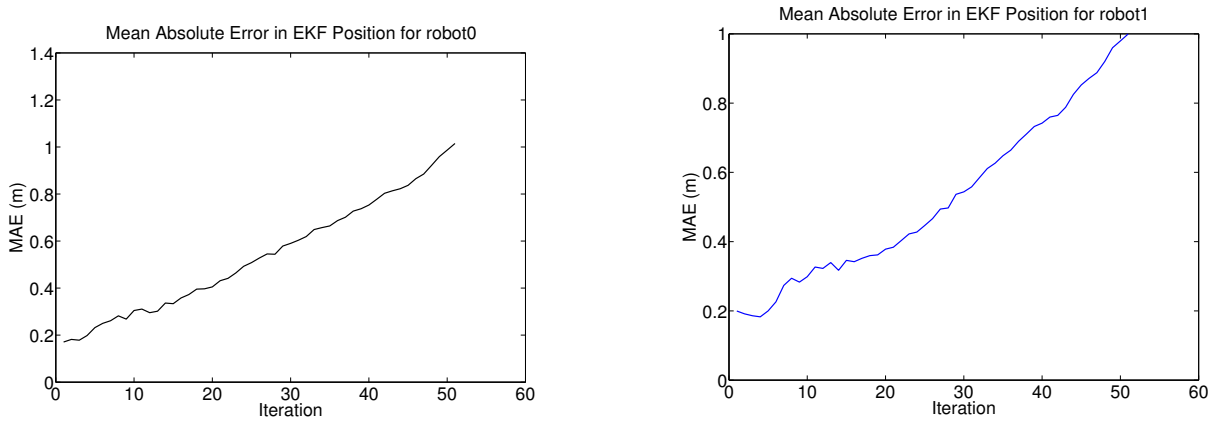


Figure 4.31: MAE for robot0 (left) and robot1 (right) before tuning.

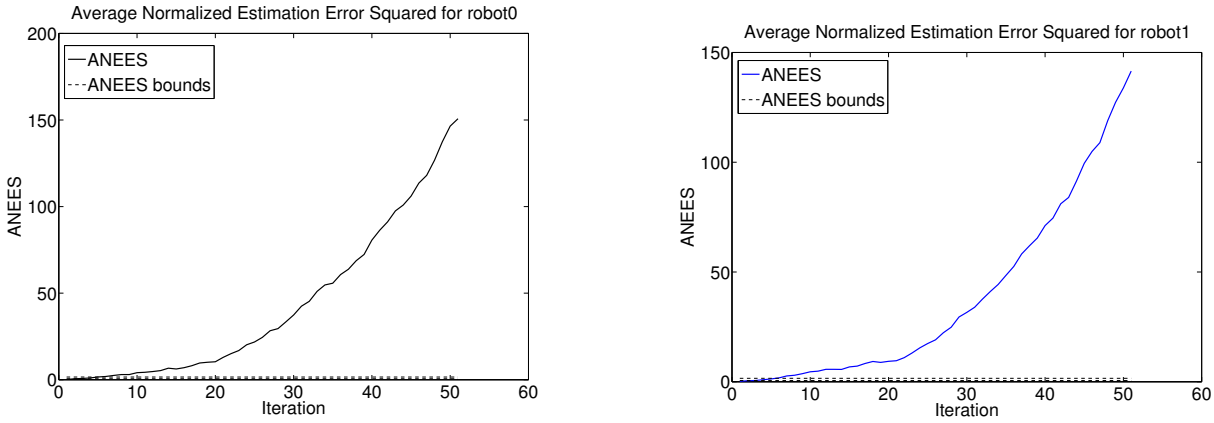


Figure 4.32: ANEES for robot0 (left) and robot1 (right) before tuning.

Figures 4.33 and 4.34 show the trajectories of robot0 and robot1 respectively, averaged over 10 runs after tuning the filter through artificial inflation of the covariance matrices. In this case, for each robot the estimated error ellipses (red) diverge at the same rate as the actual error ellipses (green), thus accurately reflecting the gradual growth Mean Absolute Error in each robot depicted in Figure 4.35. Consequently, a significant improvement in the ANEES of each robot was achieved as shown in Figure 4.36. While tunning the EKF, the best ANEES was achieved when the value of A in Equation 3.23 was 2.5. At this value of A, the percentages of consistent EKF poses computed from 10 real world experiments at 95% confidence interval were 90.20% and 88.20% for robot0 and robot1 respectively. Additionally, Figure 4.35 shows that the rate at which the Mean Absolute Error of each robot grows is smaller after tuning the filter than when compared to before it was tuned (Figure 4.31).

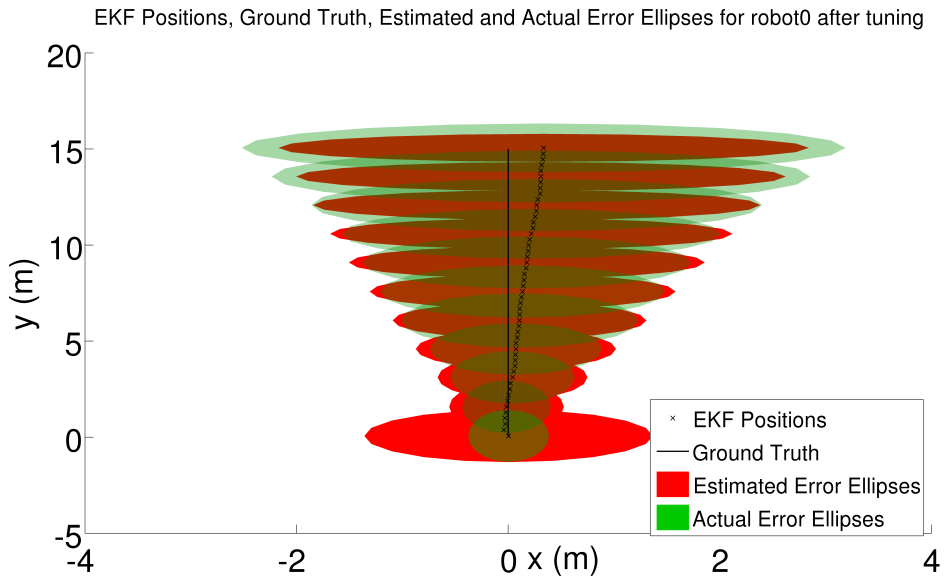


Figure 4.33: EKF positions, ground truth, estimated and actual error ellipses for robot0 after tuning.

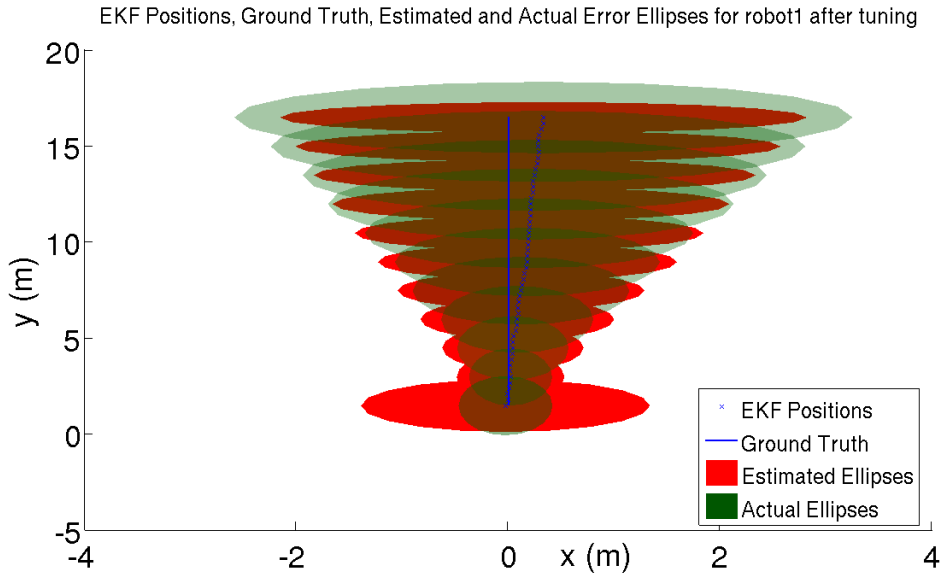


Figure 4.34: EKF positions, ground truth, estimated and actual error ellipses for robot1 after tuning.

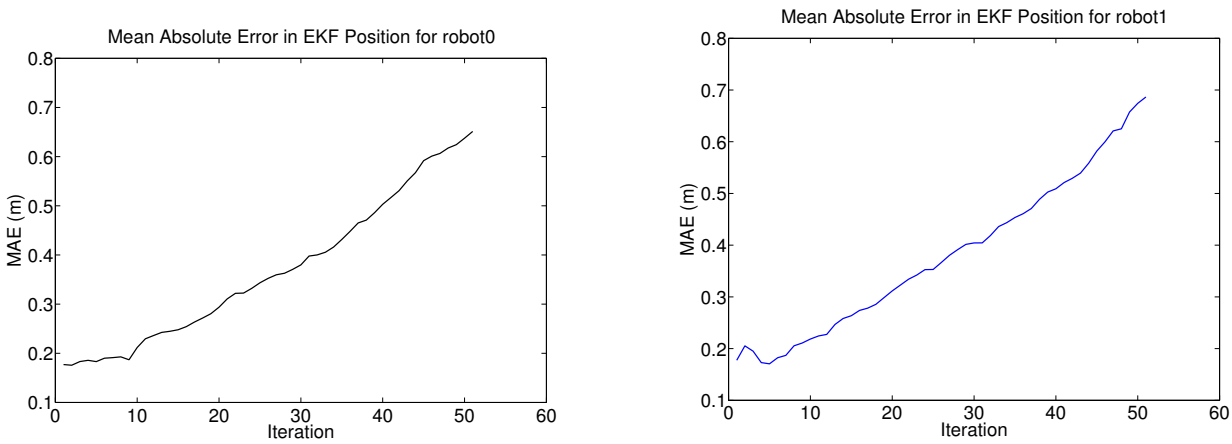


Figure 4.35: MAE for robot0 (left) and robot1 (right) after tuning.

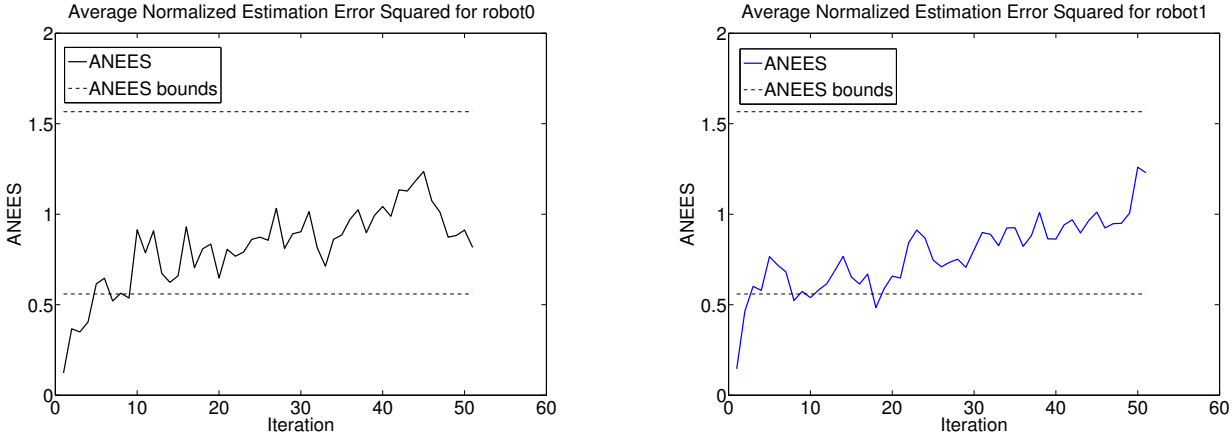


Figure 4.36: ANEES for robot0 (left) and robot1 (right) after tuning.

The experimental results show that, given that the robots avoid making tight turns throughout their journey, the consistency of poses estimated by the Extended Kalman Filter can be improved at no additional computation cost by artificially inflating the covariance matrix of the stationary robot which acts as a landmark, before propagating it to the moving robot. Table 4.1 compares the multirobot localization algorithm proposed in this thesis with some of the existing state-of-art decentralized multirobot localization techniques based on Extended Kalman Filter.

Table 4.1: Comparison of the proposed multirobot localization algorithm with some of existing state-of-art decentralized localization techniques based on EKF.

	Roumeliotis [67]	Panzieri [57]	Proposed Methodology
Year	2000	2006	2016
Algorithm	Classic Extended Kalman Filter	Interlaced Extended Kalman Filter	Heuristically Tuned Extended Kalman Filter
Performance	Optimal	Approximate	Approximate
Consistency	Consistent	Inconsistent	Consistent
Computational Complexity	$O(N^4)$	$O(N)$	$O(N)$
Communication Complexity	$O(N^2)$	$O(N)$	
Simulations		✓	✓
Real world Experiments	✓	✓	✓
Scalability	Poor	Good	Good

4.4 Chapter Summary

This chapter presented the real-world implementation and evaluation of the proposed distributed, EKF algorithm using two custom-built mobile robots. The results from the experiments validate the effectiveness of the heuristic tuning methodology in improving consistency of the EKF pose estimates of the two robots.

Chapter 5

Conclusion and Future Work

5.1 Summary and Conclusion

This thesis presented some of the common multirobot localization techniques that have been developed, and pointed out that one of the challenges that many of the existing localization methods face is getting consistent or trustworthy estimates. Historically, a common way of getting consistent EKF pose estimates involves heuristic tuning of the filter through deliberate inflation of landmark or sensor covariance matrices. However, there are no formal techniques that determine the adequate inflation of these covariances in order to achieve the desired consistency. In this thesis, a formal heuristic methodology for inflating landmark covariance matrices in single robot localization was proposed and later extended to multirobot systems. Simulation results as well as real-world experiments presented in the thesis show that, provided the robots execute smooth motions between consecutive observations, the heuristically tuned EKF outperforms the standard EKF in terms of consistency while maintaining the computational complexity that of the standard EKF.

5.2 Contributions

The main contribution of this thesis is:

- Methodical strategy for tuning a classic Extended Kalman Filter through artificial inflation of the landmark covariance matrix for consistent pose estimation in single robot localization using fixed landmarks, as well as in multi-robot localization using the robots themselves as portable landmarks.

Other contributions of the thesis include:

- A comprehensive experimental evaluation, in simulation, of the standard EKF and proposed heuristically tuned EKF for single robot localization with fixed landmarks.
- Remote controlled robotic platforms were built to support real-world experimental validation of the proposed approach.

- Implementation of an EKF-based, mutirobot, localization software in R.O.S (Robot Operating System) (Available for download at this link: https://ruslanmasinjila@bitbucket.org/ruslanmasinjila/ros_rl500xa.git)

5.3 Future Work

While this thesis demonstrates a strategy for methodical tuning of the standard EKF in order to achieve consistent pose estimates in decentralized, multirobot systems, many opportunities for improving and extending the scope of the thesis still exist. Among them we include:

- Development of a robust computer vision solution to automate the process of tracking and acquisition of distances and orientations of landmarks relative to moving robots.
- Refinement of measurement accuracy via the use of more accurate sensors other than Kinect.
- More extensive field testing using more than two robots, and over a larger area.
- Development of a solution for automatic tuning of the Extended Kalman Filter.
- Extension of the proposed tuning methodology to robotic platforms with more than two sensors.

References

[1] N. Y. Foo, *Advanced topics in artificial intelligence 12th Australian Joint Conference on Artificial Intelligence, AI'99, Sydney, Australia, December 6-10, 1999 : proceedings*, Springer, Berlin, New York, 1999.

[2] M. Laraia, *Nuclear decommissioning planning, execution and international experience*, Woodhead Pub, Philadelphia, Pa, 2012.

[3] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin, "Mars microrover navigation: Performance evaluation and enhancement," *Autonomous Robots*, vol. 2, no. 4, pp. 291–311, 1995.

[4] S. G. Tzafestas, *Introduction to mobile robot control*, Elsevier, London Waltham, MA, 2014.

[5] F. Cozman and E. Krotkov, "Automatic mountain detection and pose estimation for teleoperation of lunar rovers," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 1997, vol. 3, pp. 2452–2457.

[6] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[7] J.J. Leonard and H.F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, Jun 1991.

[8] J. Borenstein and L. Feng, *A method for measuring, comparing, and correcting dead-reckoning errors in mobile robots*, 1994.

[9] R. Negenborn, *Robot localization and Kalman filters*, Ph.D. thesis.

[10] "Meet amazon's busiest employee – the kiva robot," http://www.cnet.com/news/meet-amazons-busiest-employee-the-kiva-robot/, Accessed: 2015-09-27.

[11] Avinash Gautam and Sudeept Mohan, "A review of research in multi-robot systems," in *Proceedings of the IEEE International Conference on Industrial and Information Systems*. IEEE, 2012, pp. 1–5.

[12] Y. Jing, H. Yalou, T. Tong, and Fengchi S., "A cooperative approach for multi-robot area exploration," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 1390–1395.

[13] L.C. Carrillo-Arce, E.D. Nerurkar, J.L. Gordillo, and S.I. Roumeliotis, "Decentralized multi-robot cooperative localization using covariance intersection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1412–1417.

[14] T. R. Wanasinghe, G. K. Mann, and R. G. Gosine, "Decentralized cooperative localization for heterogeneous multi-robot system using split covariance intersection filter," in *Proceedings of the Canadian Conference on Computer and Robot Vision*. IEEE, 2014, pp. 167–174.

[15] T. R. Wanasinghe, G. K. Mann, and R. G. Gosine, "Distributed collaborative localization for a heterogeneous multi-robot system," in *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering*. IEEE, 2014, pp. 1–6.

[16] I. J Cox, "Blanche: Position estimation for an autonomous robot vehicle," in *Autonomous Mobile Robots: Control, Planning, and Architecture (Vol. 2)*, S. S. Iyengar and A. Elfes, Eds., pp. 285–292. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[17] J. Borenstein, *Navigating mobile robots : systems and techniques*, A K Peters, Wellesley, Mass, 1996.

[18] C. Adams, "The straight dope: Is "dead reckoning" short for "deduced reckoning"?," http://www.straightdope.com/columns/read/2053/is-dead-reckoning-short-for-deduced-reckoning, 2002, (Visited on 06/20/2015).

[19] J. Borenstein, H. R. Everett, and L. Feng, "Where am i? sensors and methods for mobile robot positioning," *University of Michigan*, vol. 119, no. 120, pp. 27, 1996.

[20] R. Siegwart, *Introduction to autonomous mobile robots*, MIT Press, Cambridge, Mass, 2004.

[21] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, *Mobile robot positioning: Sensors and techniques*, John Wiley & Sons, Inc., 1997.

[22] B. Barshan and H.F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 328–342, Jun 1995.

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press, 2005.

[24] S. Thrun, *A bayesian approach to landmark discovery and active perception in mobile robot navigation*, Citeseer, 1996.

[25] A. Singhal, "Issues in autonomous mobile robot navigation," Tech. Rep., 1997.

[26] A. Noureldin, *Fundamentals of Inertial Navigation, Satellite Positioning and Their Integration*, Springer, New York, 2012.

[27] G.N. Desouza and A.C. Kak, "Vision for mobile robot navigation: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237–267, Feb 2002.

[28] Y. Kim, "Localization of a mobile robot using a laser range finder in a hierarchical navigation system," in *Proceedings of the IEEE SoutheastCon*, Apr 1993, p. 4.

[29] R. Greiner and R. Isukapalli, "Learning to select useful landmarks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 3, pp. 437–449, Jun 1996.

[30] S. Thrun, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, pp. 1–35, 2002.

[31] H. Durrant-Whyte, S. Majumder, S. Thrun, M. De Battista, and S. Scheding, "A bayesian algorithm for simultaneous localisation and map building," in *Robotics Research*, pp. 49–60. Springer Berlin Heidelberg, 2003.

[32] M. W. M. G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun 2001.

[33] J. J. Leonard and H. J. S. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," in *Proceedings of the International Symposium on Robotics Research*. Citeseer, 2000, vol. 9, pp. 169–178.

[34] T. Sebastian, B. Wolfram, F. Dieter, H. Henry, and M. Maja, "A probabilistic approach to concurrent mapping and localization for mobile robots," in *Machine Learning*, 1998, pp. 29–53.

[35] B. Michael, N. Paul, L. John, and T. Seth, "Slam in large-scale cyclic environments using the atlas framework," *International Journal of Robotics Research*, pp. 1113–1139, 2004.

[36] José A. Castellanos, J. M. Martnez, José Neira, and J. D. Tardós, "Experiments in multisensor mobile robot localization and map building," in *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles*, Madrid, Spain, March 1998, pp. 173–178.

[37] K. S. Chong and L. Kleeman, "Feature-based mapping in real, large scale environments using an ultrasonic array," *The International Journal of Robotics Research*, vol. 18, no. 1, pp. 3–19, 1999.

[38] A. J. Davison, Y. G. Cid, and N. Kita, "Real-time 3D SLAM with wide-angle vision," in *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, July 2004.

[39] J. Folkesson and H. Christensen, "Graphical slam - a self-correcting map," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2004, vol. 1, pp. 383–390 Vol.1.

[40] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, 2001.

[41] J. Kim and S. Sukkarieh, "Airborne simultaneous localisation and map building," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Sept 2003, vol. 1, pp. 406–411 vol.1.

[42] J. Kim and S. Sukkarieh, "Autonomous airborne navigation in unknown terrain environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1031–1045, July 2004.

[43] S. B. Williams, P. Newman, G. Dissanayake, and H. Durrant-Whyte, "Autonomous underwater simultaneous localisation and map building," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000, vol. 2, pp. 1793–1798.

[44] S. Thrun, "An online mapping algorithm for teams of mobile robots," *International Journal of Robotics Research*, vol. 20, pp. 2001, 2001.

[45] S. Thrun, "Bayesian landmark learning for mobile robot localization," *Machine Learning*, vol. 33, no. 1, pp. 41–76, 1998.

[46] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, pp. 391–427, 1999.

[47] H. Bruyninckx, "Bayesian probability," 2002.

[48] D. Fox, J. Hightower, H. Kauz, L. Liao, and D. Patterson, "Bayesian techniques for location estimation," in *Proceedings of the Workshop on location-aware computing*. Citeseer, 2003, pp. 16–18.

[49] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the Symposium on Adaptive Systems for Signal Processing, Communications, and Control*. IEEE, 2000, pp. 153–158.

[50] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," 2004.

[51] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *AeroSense'97*. International Society for Optics and Photonics, 1997, pp. 182–193.

[52] M. S. Grewal and A. P. Andrews, "Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]," *Control Systems, IEEE*, vol. 30, no. 3, pp. 69–78, 2010.

[53] R. González, F. Rodrıguez, J. L. Guzmán, and M. Berenguel, "Comparative study of localization techniques for mobile robots based on indirect kalman filter," in *Proceedings of the IFR International Symposium on Robotics*, 2009, pp. 253–258.

[54] E. Ivanjko and I. Petrovic, "Extended kalman filter based mobile robot pose tracking using occupancy grid maps," in *Proceedings of the IEEE Mediterranean Electrotechnical Conference*, May 2004, vol. 1, pp. 311–314.

[55] L. E. Parker, "Current research in multirobot systems," *Artificial Life and Robotics*, vol. 7, no. 1-2, pp. 1–5, 2003.

[56] T. Arai, E. Pagello, and L. E. Parker, "Editorial: Advances in multi-robot systems," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 655–661, 2002.

[57] S. Panzieri, F. Pascucci, and R. Setola, "Multirobot localisation using interlaced extended kalman filter," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 2816–2821.

[58] R. Y. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 1250–1257.

[59] R. Vincent, D. Fox, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schulz, and B. Stewart, "Distributed multirobot exploration, mapping, and task allocation," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 229–255, 2008.

[60] A. Howard, M. J. Matark, and G. S. Sukhatme, "Localization for mobile robot teams using maximum likelihood estimation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2002, vol. 1, pp. 434–439.

[61] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 7–40, 2001.

[62] S. I. Roumeliotis, *Robust Mobile Robot Localization: From single-robot uncertainties to multi-robot interdependencies*, Ph.D. thesis, University of Southern California, 2000.

[63] C. Chang, S. Wang, and C. Wang, "Vision-based cooperative simultaneous localization and tracking," in *IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5191–5197.

[64] E. D. Nerurkar and S. Roumeliotis, "Asynchronous multi-centralized cooperative localization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4352–4359.

[65] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous robots*, vol. 8, no. 3, pp. 325–344, 2000.

[66] K. Y. K. Leung, *Cooperative Localization and Mapping in Sparsely-Communicating Robot Networks*, Ph.D. thesis, University of Toronto, 2012.

[67] S. I. Roumeliotis and G. A. Bekey, "Distributed multi-robot localization," in *Distributed autonomous robotic systems 4*, pp. 179–188. Springer, 2000.

[68] E. D. Nerurkar, S. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1402–1409.

[69] A. Prorok, A. Bahr, and A. Martinoli, "Low-cost collaborative localization for large-scale multi-robot systems," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 4236–4241.

[70] A. Bahr, M. R. Walter, and J. J. Leonard, "Consistent cooperative localization," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3415–3422.

[71] A. Martinelli, "Improving the precision on multi robot localization by using a series of filters hierarchically distributed," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and systems*. IEEE, 2007, pp. 1053–1058.

[72] H. Li, F. Nashashibi, and M. Yang, "Split covariance intersection filter: Theory and its application to vehicle localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1860–1871, 2013.

[73] Chai, Tianfeng, and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[74] X. R. Li, Z. Zhao, and V. P. Jilkov, "Practical measures and test for credibility of an estimator," in *Proceedings of the IFAC World Congress*, 2002.

[75] W. Niehsen, "Information fusion based on fast covariance intersection filtering," in *Proceedings of the International Conference on Information Fusion*. IEEE, 2002, vol. 2, pp. 901–904.

[76] P. S. Maybeck, *Stochastic models, estimation, and control*, vol. 3, Academic press, 1982.

[77] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 3562–3568.

[78] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[79] R. C. DuToit, J. A. Hesch, E. D. Nerurkar, and S. I. R., "Consistent map-based 3d localization on mobile devices," *arXiv preprint arXiv:1604.08087*, 2016.

[80] R. Madhavan, K. Fregene, and L. E. Parker, "Distributed heterogeneous outdoor multi-robot localization," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2002, vol. 1, pp. 374–381.

[81] MATLAB, *version 8.1.0.604 (R2013a)*, The MathWorks Inc., Natick, Massachusetts, 2013.

[82] Y. Chen, *Robustness Properties of Generalized Correlation Coefficients, with Applications to Cross-over Designs*, ProQuest, 2006.

[83] "Operating and safety manual. robomow rl500/rl550 rl800/rl850," http://olivernash.org/2010/06/26/discman-powered-mower/MANUAL_DOC0012A.pdf, Accessed: 2014-06-30.

[84] "Arduino mega 2560 rev3," http://www.robotshop.com/ca/en/arduino-mega-2560-microcontroller-rev3.html, Accessed: 2015-06-30.

[85] "Arduino compatible mega motor shield 13a, 5-28v," http://www.robotshop.com/ca/en/arduino-compatible-mega-motor-shield-1a-5-28v.html, Accessed: 2015-06-30.

[86] "Microsoft kinect for xbox 360," http://www.xbox.com/en-US/xbox-360/accessories/kinect, Accessed: 2015-06-30.

[87] "Quick start guide for kinect in ros," http://wiki.ros.org/openni_launch/Tutorials/QuickStart, Accessed: 2015-06-30.

[88] "Hokuyo laser range finders," http://www.hokuyo-aut.jp/02sensor/index.html#scanner, Accessed: 2015-06-30.

[89] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[90] R. Macknojia, A. Chavez-Aragon, P. Payeur, and R. Laganière, "Experimental characterization of two generations of kinects depth sensors," in *Proceedings of the IEEE International Symposium on Robotic and Sensors Environments*. IEEE, November 2012, pp. 150–155.

[91] "Accuracy of kinect sensor," http://wiki.ros.org/openni_kinect/kinect_accuracy, Accessed: 2015-07-30.

[92] "Ubuntu 12.04 64-bit," http://releases.ubuntu.com/12.04/, Accessed: 2015-07-30.

[93] "Robot operating system (ros)," http://www.ros.org/about-ros/, Accessed: 2015-07-30.

[94] "Openni package," http://wiki.ros.org/openni_launch/Tutorials/QuickStart#Registration:_matching_depth_to_color, Accessed: 2014-08-25.